TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Diplomarbeit

Electric Vehicles Recharge Scheduling with Logic-Based Benders Decomposition

Ausgeführt an der
Fakultät für Mathematik und Geoinformation
der Technischen Universität Wien

unter der Anleitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl

(Institut für Computergraphik und Algorithmen)

und

Univ.Ass. Dipl.-Ing. Martin Riedler

durch

Katharina Ölsböck

Bachgasse 8
3430 Staasdorf

Ort, Datum

Unterschrift

Erklärung zur Verfassung der Arbeit

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit, einschließlich Tabellen und Abbildungen, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ort, Datum

Unterschrift

Abstract

Electric vehicles represent a promising alternative to traditional internal combustion engine vehicles that might support the attainment of settled climate and energy targets. The widespread adoption of electric vehicles is complicated, however, by their need for frequent recharging and the rather long charging duration. Moreover, charging facilities are still relatively scarce and power constraints imposed by the electric grid have to be respected.

We consider the problem of scheduling the recharging of a fleet of electric vehicles at multiple facilities with limited resources. The goal is to recharge as many vehicles as possible. Every vehicle that is scheduled for charging needs a parking spot during the entire time of its stay and shall be recharged as much as possible before departure. Facilities only have a limited number of parking spots and charging machines and a limited amount of power available for charging. Machines can work at several charging rates, but higher ones should be avoided, since they reduce battery lifetime more.

First, we formulate this problem as a mixed-integer programming (MIP) model, which is a standard approach for discrete optimization problems. Moreover, a simple greedy heuristic, which quickly finds a feasible solution, is presented.

The main objective of this thesis is to investigate an alternative method for solving the problem. The so-called logic-based Benders decomposition solves optimization problems iteratively on basis of systematic trial and error. For this purpose, the problem is decomposed into a master and a subproblem. In every iteration, the master problem assigns trial values to some of the variables. Assuming those fixed, optimal values of the remaining variables are determined in the subproblem, which should be much easier to solve than the original one. From the subproblem solution we deduce constraints that are added to the master problem, in order to obtain better trial values in the next iteration.

Logic-based Benders algorithms for two different decompositions of the considered problem are formulated. The subproblems of the first one are mere feasibility problems, while in the second one they are optimization problems as well. Additionally, heuristic boosting strategies are proposed, in which either the master or subproblems are solved heuristically.

The evaluation of all algorithms on randomly generated test instances shows that the MIP model and the first decomposition work quite well. On instances with homogeneous facilities and enough resources available, the first Benders algorithm often terminates after a few iterations and achieves far better results than the compact MIP model. On the other hand, the MIP model frequently has a shorter runtime on instances with other specifications. The second decomposition shows a rather bad performance. In many cases, the algorithm does not find an optimal solution within the time limit. Heuristic boosting leads to an improved runtime on some instances, but for most offers no real advantage. Regarding larger instances, the MIP model cannot be solved at all due to high memory requirements, while the logic-based Benders algorithms at least return bounds on the optimal value. The first Benders algorithm even finds an optimal solution on many of the larger instances.

Kurzfassung

Elektrofahrzeuge stellen eine vielversprechende Alternative zu herkömmlichen Fahrzeugen mit Verbrennungsmotor dar und könnten dabei helfen, die gesetzten Klimaschutz- und Energieziele zu erreichen. Der großflächige Umstieg wird jedoch dadurch erschwert, dass sie häufig aufgeladen werden müssen und das Laden relativ lange dauert. Außerdem gibt es noch immer wenige Ladestationen, und auch die begrenzten Netzkapazitäten müssen berücksichtigt werden.

Wir betrachten das Problem, einen optimalen Plan für das Aufladen einer Flotte von Elektrofahrzeugen bei mehreren Elektrotankstellen mit begrenzten Ressourcen zu finden. Das Ziel ist, so viele Fahrzeuge wie möglich aufzuladen. Jedes Fahrzeug, das einer Tankstelle zugewiesen wird, benötigt während seines gesamten Aufenthalts einen Parkplatz und soll vor seiner Abfahrt so weit wie möglich aufgeladen werden, wobei ein minimaler Energiebedarf erfüllt werden muss. Die Tankstellen haben nur eine begrenzte Anzahl an Parkplätzen und Ladestationen zur Verfügung, und auch die verfügbare Energie ist beschränkt. Die Ladestationen können mit verschiedenen Raten betrieben werden. Höhere Raten sollten jedoch vermieden werden, da sie die Lebensdauer der Batterien stärker beeinträchtigen.

Als Erstes formulieren wir dieses Problem als ein gemischt-ganzzahliges lineares Optimierungsproblem (engl. mixed-integer programming, MIP), was eine Standardmethode für diskrete Optimierungsprobleme darstellt. Des Weiteren wird ein einfacher Greedy-Algorithmus präsentiert, der sehr schnell eine gültige heuristische Lösung findet.

Das Kernziel dieser Diplomarbeit ist, eine alternative Lösungsmethode für das Problem zu untersuchen. Die so genannte logikbasierte Benders-Dekomposition löst Optimierungsprobleme iterativ auf Grundlage von systematischem Versuch und Irrtum. Dazu wird das Problem in ein Master- und ein Subproblem zerlegt. Im Masterproblem werden in jeder Iteration einigen der Variablen Versuchswerte zugewiesen. Im Bezug auf diese Zuordnung werden dann im Subproblem optimale Werte für die restlichen Variablen bestimmt. Dieses sollte deutlich einfacher zu lösen sein als das ursprüngliche Problem. Aus der Lösung des Subproblems werden Bedingungen abgeleitet, die in das Masterproblem eingefügt werden, um in der nächsten Iteration bessere Versuchswerte zu bestimmen.

Wir formulieren logikbasierte Benders-Algorithmen für zwei verschiedene Dekompositionen des betrachteten Problems. In den Subproblemen der ersten Zerlegung muss

nur eine gültige Lösung gefunden werden, während es sich bei der zweiten Dekomposition auch hier um ein Optimierungsproblem handelt. Zusätzlich werden heuristische Beschleunigungsstrategien für die Benders-Algorithmen vorgestellt, bei denen entweder die Master- oder Subprobleme heuristisch gelöst werden.

Die Evaluierung der Algorithmen auf zufallsgenerierten Testinstanzen zeigt, dass das MIP-Modell und die erste Zerlegung recht gut funktionieren. Auf Instanzen mit gleichartigen Tankstellen und ausreichend verfügbaren Ressourcen terminiert der Benders-Algorithmus oft schon nach wenigen Iterationen und erzielt deutlich bessere Ergebnisse als das kompakte MIP-Modell. Andererseits hat das MIP-Modell auf Instanzen mit anderen Spezifikationen häufig eine kürzere Laufzeit. Die zweite Dekomposition weist ein relativ schlechtes Verhalten auf. In vielen Fällen findet der Algorithmus keine optimale Lösung innerhalb der vorgegebenen Zeit. Die heuristischen Beschleunigungsversuche führen auf manchen Instanzen zu verbesserten Laufzeiten, bringen für die meisten aber keinen wirklichen Vorteil. Auf den größeren Instanzen kann das MIP-Modell wegen des hohen Speicherbedarfs gar nicht gelöst werden, während man durch die logikbasierten Benders-Algorithmen zumindest Schranken für den optimalen Wert erhält. Der erste Benders-Algorithmus findet sogar für viele der größeren Instanzen eine optimale Lösung.

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich beim Verfassen dieser Arbeit unterstützt haben.

Mein besonderer Dank gilt meinen Betreuern, die sich viel Zeit für meine Fragen und Probleme genommen haben. Durch ihre hilfreichen Vorschläge und ihre konstruktive Kritik haben sie einen großen Teil zum Gelingen dieser Arbeit beigetragen.

Weiters möchte ich meinen Freundinnen und Freunden danken, die mich durch mein Studium begleitet haben und diese Zeit so unvergesslich machten. Mein herzlicher Dank gilt meinem Freund, der mir immer zur Seite steht.

Speziell bedanken möchte ich mich bei Anita, Judith und Matthias für das genaue Korrekturlesen und ihre hilfreichen Ratschläge zur Arbeit.

Nicht zuletzt danke ich meiner Familie, die mich schon mein ganzes Leben bei all meinen Bestrebungen unterstützt.

Contents

Abstract **v**

Kurzfassung **vii**

Danksagung **ix**

List of Tables **xiii**

1 Introduction **1**
1.1 Motivation and Background 1
1.2 Problem Description 2
1.3 Related Work 3
1.4 Objectives 5
1.5 Thesis Outline 5

2 Methods **7**
2.1 Linear and Integer Programming 7
2.1.1 Linear Programming 7
2.1.2 Duality 8
2.1.3 Integer Programming 9
2.2 Benders Decomposition 10
2.2.1 Classical Benders Decomposition 11
2.2.2 Inference Duality 12
2.2.3 Logic-Based Benders Decomposition 14
2.2.4 Correctness 14

3 The Electric Vehicles Recharge Scheduling Problem (EVRSP) **17**
3.1 General Problem Formulation 17
3.2 Mixed-Integer Programming Formulation 19
3.2.1 Variables and Domains 20
3.2.2 Mixed-Integer Programming Model 20
3.3 Greedy Heuristic 22

4 Logic-Based Benders Decomposition of the EVRSP **25**
4.1 Decomposition 1 25
4.1.1 Master Problem 26
4.1.2 Subproblems 27
4.1.3 Benders Cuts 28

4.1.4 Strengthening the Master Problem . . . 30
4.2 Decomposition 2 . . . 32
4.2.1 Master Problem . . . 32
4.2.2 Subproblems . . . 33
4.2.3 Benders Cuts . . . 34
4.2.3.1 Feasibility Cuts . . . 35
4.2.3.2 Optimality Cuts . . . 35
4.2.4 Strengthening the Master Problem . . . 37
4.3 Hierarchy of Facilities . . . 38

5 Heuristic Boosting of Logic-Based Benders Decomposition 39
5.1 Increasing the Master Problem Optimality Gap of (LBD1) . . . 40
5.2 Increasing the Master or Subproblem Optimality Gap of (LBD2) . . . 40

6 Computational Aspects 43
6.1 Removal of Trivial Inequalities . . . 43
6.2 Removal of Dominated Inequalities from the Relaxed Subproblems . . . 44
6.3 Subsets of Vehicles Responsible for Infeasibility . . . 44
6.4 Already Solved Subproblems . . . 46

7 Computational Experiments 47
7.1 Instance Generation . . . 47
7.2 Results . . . 49
7.2.1 EVRSPbasic . . . 49
7.2.1.1 Details for (LBD1) and (LBD2) . . . 52
7.2.1.2 Boosted Variants of (LBD1) and (LBD2) . . . 55
7.2.2 EVRSPhet . . . 61
7.2.3 EVRSPres . . . 62
7.2.4 EVRSPlarge . . . 62

8 Conclusions and Future Work 67
8.1 Future Work . . . 68

List of Tables

3.1 Notation overview 19

7.1 Sets of instances 49
7.2 Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPbasic 50
7.3 Results of (GREEDY) on the set EVRSPbasic 51
7.4 Detailed results of (LBD1) on the set EVRSPbasic 52
7.5 Detailed results of (LBD2) on the set EVRSPbasic 53
7.6 Results of (LBD1) with and without strengthening the master problems on the set EVRSPbasic 54
7.7 Results of (LBD1BoostM) on the set EVRSPbasic 56
7.8 Results of (LBD2BoostM) on the set EVRSPbasic 57
7.9 Results of (LBD2BoostSU) on the set EVRSPbasic 58
7.10 Results of (LBD2BoostSL) on the set EVRSPbasic 59
7.11 Comparison of the results of the boosted variants of (LBD2) with an initial optimality gap of 0.005 on the set EVRSPbasic 60
7.12 Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPhet 61
7.13 Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPres 63
7.14 Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPlarge 64
7.15 Comparison of the results of the boosted variants of (LBD2), with an initial optimality gap of 0.005, with the basic version and (GREEDY) on the set EVRSPlarge 65

Chapter 1

Introduction

1.1 Motivation and Background

Creating a sustainable environment has become a prominent issue around the world and poses great challenges to our current society. The attainment of settled climate and energy targets requires a contribution from all existing economic sectors. The transportation sector can support this aim by deployment of alternative propulsion systems for both passenger and transport vehicles. Dargay [9] estimates an increase from about 800 million vehicles worldwide in 2002 to over 2 billion in 2030. This will intensify environmental problems regarding the emission of pollutants and greenhouse gases as well as dependence on fossil fuels. [18]

To address these issues, electric vehicles (EV) could represent a promising alternative to traditional internal combustion engine vehicles. The electrification of transportation would reduce both fuel consumption and emissions, and also improve air quality, particularly in urban areas. [10]

In Austria, road transport consumes 91% of the total energy used in the transportation system. Therefore, a large-scale EV penetration in road transport would imply a significant reduction of emissions. A complete substitution of passenger cars with electric vehicles charged with renewable energy would result in an emission reduction of about 75% (93 million tons of carbon dioxide equivalent). [18]

The main disadvantage of electric vehicles compared to conventional ones is their still quite limited cruising range, resulting in the need of frequent recharging [21]. Moreover, the battery charging process takes significantly longer than refueling an internal combustion engine vehicle. Furthermore, recharging facilities are still relatively scarce

and have a limited number of charging stations, and power constraints imposed by the electric grid have to be respected. [4]

Therefore, charging stations must be used efficiently. At public facilities one possibility is to allow customers to reserve a charging station during or before their trip. Knowing the planned parking duration and required amount of energy of all vehicles arriving in the near future, the charging service provider can schedule the charging times and rates efficiently, respecting the constraints imposed by the limited resources and satisfying as many customers as possible. [5]
Also companies with a fleet of electric vehicles with known routes and timetables, e.g., a public transportation company or a carrying business, could optimize the use of available charging stations by scheduling the recharging of vehicles efficiently.

The focus of this work is the scheduling task that has to be completed in this kind of scenarios.

1.2 Problem Description

We consider the problem of scheduling the recharging of a fleet of electric vehicles. Multiple charging facilities, to which the vehicles might be assigned, are available. Figure 1.1 illustrates the scenario. From the point of view of a customer it makes no difference to which facility his or her vehicle gets allocated, since it is assumed that all facilities can be reached in about the same time. At the facility it was assigned to, each vehicle requires a free parking spot during the entire time of its stay and needs to be scheduled at a free charging station. Each vehicle must be recharged with some minimum amount of energy, in order to enable it to reach its next destination.

Scheduling the electric vehicles assigned to a facility, one must respect the constraints resulting from the resource limits. Each facility has a distinct number of parking spots and charging stations. There are at least as many parking spots available as charging stations. Moreover, the facility's power limit has to be respected at all times.

Vehicles can be charged with different rates. It is assumed that all charging machines are identical, which means that they offer the same choice of charging rates. Higher rates result in shorter charging durations but they reduce battery lifetime more. Therefore, they should be used more carefully.

The aim is to find a recharging schedule that satisfies as many customers as possible, while respecting all constraints.

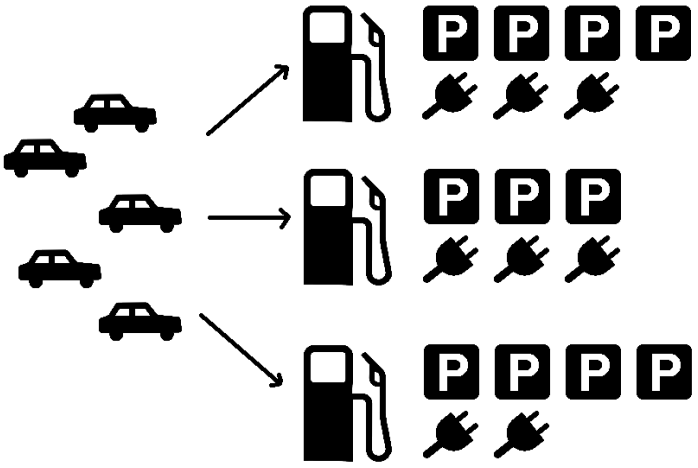

FIGURE 1.1: Electric vehicles get assigned to different charging facilities, each with an individual number of available parking spots and charging stations.

1.3 Related Work

Planning, coordinating and scheduling the recharging of electric vehicles has received much attention in the last years and a lot of work has been done in the field.

Many authors focus on the influence the large-scale adoption of electric vehicles would have on the electric grid and propose scheduling methods in order to avoid grid congestion and to minimize the cost of electricity. Clement et al. [7] consider the charging of multiple electric vehicles at home and the impact on the residential network. While uncoordinated power consumption can lead to grid problems, the coordination of charging can improve power losses and voltage deviations by flattening out peak power. In the model each vehicle is associated with one of four charging periods during the day at the end of which it must be fully recharged. Sundström and Binding [20] try to find the optimal charging schedule for a fleet of vehicles with given connection times and energy demands, while respecting the grid capacity and minimizing the total cost of electricity. Sánchez-Martín and Sánchez [19] focus on the optimal management of recharging electric vehicles at parking garages. A control system is defined that manages the power capacity and decides when each vehicle should start charging. The impact of the system on the total cost is analyzed, taking into account hourly energy prices, daily power capacity prices and non-supplied energy penalties.

Other works also take into account the possibilities offered by a smart grid. Here, vehicles not only take power from the grid by charging their batteries, but are also able to send electricity back into the grid by so-called vehicle-to-grid (V2G) operations. Mal

et al. [15] propose an algorithm that effectively schedules recharging and V2G operations in order to save cost and reduce peak loads. To maximize profit, the algorithm chooses intervals with a cheap electricity price to charge a vehicle and intervals with a high price to send energy back into the grid. Before departure, the vehicle's energy demand must be fulfilled. Zakariazadeh et al. [24] also consider a smart distribution system allowing charging and discharging of electric vehicles. Their scheduling model not only includes technical constraints and profit optimization, but also aims to reduce the emission of air pollutants.

All of the previous models assume that a vehicle is connected to the electric grid during the whole parking time and, therefore, only schedule the time and rate of recharging. They did not consider the number of available charging stations as a limited resource.

Timpner and Wolf [21] focus on the efficient use of scarce charging resources rather than the impact on the power grid. They propose different scheduling algorithms for assigning charging stations upon request, taking the specific requirements of the driver into account. Also, Qin and Zhang [16] want to improve travel efficiency and driver comfort. They aim to minimize the charging waiting time. To achieve this goal, charging stations are networked and collaborate. Vehicles interact with charging stations when approaching them to determine where they should be charged next.

Bessler and Grønbæck [3] also address the efficient operation of a public charging facility. In contrast to other works, they associate a parking time window with each vehicle during which it needs a parking place and recharging has to be completed. Under the assumption that all charging machines are identical, the problem can be split into two subproblems. First, vehicles are allocated to charging machines that they occupy during their whole parking time window, which can be done optimally by a greedy heuristic. Then, the recharging of vehicles is scheduled during the time windows, selecting a charging rate and a starting time for each vehicle while respecting a given power limit at all times. After the allocation step, this can be done independently at each charging station, reducing the task to a one machine problem. Different methods for solving this problem have been analyzed by Bučar [4, 5].

This thesis considers a scheduling task that is similar to the problem formulated by Bessler and Grønbæck [3] but generalizes it in several ways. Instead of scheduling vehicles at a single charging facility with a common power limit for all charging stations, vehicles can be allocated to different facilities, each with a specific power limit and number of charging stations. As a result, the problem cannot easily be split into an allocation and a scheduling part. Additionally, this work makes a distinction between parking and charging spots. Every facility has a number of parking spots, but there are sometimes less connection possibilities for recharging. Each vehicle needs a free parking

spot during the entire time of its stay, whereas it only occupies a charging station when being currently recharged. Another generalization is that vehicles do not have a fixed energy demand that must be fulfilled, but rather an interval of possible amounts of energy they might be charged with. Instead of merely maximizing the number of fulfilled charging requests, we aim to optimize the amount of energy a vehicle gets charged with. As in [3], high charging rates are avoided. This is, however, not done by including this goal in the objective function, but by defining limits for the number of vehicles allowed to be charged with a specific rate per facility.

To our knowledge, the problem considered in this thesis and described in more detail in chapter 3 is formulated and studied for the first time.

1.4 Objectives

The objectives of this work are:

- Introducing the Electric Vehicles Recharge Scheduling Problem and formulating it as a MIP model.
- Describing logic-based Benders decomposition and applying it to the problem.
- Heuristic boosting of the logic-based Benders algorithms.
- Evaluating the algorithms on different sets of test instances.

1.5 Thesis Outline

The thesis is organized in the following way:

- Chapter 2 introduces all methods and modeling techniques used in this work. This includes linear and integer programming as well as Benders decomposition with a special focus on its generalization to logic-based Benders decomposition.
- In Chapter 3 the Electric Vehicles Recharge Scheduling Problem (EVRSP) studied in this thesis is formulated, first in a general way, then as a mixed-integer programming model. Subsequently, a simple greedy heuristic to solve the problem is presented.
- Chapter 4 focuses on the application of logic-based Benders decomposition to the EVRSP. Two different decompositions are presented and described in detail.

