TU WIEN Informatics

Computational Methods for Fleet Scheduling in E-Mobility

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Johannes Varga, BSc BSc
Matrikelnummer 01526325

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Mitwirkung: Projektass.(FWF) Marc Huber, MSc

Wien, 1. August 2021

Johannes Varga	Günther Raidl

TU WIEN Informatics

Computational Methods for Fleet Scheduling in E-Mobility

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Johannes Varga, BSc BSc
Registration Number 01526325

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Assistance: Projektass.(FWF) Marc Huber, MSc

Vienna, 1st August, 2021

Johannes Varga	Günther Raidl

Erklärung zur Verfassung der Arbeit

Johannes Varga, BSc BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten 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.

Wien, 1. August 2021

Johannes Varga

Acknowledgements

Here I want to express my thanks to everyone who helped and supported me in composing this thesis.

First to my supervisor Günther Raidl and my co-supervisor Marc Huber for guiding me in our regular meetings when I got lost in some details. With your ideas you gave this project a direction. I am also grateful for the countless hours you spent on revising the thesis.

Furthermore I would like to thank Honda Research Institute Europe for providing funds for the project and especially Steffen Limmer and Tobias Rodemann for the valuable discussions and the feedback.

I am also very grateful to my parents for providing livelihood. Thanks to you I was able to finish this thesis and my degree without having to concern about earning my living.

Last but not least, I want to thank TU Wien and especially the research unit Algorithms and Complexity for providing the infrastructure that was needed to complete the project. That concerns in particular the cluster that was used to perform the computations.

Danksagung

An dieser Stelle möchte ich meinen Dank all jenen aussprechen, die mich unterstützt haben und mir dabei geholfen haben, diese Arbeit fertigzustellen.

Zuerst gebührt mein Dank meinen Betreuern Günther Raidl und Marc Huber. In den regelmäßigen Meetings habt ihr mir mit euren Ratschlägen und Ideen eine Richtung gegeben, wenn ich mich in Details verloren habe. Danke auch für die unzähligen Stunden, die ihr damit verbracht habt, die Arbeit Korrektur zu lesen.

Ich möchte mich zudem beim Honda Research Institute Europe für die finanzielle Förderung des Projekts bedanken und insbesondere bei Steffen Limmer und Tobias Rodemann für die wertvollen Diskussionen und das Feedback.

Mein Dank gilt auch meinen Eltern, die für meinen Lebensunterhalt gesorgt haben und mir damit das Verfassen dieser Arbeit und den Abschluss meines Studiums ermöglicht haben.

Abschließend möchte ich mich bei der TU Wien und im Speziellen der Forschungsgruppe Algorithms and Complexity für die Infrastruktur bedanken, die ich für die Arbeit verwendet habe. Das betrifft in erster Linie den Cluster, der die nötigen Berechnungen durchgeführt hat.

Abstract

In this work we investigate the Fleet Scheduling Problem, which arises in the context of E-carsharing. Given are a fleet of electric vehicles and a set of reservations. A reservation consists of the timespan in which a vehicle is needed and the amount of energy that is expected to be used. The task is to find a feasible assignment from reservations to vehicles as well as a charging plan for each vehicle that minimize a cost function. The problem can be shown to be NP-hard.

We aim to solve the problem in an exact manner. For that purpose we make use of solvers for Mixed Integer Linear Programs (MILPs). We consider two different solution approaches. First we formulate the problem directly as one MILP, which is strengthened in several ways. In particular we also propose a class of strengthening constraints that prohibit the assignment of subsets of reservations to a vehicle, if that assignment is not possible due to charging constraints. In our second approach we perform a Benders decomposition which decomposes the problem into a master problem (MP) and a subproblem and solves them iteratively in an alternating manner. We perform the decomposition in two different ways. First we do it in a more classical way, then we choose different subsets of variables making the MP stronger and more complex and the subproblem smaller. It turns out that the MP is the bottleneck of the Benders decomposition. We therefore investigate different measures that aim to improve the performance of solving the MP instances. The first measure implements Branch-and-Check which embeds the Benders decomposition in a single Branch-and-Bound tree. That way the MP does not need to be solved from scratch in each iteration. For the second and third measure we speed up the solving process of the MP by solving it in an inexact manner, either allowing a gap for the MILP solver or applying a heuristic. The heuristic is based on General Variable Neighborhood Search and we are able to make use of delta evaluation in order to speed up the search of the neighborhoods. We also propose a network flow formulation that can be used to solve the subproblem. However, we found that the network flow formulation solved with a network simplex algorithm does not perform better than the linear program solved with a general purpose linear program solver. Therefore we do not investigate further here.

As it turns out directly solving the single MILP usually performs better than the Benders decomposition approach for small to medium sized problem instances. For large instances however the MILP solvers are not able to find any reasonable primal solutions while the

Benders decomposition was able to do so and therefore achieved smaller gaps. Especially the variant that uses a heuristic for the MP is able to find good feasible solutions for large instances, altough at the cost of not finding reasonable dual bounds for many of these instances.

Kurzfassung

In dieser Arbeit untersuchen wir das Fleet Scheduling Problem (FSP), das bei E-Carsharing auftritt. Dabei sind eine Flotte von Fahrzeugen und eine Menge von Reservierungen gegeben. Jede Reservierung besteht aus dem Zeitraum in dem ein Fahrzeug gebraucht wird und der Menge an Energie, die voraussichtlich verwendet wird. Aufgabe ist es Reservierungen Fahrzeugen zuzuweisen und für jedes Fahrzeug einen Ladeplan zu erstellen. Dabei soll eine Kostenfunktion minimiert werden. Es kann gezeigt werden, dass das Problem NP-hart ist.

Wir streben eine exakte Lösung des Problems an. Dafür verwenden wir Software zum Lösen von Mixed Integer Linear Programs (MILPs). Die Lösung des Problems wird auf zwei verschiedene Arten angegangen. Zuerst formulieren wir das Problem als MILP und entwerfen eine Klasse von stärkenden Ungleichungen, die die Zuweisung einer Teilmenge von Reservierungen an ein Fahrzeug verhindert, falls das nicht mit dem Ladeplan vereinbar ist. Für den zweiten Ansatz führen wir eine Benders decomposition durch. Dabei wird das MILP in ein Master Problem (MP) und eine Subproblem (SP) unterteilt, welche in mehreren Iterationen abwechselnd gelöst werden. Die Benders decomposition wird auf zwei verschiedene Arten durchgeführt. Einmal auf eine klassischere Art, das andere Mal mit mehr Variablen im MP und weniger Variablen im SP. Wie sich herausstellt wird, verglichen mit dem SP, deutlich mehr Zeit für das Lösen des MPs aufgebracht. Wir untersuchen daher verschiedene Möglichkeiten, den Lösungsprozess für Instanzen des MPs