- Chapter 5 proposes an approach for heuristic boosting of the logic-based Benders algorithms and explains the necessary adaptations.
- Chapter 6 gives details of the implementation of the algorithms, describing how this can be done efficiently.
- In Chapter 7 the results of computational experiments, which were conducted on different sets of randomly generated instances, are presented and analyzed.
- Finally, Chapter 8 provides a summary of the thesis and discusses possible future work.

Chapter 2

Methods

In this thesis two different methods are applied to the problem of scheduling the recharging of electric vehicles, which is formulated in detail in Chapter 3. Mixed-integer programming (MIP) is a powerful modeling framework for discrete optimization problems [2]. As it has become a standard modeling tool, various commercial solvers are available. Benders decomposition [1] was proposed for large MIP problems of a special structure. The idea was extended to logic-based Benders decomposition (LBD) [13] that allows a much larger class of problems including combinatorial optimization problems. Both modeling techniques are introduced in this chapter.

2.1 Linear and Integer Programming

We give a short introduction to the concepts of linear and integer programming. The first two sections are mainly based on the book of Bertsimas and Tsitsiklis [2], and the third one on the book of Wolsey [22].

2.1.1 Linear Programming

In a *linear programming (LP) problem*, or *linear program*, the goal is to maximize or minimize a given linear function, called the *objective function*, subject to a set of linear equality and inequality *constraints*.

In particular, let $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ be the vector of variables of the problem whose values shall be assigned in an optimal way. They are referred to as *decision variables*. A given *cost vector* $\mathbf{c} = (c_1, c_2, \dots, c_n)^T \in \mathbb{R}^n$ defines the objective function

$\mathbf{c^T x} = \sum_{i=1}^{n} c_i x_i$. Every constraint is defined by a vector $\mathbf{a} \in \mathbb{R}^n$ and a scalar $b \in \mathbb{R}$ and has one of the following forms:

$$\mathbf{a^T x} = \sum_{i=1}^{n} a_i x_i \leq b, \quad \mathbf{a^T x} = \sum_{i=1}^{n} a_i x_i = b \quad \text{or} \quad \mathbf{a^T x} = \sum_{i=1}^{n} a_i x_i \geq b.$$

A vector $\mathbf{x}$ satisfying all constraints is called a *feasible solution*. The aim is to find an *optimal solution*, i.e., a feasible solution $\mathbf{x}^*$ which, depending on the problem, maximizes or minimizes the objective function.

An equality $\mathbf{a^T x} = b$ can be reformulated with two inequality constraints $\mathbf{a^T x} \leq b$ and $\mathbf{a^T x} \geq b$. On the other hand, any constraint of the form $\mathbf{a^T x} \geq b$ can be rewritten as $-\mathbf{a^T x} \leq -b$. Hence, the feasible set of a linear program can be formulated only with inequalities of the form $\mathbf{a^T x} \leq b$. Moreover, minimizing an objective function $\mathbf{c^T x}$ is equivalent to maximizing $-\mathbf{c^T x}$.

Suppose that there are m inequality constraints $\mathbf{a_i^T x_i} \leq b_i$, indexed by $i = 1, ..., m$. Let $\mathbf{b} = (b_1, b_2, \dots, b_m)^T$ and let A be the $m \times n$ matrix whose rows are the vectors $\mathbf{a_i^T}$. It follows from the remarks above that every linear program can be written compactly as

$$\begin{aligned} &\text{maximize} && \mathbf{c^T x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b}. \end{aligned} \tag{2.1}$$

Regarding the solution of linear programs, various effective algorithms have been formulated. The most prominent one is the *simplex method* developed by Dantzig [8]. It is incorporated in most commercial linear programming solvers. Although the simplex method itself has in principle exponential worst case runtime, modern implementations are highly efficient in practice. There are also polynomial time algorithms for linear programming, e.g., the ellipsoid method [14] or so-called interior point methods [23].

2.1.2 Duality

Every linear program can be associated with a related linear program, called its *dual*. Consider the linear program (2.1), which we will refer to as *primal* problem. Let $\mathbf{x}^*$ be an optimal solution, which is assumed to exist. Replacing the set of constraints $A\mathbf{x} \leq \mathbf{b}$ with a penalty term $-\mathbf{u^T}(A\mathbf{x} - \mathbf{b})$ in the objective function, where $\mathbf{u} \geq 0$ is a price vector, leads to a relaxed version of the problem:

$$\begin{aligned} &\text{maximize} && \mathbf{c^T x} - \mathbf{u^T}(A\mathbf{x} - \mathbf{b}) \\ &\text{subject to} && \mathbf{u} \geq 0. \end{aligned} \tag{2.2}$$

The optimal cost $g(\mathbf{u})$ of the relaxed problem, as a function of the price vector $\mathbf{u}$, represents an upper bound on the optimal cost $\mathbf{c^T x^*}$ of the primal problem, since $\mathbf{x}^*$ is a feasible solution to (2.1) and thus $(A\mathbf{x}^* - \mathbf{b}) \leq 0$:

$$g(\mathbf{u}) = \max_{\mathbf{x}} \ \mathbf{c^T x} - \mathbf{u^T}(A\mathbf{x} - \mathbf{b}) \geq \mathbf{c^T x^*} - \mathbf{u^T}(A\mathbf{x}^* - \mathbf{b}) \geq \mathbf{c^T x^*}.$$

The dual problem then consists in searching for the tightest upper bound of this type. For $g(\mathbf{u})$ we further get

$$g(\mathbf{u}) = \max_{\mathbf{x}} \ \mathbf{c^T x} - \mathbf{u^T}(A\mathbf{x} - \mathbf{b}) = \mathbf{u^T b} + \max_{\mathbf{x}} \ (\mathbf{c^T} - \mathbf{u^T}A)\mathbf{x}.$$

Minimizing $g(\mathbf{u})$, we can omit those values of $\mathbf{u}$ for which $(\mathbf{c^T} - \mathbf{u^T}A) \neq 0$, because the respective $g(\mathbf{u})$ is equal to ∞. Therefore, the dual problem of (2.1) can be simply written as

$$\begin{aligned} \text{minimize} \quad & \mathbf{u^T b} \\ \text{subject to} \quad & \mathbf{u^T A} = \mathbf{c^T} \\ & \mathbf{u} \geq 0. \end{aligned} \tag{2.3}$$

The *strong duality theorem* states that if a linear program has an optimal solution, so does its dual, and the respective optimal costs are equal (for a proof see [2]).

2.1.3 Integer Programming

A wide range of real-world problems could be formulated as linear programs, except that in many cases we would have to add the restriction that some or all of the variables must be integers. These problems are referred to as *integer programs*.

Similarly to the linear programming formulation (2.1), any general integer program can be written as

$$\begin{aligned} \text{maximize} \quad & \mathbf{c^T x} + \mathbf{d^T y} \\ \text{subject to} \quad & A\mathbf{x} + B\mathbf{y} \leq \mathbf{b} \\ & \mathbf{y} \text{ integer}, \end{aligned} \tag{2.4}$$

where $\mathbf{x}$ is the vector of real decision variables and $\mathbf{y}$ is the vector of integer variables. If all variables are integers, the problem is called an *integer programming (IP) problem* or *integer program*. A *mixed-integer programming (MIP) problem* denominates a problem, where some variables may take real values.

Although integer programs seem quite similar to linear programs, they are in general much harder to solve. Many integer programs belong to the class of *NP-hard problems* for which no polynomial-time algorithm has been found so far.

Various algorithms for integer programming have been formulated, either solving the problem exactly, approximately or heuristically. Exact algorithms are guaranteed to find an optimal solution but may have exponential runtime. [2]

A common way to solve integer programs are *branch and bound* algorithms. They use a divide-and-conquer approach to explore the set of feasible solutions. Upper bounds (for a maximization problem), also called dual bounds, are obtained by solving a relaxed version of the problem, usually dropping the integrality constraints and thus leading to a linear program. To split the feasible set, a variable y_i is chosen for which the optimal solution y_i^* of the linear program is not integer and either of the two inequalities

$$y_i \leq \lfloor y_i^* \rfloor \quad \text{and} \quad y_i \geq \lceil y_i^* \rceil$$

is added, essentially dividing the problem into two smaller subproblems, which is called *branching*. One after the other, the open subproblems, or *nodes*, are examined, solving the corresponding linear program. If the solution happens to be integer, we have found a feasible integer solution to the problem and can update the value of the incumbent solution, acting as a lower, also called primal, bound. Otherwise, new subproblems are created by branching. To avoid searching the whole feasible set, branches are pruned whenever possible, depending on the current lower and upper bounds.

Various strategies can be used to perform the different steps of the algorithm such as how to choose a node or a branching variable. Commercial systems additionally include a preprocessor which simplifies the problem by reducing the number of constraints and variables.

2.2 Benders Decomposition

Benders decomposition, as introduced in [13], is a method of solving optimization problems iteratively on a trial and error basis. In every iteration, some variables get assigned trial values. Assuming those fixed, optimal values of the remaining variables are determined. From this solution we might deduce information about the quality of other trial solutions. It is used in form of new constraints, called *Benders cuts*, that are added to the original problem. New trial values consistent with all Benders cuts are determined. The algorithm terminates when a trial solution was found to be optimal. In the ideal case, it enumerates only a few of the possible trial values.

The method, first proposed by Benders [1], was suggested for large MIP problems with "complicating variables" in the sense that by assuming them fixed the problem becomes significantly easier to solve, e.g., it decouples into multiple independent problems. Classical Benders decomposition requires the subproblem to be linear (with only continuous variables) because Benders cuts are generated by solving its dual. Geoffrion [11] generalized Benders' algorithm to certain non-linear problems by employing non-linear convex duality theory. Hooker and Ottosson [13] extended the approach to an even larger class of problems by defining a generalized concept of duality for any optimization problem. This method is called *logic-based Benders decomposition (LBD)*.

In the following section, which is based on [13], first, classical Benders decomposition is presented. Then, the inference dual of an optimization problem is defined, required for the logic-based Benders algorithm, which is subsequently formulated. Finally, it is shown that the procedure always terminates with the optimal solution value.

2.2.1 Classical Benders Decomposition

Benders decomposition partitions the variables of an optimization problem into two vectors $\mathbf{x} \in \mathbb{R}^n$ and $y \in D_y$, e.g., the continuous and the integer variables of a MIP problem. It applies to problems of the form

$$\begin{aligned} \text{maximize} \quad & \mathbf{c^T x} + f(y) \\ \text{subject to} \quad & A\mathbf{x} + g(y) \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n,\ y \in D_y \end{aligned} \tag{2.5}$$

with vectors $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, a matrix $A \in \mathbb{R}^{m \times n}$, a real-valued function $f(y)$ and a m-component vector-valued function $g(y)$. Assuming y fixed to some trial value $\bar{y} \in D_y$, the following linear subproblem remains:

$$\begin{aligned} \text{maximize} \quad & \mathbf{c^T x} + f(\bar{y}) \\ \text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} - g(\bar{y}) \\ & \mathbf{x} \in \mathbb{R}^n. \end{aligned} \tag{2.6}$$

The classical dual problem (2.3) is

$$\begin{aligned} \text{minimize} \quad & \mathbf{u^T}(\mathbf{b} - g(\bar{y})) + f(\bar{y}) \\ \text{subject to} \quad & \mathbf{u^T A} = \mathbf{c^T} \\ & \mathbf{u} \geq 0. \end{aligned} \tag{2.7}$$

If the feasible set of (2.7) is bounded and non-empty, the dual problem has some finite solution $\mathbf{u}^*$. Its optimal value not only provides a valid bound on the objective value z of (2.5) if $y = \bar{y}$, but it also generates a valid cut for any value of y,

$$z \leq (\mathbf{u}^*)^{\mathbf{T}}(\mathbf{b} - g(y)) + f(\bar{y}) =: \beta_{\bar{y}}(y),$$

which is called a *Benders cut*, or in particular an *optimality cut*. In the case of an infeasible subproblem (an unbounded dual) a *feasibility cut* is derived that cuts away the current $\bar{y}$ from the feasible set of (2.5). An unbounded subproblem implies that the original problem is likewise unbounded.

In every iteration of the algorithm the so-called *master problem* is solved whose constraints include the Benders cuts generated so far. In iteration h' we have

$$\begin{aligned} \text{maximize} \quad & z \\ \text{subject to} \quad & z \leq \beta_{y^h}(y) \quad \forall h \in \{1, \dots, h' - 1\} \\ & y \in D_y, \end{aligned} \tag{2.8}$$

where $y^1, \dots, y^{h'-1}$ are the solution values of the previous master problems. Solving master problem (2.8), we get the next trial value $\bar{y}$ for subproblem (2.6). The solution of its dual leads to a new Benders cut that is added to the master problem and the procedure repeats.

The algorithm terminates if the optimal value z^* of the current master problem is equal to the optimal value β^* of the resulting subproblem dual (2.7). It is summarized as Algorithm 2.1.

2.2.2 Inference Duality

We now consider a general optimization problem

$$\begin{aligned} \text{maximize} \quad & f(x) \\ \text{subject to} \quad & x \in S \\ & x \in D \end{aligned} \tag{2.9}$$

with domain D, feasible set S and where f is a real-valued function. The domain D might be, for example, the set of real vectors or binary vectors of dimension n. The feasible set S is defined by a collection of constraints.

Let $P(x)$ and $Q(x)$ be two propositions whose truth value is a function of x. We say that $P(x)$ *implies* $Q(x)$ *with respect to* D, if $Q(x)$ is true for any $x \in D$ for which $P(x)$

Algorithm: Benders Decomposition

choose an initial $\bar{y} \in D_y$;
$z^* := \infty$; $h := 0$;
while the subproblem dual has a feasible solution $\beta^* < z^*$ **do**
  derive a bounding function $\beta_{\bar{y}}(y)$;
  $h := h + 1$;
  $y^h := \bar{y}$;
  add the Benders cut $z \leq \beta_{y^h}(y)$ to the master problem;
  solve the master problem;
  if the master problem is infeasible **then**
    return "infeasible";
  else
    let $\bar{y}$ be an optimal solution of the master problem
      with objective value z^*;
  end
  solve the subproblem dual;
end
return z^*;

ALGORITHM 2.1: The generic Benders algorithm.

is true, and denote this by

$$P(x) \xrightarrow{D} Q(x).$$

We can now define the *inference dual* of (2.9):

$$\begin{aligned} &\text{minimize} \quad \beta \\ &\text{subject to} \quad x \in S \xrightarrow{D} f(x) \leq \beta. \end{aligned} \tag{2.10}$$

The dual searches for the smallest $\beta \in \mathbb{R}$ for which $f(x) \leq \beta$ can be inferred from the set of constraints, i.e., the tightest possible upper bound on the objective function $f(x)$, which is obviously identical with the optimal objective value. This implies that an optimization problem (2.9) always has the same optimal value as its inference dual (2.10), which is referred to as *strong duality* property.

It is convenient to let the optimal objective value of a maximization problem be ∞ when it is unbounded and $-\infty$ when it is infeasible, and vice versa for a minimization problem.

Hooker and Ottosson [13] have shown that the classical dual problem of a linear program has the same optimal value as its inference dual.

2.2.3 Logic-Based Benders Decomposition

By means of the inference dual, we can now extend Benders decomposition to general optimization problems. As in the classical case, the variables of problem (2.9) are partitioned into two vectors x and y that belong to the domains D_x and D_y, respectively. This leads to an optimization problem of the form

$$\begin{aligned} \text{maximize} \quad & f(x,y) \\ \text{subject to} \quad & (x,y) \in S \\ & x \in D_x,\ y \in D_y \end{aligned} \tag{2.11}$$

with objective function f and feasible set S. Fixing y to some trial value $\bar{y} \in D_y$, we get the following subproblem,

$$\begin{aligned} \text{maximize} \quad & f(x,\bar{y}) \\ \text{subject to} \quad & (x,\bar{y}) \in S \\ & x \in D_x, \end{aligned} \tag{2.12}$$

with inference dual

$$\begin{aligned} \text{minimize} \quad & \beta \\ \text{subject to} \quad & (x,\bar{y}) \in S \xrightarrow{D_x} f(x,\bar{y}) \leq \beta. \end{aligned} \tag{2.13}$$

Solving the dual consists in finding the tightest upper bound β^* on the objective value that can be inferred from the constraints, assuming y is fixed to $\bar{y}$. The solution can be viewed as a proof that β^* is the tightest bound, given that $y = \bar{y}$. The basic idea of logic-based Benders decomposition is to use the same line of argument to deduce a function $\beta_{\bar{y}}(y)$ that gives a valid upper bound on the objective value of the original problem (2.11) for any fixed value of y. This results in a generalized Benders cut

$$z \leq \beta_{\bar{y}}(y),$$

where the subscript $\bar{y}$ denotes which trial value of y has led to the bounding function. Otherwise, logic-based Benders decomposition proceeds in the same way as the classical algorithm of Section 2.2.1.

2.2.4 Correctness

The following theorems ensure that, in the case of a feasible problem, the logic-based Benders algorithm always terminates with the optimal value if all Benders cuts satisfy

certain properties and D_y is finite.

Theorem 2.1. *Suppose that in every iteration of the logic-based Benders algorithm, the bounding function $\beta_{\bar{y}}(y)$ satisfies the following property:*

(B1) *$\beta_{\bar{y}}(y)$ is a valid upper bound on $f(x, y)$, i.e., $\beta_{\bar{y}}(y) \geq f(x, y)$, for any feasible solution (x, y) of (2.11).*

Then, if the algorithm terminates with an optimal solution y^* and objective value z^* in the master problem, the original problem (2.11) has an optimal solution (x^*, y^*) with objective value $f(x^*, y^*) = z^*$.
If it terminates with an infeasible master problem, then (2.11) is infeasible. If it terminates with an infeasible subproblem dual, then (2.11) is unbounded.

Proof. The optimal value of each subproblem (2.12) always provides a lower bound on the optimal value of the original maximization problem (2.11), since it is the same problem, only with the additional constraint that y is fixed to some trial value $\bar{y}$. Due to strong duality, the same holds for the optimal value of each subproblem dual (2.13).
On the other hand, (B1) implies that the objective values of all feasible solutions of (2.11) respect the constraints of every master problem (2.8) (corresponding to the Benders cuts). Hence, the optimal value of every master problem constitutes an upper bound on the optimal objective value of (2.11).
Now, first suppose that the algorithm has terminated with an optimal solution y^* and corresponding objective value $z^* \in \mathbb{R}$ in the master problem. Then, because of the termination criterion, z^* is equal to the optimal value β^* of the last subproblem dual. It follows from the considerations above that z^* is an upper bound and β^* is a lower bound on the optimal value of (2.11). Thus, $z^* = \beta^*$ is the optimal value of the problem. Since the last subproblem dual was feasible, the primal subproblem has some optimal solution x^* with objective value β^*. Then, (x^*, y^*) is an optimal solution of (2.11).
An infeasible master problem implies infeasibility of the original problem, since all Benders cuts are valid.
In the case of an infeasible subproblem dual, the primal subproblem is unbounded. Hence, (2.11) is also unbounded. □

Theorem 2.2. *Suppose that in every iteration of the logic-based Benders algorithm, the bounding function $\beta_{\bar{y}}(y)$ satisfies* (B1) *and*

(B2) *$\beta_{\bar{y}}(\bar{y}) = \beta^*$, where β^* is the optimal value of the subproblem dual for trial value $\bar{y}$.*

Then, if the subproblem dual is solved to optimality and D_y is finite, the logic-based Benders algorithm terminates.

Proof. Since D_y is finite, there is only a finite number of possible subproblems (2.12), each one arising from fixing y to some $\bar{y} \in D_y$. In the worst case, the algorithm iterates over all values $\bar{y} \in D_y$. We show that any subproblem never gets examined twice, except maybe in the last iteration.

If the same subproblem is ever solved a second time, the solution y^* of the current master problem must be equal to the solution y^h of an earlier iteration. Therefore, the master problem must already contain a Benders cut $z \leq \beta_{y^h=y^*}(y)$. Due to (B2), the objective value z^* of the current master solution y^* must fulfill $z^* \leq \beta_{y^*}(y^*) = \beta^*$, where β^* is the optimal value of the corresponding subproblem. Therefore, z^* must be equal to β^*, which is the termination criterion of the algorithm. □

Chapter 3

The Electric Vehicles Recharge Scheduling Problem (EVRSP)

This chapter provides a formal definition of the *Electric Vehicles Recharge Scheduling Problem (EVRSP)* studied in this thesis. First, it is formulated in a general way. Then, a MIP model of the problem is described and finally a greedy heuristic to solve the EVRSP is presented.

3.1 General Problem Formulation

We consider a set J of n electric vehicles to be recharged, a set I of m charging facilities, a set $R = \{r_{\min}, \dots, r_{\max}\}$ of possible charging rates and a time horizon $[0, t_{\max}]$. In the models the time horizon is discretized in $t_{\max} + 1$ time intervals, $T := \{0, 1, \dots, t_{\max}\}$.

Each electric vehicle $j \in J$ has a specific parking time and energy demand. Vehicle j will arrive at a facility at time $t_{\text{arr},j}$ and will depart at time $t_{\text{dep},j}$. These are the same for all charging facilities. During the whole time interval $[t_{\text{arr},j}, t_{\text{dep},j}]$ the vehicle needs a parking spot. Vehicle j needs to be charged with an energy amount of at least $e_{\min,j}$ and at most $e_{\max,j}$, the latter corresponds to the amount of energy required to recharge the vehicle completely.

Each charging facility $i \in I$ is characterized by its limited number of resources. There are a_i parking spots and b_i charging spots available, with $b_i \leq a_i$. The maximum total power available at facility i is P_i, which is assumed to be constant over time.

A solution to the problem consists of a subset of vehicles $\hat{J} \subseteq J$ to be recharged and a feasible charging schedule for each of them. This schedule defines the facility i, to which

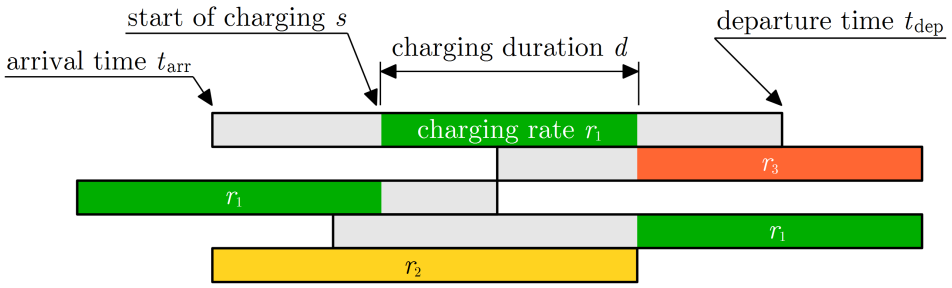

FIGURE 3.1: Example of charging schedules of five vehicles assigned to the same facility.

the vehicle is assigned, the starting time s and duration d of charging, as well as the charging rate r. An example of charging schedules is depicted in Figure 3.1.

Starting time s and duration d have to be chosen in such a way that the vehicle is charged during its parking time $[t_{\text{arr},j}, t_{\text{dep},j}]$. Charging periods are necessarily contiguous. For each facility it has to be ensured that there are sufficient parking and charging spots for all assigned vehicles. The minimum energy demand of each vehicle j that was assigned to some facility has to be satisfied. It is assumed that this is theoretically possible for all vehicles $j \in J$, i.e., they must satisfy $r_{\max} \cdot (t_{\text{dep},j} - t_{\text{arr},j}) \geq e_{\min,j}$.

All charging machines offer the same choice of charging rates. During a charging process the rate cannot be changed, which means that each vehicle is recharged with a specific rate. The sum of currently used charging rates must not exceed the total power limit of a facility at all times.

To avoid high charging rates, the number of vehicles charged with rate r at each facility is limited by the value $\rho(r)$ of some monotonically decreasing function $\rho : R \to [0, n]$, which is the same for all facilities. We define $\rho(r)$ as

$$\rho(r) := \begin{cases} n & \text{if } r = r_{\min} \\ \max\left(1, \left(\frac{-(1-\alpha)\cdot r}{r_{\max}-r_{\min}} + 1 + \frac{(1-\alpha)\cdot r_{\min}}{r_{\max}-r_{\min}}\right) \cdot \frac{n}{m}\right) & \text{else} \end{cases}$$

with $0 \leq \alpha \leq 1$. The second term in the maximum function defines a decreasing affine function with value $\frac{n}{m}$ at $r_{\min}$ and value $\alpha \cdot \frac{n}{m}$ at $r_{\max}$. This means that about $(\alpha \cdot 100)\%$ of all vehicles are allowed to be charged with the highest rate, while quotas are equally divided between all facilities. If the second term is lower than 1, the function takes 1 as value instead. In this way, every rate can be used at least once. As we do not want to limit the number of vehicles charged with the lowest rate, the value of $\rho(r_{\min})$ is set to n.

A solution fulfilling all the constraints defined above is called *feasible*. It is not necessary that all vehicles are being scheduled. The aim is to assign as many vehicles as possible and to satisfy their energy demands as much as possible. Let e_j be the fulfilled energy demand of vehicle j. The *profit* earned for charging vehicle $j \in \hat{J}$ is defined as

$$p_j := \frac{e_j}{e_{\max,j}} \ .$$

The goal is to find a feasible solution with maximum total profit

$$\max \sum_{j \in \hat{J}} p_j \ .$$

Table 3.1 summarizes the notation of variables and parameters.

Symbol	Meaning
J	set of vehicles
n	number of vehicles
I	set of charging facilities
m	number of charging facilities
R	set of charging rates (in kW)
n_R	number of charging rates
$r_{\min}$	lowest charging rate available
$r_{\max}$	highest charging rate available
$[0, t_{\max}]$	considered time horizon
T	set of discrete time intervals $\{0, 1, ..., t_{\max}\}$
$t_{\text{arr},j}$	arrival time of vehicle j
$t_{\text{dep},j}$	departure time of vehicle j
$e_{\min,j}$	minimum energy demand (in kWh) of vehicle j
$e_{\max,j}$	maximum energy demand (in kWh) of vehicle j
a_i	number of parking spots at facility i
b_i	number of charging spots at facility i
P_i	available power (in kW) at facility i
$\rho(r)$	maximum number of vehicles charged with rate r per facility
$\hat{J} \subseteq J$	subset of vehicles scheduled for recharging
i_j	facility to which vehicle j is allocated for recharging
s_j	starting time of charging vehicle j
d_j	duration of charging vehicle j
r_j	charging rate for vehicle j

TABLE 3.1: Notation overview.

3.2 Mixed-Integer Programming Formulation

We now formulate the Electric Vehicles Recharge Scheduling Problem (EVRSP), which was defined in the last section, as a mixed-integer program.

3.2.1 Variables and Domains

The model will be formulated in terms of binary variables x_{ijrds} which indicate whether vehicle $j \in J$ is scheduled to be charged at facility $i \in I$, starting at time $s \in T$ with duration $d \in T$ and charging rate $r \in R$. Therefore, the number of decision variables is $\mathcal{O}(|I| \cdot |J| \cdot |R| \cdot t_{\max}^2)$.

Not all variables x_{ijrds} need to be considered in the model. We can immediately exclude a great number of infeasible combinations from the indices' domains. For each vehicle j and fixed charging rate r, the charging duration (rounded to the next integer) must lie in $D_{jr} := \{d \in T \,|\, \lceil e_{\min,j}/r \rceil \leq d \leq \lceil e_{\max,j}/r \rceil\}$. Now, let d be fixed additionally, then the only possible starting times are in $S_{jd} := \{s \in T \,|\, t_{\text{arr},j} \leq s \leq t_{\text{dep},j} - d\}$.

In addition to the binary variables x_{ijrds}, auxiliary variables $e_j \in \mathbb{R}$ that represent the fulfilled energy demand of vehicle j are needed to formulate the model.

To summarize, the variables of the model are

$$
\begin{aligned}
x_{ijrds} \in \{0,1\} \quad &\ldots \text{ vehicle } j \text{ is charged at facility } i, \text{ starting at time } s \\
&\quad\ \text{with duration } d \text{ and charging rate } r, \\
e_j \in [0, e_{\max,j}] \ &\ldots \text{ amount of energy vehicle } j \text{ gets recharged with,}
\end{aligned}
$$

where $\quad i \in I$, $j \in J$, $r \in R$, $d \in D_{jr}$ and $s \in S_{jd}$.

3.2.2 Mixed-Integer Programming Model

Let $J(t) := \{j \in J \,|\, t_{\text{arr},j} \leq t \leq t_{\text{dep},j}\} \subseteq J$ be the set of vehicles parked at time t and $S_{jd}(t) := \{s \in T \,|\, \max(0, t-d) \leq s \leq t\} \subseteq S_{jd}$ the set of possible starting times so that t lies within the charging time of vehicle j, fixed to be of duration d. The values of constants $e^*_{jr} \in \mathbb{R}$, which are used in the model, will be defined later. We can then formulate our problem as the following compact mixed-integer program, referred to as (MIP1):

$$
\text{maximize} \quad \sum_{j \in J} \frac{e_j}{e_{\max,j}} \tag{3.1}
$$

$$
\text{subject to} \quad \sum_{\substack{i \in I,\, r \in R,\\ d \in D_{jr},\, s \in S_{jd}}} x_{ijrds} \leq 1 \qquad \forall j \in J \tag{3.2}
$$

$$
\sum_{\substack{j \in J(t),\, r \in R,\\ d \in D_{jr},\, s \in S_{jd}}} x_{ijrds} \leq a_i \qquad \forall i \in I,\, t \in T \tag{3.3}
$$

$$\sum_{\substack{j\in J,\, r\in R,\\ d\in D_{jr},\, s\in S_{jd}(t)}} x_{ijrds} \leq b_i \qquad \forall i \in I,\, t \in T \tag{3.4}$$

$$\sum_{\substack{j\in J,\, r\in R,\\ d\in D_{jr},\, s\in S_{jd}(t)}} x_{ijrds} \cdot r \leq P_i \qquad \forall i \in I\,, t \in T \tag{3.5}$$

$$e_{\min,j} \cdot \sum_{\substack{i\in I,\, r\in R,\\ d\in D_{jr},\, s\in S_{jd}}} x_{ijrds} \leq \sum_{\substack{i\in I,\, r\in R,\\ d\in D_{jr},\, s\in S_{jd}}} x_{ijrds} \cdot r \cdot d \qquad \forall j \in J \tag{3.6}$$

$$\sum_{\substack{i\in I,\, r\in R,\\ d\in D_{jr},\, s\in S_{jd}}} x_{ijrds} \cdot r \cdot d \leq \sum_{\substack{i\in I,\, r\in R,\\ d\in D_{jr},\, s\in S_{jd}}} x_{ijrds} \cdot e^*_{jr} \qquad \forall j \in J \tag{3.7}$$

$$e_j \leq \sum_{\substack{i\in I,\, r\in R,\\ d\in D_{jr},\, s\in S_{jd}}} x_{ijrds} \cdot r \cdot d \qquad \forall j \in J \tag{3.8}$$

$$\sum_{\substack{j\in J,\, d\in D_{jr},\\ s\in S_{jd}}} x_{ijrds} \leq \rho(r) \qquad \forall i \in I,\, r \in R \tag{3.9}$$

$$x_{ijrds} \in \{0,1\} \qquad \forall i \in I,\, j \in J,\, r \in R,\, d \in D_{jr},\, s \in S_{jd} \tag{3.10}$$

$$e_j \in [0, e_{\max,j}] \qquad \forall j \in J. \tag{3.11}$$

The objective function (3.1) sums up the relative values of fulfilled energy demand of all vehicles. As the contribution of each vehicle to the sum is at most 1, it is bounded from above by the total number of vehicles n. This value is reached when all vehicles in J get fully recharged.

The first set of inequalities (3.2) states that every vehicle can be scheduled at most once.

The following inequalities (3.3)-(3.5) ensure that for all facilities i and all time intervals t the facility's resource limits are respected. Firstly, the number of vehicles parked at facility i at time t must not exceed the number of parking spots a_i. Secondly, the number of vehicles currently charged, which are those for which t lies within the planned charging time $\{t' \in T \mid s \leq t' \leq s + d\}$, must not be greater than the number of charging spots b_i. Thirdly, the sum of currently used charging rates r must not exceed the facility's power limit P_i.

The set of inequalities (3.6) states that for every vehicle j, which is scheduled for recharging, the minimum energy demand $e_{\min,j}$ has to be satisfied.

Recharging should be stopped when a vehicle's battery is fully charged, i.e., when its maximum energy demand $e_{\max,j}$ is fulfilled. Formulating the corresponding inequality constraint, one must pay attention to the fact that only integer values are allowed for the duration of charging d in the model. However, the exact duration to satisfy a vehicle's maximum energy demand $e_{\max,j}$ might not be integer. In this case d should take the

next integer value, i.e., be rounded up. Then, the amount of theoretically charged energy might be a bit higher than $e_{\max,j}$, namely $\left\lceil \frac{e_{\max,j}}{r} \right\rceil \cdot r =: e^*_{jr}$ for charging with rate r. This value is used instead of $e_{\max,j}$ as upper limit for the amount of recharged energy in (3.7).

To measure the profit generated by a vehicle, we, however, need to set $e_{\max,j}$ as upper bound for the fulfilled energy demand in order to ensure that the objective value for a single vehicle, corresponding to its relative value of fulfilled energy demand, is less than or equal to 1. This is why we need the auxiliary variables e_j that represent the actually fulfilled energy demand of vehicle j. The value of e_j is the minimum of $e_{\max,j}$ and the amount of theoretically charged energy of vehicle j (corresponding to the sum on the right-hand side of (3.8)).

In the last set of inequalities (3.9), for all rates r the maximum number of vehicles allowed to be charged with this rate is limited by $\rho(r)$ at all facilities.

3.3 Greedy Heuristic

A greedy heuristic for solving the EVRSP was designed in order to be able to compare the results of the other algorithms developed in this thesis with a much simpler approach. We need to check whether the additional time spent on solving the problem really leads to a better solution.

The greedy heuristic solves the EVRSP in a straightforward way. In a preliminary step, the vehicles get sorted in such a way that the algorithm first tries to allocate the most "difficult" ones, which are those vehicles j where the minimum energy demand is high in comparison to the parking duration, i.e., where the value $(t_{\text{dep},j} - t_{\text{arr},j})/e_{\min,j}$ is low. The algorithm then considers the vehicles one by one, trying to assign a feasible charging schedule.

Since high charging rates should be avoided, the lowest possible rate is assigned to each vehicle. For vehicle j this is the lowest $r \in R$, for which the minimum energy demand can be fulfilled during the parking time, i.e., $(t_{\text{dep},j} - t_{\text{arr},j}) \cdot r \geq e_{\min,j}$, and which has not been used too often yet. We start at the first facility and check whether there are still enough parking spots available. If not, we try to allocate the vehicle to the next facility. As its energy demand should be satisfied as much as possible, the algorithm searches for a charging schedule of longest possible duration. For each charging duration $d \in D_{jr} = \{d \in T \mid \lceil e_{\min,j}/r \rceil \leq d \leq \lceil e_{\max,j}/r \rceil\}$ (starting with the longest one), for one starting time $s \in S_{jd}$ after the other, it is tested whether there are still enough charging resources available during the corresponding charging time. If a feasible schedule is

found, we assign the vehicle and update the resource limits. If none is found, we go on to the next facility and search for a feasible charging schedule there. The next vehicle is considered when the current one was successfully assigned or it was not possible to charge it at any of the facilities.

The complete procedure is given as Algorithm 3.1. It will be referred to as (GREEDY).

Algorithm: Greedy Heuristic

```
1  initialize variables for the resource limits (parking and charging
   spots, power) ∀i ∈ I, t ∈ T;
2  sort list of vehicles J by (t_dep,j − t_arr,j)/e_min,j in ascending order;
3  for j ∈ J do
4      schedule_found := false;
5      determine lowest rate r_j ∈ R for which (t_dep,j − t_arr,j) · r_j ≥ e_min,j and
       which has not been used up to the limit yet;
6      for i ∈ I do
7          if not enough parking spots at facility i then
8              continue;
9          end
10         for d := ⌈e_max,j/r⌉, ..., ⌈e_min,j/r⌉ do
11             for s ∈ S_jd do
12                 if all resource limits are respected for this charging
                   schedule of vehilce j then
13                     schedule_found := true;
14                     s_j := s;
15                     break;
16                 end
17             end
18             if schedule_found then
19                 d_j := d;
20                 break;
21             end
22         end
23         if schedule_found then
24             i_j := i;
25             break;
26         end
27     end
28     if schedule_found then
29         allocate vehicle j to facility i_j and assign charging rate r_j,
           duration d_j and starting time s_j;
30         update resource limits;
31     end
32 end
```

Algorithm 3.1: Greedy heuristic for solving the EVRSP.

Chapter 4

Logic-Based Benders Decomposition of the EVRSP

Hooker [12] explored how logic-based Benders decomposition can be applied to planning and scheduling problems and achieved substantial speedups relative to the state of the art. In this chapter, we develop logic-based Benders algorithms for solving the EVRSP. Two different decompositions are proposed and described in detail.

In the first one, vehicles get allocated to a facility and also get a charging rate and duration assigned in the master problem, while the subproblem merely determines feasible starting times for the given assignment. In the second decomposition, the master problem only allocates vehicles to facilities. Optimal recharging schedules are calculated in the subproblem.

For both decompositions, first the master problem and the subproblem are formulated. Then, Benders cuts are derived and finally a relaxation of the subproblem is given that can be included within the master problem to strengthen it.

4.1 Decomposition 1

In the first decomposition of the EVRSP vehicles get assigned a facility, a charging rate and a charging duration in the master problem. It only remains to calculate the starting times of charging in the subproblem. Since the resulting total profit is already determined in the master problem, the subproblem is a mere feasibility problem, where a feasible schedule needs to be calculated for each facility. The algorithm corresponding to this decomposition will be referred to as (LBD1).

4.1.1 Master Problem

The decision variables of the master problem are $x_{ijrd} \in \{0,1\}$ indicating whether vehicle j is assigned to facility i with charging rate r and duration d. We consider the master problem of iteration h'. Let $H := \{1, 2, \dots, h'-1\}$ be the set of previous iterations. We get the following problem,

$$\text{maximize} \quad z \tag{4.1}$$

$$\text{subject to} \quad \sum_{j \in J} \frac{e_j}{e_{\max,j}} = z \tag{4.2}$$

$$\sum_{\substack{i \in I,\, r \in R,\\ d \in D_{jr}}} x_{ijrd} \leq 1 \qquad \forall j \in J \tag{4.3}$$

$$\sum_{\substack{j \in J(t),\, r \in R,\\ d \in D_{jr}}} x_{ijrd} \leq a_i \qquad \forall i \in I,\, t \in T \tag{4.4}$$

$$e_{\min,j} \cdot \sum_{\substack{i \in I,\, r \in R,\\ d \in D_{jr}}} x_{ijrd} \leq \sum_{\substack{i \in I,\, r \in R,\\ d \in D_{jr}}} x_{ijrd} \cdot r \cdot d \qquad \forall j \in J \tag{4.5}$$

$$\sum_{\substack{i \in I,\, r \in R,\\ d \in D_{jr}}} x_{ijrd} \cdot r \cdot d \leq \sum_{\substack{i \in I,\, r \in R,\\ d \in D_{jr}}} x_{ijrd} \cdot e^*_{jr} \qquad \forall j \in J \tag{4.6}$$

$$e_j \leq \sum_{\substack{i \in I,\, r \in R,\\ d \in D_{jr}}} x_{ijrd} \cdot r \cdot d \qquad \forall j \in J \tag{4.7}$$

$$\sum_{j \in J,\, d \in D_{jr}} x_{ijrd} \leq \rho(r) \qquad \forall i \in I, r \in R \tag{4.8}$$

$$z \leq \beta_{\mathbf{x}^h}(\mathbf{x}) \qquad \forall h \in H \tag{4.9}$$

$$z \in \mathbb{R} \tag{4.10}$$

$$e_j \in [0, e_{\max,j}] \qquad \forall j \in J \tag{4.11}$$

$$x_{ijrd} \in \{0,1\} \qquad \forall i \in I,\, j \in J,\, r \in R,\, d \in D_{jr}, \tag{4.12}$$

where the domain D_{jr}, the set $J(t)$ and the constants e^*_{jr} are defined as in Section 3.2.

The set of inequalities (4.3) states that every vehicle can be allocated at most once. The parking spots limit of all facilities is provided by (4.4). Inequalities (4.5)-(4.7) ensure that the energy demands of all vehicles that are allocated to some facility are fulfilled. As explained in Section 3.2.2, the sum on the right-hand side of (4.7) corresponds to the amount of energy vehicle j is theoretically recharged with during the whole charging

duration, whereas the variables e_j represent the actually fulfilled energy demand (additionally bounded above by $e_{\max,j}$). The set of inequalities (4.8) enforces the limit on the number of charging rates being used at each facility.

The value of the objective function (4.1) corresponds to the sum of relative values of fulfilled energy demand per vehicle (4.2). The Benders cuts (4.9) generated in the previous iterations of the algorithm provide a valid bound on the objective value. They are formulated in Section 4.1.3.

A solution to the master problem corresponds to an allocation of vehicles to facilities and an assignment of charging rates and durations to each of these vehicles that generates the highest possible profit (vehicles get recharged as much as possible) among those not proven to be infeasible yet (excluded by a Benders cut). Feasibility of this trial assignment is examined in the subproblems.

4.1.2 Subproblems

Assuming that in iteration h every vehicle j of a subset $\hat{J}^h \subseteq J$ was allocated to a facility and was assigned a charging rate r_j^h and a duration d_j^h in the master problem, the remaining problem decouples into multiple independent subproblems, one for each facility.

We consider the subproblem of facility i. Let J_i^h be the set of vehicles assigned to i in iteration h. It remains to determine starting times of charging for all vehicles $j \in J_i^h$ in such a way that the facility's resource limits are not exceeded. We formulate the subproblem in terms of binary variables x_{js} indicating whether the charging of vehicle $j \in J_i^h$ is scheduled to start at time $s \in S_j^h := \{s \in T \mid t_{\text{arr},j} \leq s \leq t_{\text{dep},j} - d_j^h\}$. The subproblem of facility i can then be written as

$$\text{maximize} \quad \sum_{j \in J_i^h} \frac{\min(r_j^h \cdot d_j^h, e_{\max,j})}{e_{\max,j}} \tag{4.13}$$

$$\text{subject to} \quad \sum_{s \in S_j^h} x_{js} = 1 \qquad \forall j \in J_i^h \tag{4.14}$$

$$\sum_{j \in J_i^h,\, s \in S_j^h(t)} x_{js} \leq b_i \qquad \forall t \in T \tag{4.15}$$

$$\sum_{j \in J_i^h,\, s \in S_j^h(t)} x_{js} \cdot r_j^h \leq P_i \qquad \forall t \in T \tag{4.16}$$

$$x_{js} \in \{0,1\} \qquad \forall j \in J_i^h,\, s \in S_j^h, \tag{4.17}$$

where $S_j^h(t) := \{s \in T \mid \max(0, t - d_j^h) \leq s \leq t\} \subseteq S_j^h$ is the set of possible starting times so that time t lies within the charging time of vehicle j.

The objective function (4.13) is constant since the profit of facility i is already determined in the master problem. It is sufficient to find a feasible schedule. If the subproblem is infeasible its optimal objective value is $-\infty$ by convention.

The set of equalities (4.14) states that every vehicle $j \in J_i^h$ must get exactly one starting time $s \in S_j^h$ assigned. Inequalities (4.15) ensure that not more vehicles are being charged than there are charging spots available at all times. Finally, the power limit constraint of facility i is provided by the set of inequalities (4.16).

4.1.3 Benders Cuts

In iteration h the decision variables of the master problem x_{ijrd} are fixed to values x_{ijrd}^h, which are obtained by solving the current problem. Then, the resulting subproblem for each facility is solved. For the subproblem of some facilities we get a feasible solution, whereas others prove to be infeasible.

From the solution of the subproblems we must infer a Benders cut of the form

$$z \leq \beta_{\mathbf{x}^h}(\mathbf{x})$$

with a linear bounding function $\beta_{\mathbf{x}^h}(\mathbf{x})$, where z is the objective value of the master problem, $\mathbf{x}$ denotes the array of decision variables $(x_{ijrd})_{i \in I,\, j \in J_i^h,\, r \in R,\, d \in D_{jr}}$ and $\mathbf{x}^h$ the array of trial values in iteration h.

To be valid, the cut must satisfy properties (B1) and (B2) of Section 2.2.4. In particular, it has to fulfill the following conditions:

(B1) For all feasible values of $\mathbf{x}$ the term $\beta_{\mathbf{x}^h}(\mathbf{x})$ provides a valid upper bound on the objective value z.

(B2) $\beta_{\mathbf{x}^h}(\mathbf{x}^h) = \beta^*$, where β^* is the optimal value of the subproblem (in our case the sum of optimal values of all subproblems) in iteration h.

Since we have independent subproblems, we derive a bounding function $\beta_{\mathbf{x}_i^h}(\mathbf{x}_i)$ for every facility i and combine them to a bounding function on the total profit,

$$z \leq \beta_{\mathbf{x}^h}(\mathbf{x}) := \sum_{i \in I} \beta_{\mathbf{x}_i^h}(\mathbf{x}_i), \tag{4.18}$$

where $\mathbf{x}_i$ denotes the array of variables $(x_{ijrd})_{j \in J_i^h, r \in R, d \in D_{jr}}$ and $\mathbf{x}_i^h$ the array of trial values for facility i in iteration h.

We define bounding functions similar to those in [12]. In the case of a feasible subproblem at facility i we get the following trivial bound,

$$\beta_{\mathbf{x}_i^h}(\mathbf{x}_i) := \sum_{j \in J_i^h} \frac{e_j}{e_{\max,j}},$$

which is the profit generated at facility i for the assignment $\mathbf{x}_i$. If the subproblem is infeasible, the most obvious cut simply excludes the assignment that led to this subproblem from the feasible set of the master problem. The corresponding bounding function is

$$\beta_{\mathbf{x}_i^h}(\mathbf{x}_i) := \begin{cases} -\infty & \text{if } \mathbf{x}_i = \mathbf{x}_i^h, \\ \sum_{j \in J_i^h} \frac{e_j}{e_{\max,j}} & \text{else.} \end{cases}$$

This bound cannot be satisfied in the case $\mathbf{x}_i = \mathbf{x}_i^h$ and is trivial otherwise.

Let $I^h \subseteq I$ be the set of facilities for which the corresponding subproblem is infeasible in iteration h. We can then rewrite the Benders cut (4.18) as the following set of inequalities,

$$\sum_{j \in J_i^h} (1 - x_{ijr_j^h d_j^h}) \geq 1 \quad \forall i \in I^h, \tag{4.19}$$

which requires that the assignment of at least one vehicle must be changed at each facility where the subproblem was infeasible.

As explained in [12], the bound for facility i can be strengthened in many cases by identifying a subset $\tilde{J}_i^h \subset J_i^h$ of vehicles assigned to i that is responsible for infeasibility and cannot be reduced further. To calculate such a subset, we iteratively reduce the set using a simple greedy heuristic which is formulated as Algorithm 4.1. Replacing J_i^h by $\tilde{J}_i^h$ in (4.19), the Benders cut remains valid. One must pay attention to the fact that the subset $\tilde{J}_i^h$ of vehicles responsible for infeasibility is not unique. The set computed by Algorithm 4.1 depends on the order in which it iterates over J_i^h. We might also calculate multiple subsets $\tilde{J}_i^h \subset J_i^h$ and add a cut (4.19) for each of them in iteration h. Details are given in Section 6.3.

If all subproblems are feasible, the sum of their objective values is equal to the optimal value of the current master problem, which means that the termination criterion of the algorithm is met. Indeed, we have found a feasible recharging schedule that generates the highest possible profit.

Algorithm: Subset of Vehicles Responsible for Infeasibility

```
1 J̃_i^h := J_i^h;
2 for j ∈ J_i^h do
3 |   resolve the subproblem of facility i over the set of
  |   vehicles J̃_i^h\{j};
4 |   if the subproblem is still infeasible then
5 |   |   J̃_i^h := J̃_i^h\{j};
6 |   end
7 end
8 return J̃_i^h
```

ALGORITHM 4.1: Computing a subset $\tilde{J}_i^h \subseteq J_i^h$ of vehicles responsible for infeasibility of subproblem i.

4.1.4 Strengthening the Master Problem

Hooker [12] suggests that the master problem can frequently be strengthened by including inequalities derived from a relaxation of the subproblem within the master problem.

First, we consider the subproblems' charging spots limit (4.15). For $t_1, t_2 \in T$, let $J(t_1, t_2) := \{j \in J \mid [t_{\text{arr},j}, t_{\text{dep},j}] \subseteq [t_1, t_2]\}$ be the set of vehicles to be scheduled within the time window $[t_1, t_2]$. A charging duration of d indicates that the vehicle occupies a charging spot for d time intervals. Altogether, there are $b_i \cdot (t_2 - t_1)$ slots available within the time window $[t_1, t_2]$, since facility i has b_i charging stations. Following these considerations, we can formulate an aggregated charging spots constraint for each time window $[t_1, t_2]$ with $0 \leq t_1 < t_2 \leq t_{\max}$ and every facility $i \in I$:

$$\sum_{\substack{j \in J(t_1,t_2),\\ r \in R,\, d \in D_{jr}}} x_{ijrd} \cdot d \leq b_i \cdot (t_2 - t_1). \tag{4.20}$$

A relaxed version of the power limit constraint (4.16) can be defined in a similar way. A vehicle that was assigned a charging rate r and a duration d is recharged with an energy amount of $r \cdot d$. Resulting from the power limit P_i at facility i, the maximum amount of energy that can be consumed within the time window $[t_1, t_2]$ is $P_2 \cdot (t_2 - t_1)$. This leads to the following aggregated power limit constraint for time window $[t_1, t_2]$ with $0 \leq t_1 < t_2 \leq t_{\max}$ and facility $i \in I$:

$$\sum_{\substack{j \in J(t_1,t_2),\\ r \in R,\, d \in D_{jr}}} x_{ijrd} \cdot r \cdot d \leq P_i \cdot (t_2 - t_1). \tag{4.21}$$

It is not necessary to add the inequalities (4.20) and (4.21) for all $t_1, t_2 \in T$ to the master problem. For t_1 we only need to consider the distinct values of the set of arrival times $\{t_{\text{arr},j} \mid j \in J\}$ and for t_2 the departure times $\{t_{\text{dep},j} \mid j \in J\}$.

Moreover, many inequalities may be redundant with the others. We define

$$T^i_{\text{cspots}}(t_1, t_2) := \frac{1}{b_i} \sum_{\substack{j \in J(t_1,t_2),\\ r \in R,\, d \in D_{jr}}} d - t_2 + t_1 \qquad \text{and} \tag{4.22}$$

$$T^i_{\text{power}}(t_1, t_2) := \frac{1}{P_i} \sum_{\substack{j \in J(t_1,t_2),\\ r \in R,\, d \in D_{jr}}} r \cdot d - t_2 + t_1 \tag{4.23}$$

as the *tightness* of inequality (4.20) and (4.21), respectively.

Lemma 4.1. *The relaxed charging spots (4.20) or power limit constraint (4.21) for time window $[t_1, t_2]$ is dominated by the same constraint for a different time window $[u_1, u_2]$, if $[u_1, u_2] \subseteq [t_1, t_2]$ and if the inequality corresponding to $[u_1, u_2]$ has a higher tightness.*

Proof. We provide a proof for the relaxed charging spots constraint. The statement for the power limit constraint follows analogously. Let (4.20) hold for time window $[u_1, u_2]$, let $[u_1, u_2] \subseteq [t_1, t_2]$ and let the tightness (4.22) be higher for $[u_1, u_2]$. We will show that inequality (4.20) for time window $[t_1, t_2]$ is then also fulfilled.

Since the tightness is higher for $[u_1, u_2]$, we get

$$\frac{1}{b_i} \sum_{\substack{j \in J(t_1,t_2),\\ r \in R,\, d \in D_{jr}}} d - t_2 + t_1 \leq \frac{1}{b_i} \sum_{\substack{j \in J(u_1,u_2),\\ r \in R,\, d \in D_{jr}}} d - u_2 + u_1,$$

which we can rewrite as

$$\sum_{\substack{j \in J(t_1,t_2),\\ r \in R,\, d \in D_{jr}}} d - \sum_{\substack{j \in J(u_1,u_2),\\ r \in R,\, d \in D_{jr}}} d \leq -b_i \cdot (u_2 - u_1) + b_i \cdot (t_2 - t_1). \tag{4.24}$$

We note that $J(u_1, u_2) \subseteq J(t_1, t_2)$ and thus $J(t_1, t_2) = J(u_1, u_2) \,\dot\cup\, (J(t_1, t_2) \backslash J(u_1, u_2))$. The proof can now be completed as follows:

$$\begin{aligned}
\sum_{\substack{j \in J(t_1,t_2),\\ r \in R,\, d \in D_{jr}}} x_{ijrd} \cdot d &= \sum_{\substack{j \in J(u_1,u_2),\\ r \in R,\, d \in D_{jr}}} x_{ijrd} \cdot d + \sum_{\substack{j \in J(t_1,t_2) \backslash J(u_1,u_2),\\ r \in R,\, d \in D_{jr}}} x_{ijrd} \cdot d \\
&\leq \sum_{\substack{j \in J(u_1,u_2),\\ r \in R,\, d \in D_{jr}}} x_{ijrd} \cdot d + \sum_{\substack{j \in J(t_1,t_2) \backslash J(u_1,u_2),\\ r \in R,\, d \in D_{jr}}} d
\end{aligned}$$

$$= \sum_{\substack{j\in J(u_1,u_2),\\ r\in R,\, d\in D_{jr}}} x_{ijrd}\cdot d + \Big(\sum_{\substack{j\in J(t_1,t_2),\\ r\in R,\, d\in D_{jr}}} d - \sum_{\substack{j\in J(u_1,u_2),\\ r\in R,\, d\in D_{jr}}} d \Big)$$

$$\overset{(4.20)}{\leq} b_i\cdot(u_2-u_1) + \Big(\sum_{\substack{j\in J(t_1,t_2),\\ r\in R,\, d\in D_{jr}}} d - \sum_{\substack{j\in J(u_1,u_2),\\ r\in R,\, d\in D_{jr}}} d \Big)$$

$$\overset{(4.24)}{\leq} b_i\cdot(u_2-u_1) - b_i\cdot(u_2-u_1) + b_i\cdot(t_2-t_1) = b_i\cdot(t_2-t_1).$$

□

All dominated inequalities are removed in a preprocessing step (see Section 6.2).

4.2 Decomposition 2

In the second decomposition the master problem simply allocates vehicles to charging facilities. An optimal charging schedule for each vehicle is determined in the subproblem. This can be done independently for each facility. The algorithm corresponding to this decomposition will be referred to as (LBD2).

4.2.1 Master Problem

The master problem is formulated in terms of binary variables x_{ij} indicating whether vehicle j is allocated to facility i. Additionally, continuous variables p_i represent the maximum profit that can be generated at facility i. In iteration h' the master problem takes the following form,

$$\text{maximize} \quad \sum_{i\in I} p_i \tag{4.25}$$

$$\text{subject to} \quad \sum_{j\in J} x_{ij} \geq p_i \qquad \forall i\in I \tag{4.26}$$

$$\sum_{i\in I} x_{ij} \leq 1 \qquad \forall j\in J \tag{4.27}$$

$$\sum_{j\in J(t)} x_{ij} \leq a_i \qquad \forall i\in I,\, t\in T \tag{4.28}$$

$$p_i \leq \beta_{\mathbf{x}_i^h}(\mathbf{x}_i) \qquad \forall h\in H \tag{4.29}$$

$$p_i \in [0,n] \qquad \forall i\in I \tag{4.30}$$

$$x_{ij}\in\{0,1\} \qquad \forall i\in I,\, j\in J, \tag{4.31}$$

where $H := \{1, 2, \dots, h' - 1\}$ is the set of previous iterations and $J(t)$ is defined as in Section 3.2.

The objective function (4.25) is the sum of the maximum profit values p_i generated at the facilities $i \in I$. Firstly, these profits are bounded by the number of vehicles assigned to the facility, as expressed by the inequalities (4.26). Secondly, the Benders cuts (4.29) that were derived in the previous iterations of the algorithm (see Section 4.2.3) provide an upper bound.

Every vehicle is allocated to at most one facility, as stated by the set of inequalities (4.27). It is only possible to check the parking spots limit (4.28) in the master problem. All other resource constraints have to be verified in the subproblems.

4.2.2 Subproblems

Having assigned vehicles to facilities in the master problem, the rest of the problem decomposes into a set of independent subproblems, one for each facility. For the given allocation of vehicles we search for charging schedules that maximize the profit at each facility.

We consider the subproblem of facility i in iteration h. Let J_i^h be the set of vehicles assigned to i. Variables x_{jrds} indicate whether vehicle j starts charging at time s with duration d and rate r. The subproblem of facility i is then formulated as follows,

$$\text{maximize} \quad \sum_{j \in J_i^h} \frac{e_j}{e_{\max,j}} \tag{4.32}$$

$$\text{subject to} \sum_{\substack{r \in R,\, d \in D_{jr},\\ s \in S_{jd}}} x_{jrds} = 1 \qquad \forall j \in J_i^h \tag{4.33}$$

$$\sum_{\substack{j \in J_i^h,\, r \in R,\\ d \in D_{jr},\, s \in S_{jd}(t)}} x_{jrds} \leq b_i \qquad \forall t \in T \tag{4.34}$$

$$\sum_{\substack{j \in J_i^h,\, r \in R,\\ d \in D_{jr},\, s \in S_{jd}(t)}} x_{jrds} \cdot r \leq P_i \qquad \forall t \in T \tag{4.35}$$

$$e_{\min,j} \leq \sum_{\substack{r \in R,\, d \in D_{jr},\\ s \in S_{jd}}} x_{jrds} \cdot r \cdot d \qquad \forall j \in J_i^h \tag{4.36}$$

$$\sum_{\substack{r \in R,\, d \in D_{jr},\\ s \in S_{jd}}} x_{jrds} \cdot r \cdot d \leq \sum_{\substack{r \in R,\, d \in D_{jr},\\ s \in S_{jd}}} x_{jrds} \cdot e^*_{jr} \qquad \forall j \in J_i^h \tag{4.37}$$

$$e_j \leq \sum_{\substack{r \in R,\, d \in D_{jr},\\ s \in S_{jd}}} x_{jrds} \cdot r \cdot d \qquad \forall j \in J_i^h \tag{4.38}$$

$$\sum_{\substack{j \in J_i^h,\, d \in D_{jr},\\ s \in S_{jd}}} x_{jrds} \leq \rho(r) \qquad \forall r \in R \tag{4.39}$$

$$e_j \in [0, e_{\max,j}] \qquad \forall j \in J_i^h \tag{4.40}$$

$$x_{jrds} \in \{0,1\} \qquad \forall j \in J_i^h,\, r \in R,\; d \in D_{jr},\, s \in S_{jd}, \tag{4.41}$$

where the domains D_{jr} and S_{jd}, the set $S_{jd}(t)$ and the constants e^*_{jr} are defined as in Section 3.2.

The formulation of the subproblem is quite similar to the model (MIP1) of Section 3.2. The major changes are that the variable index i is fixed (and therefore omitted) and that only the vehicles $j \in J_i^h$ are considered. Moreover, since all assigned vehicles must be scheduled, the sum on the left-hand side of the constraint of (MIP1) corresponding to (4.36) does not need to be included here.

4.2.3 Benders Cuts

After solving the master problem, in iteration h the variables x_{ij} are fixed to values x_{ij}^h. Subsequently, for each facility an optimal schedule for the given allocation of vehicles is computed in the subproblems.

Since the subproblems are independent of one another, we derive a Benders cut that consists of several inequalities, one for each facility,

$$p_i \leq \beta_{\mathbf{x}_i^h}(\mathbf{x}_i) \quad \forall i \in I,$$

where $\mathbf{x}_i$ denotes the vector of decision variables $(x_{ij})_{i \in I}$ and $\mathbf{x}_i^h$ the vector of trial values $(x_{ij}^h)_{i \in I}$ in iteration h.

The bounding function of subproblem i needs to fulfill the following requirements:

(B1) For all feasible values of $\mathbf{x}_i$ the term $\beta_{\mathbf{x}_i^h}(\mathbf{x}_i)$ provides a valid upper bound on the maximum profit p_i of facility i.

(B2) In particular, $\beta_{\mathbf{x}_i^h}(\mathbf{x}_i^h) = p_i^h$, where p_i^h is the optimal value of subproblem i in iteration h.

4.2.3.1 Feasibility Cuts

In the case of an infeasible subproblem i, we derive cuts similar to those in Section 4.1.3. We infer the valid inequality

$$\sum_{j \in J_i^h} (1 - x_{ij}) \geq 1, \tag{4.42}$$

which states that at least one of the currently assigned vehicles must be removed from facility i.

As in (LBD1), this bound can be strengthened by replacing J_i^h with a subset $\tilde{J}_i^h$ of vehicles responsible for the infeasibility of the problem.

4.2.3.2 Optimality Cuts

If subproblem i is feasible with optimal profit value p_i^h, we get the trivial bound $p_i \leq p_i^h$, if $x_i = x_i^h$. We also want to derive a bound for other values of x_i.

The profit p_i is increased by at most 1 if we add a vehicle to facility i. This can be formulated by the following inequality:

$$p_i \leq p_i^h + \sum_{j \in J \setminus J_i^h} x_{ij}.$$

Although this seems reasonable, the cut is not valid in some special cases. It is possible that the profit of a facility increases when we remove a vehicle. This is due to the definition of the objective function and can happen when the vehicle occupies resources that are required by other vehicles to get charged more. Since the vehicles differ in their energy demand and the profit generated by the charging of a vehicle only depends on the ratio of fulfilled demand and maximum demand, the profit generated by a certain amount of energy consumed for recharging need not be the same for different vehicles. Therefore, if we remove a vehicle, others might use the freed resources more efficiently and in total a higher profit is generated. An example is given in Figure 4.1.

To overcome this problem, we resolve subproblem i with the modification that not all of the vehicles must be scheduled. The equality sign in constraint (4.33) is changed to "$\leq$" and the left-hand side of the minimum energy demand constraint (4.36) is adapted in analogy to (3.6) of (MIP1). We denote the objective value of this relaxed subproblem by q_i^h. It holds that $p_i^h \leq q_i^h$, in most cases they are equal. We use q_i^h instead of p_i^h to

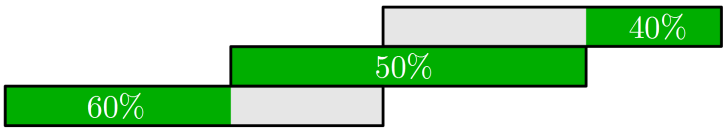

FIGURE 4.1: Example of recharging schedules of three vehicles assigned to the same facility, where the removal of the second vehicle would result in an increase in profit. When all three vehicles must be charged, the total profit is 1.5. Removing the second vehicle (the recharging of which could only be half completed during its parking time), the other vehicles can be fully recharged, resulting in a profit of 2.

get a valid bound:

$$p_i \leq q_i^h + \sum_{j \in J \setminus J_i^h} x_{ij}. \tag{4.43}$$

If $q_i^h \neq p_i^h$ the bounding function does not fulfill property (B2). Therefore, in this case we must add an additional inequality of the form

$$p_i \leq \tilde{\beta}_{\mathbf{x}_i^h}(\mathbf{x}_i)$$

with $\tilde{\beta}_{\mathbf{x}_i^h}(\mathbf{x}_i^h) = p_i^h$. Otherwise, the algorithm might not terminate. We use the following inequality:

$$p_i \leq p_i^h + \sum_{j \in J \setminus J_i^h} x_{ij} \cdot n + \sum_{j \in J_i^h} (1 - x_{ij}) \cdot n. \tag{4.44}$$

It becomes $p_i \leq p_i^h$ if exactly the same set of vehicles is allocated to facility i again and is trivially fulfilled for all other assignments.

To strengthen inequality (4.43), we consider how the profit changes when a vehicle $j \in J_i^h$ is unallocated. It is reduced by at most $\min(d_j^h \cdot r_j^h, e_{\max,j})/e_{\max,j}$, which is the vehicle's contribution to the profit function. As explained above, this value is not subtracted in general, as the removal of a vehicle can even lead to an increase in profit. Only if the vehicle does not occupy any resources that can be used by other vehicles, its removal leads to a decrease of the profit by $\min(d_j^h \cdot r_j^h, e_{\max,j})/e_{\max,j}$. Two sufficient conditions for this are:

1. All other vehicles are fully recharged in the current solution of the subproblem.
2. All resources used by vehicle j (charging rate, charging spot and power) are not consumed up to the limit, i.e., neither of the concerned resource constraints is

tight. This can be expressed as follows:

$$r_j^h = r_{\min} \quad \text{or} \quad \rho(r_j^h) \; - \sum_{\substack{k \in J_i^h,\, d \in D_{jr},\\ s \in S_{jd}}} x_{k r_j^h d s} \geq 1$$

$$P_i - \sum_{\substack{k \in J_i^h,\, r \in R,\\ d \in D_{jr},\, s \in S_{jd}(t)}} x_{ksdr} \cdot r \geq r_{\max} \qquad \forall t \in T_j^h$$

$$b_i - \sum_{\substack{k \in J_i^h,\, r \in R,\\ d \in D_{jr},\, s \in S_{jd}(t)}} x_{ksdr} \geq 1 \qquad \forall t \in T_j^h,$$

where r_j^h is the charging rate that was assigned to vehicle j in iteration h and $T_j^h := \{t \in T \,|\, s_j^h \leq t \leq s_j^h + d_j^h\}$ the set of time intervals during which vehicle j is scheduled to be charged.

If either of the two conditions is fulfilled we add

$$-(1 - x_{ij}) \cdot \min(d_j^h \cdot r_j^h, e_{\max,j})/e_{\max,j}$$

to the right-hand side of inequality (4.43).

4.2.4 Strengthening the Master Problem

As in Section 4.1.4, we formulate relaxed versions of the subproblem constraints which can be included within the master problem to strengthen it.

Similar to (4.21) of (LBD1), an aggregated power limit constraint for time window $[t_1, t_2]$ and facility i can be written as follows,

$$\sum_{j \in J(t_1,t_2)} x_{ij} \cdot e_{\min,j} \leq P_i \cdot (t_2 - t_1), \tag{4.45}$$

where $J(t_1, t_2) := \{j \in J \,|\, [t_{\text{dep},j}, t_{\text{dep},j}] \subseteq [t_1, t_2]\}$ is the set of vehicles to be scheduled within time window $[t_1, t_2]$. In contrast to (LBD1), the master problem does not determine how much power will be consumed by charging vehicle j, only the vehicle's minimum energy demand $e_{\min,j}$ is known.

We define the tightness of (4.45) as

$$T^i_{\text{power}}(t_1, t_2) := \frac{1}{P_i} \sum_{j \in J(t_1,t_2)} e_{\min,j} - t_2 + t_1.$$

Again, only those inequalities (4.45) with $t_1 \in \{t_{\text{arr},j} \mid j \in J\}$ and $t_2 \in \{t_{\text{dep},j} \mid j \in J\}$ need to be added to the master problem, which are not dominated by others, i.e., there is no such constraint for a time window $[u_1, u_2]$ with a higher tightness and $[u_1, u_2] \subseteq [t_1, t_2]$.

Furthermore, we formulate a relaxation of the charging spots limit. If the minimum charging duration of a vehicle is longer than half of its parking duration, then there is a non-empty time interval, in which it must be charged when allocated to some facility. Let $d_{\min,j} := \lceil e_{\min,j}/r_{\max} \rceil$ be the minimum charging duration of vehicle j and $C(t) := \{j \in J \mid t_{\text{dep},j} - d_{\min,j} \leq t \leq t_{\text{arr},j} + d_{\min,j}\}$ be the set of all vehicles that must be charged at time t. We can then define a relaxed charging spots limit,

$$\sum_{j \in C(t)} x_{ij} \leq b_i \quad \forall i \in I,\, t \in T, \tag{4.46}$$

that can be added to the master problem.

4.3 Hierarchy of Facilities

In Sections 4.1.3 and 4.2.3 individual bounds were derived for every facility as Benders cuts, since the subproblems are independent. They are, however, not unrelated. Having computed the optimal profit for a given assignment at one facility, we might infer a bound on the maximal profit for the same assignment at other facilities.

We call a facility i *dominated* by another facility $\bar{i}$, if all resources are at least as limited, i.e., it has less or the same number of parking and charging spots, and less or the same amount of power available. This defines a partial order on the set of facilities. Every feasible charging schedule for a given assignment at facility i would also be feasible at facility $\bar{i}$. Hence, the optimal profit for a given assignment of vehicles to facility $\bar{i}$ is an upper bound on the optimal profit for the same assignment at facility i.

In a preprocessing step, we determine for all facilities the set of dominated ones. In the case of homogeneous facilities, these sets simply contain all other facilities. Whenever we add a Benders cut with respect to facility i to the master problem, we also add the same inequality for all facilities dominated by i.

Chapter 5

Heuristic Boosting of Logic-Based Benders Decomposition

There are different strategies for trying to improve the performance of decomposition algorithms like those described in the last chapter. One approach is to solve either the master or subproblems heuristically, getting a possibly suboptimal solution for them. By adapting the algorithms in an appropriate way, we ensure that they still return an optimal solution eventually.

When solving a MIP model with a branch and bound algorithm, it can happen that an optimal integer solution is already found after a while, but the procedure takes quite a long time to prove its optimality. For a maximization problem, branches are explored and pruned until the global lower bound, which is the value of the best integer solution found so far, and the local upper bound coincide at the optimal value. In practice, due to numerical reasons, a solution is accepted when the difference between upper and lower bounds falls below a previously specified threshold, called the *optimality gap*. One can try to set this gap to a higher value to obtain a solution more quickly, which is, however, not guaranteed to be optimal. Using appropriate extensions that reconsider heuristic results in a later phase, we can use this approach for the master or subproblems but still get an optimal solution in the end.

This is just one possibility for solving the master or subproblems heuristically. For the same purpose, other heuristics or advanced metaheuristics could be used as well. We confine ourselves, however, to the method described above. Furthermore, we want to emphasize that our focus lies on finding provably optimal solutions to the original problem instead of just solving it heuristically.

5.1 Increasing the Master Problem Optimality Gap of (LBD1)

In the first decomposition we only consider the master problems since the subproblems are mere feasibility problems that are solved rather quickly. At the beginning we set the optimality gap for solving the master problems to a higher value. Consequently, they are solved faster but the solutions are not guaranteed to be optimal. Instead of terminating the algorithm when all subproblems turn out to be feasible, we reset the optimality gap to zero (or in practice a very small constant) and restart the iterations. Only when also this final exact phase terminates we can be sure to have obtained an optimal solution. We will refer to this variant as (LBD1BoostM).

5.2 Increasing the Master or Subproblem Optimality Gap of (LBD2)

The same approach that was used for boosting (LBD1) can be applied to the second decomposition as well. We increase the initial optimality gap for solving the master problems. When the termination criterion of the logic-based Benders algorithm is met, iterations are restarted with the gap reset to zero. This will be referred to as (LBD2BoostM).

Furthermore, in the second decomposition the subproblems are rather large. We can also try to set the optimality gap for solving them to a higher value instead. This might lead to a substantial speedup, but we only get lower bounds on the optimal profit of a subproblem. The Benders cuts that are derived might be invalid, possibly cutting away an optimal solution.

To counter this problem, we have to adapt the Benders cuts derived from the suboptimally solved subproblems. This only concerns the optimality cuts since feasibility is not affected by a change of the optimality gap. The MIP solver not only returns the best integer solution found so far, which is a lower bound on the optimal solution value, but also the maximum objective value of all remaining unexplored nodes, which is an upper bound. In (4.44) we replace the solution value p_i^h of subproblem i of iteration h and in (4.43) the optimal value q_i^h of the modified subproblem, in which not all vehicles have to be scheduled, with the corresponding upper bounds. In this way, the cuts remain valid but might be weaker than those inferred from subproblems that were solved to optimality. This possibly leads to more iterations of the algorithm and it might terminate even when no optimal solution was found since only upper bounds are used for the profit values of the subproblems. We can, however, easily handle this problem by resetting

the optimality gap to zero and restarting the iterations. This variant will be referred to as (LBD2BoostSU).

Alternatively, we could just use an increased optimality gap for solving the subproblems without further adapting the algorithm. Eventually, it will terminate with a solution, which is not guaranteed to be optimal anymore. The hope is, however, to find a reasonably good heuristic solution in shorter time, since the Benders cuts are stronger than those of the adaptation described above. We will refer to this approach as (LBD2BoostSL).

In order to obtain provably optimal solutions with this method as well, one could proceed by exactly verifying and possibly correcting all heuristically derived Benders cuts, as it was done in [17]. Since this approach did, however, not produce good results for the given problem, it will not be considered further in this thesis.

Chapter 6

Computational Aspects

This chapter provides details about the implementation of the models that were defined above.

First, we explain how to avoid adding trivially valid inequalities to the model. Then, an algorithm is described which removes redundant inequalities from the relaxed subproblems. Afterwards, details are given on how to compute a subset of vehicles responsible for infeasibility of a subproblem, which is needed to strengthen the feasibility cuts. Finally, a method to avoid resolving subproblems that were already considered before is presented.

6.1 Removal of Trivial Inequalities

Some of the inequalities specified in the definitions of the models might be trivially true for specific instances. For example, consider the parking spots limit (3.3) of (MIP1) for a given facility $i \in I$ and a given time $t \in T$:

$$\sum_{\substack{j \in J(t), r \in R, \\ d \in D_{jr}, s \in S_{jd}}} x_{ijrds} \leq a_i.$$

If the number of x-variables is smaller than or equal to a_i, then the constraint is fulfilled for any assignment and need not be added to the model. Implementing (MIP1) as well as the master and subproblems of (LBD1) and (LBD2), this is checked before adding any constraint of this form.

The same approach works for inequalities, where the sum not only contains boolean variables but also constant coefficients. An example is the power limit constraint (3.5)

for a given facility $i \in I$ and a given time $t \in T$:

$$\sum_{\substack{j \in J,\, r \in R,\\ d \in D_{jr},\, s \in S_{jd}(t)}} x_{ijrds} \cdot r \leq P_i.$$

Here, instead of counting the variables x_{ijrds}, we add up all the coefficients r and check if the sum is greater than P_i. If not, once again the constraint is trivially fulfilled and need not be added to the model.

6.2 Removal of Dominated Inequalities from the Relaxed Subproblems

It was already mentioned in Sections 4.1.4 and 4.2.4 that, when including relaxed versions of the subproblem constraints within the master problem, we should avoid adding redundant inequalities. This applies to the aggregated charging spots (4.20) and power limit constraints (4.21) of (LBD1), as well as to the aggregated power limit constraints (4.45) of (LBD2).

An aggregated constraint for facility i and time window $[t_1, t_2]$ is dominated by the same constraint for time window $[u_1, u_2]$, if $[u_1, u_2] \subseteq [t_1, t_2]$ and if the constraint for $[t_1, t_2]$ has a lower tightness. The inequality is then redundant and can be removed. Let $T^i(t_1, t_2)$ be the tightness of an aggregated constraint for facility i and time window $[t_1, t_2]$, as defined in Sections 4.1.4 and 4.2.4.

A procedure [12] for the removal of dominated inequalities is given as Algorithm 6.1. For each facility $i \in I$, it returns a set R^i of time windows corresponding to undominated inequalities. Let $(\text{arr}_1, \ldots, \text{arr}_{n_a})$ and $(\text{dep}_1, \ldots, \text{dep}_{n_b})$ be the sorted lists of n_a distinct arrival and n_b distinct departure times of the vehicles $j \in J$. For each arrival time arr_a, the algorithm iterates over the list of departure times dep_b. Only if $[\text{arr}_a, \text{dep}_b]$ is a proper time window and there was no tighter constraint found with an earlier departure time, the time interval is added to the set R^i. Besides, all time windows corresponding to constraints dominated by the one that is currently considered are removed from R^i.

6.3 Subsets of Vehicles Responsible for Infeasibility

In Section 4.1.3 it was explained how to strengthen the feasibility cut derived from subproblem i in iteration h by computing a subset $\tilde{J}_i^h \subset J_i^h$ of vehicles assigned to facility i that is responsible for infeasibility. The subset computed by Algorithm 4.1 depends on

Algorithm: Time Windows for Relaxed Inequalities

```
1  create sorted lists (arr_1, ..., arr_{n_a}) and (dep_1, ..., dep_{n_b}) of unique
   arrival and departure times;
2  for i ∈ I do
3      R^i := ∅;
4      for a := 1, ..., n_a do
5          b' := 0;
6          for b := 1, ..., n_b do
7              if dep_b ≥ arr_a and (b' = 0 or T^i(arr_a, dep_b) > T^i(arr_a, dep_{b'})) then
8                  for [arr_ā, dep_b̄] ∈ R^i do
9                      if b̄ = b and ā < a and T^i(arr_a, dep_b) ≥ T^i(arr_ā, dep_b̄)
                       then
10                         R^i := R^i \ {[arr_ā, dep_b̄]};
11                     end
12                 end
13                 R^i := R^i ∪ {[arr_a, dep_b]};
14                 b' := b;
15             end
16         end
17     end
18 end
```

Algorithm 6.1: Generating a set of time windows corresponding to undominated inequalities of the subproblem relaxation.

the order in which the vehicles in J_i^h are considered. This can be done randomly or in a predefined way depending on the vehicles' attributes. Moreover, we can try to run the algorithm several times for different orders to obtain multiple subsets $\tilde{J}_i^h \subset J_i^h$ of vehicles responsible for infeasibility, since they are not unique. The computed sets are minimal elements of the partial order defined by the subset relation, but need not be of minimum cardinality. It may, however, also be the case that J_i^h cannot be reduced at all.

In our implementation of (LBD1), the set J_i^h is sorted by the vehicles' *difficulty* to be scheduled, which we define as the ratio of parking duration and minimum energy demand, i.e., $(t_{\mathrm{dep},j} - t_{\mathrm{arr},j})/e_{\min,j}$ for vehicle j (cf. Section 3.3). We run Algorithm 4.1 two times. First, the set J_i^h is sorted by ascending difficulty, starting with the vehicle that is the easiest to schedule. Then, it is sorted in the reverse order, starting with the vehicle which is most difficult to schedule. Therefore, for facility i two inequalities of the form (4.19), where the set J_i^h is replaced by the corresponding subset, are added to the master problem. It has been found empirically that adding a second inequality resulting from another subset is by far better than defining just one. On the other hand,

there is no benefit to be gained from calculating more than two subsets. Ordering J_i^h by difficulty instead of using a random order achieved slightly better results.

For (LBD2) only one subset $\tilde{J}_i^h \subseteq J_i^h$ is computed to define a feasibility cut (4.42) for subproblem i in iteration h. As the subproblems are rather large, resolving them may take quite a long time. Therefore, it is better to run Algorithm 4.1 just once. Instead of sorting J_i^h in a special way, the order of vehicles given by the problem definition was used.

6.4 Already Solved Subproblems

In the proof of Theorem 2.2 in Section 2.2.4 it was stated that for a logic-based Benders algorithm no subproblem gets examined twice, except maybe in the last iteration. This only holds, however, for the subproblem as a whole and not for its individual parts when it decomposes into multiple problems. Thus, it can happen that the current subproblem only changes for some of the facilities and for the remaining ones stays the same as in a previous iteration. Moreover, during the generation of a feasibility cut when the subproblem of a facility gets resolved after removing one or more vehicles, a subproblem that was already solved before might be solved another time.

This does not matter so much for (LBD1), since the subproblems of the first decomposition are solved rather quickly. In (LBD2), however, resolving a subproblem may take quite a long time and should, therefore, be avoided. For this purpose, the subproblem results are stored together with the problem specification (the set of vehicles that were assigned to the facility) in a hash table. There is an individual table for every facility. As hash value we use the sum of indices of the assigned vehicles, i.e., $\sum_{j \in V_i} j$. Subproblem results with the same hash value are stored in a list.

Whenever a new subproblem is defined, we first check whether the same problem was solved before. If this is the case, there is no need to resolve it. We simply look up the result stored in the hash table. Furthermore, no new Benders cut needs to be generated as it was already added before.

Chapter 7

Computational Experiments

In order to evaluate the algorithms formulated in this thesis, multiple experiments are performed. The first section describes how instances are generated. Subsequently, results are given in the form of several tables, for different sets of instances as well as algorithm variants, and analyzed in detail.

All algorithms are implemented in C++ and compiled with GCC 4.8. For solving the MIP models, CPLEX version 12.6.1 is used. Test runs are performed on a single core of an Intel Xeon E5540 with 2.53 GHz and 24 GB RAM. The time limit is set to 2 hours.

7.1 Instance Generation

To create problem instances for testing the algorithms, a random generator is used. It receives as input the desired number of facilities m, number of vehicles n, number of charging rates n_R and considered time horizon $t_{\max}$ and returns a problem instance as defined in Section 3.1.

Regarding the choice of rates offered by the charging machines, it would not make sense to generate them randomly. Instead, a basic charging rate $r_{\min}$ is chosen and all other rates are defined to be $2 \cdot r_{\min}$, $4 \cdot r_{\min}$, ..., $2^{n_R-1} \cdot r_{\min}$. We set the minimum charging rate to 6 kW, which (or in practice rather 6.6 kW) seems to be quite typical for charging EVs [6]. For a cardinality of $n_R := 3$ we get the following set of rates: $R = \{6\,\text{kW}, 12\,\text{kW}, 24\,\text{kW}\}$.

We choose $\alpha := 0.2$ for the parameter needed to define the function $\rho(r)$ which limits the number of vehicles charged with rate r at each facility. Considering for example an instance with 3 facilities, 10 vehicles and 3 rates, the lowest charging rate can be used freely, the medium rate at most twice and the highest rate at most once per facility.

The facilities' resource limits are generated randomly, based on a discrete uniform distribution within reasonable limits depending on the number of facilities and vehicles and the available charging rates. To get the number of parking spots a_i and charging spots b_i of facility i, two random integers are selected from the set $\{1, 2, \dots, 2 \cdot \lceil n/m \rceil\}$, on condition that $a_i \geq b_i$. In this way, the expected number of parking spots of a facility is about n/m, providing enough space for all vehicles to be recharged. The power limit P_i is drawn randomly from the set $\{r_{\min}, r_{\min} + 1, \dots, \lceil r_{\max} \cdot b_i \rceil\}$ with the extreme cases being that only one vehicle can be charged with minimum rate at any time or that all available machines are able to charge simultaneously at maximum rate.
Also, instances with tighter resource limits at each facility are generated. For this purpose, the numbers of spots is chosen randomly from the set $\{1, 2, \dots, \lceil n/m \rceil\}$ and the power limits from $\{r_{\min}, r_{\min} + 1, \dots, \lceil r_{\max} \cdot b_i/2 \rceil\}$.

Similarly, the specifications of the vehicles to be recharged are generated randomly based on a discrete uniform distribution. The discrete time intervals $T = \{0, \dots, t_{\max}\}$ are viewed to have a duration of 10 minutes each. Thus, if we set $t_{\max} := 60$, the considered time horizon is about 10 hours. The parking duration dur_j of a vehicle j is selected randomly from the set $\{3, 4, \dots, t_{\max}\}$, which means that every vehicle is parked for at least 30 minutes and the duration must not exceed the considered time horizon. The arrival time $t_{\text{arr},j}$ is then chosen to be a random integer between 0 and $(t_{\max} - \text{dur}_j)$. As a result, the departure time $t_{\text{dep},j}$ is set to $(t_{\text{arr},j} + \text{dur}_j)$.

Regarding the energy demand of a vehicle j, values for $e_{\min,j}$ and $e_{\max,j}$ are drawn randomly from the set $\{5\,\text{kWh}, 6\,\text{kWh}, \dots, 30\,\text{kWh}\}$, which seems to be a typical range when considering the battery size of currently available EV models [6]. The values for minimum and maximum energy demand of a vehicle are chosen on condition that $e_{\min,j} \leq e_{\max,j}$ and $r_{\max} \cdot (t_{\text{dep},j} - t_{\text{arr},j}) \geq e_{\min,j}$.

In the following experiments, the time horizon $t_{\max}$ is set to 60 and the number of charging rates n_R to 3. The numbers of facilities and vehicles vary from instance to instance and are indicated by its name. For example, instance "3-10-a" specifies 3 facilities and 10 vehicles to be scheduled. The letter at the end makes it possible to distinguish instances with the same number of facilities and vehicles.

Experiments are performed on four different sets of instances. The set EVRSPbasic constains instances with homogeneous facilities and normal resource limits. The number of facilities m ranges from 3 to 20 and the number of vehicles n from 10 to 200. The set EVRSPhet has the same specifications except that the instances have heterogeneous facilities. EVRSPres considers homogeneous facilities, but the resource limits are tighter. Finally, EVRSPlarge contains larger instances than the other sets, with the number of

facilities m ranging from 20 to 50 and the number of vehicles n from 300 to 1000. The specifications of the sets of instances are summarized in Table 7.1.

Set of Instances	m	n	n_R	$t_{\max}$	hom. facilities	resource limits
EVRSPbasic	$[3, 20]$	$[10, 200]$	3	60	yes	normal
EVRSPhet	$[3, 20]$	$[10, 200]$	3	60	no	normal
EVRSPres	$[3, 20]$	$[10, 200]$	3	60	yes	tighter
EVRSPlarge	$[20, 50]$	$[300, 1000]$	3	60	yes	normal

TABLE 7.1: Sets of instances.

7.2 Results

Test runs are performed on the sets of instances described above. First, the results achieved on EVRSPbasic are analyzed for all algorithms and their variants. Then, we evaluate their performance on the other sets as well.

If not stated otherwise, the relaxed subproblem inequalities derived in Section 4.1.4 are not included within the master problems of (LBD1), whereas the subproblem relaxation described in Section 4.2.4 is used for strengthening the master problems of (LBD2), since these variants show the best performance.

7.2.1 EVRSPbasic

Table 7.2 compares the results achieved by (MIP1), (LBD1) and (LBD2) on the set EVRSPbasic. The second column displays the optimal objective value for every instance. If no method is able to find an optimal solution within the time limit, the best known upper bound ("UB") on the objective value is listed instead. The superscript "^{M}" marks the bounds computed by (MIP1). For every algorithm, the optimality gap (abbreviated "g"), i.e., the relative difference between the best integer solution found by the algorithm and the optimal value of the instance (or the best known upper bound), and the total runtime are given. If termination is caused by the time limit, "TL" is written instead. Additionally, the number of iterations (abbreviated "it") is displayed for the logic-based Benders algorithms. The best runtime is printed bold for every instance that is solved to optimality.

We observe that (MIP1) and (LBD1) show a good performance on most instances, while (LBD2) often takes a very long time to find an optimal solution.

- (MIP1) generally terminates after a reasonable amount of time, except for some instances where it seems to be much harder to find an optimal charging schedule.

Instance	UB	(MIP1)		(LBD1)			(LBD2)		
		g[%]	time[s]	g[%]	time[s]	#it	g[%]	time[s]	#it
3-10-a	7.72	0.00	**0.94**	0.00	2.53	50	0.00	5.28	17
3-10-b	7.00	0.00	**0.63**	0.00	0.65	23	0.00	1.84	4
3-10-c	10.00	0.00	1.41	0.00	**0.07**	2	0.00	0.44	1
3-30-a	28.72^M	0.01	TL	-	TL	546	7.05	TL	330
3-30-b	30.00	0.00	3.24	0.00	**0.58**	4	0.00	0.76	1
3-30-c	29.92	0.00	6.41	0.00	**2.74**	10	0.00	1762.28	119
5-30-a	26.12	0.00	**638.99**	-	TL	149	28.07	TL	281
5-30-b	30.00	0.00	**7.85**	0.00	16.97	72	0.00	1236.55	270
5-30-c	29.00	0.00	3.60	0.00	**0.50**	4	0.00	0.51	1
5-50-a	47.25	0.00	**18.13**	-	TL	2631	6.23	TL	187
5-50-b	46.00	0.00	**5.83**	0.00	15.69	53	0.00	TL	311
5-50-c	48.43	0.00	7.55	0.00	**4.26**	15	0.00	595.43	225
10-50-a	35.70	0.00	27.13	0.00	**5.45**	15	0.00	TL	631
10-50-b	46.51	0.00	15.88	0.00	**10.20**	30	0.00	TL	280
10-50-c	48.21	0.00	43.43	0.00	**38.14**	70	2.27	TL	175
10-100-a	99.80	0.00	39.88	0.00	**13.01**	16	0.03	TL	359
10-100-b	99.90	0.00	42.24	0.00	**8.72**	10	0.11	TL	230
10-100-c	99.67	0.00	52.25	0.00	**9.10**	9	0.13	TL	233
10-200-a	199.78	0.00	102.65	0.00	**29.57**	14	0.42	TL	100
10-200-b	199.51	0.00	109.88	0.00	**67.83**	20	0.24	TL	135
10-200-c	198.82	0.00	92.69	0.00	**34.87**	14	0.54	TL	105
20-100-a	95.71	0.00	**72.93**	0.00	104.36	65	30.32	TL	18
20-100-b	94.95	0.00	**498.58**	0.00	641.82	250	38.55	TL	23
20-100-c	91.68	0.00	76.89	0.00	**72.73**	53	1.71	TL	36
20-200-a	198.49	0.00	222.54	0.00	**58.92**	15	1.71	TL	42
20-200-b	199.80	0.00	**3890.90**	-	TL	627	38.95	TL	20
20-200-c	70.27	0.00	**823.15**	-	TL	144	-	TL	1

TABLE 7.2: Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPbasic.

Apparently the difficulty of a problem not only depends on its size, but also on the instance properties.

- (LBD1) outperforms the compact MIP model on the majority of instances, but for some it takes much more time to finish. Considering the respective objective values, we observe that (LBD1) performs especially well on instances where all or nearly all vehicles can be fully recharged. It then only takes a few iterations to terminate. For instances where several vehicles cannot be fully recharged, the algorithm generally needs much more iterations. A disadvantage of this decomposition is that it only returns a feasible solution when it finds an optimal one. It does not produce any feasible intermediate solutions.

- As mentioned before, (LBD2) does not perform well in general. Only when it already finishes after one iteration, the algorithm is competitive with the others.

Unlike (LBD1), it usually returns a feasible solution when interrupted early. In some cases it does not terminate in time, although it has found a solution with optimal value, because the algorithm cannot yet exclude that there may exist a feasible solution with a better value.

Table 7.3 shows the results of the greedy algorithm described in Section 3.3 on the set EVRSPbasic. Its runtime is often less than a hundredth of a second, but it does not find an optimal solution for any instance. In many cases the value of the heuristically found solution is quite far from the optimal one. This shows that the EVRSP is indeed a challenging problem that cannot be solved satisfactorily with such a simple greedy approach.

Instance	UB	**(GREEDY)**	
		g[%]	time[s]
3-10-a	7.72	15.47	< 0.01
3-10-b	7.00	0.55	< 0.01
3-10-c	10.00	5.02	< 0.01
3-30-a	28.72	37.27	< 0.01
3-30-b	30.00	7.84	< 0.01
3-30-c	29.92	14.05	0.01
5-30-a	26.12	33.13	0.01
5-30-b	30.00	24.92	0.01
5-30-c	29.00	6.81	< 0.01
5-50-a	47.25	25.56	0.01
5-50-b	46.00	24.01	< 0.01
5-50-c	48.43	10.79	< 0.01
10-50-a	35.70	12.97	< 0.01
10-50-b	46.51	14.42	< 0.01
10-50-c	48.21	18.43	0.01
10-100-a	99.80	15.63	0.01
10-100-b	99.90	18.83	0.01
10-100-c	99.67	20.81	0.01
10-200-a	199.78	17.67	< 0.01
10-200-b	199.51	25.71	0.05
10-200-c	198.82	18.39	< 0.01
20-100-a	95.71	10.84	< 0.01
20-100-b	94.95	20.71	0.08
20-100-c	91.68	8.90	< 0.01
20-200-a	198.49	16.71	< 0.01
20-200-b	199.80	28.07	0.15
20-200-c	70.27	28.97	0.34

Table 7.3: Results of (GREEDY) on the set EVRSPbasic.

7.2.1.1 Details for (LBD1) and (LBD2)

After the comparison of the algorithms, detailed results are now given for the logic-based Benders algorithms.

Table 7.4 displays additional information about the test runs with (LBD1). It separately lists the total time "t_{all}" spent on either the master or subproblems, respectively, as well as the average solving time "t_{avg}". The number of solved master problems corresponds to the number of iterations. The total number of subproblems, which contains those arising from the solution of a master problem as well as those defined during the generation of a feasibility cut, is provided additionally. Finally, the number of feasibility cuts that are added to the master problem is given.

Instance	**(LBD1)**			Master Problems		Subproblems			Cuts
	g[%]	time[s]	#it	t_{all}[s]	t_{avg}[s]	#	t_{all}[s]	t_{avg}[s]	#
3-10-a	0.00	2.53	50	1.43	0.03	796	0.87	< 0.01	98
3-10-b	0.00	0.65	23	0.22	0.01	365	0.32	< 0.01	42
3-10-c	0.00	**0.07**	2	0.03	0.01	16	0.01	< 0.01	2
3-30-a	-	TL	546	7060.80	12.93	32785	131.19	< 0.01	1530
3-30-b	0.00	**0.58**	4	0.12	0.03	174	0.35	< 0.01	8
3-30-c	0.00	**2.74**	10	0.72	0.07	432	1.82	< 0.01	18
5-30-a	-	TL	149	7181.15	48.20	8846	15.97	< 0.01	651
5-30-b	0.00	16.97	72	7.27	0.10	2690	8.54	< 0.01	153
5-30-c	0.00	**0.50**	4	0.21	0.05	118	0.16	< 0.01	8
5-50-a	-	TL	2631	6191.84	2.35	190858	935.85	< 0.01	7097
5-50-b	0.00	15.69	53	5.98	0.11	3061	8.42	< 0.01	98
5-50-c	0.00	**4.26**	15	1.69	0.11	877	2.10	< 0.01	33
10-50-a	0.00	**5.45**	15	4.32	0.29	430	0.35	< 0.01	34
10-50-b	0.00	**10.20**	30	7.38	0.25	1168	1.57	< 0.01	53
10-50-c	0.00	**38.14**	70	25.22	0.36	4554	9.96	< 0.01	319
10-100-a	0.00	**13.01**	16	8.51	0.53	1194	2.82	< 0.01	48
10-100-b	0.00	**8.72**	10	5.11	0.51	1070	2.30	< 0.01	44
10-100-c	0.00	**9.10**	9	5.66	0.63	1204	2.06	< 0.01	53
10-200-a	0.00	**29.57**	14	17.38	1.24	3454	8.87	< 0.01	81
10-200-b	0.00	**67.83**	20	27.96	1.40	4898	34.85	0.01	95
10-200-c	0.00	**34.87**	14	17.42	1.24	3736	14.03	< 0.01	84
20-100-a	0.00	104.36	65	90.71	1.40	3944	5.09	< 0.01	189
20-100-b	0.00	641.82	250	554.17	2.22	22628	57.01	< 0.01	1330
20-100-c	0.00	**72.73**	53	61.13	1.15	3680	4.25	< 0.01	196
20-200-a	0.00	**58.92**	15	48.77	3.25	2536	3.79	< 0.01	108
20-200-b	-	TL	627	6316.07	10.07	135680	714.80	0.01	5222
20-200-c	-	TL	144	7064.05	49.06	49934	94.57	< 0.01	2855

TABLE 7.4: Detailed results of (LBD1) on the set EVRSPbasic.

For those instances which are solved to optimality, the algorithm spends a similar amount of time on the master and subproblems. Although it takes substantially less time to solve a single subproblem, this is offset by their large number. When the algorithm is unable to find an optimal solution within the time limit, most time is spent on solving the master problems. One can see that not only the relatively high number of iterations is responsible for the long runtime of (LBD1) in these cases, but also the increased amount of time needed to solve a single master problem.

Table 7.5 shows the same information about the test runs with (LBD2), except that for the Benders cuts we distinguish between feasibility ("feas") and optimality cuts ("opt").

Instance	**(LBD2)**			Master Problems		Subproblems			# Cuts	
	g[%]	time[s]	#it	t_{all}[s]	t_{avg}[s]	#	t_{all}[s]	t_{avg}[s]	feas	opt
3-10-a	0.00	5.28	17	0.20	0.01	92	4.46	0.05	3	39
3-10-b	0.00	1.84	4	0.01	< 0.01	32	1.67	0.05	4	3
3-10-c	0.00	0.44	1	< 0.01	< 0.01	3	0.40	0.13	0	0
3-30-a	7.05	TL	330	216.83	0.66	6348	6860.44	1.08	328	633
3-30-b	0.00	0.76	1	0.01	0.01	3	0.67	0.22	0	0
3-30-c	0.00	1762.28	119	8.42	0.07	603	1736.80	2.88	0	301
5-30-a	28.07	TL	281	5827.44	20.74	7698	1304.12	0.17	645	678
5-30-b	0.00	1236.55	270	100.31	0.37	4439	1064.49	0.24	213	912
5-30-c	0.00	0.51	1	0.01	0.01	5	0.42	0.08	0	0
5-50-a	6.23	TL	187	732.73	3.92	6483	6323.09	0.98	257	561
5-50-b	0.00	TL	311	6724.34	21.62	4420	385.65	0.09	73	1295
5-50-c	0.00	595.43	225	303.33	1.35	1611	247.84	0.15	2	777
10-50-a	0.00	TL	631	6617.91	10.49	11480	480.02	0.04	48	5567
10-50-b	0.00	TL	280	6840.82	24.43	4852	297.77	0.06	14	2330
10-50-c	2.27	TL	175	6233.76	35.62	4028	921.18	0.23	122	1457
10-100-a	0.03	TL	359	5784.97	16.11	6911	1243.01	0.18	10	3312
10-100-b	0.11	TL	230	5586.66	24.29	6751	1470.37	0.22	102	2058
10-100-c	0.13	TL	233	6024.87	25.86	5593	1053.94	0.19	58	2179
10-200-a	0.42	TL	100	6127.40	61.27	3768	960.19	0.25	44	861
10-200-b	0.24	TL	135	3855.39	28.56	9797	3104.81	0.32	125	1052
10-200-c	0.54	TL	105	5928.39	56.46	3791	1112.55	0.29	32	815
20-100-a	30.32	TL	18	7124.75	395.82	1222	68.07	0.06	68	220
20-100-b	38.55	TL	23	6947.70	302.07	1859	232.11	0.12	101	258
20-100-c	1.71	TL	36	7047.11	195.75	2007	135.32	0.07	82	428
20-200-a	1.71	TL	42	6732.02	160.29	2764	410.35	0.15	57	592
20-200-b	38.95	TL	20	4172.66	208.63	2745	2965.05	1.08	88	199
20-200-c	-	TL	1	TL	TL	0	-	-	0	0

TABLE 7.5: Detailed results of (LBD2) on the set EVRSPbasic.

Considering the smaller instances that are solved to optimality, we observe that the algorithm spends more time on the solution of the subproblems. In addition to the fact that there is a higher number of subproblems, they are also more difficult to solve than the master problems. However, for larger instances where the algorithm cannot finish in time, the master problems are responsible for the long runtime. For the last instance,

not even a single master problem can be solved within the time limit. In general, the algorithm generates far more optimality cuts than feasibility cuts, which indicates that it often finds a feasible solution after a few iterations but takes longer to compute an optimal one or prove optimality.

Finally, Table 7.6 compares the results achieved by (LBD1) when strengthening the master problems as described in Section 4.1.4 with those of the basic algorithm as seen before.

Instance	g[%]	**(LBD1) w. Strengthened Master**				**(LBD1)**			
		time[s]	#it	Master		time[s]	#it	Master	
				$t_{all}[s]$	$t_{avg}[s]$			$t_{all}[s]$	$t_{avg}[s]$
3-10-a	0.00	**1.80**	17	1.35	0.08	2.53	50	1.43	0.03
3-10-b	0.00	**0.64**	23	0.21	0.01	0.65	23	0.22	0.01
3-10-c	0.00	0.16	2	0.04	0.02	**0.07**	2	0.03	0.01
3-30-a	-	TL	534	7062.24	13.23	TL	546	7060.80	12.93
3-30-b	0.00	**0.55**	4	0.12	0.03	0.58	4	0.12	0.03
3-30-c	0.00	**2.74**	10	0.72	0.07	**2.74**	10	0.72	0.07
5-30-a	-	TL	151	7182.27	47.56	TL	149	7181.15	48.20
5-30-b	0.00	36.55	59	26.56	0.45	**16.97**	72	7.27	0.10
5-30-c	0.00	1.04	4	0.24	0.06	**0.50**	4	0.21	0.05
5-50-a	-	TL	2641	6171.49	2.34	TL	2631	6191.84	2.35
5-50-b	0.00	18.44	53	6.53	0.12	**15.69**	53	5.98	0.11
5-50-c	0.00	10.15	15	4.19	0.28	**4.26**	15	1.69	0.11
10-50-a	0.00	**5.44**	15	4.30	0.29	5.45	15	4.32	0.29
10-50-b	0.00	10.33	30	7.49	0.25	**10.20**	30	7.38	0.25
10-50-c	0.00	**36.42**	70	24.28	0.35	38.14	70	25.22	0.36
10-100-a	0.00	13.05	16	8.52	0.53	**13.01**	16	8.51	0.53
10-100-b	0.00	182.45	9	101.39	11.27	**8.72**	10	5.11	0.51
10-100-c	0.00	**9.07**	9	5.71	0.63	9.10	9	5.66	0.63
10-200-a	0.00	30.70	14	18.09	1.29	**29.57**	14	17.38	1.24
10-200-b	0.00	68.78	20	28.76	1.44	**67.83**	20	27.96	1.40
10-200-c	0.00	**34.56**	14	17.98	1.28	34.87	14	17.42	1.24
20-100-a	0.00	110.12	65	95.41	1.47	**104.36**	65	90.71	1.40
20-100-b	0.00	656.10	250	567.38	2.27	**641.82**	250	554.17	2.22
20-100-c	0.00	1501.43	52	1290.33	24.81	**72.73**	53	61.13	1.15
20-200-a	0.00	59.03	15	48.47	3.23	**58.92**	15	48.77	3.25
20-200-b	-	TL	642	6344.51	9.88	TL	627	6316.07	10.07
20-200-c	-	TL	144	7073.93	49.12	TL	144	7064.05	49.06

TABLE 7.6: Results of (LBD1) with and without strengthening the master problem on the set EVRSPbasic.

We observe that (LBD1) performs better without strengthening the master problems on nearly all instances. The introduced time overhead of the strengthened formulation is not compensated by the only occasionally reduced number of iterations.

7.2.1.2 Boosted Variants of (LBD1) and (LBD2)

We now compare the results of (LBD1) and (LBD2) with those achieved by the heuristically boosted versions of the algorithms, as described in Chapter 5. In general, an optimality gap of 1e-04 = 0.0001 is used for solving a MIP with CPLEX. In these alternative variants of the logic-based Benders algorithms the gap for solving the master or subproblems is initially set to a higher value. In the following, results are given for an initial optimality gap of either 0.005 or 0.05.

Table 7.7 shows the results of (LBD1BoostM) and compares them with those of the basic version of (LBD1). The best runtime for each instance is printed bold.

Generally, increasing the optimality gap leads to a faster solution of the master problems, resulting in a decreased average solving time. A higher number of iterations is the price that has to be paid. This does not hold for all test runs, as an altered solution to a master problem might lead to the formulation of different Benders cuts and thus other trial values might be tested. We observe that for most instances eventually no benefit is to be gained from the heuristic approach, since the total runtime is longer than for the basic version of (LBD1). The boosted variant only performs better occasionally.

Table 7.8 compares the performance of (LBD2BoostM) with the basic algorithm. When no optimal solution is found within the time limit, the best solution gap is printed bold instead of the runtime.

Similarly to above, the average time for solving a master problem is generally shorter with a higher optimality gap. Additionally, the number of iterations and the time spent for solving the subproblems might change and this may have a significant impact on the total runtime. Therefore, increasing the master optimality gap can result in a better performance, but it could also be worse. For most instances, (LBD2BoostM) with an initial optimality gap of 0.005 achieves the best results. Compared to the basic algorithm, the runtime is substantially shorter on some instances, terminating after considerably fewer iterations. Moreover, it finds a better feasible solution within the time limit for most instances that are not solved to optimality.

Next, we consider the variants of (LBD2), where the initial subproblem optimality gap is increased. Table 7.9 shows the results of (LBD2BoostSU).

As intended, the average time for solving a single subproblem is in general shorter for a higher optimality gap. An increase is, however, also possible, since the adapted Benders cuts may lead to other trial values and consequently the formulation of different subproblems. Additionally, the number of iterations might change considerably. Although the results are not as clear as above, (LBD2BoostSU) with a gap of 0.005 often shows the

Instance	(LBD1BoostM)			Master	Instance	(LBD1BoostM)			Master
	Gap	time[s]	#it	t_{avg}[s]		**Gap**	time[s]	#it	t_{avg}[s]
3-10-a	-	2.53	50	0.029	10-50-c	-	**38.14**	70	0.360
	0.005	**2.36**	51	0.025		0.005	42.91	80	0.365
	0.05	3.29	77	0.022		0.05	75.54	140	0.374
3-10-b	-	0.65	23	0.009	10-100-a	-	**13.01**	16	0.532
	0.005	**0.63**	24	0.009		0.005	13.58	17	0.507
	0.05	**0.63**	24	0.009		0.05	13.57	17	0.509
3-10-c	-	**0.07**	2	0.013	10-100-b	-	**8.72**	10	0.511
	0.005	**0.07**	3	0.009		0.005	9.01	11	0.475
	0.05	**0.07**	3	0.009		0.05	8.71	11	0.470
3-30-a	-	TL	546	12.932	10-100-c	-	**9.10**	9	0.629
	0.005	TL	1491	4.555		0.005	12.50	14	0.607
	0.05	TL	2736	2.367		0.05	13.25	14	0.627
3-30-b	-	0.58	4	0.030	10-200-a	-	**29.57**	14	1.241
	0.005	0.57	5	0.024		0.005	29.98	15	1.179
	0.05	**0.56**	5	0.024		0.05	29.99	15	1.179
3-30-c	-	**2.74**	10	0.072	10-200-b	-	**67.83**	20	1.398
	0.005	2.81	11	0.066		0.005	74.01	21	1.494
	0.05	2.81	11	0.065		0.05	75.10	23	1.417
5-30-a	-	TL	149	48.196	10-200-c	-	34.87	14	1.244
	0.005	TL	232	30.907		0.005	**33.97**	15	1.173
	0.05	TL	896	7.909		0.05	38.18	17	1.186
5-30-b	-	16.97	72	0.101	20-100-a	-	**104.36**	65	1.396
	0.005	105.95	325	0.162		0.005	106.16	66	1.386
	0.05	**10.63**	53	0.086		0.05	108.09	66	1.420
5-30-c	-	**0.50**	4	0.053	20-100-b	-	641.82	250	2.217
	0.005	0.51	5	0.041		0.005	624.51	238	2.263
	0.05	0.51	5	0.041		0.05	**460.64**	196	1.981
5-50-a	-	TL	2631	2.353	20-100-c	-	**72.73**	53	1.153
	0.005	TL	1448	4.65		0.005	75.43	54	1.171
	0.05	TL	2743	2.245		0.05	75.37	56	1.128
5-50-b	-	15.69	53	0.113	20-200-a	-	**58.92**	15	3.251
	0.005	**15.60**	54	0.111		0.005	60.97	16	3.148
	0.05	16.05	55	0.109		0.05	61.00	16	3.141
5-50-c	-	4.26	15	0.113	20-200-b	-	TL	627	10.074
	0.005	4.24	16	0.106		0.005	TL	603	10.57
	0.05	**4.17**	16	0.105		0.05	TL	699	8.84
10-50-a	-	**5.45**	15	0.288	20-200-c	-	TL	144	49.056
	0.005	5.56	16	0.277		0.005	TL	145	48.72
	0.05	5.57	16	0.273		0.05	TL	148	47.767
10-50-b	-	10.20	30	0.246					
	0.005	**9.87**	30	0.235					
	0.05	9.88	31	0.229					

TABLE 7.7: Results of (LBD1BoostM) on the set EVRSPbasic.

Instance	(LBD2BoostM)				Master	Instance	(LBD2BoostM)				Master
	Gap	g[%]	time[s]	#it	t_{avg}[s]		**Gap**	g[%]	time[s]	#it	t_{avg}[s]
3-10-a	-	0.00	**5.28**	17	0.01	10-50-c	-	2.27	TL	175	35.62
	0.005	0.00	5.42	18	0.01		0.005	**1.90**	TL	198	30.75
	0.05	0.00	7.11	23	0.01		0.05	3.22	TL	436	8.50
3-10-b	-	0.00	**1.84**	4	< 0.01	10-100-a	-	0.03	TL	359	16.11
	0.005	0.00	1.86	5	< 0.01		0.005	0.00	**1357.90**	143	4.37
	0.05	0.00	1.85	5	< 0.01		0.05	0.03	TL	366	15.61
3-10-c	-	0.00	**0.44**	1	< 0.01	10-100-b	-	0.11	TL	230	24.29
	0.005	0.00	**0.44**	2	< 0.01		0.005	**0.04**	TL	231	24.09
	0.05	0.00	**0.44**	2	< 0.01		0.05	0.11	TL	233	24.03
3-30-a	-	7.05	TL	330	0.66	10-100-c	-	0.13	TL	233	25.86
	0.005	7.27	TL	308	0.26		0.005	**0.07**	TL	273	21.40
	0.05	**6.99**	TL	294	0.20		0.05	0.13	TL	248	24.00
3-30-b	-	0.00	**0.76**	1	0.01	10-200-a	-	**0.42**	TL	100	61.27
	0.005	0.00	0.77	2	< 0.01		0.005	0.46	TL	99	62.38
	0.05	0.00	0.78	2	< 0.01		0.05	0.46	TL	101	60.35
3-30-c	-	0.00	1762.28	119	0.07	10-200-b	-	0.24	TL	135	28.56
	0.005	0.00	**385.08**	28	0.01		0.005	0.34	TL	161	16.07
	0.05	0.00	1866.96	120	0.07		0.05	**0.18**	TL	161	18.26
5-30-a	-	**28.07**	TL	281	20.74	10-200-c	-	0.54	TL	105	56.46
	0.005	28.69	TL	277	20.83		0.005	0.28	TL	103	59.69
	0.05	29.32	TL	889	2.27		0.05	**0.27**	TL	100	63.09
5-30-b	-	0.00	1236.55	270	0.37	20-100-a	-	30.32	TL	18	395.82
	0.005	0.00	**236.42**	65	0.08		0.005	3.40	TL	36	195.46
	0.05	0.00	2474.70	488	0.60		0.05	**2.34**	TL	42	167.06
5-30-c	-	0.00	**0.51**	1	0.01	20-100-b	-	38.55	TL	23	302.07
	0.005	0.00	**0.51**	2	< 0.01		0.005	38.55	TL	28	246.36
	0.05	0.00	**0.51**	2	< 0.01		0.05	**36.09**	TL	74	84.69
5-50-a	-	6.23	TL	187	3.92	20-100-c	-	**1.71**	TL	36	195.75
	0.005	**1.95**	TL	223	2.52		0.005	**1.71**	TL	50	139.81
	0.05	10.46	TL	231	0.43		0.05	1.74	TL	178	35.89
5-50-b	-	0.00	TL	311	21.62	20-200-a	-	**1.71**	TL	42	160.29
	0.005	0.00	TL	495	13.19		0.005	2.32	TL	81	76.59
	0.05	0.00	TL	306	22.09		0.05	**1.71**	TL	39	173.50
5-50-c	-	0.00	595.43	225	1.35	20-200-b	-	**38.95**	TL	20	208.63
	0.005	0.00	**218.00**	137	0.28		0.005	**38.95**	TL	24	156.58
	0.05	0.00	622.83	226	1.45		0.05	**38.95**	TL	22	177.76
10-50-a	-	**0.00**	TL	631	10.49	20-200-c	-	-	TL	1	TL
	0.005	**0.00**	TL	641	10.32		0.005	-	TL	1	TL
	0.05	**0.00**	TL	628	10.53		0.05	95.27	TL	14	504.11
10-50-b	-	**0.00**	TL	280	24.43						
	0.005	0.02	TL	324	20.92						
	0.05	0.02	TL	262	26.21						

TABLE 7.8: Results of (LBD2BoostM) on the set EVRSPbasic.

Instance	(LBD2BoostSU)				Sub	Instance	(LBD2BoostSU)				Sub
	Gap	g[%]	time[s]	#it	t_{avg}[s]		**Gap**	g[%]	time[s]	#it	t_{avg}[s]
3-10-a	-	0.00	**5.28**	17	0.049	10-50-c	-	2.27	TL	175	0.229
	0.005	0.00	5.30	18	0.047		0.005	2.55	TL	157	0.258
	0.05	0.00	5.35	18	0.048		0.05	**1.63**	TL	155	0.212
3-10-b	-	0.00	**1.84**	4	0.052	10-100-a	-	0.03	TL	359	0.180
	0.005	0.00	2.04	5	0.053		0.005	0.00	**916.48**	100	0.210
	0.05	0.00	2.05	5	0.053		0.05	0.03	TL	201	0.221
3-10-c	-	0.00	**0.44**	1	0.134	10-100-b	-	**0.11**	TL	230	0.218
	0.005	0.00	0.87	2	0.135		0.005	2.26	TL	245	0.235
	0.05	0.00	0.85	2	0.131		0.05	0.25	TL	198	0.231
3-30-a	-	7.05	TL	330	1.081	10-100-c	-	**0.13**	TL	233	0.188
	0.005	**6.49**	TL	351	1.046		0.005	**0.13**	TL	248	0.198
	0.05	9.18	TL	455	0.772		0.05	0.36	TL	208	0.181
3-30-b	-	0.00	**0.76**	1	0.223	10-200-a	-	0.42	TL	100	0.255
	0.005	0.00	**0.76**	1	0.220		0.005	0.16	TL	237	0.260
	0.05	0.00	**0.76**	1	0.222		0.05	**0.09**	TL	229	0.244
3-30-c	-	0.00	1762.28	119	2.880	10-200-b	-	0.24	TL	135	0.317
	0.005	0.00	718.76	141	0.995		0.005	**0.23**	TL	137	0.299
	0.05	0.00	**134.48**	42	0.599		0.05	0.46	TL	108	0.310
5-30-a	-	28.07	TL	281	0.169	10-200-c	-	0.54	TL	105	0.293
	0.005	16.28	TL	338	0.165		0.005	**0.05**	TL	197	0.280
	0.05	**11.19**	TL	313	0.181		0.05	0.16	TL	186	0.320
5-30-b	-	0.00	1236.55	270	0.240	20-100-a	-	30.32	TL	18	0.056
	0.005	0.00	**919.01**	220	0.233		0.005	30.32	TL	18	0.056
	0.05	0.00	1224.97	277	0.224		0.05	**25.48**	TL	18	0.058
5-30-c	-	0.00	**0.51**	1	0.084	20-100-b	-	38.55	TL	23	0.125
	0.005	0.00	0.98	2	0.082		0.005	47.16	TL	26	0.142
	0.05	0.00	1.02	2	0.084		0.05	**30.62**	TL	43	0.115
5-50-a	-	6.23	TL	187	0.975	20-100-c	-	**1.71**	TL	36	0.067
	0.005	5.79	TL	217	0.904		0.005	18.62	TL	33	0.065
	0.05	**3.40**	TL	289	0.588		0.05	17.14	TL	24	0.070
5-50-b	-	**0.00**	TL	311	0.087	20-200-a	-	1.71	TL	42	0.148
	0.005	**0.00**	TL	344	0.078		0.005	1.79	TL	45	0.139
	0.05	0.19	TL	302	0.084		0.05	**1.02**	TL	43	0.141
5-50-c	-	0.00	595.43	225	0.154	20-200-b	-	**38.95**	TL	20	1.080
	0.005	0.00	**168.49**	96	0.144		0.005	38.96	TL	20	0.963
	0.05	0.00	284.33	125	0.155		0.05	40.00	TL	20	0.483
10-50-a	-	**0.00**	TL	631	0.042	20-200-c	-	-	TL	1	TL
	0.005	**0.00**	TL	638	0.042		0.005	-	TL	1	TL
	0.05	**0.00**	TL	596	0.042		0.05	-	TL	1	TL
10-50-b	-	**0.00**	TL	280	0.061						
	0.005	**0.00**	TL	230	0.061						
	0.05	**0.00**	TL	260	0.063						

TABLE 7.9: Results of (LBD2BoostSU) on the set EVRSPbasic.

best performance. Again, on some instances the adapted algorithm has a substantially shorter runtime than the basic version of (LBD2).

Table 7.10 compares the results of the third variant (LBD2BoostSL) with the basic algorithm. Here, in contrast to the previous method, the Benders cuts are not adapted.

Instance	(LBD2BoostSL)				Sub	Instance	(LBD2BoostSL)				Sub
	Gap	g[%]	time[s]	#it	t_{avg}[s]		**Gap**	g[%]	time[s]	#it	t_{avg}[s]
3-10-a	-	0.00	5.28	17	0.049	10-50-c	-	**2.27**	TL	175	0.229
	0.005	0.00	**5.16**	17	0.048		0.005	2.30	TL	168	0.258
	0.05	0.00	5.17	17	0.048		0.05	3.19	TL	161	0.180
3-10-b	-	0.00	**1.84**	4	0.052	10-100-a	-	0.03	TL	359	0.180
	0.005	0.00	1.90	4	0.054		0.005	0.00	**915.83**	100	0.208
	0.05	0.00	**1.84**	4	0.052		0.05	0.03	TL	202	0.219
3-10-c	-	0.00	**0.44**	1	0.134	10-100-b	-	0.11	TL	230	0.218
	0.005	0.00	0.45	1	0.137		0.005	**0.07**	TL	185	0.228
	0.05	0.00	0.96	2	0.097		0.05	0.31	TL	133	0.255
3-30-a	-	7.05	TL	330	1.081	10-100-c	-	**0.13**	TL	233	0.188
	0.005	**6.49**	TL	354	1.040		0.005	**0.13**	TL	248	0.193
	0.05	9.18	TL	454	0.775		0.05	0.36	TL	206	0.185
3-30-b	-	0.00	**0.76**	1	0.223	10-200-a	-	0.42	TL	100	0.255
	0.005	0.00	0.77	1	0.226		0.005	**0.16**	TL	235	0.266
	0.05	0.00	0.77	1	0.226		0.05	0.17	TL	269	0.247
3-30-c	-	0.00	1762.28	119	2.880	10-200-b	-	0.24	TL	135	0.317
	0.005	0.00	**150.59**	14	1.926		0.005	**0.23**	TL	136	0.302
	0.05	0.00	237.62	75	0.572		0.05	0.46	TL	108	0.315
5-30-a	-	28.07	TL	281	0.169	10-200-c	-	0.54	TL	105	0.293
	0.005	16.28	TL	336	0.166		0.005	**0.07**	TL	300	0.269
	0.05	**11.19**	TL	317	0.181		0.05	0.23	TL	146	0.317
5-30-b	-	0.00	**1236.55**	270	0.240	20-100-a	-	30.32	TL	18	0.056
	0.005	0.00	2425.18	521	0.226		0.005	30.32	TL	18	0.057
	0.05	0.56	TL	1137	0.228		0.05	**25.48**	TL	18	0.058
5-30-c	-	0.00	**0.51**	1	0.084	20-100-b	-	38.55	TL	23	0.125
	0.005	0.00	0.52	1	0.085		0.005	**28.23**	TL	25	0.139
	0.05	0.00	0.52	1	0.085		0.05	47.20	TL	39	0.144
5-50-a	-	6.23	TL	187	0.975	20-100-c	-	**1.71**	TL	36	0.067
	0.005	3.19	TL	232	0.834		0.005	**1.71**	TL	36	0.070
	0.05	**2.39**	TL	277	0.563		0.05	3.86	TL	34	0.065
5-50-b	-	**0.00**	TL	311	0.087	20-200-a	-	1.71	TL	42	0.148
	0.005	**0.00**	TL	336	0.090		0.005	**0.88**	TL	54	0.139
	0.05	0.19	TL	310	0.085		0.05	2.48	TL	45	0.137
5-50-c	-	0.00	595.43	225	0.154	20-200-b	-	38.95	TL	20	1.080
	0.005	0.00	626.70	225	0.157		0.005	38.96	TL	18	1.035
	0.05	0.00	**216.51**	108	0.171		0.05	**35.93**	TL	23	0.536
10-50-a	-	**0.00**	TL	631	0.042	20-200-c	-	-	TL	1	TL
	0.005	**0.00**	TL	630	0.042		0.005	-	TL	1	TL
	0.05	**0.00**	TL	595	0.043		0.05	-	TL	1	TL
10-50-b	-	**0.00**	TL	280	0.061						
	0.005	**0.00**	TL	275	0.061						
	0.05	0.02	TL	283	0.063						

TABLE 7.10: Results of (LBD2BoostSL) on the set EVRSPbasic.

Although the subproblem optimality gap is increased, the average solving time is longer for many instances. As explained above, this can be caused by the formulation of different Benders cuts, where only a lower bound on the optimal profit of the considered subproblems is used. For those instances where (LBD2BoostSL) finishes within the time limit, it returns an optimal solution, even though possibly invalid Benders cuts are generated by using suboptimal solutions of the subproblems. Since there need not be a unique optimal solution, some optima might be cut away, but at least one remains that is found by the algorithm. Similarly to above, heuristic boosting of (LBD2) leads to a significantly improved performance on some instances, especially with an initial gap of 0.005.

Finally, the three boosted variants of (LBD2), all with an initial optimality gap of 0.005, are compared in Table 7.11.

Instance	**(LBD2)**		**(LBD2BoostM)**		**(LBD2BoostSU)**		**(LBD2BoostSL)**	
	g[%]	time[s]	g[%]	time[s]	g[%]	time[s]	g[%]	time[s]
3-10-a	0.00	5.28	0.00	5.42	0.00	5.30	0.00	**5.16**
3-10-b	0.00	**1.84**	0.00	1.86	0.00	2.04	0.00	1.90
3-10-c	0.00	**0.44**	0.00	**0.44**	0.00	0.87	0.00	0.45
3-30-a	7.05	TL	7.27	TL	**6.49**	TL	**6.49**	TL
3-30-b	0.00	**0.76**	0.00	0.77	0.00	**0.76**	0.00	0.77
3-30-c	0.00	1762.28	0.00	385.08	0.00	718.76	0.00	**150.59**
5-30-a	28.07	TL	28.69	TL	**16.28**	TL	**16.28**	TL
5-30-b	0.00	1236.55	0.00	**236.42**	0.00	919.01	0.00	2425.18
5-30-c	0.00	**0.51**	0.00	**0.51**	0.00	0.98	0.00	0.52
5-50-a	6.23	TL	**1.95**	TL	5.79	TL	3.19	TL
5-50-b	**0.00**	TL	**0.00**	TL	**0.00**	TL	**0.00**	TL
5-50-c	0.00	595.43	0.00	218.00	0.00	**168.49**	0.00	626.70
10-50-a	**0.00**	TL	**0.00**	TL	**0.00**	TL	**0.00**	TL
10-50-b	**0.00**	TL	0.02	TL	**0.00**	TL	**0.00**	TL
10-50-c	2.27	TL	**1.90**	TL	2.55	TL	2.30	TL
10-100-a	0.03	TL	0.00	1357.90	0.00	916.48	0.00	**915.83**
10-100-b	0.11	TL	**0.04**	TL	2.26	TL	0.07	TL
10-100-c	0.13	TL	**0.07**	TL	0.13	TL	0.13	TL
10-200-a	0.42	TL	0.46	TL	**0.16**	TL	**0.16**	TL
10-200-b	0.24	TL	0.34	TL	**0.23**	TL	**0.23**	TL
10-200-c	0.54	TL	0.28	TL	**0.05**	TL	0.07	TL
20-100-a	30.32	TL	**3.40**	TL	30.32	TL	30.32	TL
20-100-b	38.55	TL	38.55	TL	47.16	TL	**28.23**	TL
20-100-c	**1.71**	TL	**1.71**	TL	18.62	TL	**1.71**	TL
20-200-a	1.71	TL	2.32	TL	1.79	TL	**0.88**	TL
20-200-b	38.95	TL	38.95	TL	38.96	TL	38.96	TL
20-200-c	-	TL	-	TL	-	TL	-	TL

TABLE 7.11: Comparison of the results of the boosted variants of (LBD2) with an initial optimality gap of 0.005 on the set EVRSPbasic.

We observe that the different strategies for boosting (LBD2) all lead to similar results. Depending on the instance, one variant or another shows a better performance. Considerable improvements compared to the basic algorithm can be achieved in many cases. It has to be mentioned, however, that the results are still much worse than those of (MIP1) and (LBD1).

7.2.2 EVRSPhet

Having described and analyzed the results of all algorithms and their variants on the set EVRSPbasic, we now compare the performance of (MIP1), (LBD1) and (LBD2) on the other sets of instances. Table 7.12 shows the results on the set EVRSPhet with heterogeneous facilities.

Instance	UB	**(MIP1)**		**(LBD1)**			**(LBD2)**		
		g[%]	time[s]	g[%]	time[s]	#it	g[%]	time[s]	#it
3-10-a	10.00	0.00	0.65	0.00	**0.19**	6	0.00	2.81	4
3-10-b	10.00	0.00	0.93	0.00	**0.47**	15	0.00	15.62	37
3-10-c	9.88	0.00	**0.95**	0.00	1.41	35	0.00	15.50	36
3-30-a	29.91	0.00	**3.12**	0.00	6.00	50	0.00	TL	2139
3-30-b	29.69	0.00	**5.81**	0.00	16.69	92	0.00	TL	2589
3-30-c	28.74	0.00	**6.00**	-	TL	1289	0.00	TL	1048
5-30-a	30.00	0.00	9.79	0.00	**9.39**	59	0.00	2751.87	907
5-30-b	30.00	0.00	8.87	0.00	**3.37**	18	0.00	117.74	76
5-30-c	29.85	0.00	**4.64**	0.00	6.50	49	0.00	5416.61	641
5-50-a	48.67	0.00	**9.89**	0.00	38.29	137	0.04	TL	102
5-50-b	49.52	0.00	**10.44**	0.00	28.82	121	0.00	TL	1090
5-50-c	49.54	0.00	**8.03**	0.00	18.52	55	0.00	3432.58	797
10-50-a	48.02	0.00	**12.51**	0.00	61.82	184	0.02	TL	392
10-50-b	45.63	0.00	**36.09**	0.00	110.40	234	0.02	TL	702
10-50-c	47.93	0.00	40.42	0.00	**25.03**	57	0.00	897.79	184
10-100-a	91.76	0.00	**144.66**	-	TL	2374	0.23	TL	3
10-100-b	88.85	0.00	**70.18**	0.00	746.47	644	0.17	TL	188
10-100-c	88.91	0.00	**86.06**	-	TL	2833	0.19	TL	188
10-200-a	187.19	0.00	**573.75**	-	TL	1817	0.28	TL	58
10-200-b	185.57	0.00	**135.43**	-	TL	1611	0.40	TL	15
10-200-c	187.03	0.00	**203.65**	-	TL	1959	0.30	TL	86
20-100-a	93.48	0.00	**329.89**	0.00	1289.61	584	0.31	TL	141
20-100-b	92.13	0.00	618.67	0.00	**567.72**	304	0.34	TL	214
20-100-c	94.23	0.00	**231.13**	0.00	1009.42	468	0.39	TL	168
20-200-a	189.37	0.00	**2902.55**	-	TL	1004	0.45	TL	36
20-200-b	186.84	0.00	**1351.91**	-	TL	1029	0.48	TL	26
20-200-c	181.78	0.00	**463.43**	-	TL	1454	0.27	TL	73

TABLE 7.12: Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPhet.

- (MIP1) clearly achieves the best results here, it solves all instances to optimality within the time limit and has the shortest runtime for most.
- (LBD1) still performs better on some instances, but often it takes substantially more time than (MIP1). Compared to the test runs on instances with homogeneous facilities (see Table 7.2), the algorithm needs more iterations until it finds an optimal solution. This is due to the fact that a Benders cut, which is derived from the solution of a subproblem at one facility, cannot be generalized to all other facilities, as it was previously the case. Now it can only be extended to dominated ones. Therefore, it may be necessary to try to assign the same set of vehicles to different facilities, which requires multiple iterations.
- Again, (LBD2) shows a poor performance, taking much more time to find an optimal solution than the other algorithms.

7.2.3 EVRSPres

We have seen that (LBD1) performs especially well on instances where all or nearly all vehicles can be fully recharged, which indicates that there are enough charging resources available. In order to verify this assumption, the algorithms are tested on the set EVRSPres which contains instances with tighter resource limits, with on average only half as many parking and charging spots and half the amount of power available at each facility compared to the instances of EVRSPbasic. Results are given in Table 7.13.

(LBD1) indeed takes much longer to find an optimal solution than for the instances of EVRSPbasic, needing a lot more iterations until it terminates. Similarly, (LBD2) has a significantly longer runtime or does not finish within the time limit. The performance of (MIP1), on the other hand, does not change substantially. This shows that (MIP1) is much more robust for instances with varying specifications than the logic-based Benders algorithms.

7.2.4 EVRSPlarge

Finally, we analyze the performance of the algorithms on the set EVRSPlarge which contains much larger instances than those considered before. Now an optimal schedule for the recharging of several hundreds of vehicles shall be determined. This is a task which is not improbable to occur in reality.

Table 7.14 compares the results of (MIP1), (LBD1) and (LBD2). Some instances are not solved to optimality within the time limit by any of the algorithms. As mentioned before,

Instance	UB	(MIP1)		(LBD1)			(LBD2)		
		g[%]	time[s]	g[%]	time[s]	#it	g[%]	time[s]	#it
3-10-a	8.15	0.00	**1.04**	0.00	5.42	113	0.00	15.93	30
3-10-b	8.52	0.00	0.86	0.00	**0.77**	24	0.00	13.19	36
3-10-c	5.69	0.00	**1.20**	0.00	3.49	61	0.00	4.81	8
3-30-a	4.28	0.00	**1.36**	0.00	48.82	408	0.00	3.62	12
3-30-b	8.97	0.00	**2.13**	-	TL	386	31.32	TL	190
3-30-c	4.94	0.00	**1.51**	-	TL	972	0.00	7.45	20
5-30-a	12.35	0.00	**7.37**	0.00	856.73	409	9.65	TL	35
5-30-b	7.07	0.00	**3.75**	-	TL	632	2.56	TL	77
5-30-c	5.54	0.00	**2.18**	0.00	375.89	949	0.00	5.43	16
5-50-a	44.98^{M}	1.69	TL	-	TL	328	7.82	TL	562
5-50-b	39.59^{M}	0.01	TL	-	TL	470	3.95	TL	756
5-50-c	43.31	0.00	11.04	0.00	**10.93**	48	0.27	TL	394
10-50-a	42.58^{M}	1.71	TL	-	TL	454	29.59	TL	75
10-50-b	44.46	0.00	63.19	0.00	**11.91**	23	0.57	TL	225
10-50-c	46.74	0.00	23.30	0.00	**14.03**	36	0.33	TL	116
10-100-a	36.15^{M}	0.28	TL	-	TL	89	34.08	TL	146
10-100-b	95.06	0.00	**1780.30**	-	TL	443	10.87	TL	208
10-100-c	94.09	0.00	103.24	0.00	**97.13**	94	1.32	TL	137
10-200-a	187.07	0.00	**2075.32**	-	TL	839	1.50	TL	56
10-200-b	137.23^{M}	3.04	TL	-	TL	127	82.41	TL	44
10-200-c	189.14	0.00	**141.83**	0.00	151.25	64	2.03	TL	61
20-100-a	96.06	0.00	**2065.00**	-	TL	625	10.16	TL	36
20-100-b	69.15	0.00	364.81	0.00	**180.42**	69	1.14	TL	133
20-100-c	93.89	0.00	446.46	0.00	**324.59**	126	22.31	TL	23
20-200-a	67.77	0.00	**1050.62**	-	TL	131	-	TL	1
20-200-b	113.97^{M}	0.86	TL	-	TL	184	92.18	TL	15
20-200-c	194.50^{M}	3.94	TL	-	TL	404	35.61	TL	19

TABLE 7.13: Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPres.

the best known upper bound on the objective value is listed instead of the optimal value in these cases. A superscript "^{L}" marks the bounds obtained by (LBD1). Furthermore, for many instances the compact MIP model reaches a size CPLEX is not able to process any more. The program terminates because the memory limit ("ML") is exceeded.

Up to a certain problem size, the results are similar to those on smaller instances (see Table 7.2 and Section 7.2.1). (MIP1) and (LBD1) perform quite well on the majority of instances, while (LBD2) is not able to compute an optimal solution for any of them within the time limit. In many cases, (LBD1) terminates much faster than (MIP1).

The more interesting instances are those exceeding a certain size. When the recharging of 400 vehicles needs to be scheduled on 40 facilities, the compact MIP model becomes too large to be solved by CPLEX. This is where a big advantage of logic-based Benders decomposition lies. Decoupling the problem into smaller subproblems, the arising MIP

Instance	UB	(MIP1)		(LBD1)			(LBD2)		
		g[%]	time[s]	g[%]	time[s]	#it	g[%]	time[s]	#it
20-300-a	284.83	0.00	2500.42	0	**278.84**	24	6.53	TL	77
20-300-b	285.96	0.00	961.80	0	**575.08**	68	20.05	TL	24
20-300-c	287.00	0.00	**403.54**	0	557.89	96	13.58	TL	38
30-300-a	299.86	0.00	1409.47	0	**123.22**	9	2.53	TL	25
30-300-b	299.75	0.00	649.02	0	**108.43**	10	2.55	TL	39
30-300-c	299.86	0.00	796.88	0	**98.52**	9	0.83	TL	32
30-400-a	373.87	0.00	**769.10**	0	1066.34	89	24.48	TL	22
30-400-b	373.05	0.00	**849.19**	0	931.66	75	13.81	TL	24
30-400-c	370.36	0.00	**903.20**	0	943.96	77	26.45	TL	34
40-300-a	249.60^{L}	31.44	TL	-	TL	243	81.74	TL	6
40-300-b	278.27	-	ML	0	**1103.96**	69	18.91	TL	12
40-300-c	248.71^{L}	51.87	TL	-	TL	187	90.79	TL	6
40-400-a	398.75	-	ML	0	**275.09**	13	8.03	TL	48
40-400-b	399.32^{L}	-	ML	-	TL	157	70.68	TL	12
40-400-c	398.56^{L}	-	ML	-	TL	127	74.05	TL	10
40-500-a	470.20	-	ML	0	**1844.75**	80	23.96	TL	14
40-500-b	476.07	-	ML	0	**2219.11**	94	9.01	TL	13
40-500-c	471.02	-	ML	0	**1665.18**	71	37.32	TL	11
50-300-a	262.32^{L}	-	ML	-	TL	233	65.84	TL	5
50-300-b	246.94^{L}	-	ML	-	TL	147	68.95	TL	4
50-300-c	281.96	-	ML	0	**1287.56**	78	18.65	TL	13
50-500-a	387.33^{L}	-	ML	-	TL	10	18.72	TL	16
50-500-b	246.61^{L}	-	ML	-	TL	23	42.16	TL	15
50-500-c	499.42	-	ML	0	**497.82**	13	4.15	TL	26
50-1000-a	959.30^{L}	-	ML	-	TL	101	37.27	TL	9
50-1000-b	955.88	-	ML	0	**6719.15**	97	32.53	TL	9
50-1000-c	955.67	-	ML	0	**2795.20**	39	15.69	TL	5

TABLE 7.14: Results of (MIP1), (LBD1) and (LBD2) on the set EVRSPlarge.

models can be solved successfully. For many instances, (LBD1) is still able to find an optimal solution within the time limit. For the others, the logic-based Benders algorithms at least return bounds on the optimal objective value. The value of the last master problem solved by (LBD1) serves as an upper bound, while the best integer solution value found by (LBD2) constitutes a lower bound.

Although (LBD2) generally shows a much worse performance than the other algorithms, it at least gives us a feasible solution when interrupted early. This is useful for those instances where the compact MIP model is too large to be solved and (LBD1) does not finish within the time limit. On smaller instances it is not worth to use boosting strategies for (LBD2), since they cannot improve the runtime to such an extent to be comparable with the other algorithms. But for larger instances, we might obtain a better feasible solution in this way.

Table 7.15 compares the gap of the best feasible solution found by (LBD2) with those of the boosted variants (LBD2BoostM), (LBD2BoostSU) and (LBD2BoostSL), all with an initial optimality gap of 0.005. Furthermore, the results of the greedy algorithm are given.

Instance	**(LBD2)**	**(LBD2BoostM)**	**(LBD2BoostSU)**	**(LBD2BoostSL)**	**(GREEDY)**
	g[%]	g[%]	g[%]	g[%]	g[%]
20-300-a	**6.53**	**6.53**	**6.53**	**6.53**	20.65
20-300-b	**20.05**	32.86	29.74	23.37	27.62
20-300-c	**13.58**	**13.58**	**13.58**	**13.58**	21.09
30-300-a	**2.53**	8.05	8.75	7.01	11.31
30-300-b	2.55	**1.24**	2.90	2.90	11.25
30-300-c	**0.83**	**0.83**	**0.83**	**0.83**	15.17
30-400-a	24.48	**7.64**	24.48	24.48	17.86
30-400-b	**13.81**	15.87	**13.81**	**13.81**	17.07
30-400-c	26.45	25.87	26.45	26.45	**15.89**
40-300-a	81.74	81.74	84.40	84.40	**28.61**
40-300-b	**18.91**	**18.91**	**18.91**	**18.91**	21.47
40-300-c	90.79	90.79	90.79	90.79	**54.47**
40-400-a	**8.03**	13.57	10.05	10.05	11.87
40-400-b	70.68	70.68	70.72	70.76	**27.45**
40-400-c	74.05	68.69	74.06	69.32	**31.30**
40-500-a	**23.96**	**23.96**	**23.96**	**23.96**	26.51
40-500-b	9.01	9.81	**5.35**	13.65	15.64
40-500-c	37.32	30.13	32.41	32.41	**25.66**
50-300-a	65.84	69.44	69.44	64.08	**23.83**
50-300-b	68.95	78.23	68.95	68.95	**29.79**
50-300-c	18.65	18.65	18.65	18.65	**5.98**
50-500-a	18.72	19.65	**18.15**	**18.15**	23.08
50-500-b	**42.16**	42.37	**42.16**	**42.16**	46.14
50-500-c	4.15	4.15	**3.50**	4.15	11.72
50-1000-a	37.27	29.34	30.22	31.77	**27.24**
50-1000-b	32.53	22.06	**21.43**	**21.43**	31.60
50-1000-c	**15.69**	15.84	21.76	21.76	27.78

TABLE 7.15: Comparison of the results of the boosted variants of (LBD2), with an initial optimality gap of 0.005, with the basic version and (GREEDY) on the set EVRSPlarge.

For some instances, heuristic boosting of (LBD2) indeed yields a better feasible solution. This is, however, not true in general. Moreover, the greedy algorithm finds a better solution than all variants of (LBD2) for many instances, while terminating within a few seconds. Hence, for the purpose of obtaining sufficiently good feasible solutions for larger instances that cannot be solved by (LBD1), it seems more promising to develop a better heuristic than trying to boost (LBD2).

Chapter 8

Conclusions and Future Work

Managing the recharging of electric vehicles poses great challenges, since the charging process takes a relatively long time and charging facilities are still very scarce. In this work, we considered the problem of scheduling the recharging of a fleet of electric vehicles to multiple facilities with limited resources, designated as "Electric Vehicles Recharge Scheduling Problem (EVRSP)". The objective was to find a feasible charging schedule that fulfills the vehicles' energy demands as much as possible.

The problem was formulated as a MIP model, denoted (MIP1), and also a simple greedy approach for solving it was presented.

The focus of this thesis, however, was to apply logic-based Benders decomposition to the EVRSP. It decomposes the problem into a master and subproblem and solves them iteratively. The solution of the master problem specifies a trial assignment for some of the variables, which leads to a subproblem that is in general much easier to solve than the original one. From the solution of the subproblem we infer bounds on the optimal objective value for other assignments, called Benders cuts, that are added to the master problem.

In this work, decompositions of the EVRSP were defined in such a way that the subproblem decomposes into multiple independent problems, one for each facility. The master problem of the first decomposition already determines the amount of energy a vehicle will be charged with, while the subproblems only need to compute a feasible schedule for the given assignment. In the second decomposition, the master problem merely allocates vehicles to facilities. Optimal charging schedules are calculated in the subproblems. The corresponding algorithms were designated (LBD1) and (LBD2).

Moreover, an approach for heuristically boosting the logic-based Benders algorithms was presented. The optimality gap for solving either the master or subproblems was

increased, obtaining possibly suboptimal solutions for them, but in a shorter time. It was described how the algorithms can be adapted, in order to still get an optimal solution to the whole problem eventually.

To evaluate their performance, all algorithms were tested on different sets of randomly generated instances. (MIP1) and (LBD1) generally achieved good results, while (LBD2) often needed much more time to find an optimal solution. (LBD1) worked especially well on instances with homogeneous facilities and enough charging resources available. This stems from the fact that logic-based Benders decomposition solves the problem on a trial and error basis. When there are plenty of solutions with optimal value in the feasible set, the algorithm finds an optimal schedule in a relatively short time. If there are, however, only a few possible assignments with optimal profit, it generally needs much more iterations to terminate, resulting in a longer runtime. The second decomposition led to rather poor results. Obviously, it is more beneficial to define subproblems that are mere feasibility problems. Although the boosted versions of the logic-based Benders algorithms showed a better performance on some instances, they did not produce the desired improvements in general.

While the compact MIP model exceeded the memory limit on larger instances, it was still possible to run the logic-based Benders algorithms, as they decompose the whole problem into smaller ones. On many of the larger instances, (LBD1) found an optimal solution within the time limit or at least returned an upper bound on the optimal profit. On the other hand, feasible solutions were obtained by (LBD2) for all instances.

8.1 Future Work

In this work, both master and subproblems of the EVRSP were formulated as MIP models. Hooker [12] emphasizes that one advantage of logic-based Benders decomposition is that it enables the combination of mixed-integer programming and constraint programming (CP). Their relative strengths can be exploited by applying MIP to the allocation task of the master problem and using CP in the subproblems for scheduling. We could use this idea for the EVRSP, formulating the subproblems as CP problems.

Another issue that can be addressed by future work are strategies for repairing the infeasible intermediate solutions of the first decomposition, generating feasible ones. Of course, this could easily be done by just unallocating vehicles until a feasible schedule is found for every facility. This would, however, strongly reduce the objective value and probably lead to solutions with low profit. It should be investigated how this can be done more sophisticatedly, in order to produce sufficiently good feasible solutions.

Bibliography

[1] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[2] D. Bertsimas and J. Tsitsiklis. *Introduction to linear optimization.* Athena Scientific, 1st edition, 1997.

[3] S. Bessler and J. Grønbæk. Routing EV users towards an optimal charging plan. In *2012 EVS26 Online Conference Proceedings*, Los Angeles, California, 2012.

[4] D. Bučar. Electric vehicles recharge scheduling with time windows. Master thesis, Vienna University of Technology, Vienna, Austria, 2014.

[5] D. Bučar, S. Bessler, N. Musliu, and J. Grønbæk. Scheduling of electric vehicle charging operations. *Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA)*, Ghent, Belgium, 2013.

[6] ChargePoint. EV models. `http://www.chargepoint.com/evs`. Accessed: June 2015.

[7] K. Clement, E. Haesen, and J. Driesen. Coordinated charging of multiple plug-in hybrid electric vehicles in residential distribution grids. In *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pages 1–7, Seattle, Washington, 2009.

[8] G. B. Dantzig. *Linear programming and extensions.* Princeton University Press, Princeton, New Jersey, 1963.

[9] J. Dargay, D. Gately, and M. Sommer. Vehicle ownership and income growth, worldwide: 1960-2030. *Energy Journal*, 28(4), 2007.

[10] Electric Power Research Institute. Environmental assessment of plug-in hybrid electric vehicles. volume 1: Nationwide greenhouse gas emissions. Technical Report 1015325, Palo Alto, California, 2007.

[11] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[12] J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55(3):588–602, 2007.

[13] J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming. A Publication of the Mathematical Programming Society*, 96(1, Ser. A):33–60, 2003.

[14] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademiia Nauk SSSR*, 224:1093–1096, 1979. English Translation: *Soviet Mathematics Doklady*, 20:191–194, 1979.

[15] S. Mal, A. Chattopadhyay, A. Yang, and R. Gadh. Electric vehicle smart charging and vehicle-to-grid operation. *International Journal of Parallel, Emergent and Distributed Systems*, 28(3):249–265, 2013.

[16] H. Qin and W. Zhang. Charging scheduling with minimal waiting in a network of electric vehicles and charging stations. In *Proceedings of the Eighth ACM International Workshop on Vehicular Inter-networking*, VANET '11, pages 51–60, Las Vegas, Nevada, 2011.

[17] G. R. Raidl, T. Baumhauer, and B. Hu. Boosting an exact logic-based benders decomposition approach by variable neighborhood search. *Electronic Notes in Discrete Mathematics*, 47(0):149–156, 2015. The 3rd International Conference on Variable Neighborhood Search (VNS'14).

[18] R. Rezania. *Integration of electric vehicles in the Austrian electricity system.* PhD thesis, Vienna University of Technology, Vienna, Austria, 2013.

[19] P. Sánchez-Martín and G. Sánchez. Optimal electric vehicles consumption management at parking garages. In *PowerTech, 2011 IEEE Trondheim*, pages 1–7, Trondheim, Norway, 2011.

[20] O. Sundström and C. Binding. Planning electric-drive vehicle charging under constrained grid conditions. In *Power System Technology (POWERCON), 2010 International Conference on*, pages 1–6, Hangzhou, China, 2010.

[21] J. Timpner and L. Wolf. Design and evaluation of charging station scheduling strategies for electric vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):579–588, 2014.

[22] L. A. Wolsey. *Integer programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York, 1998. A Wiley-Interscience Publication.

[23] S. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.

[24] A. Zakariazadeh, S. Jadid, and P. Siano. Multi-objective scheduling of electric vehicles in smart distribution system. *Energy Conversion and Management*, 79(0):43–53, 2014.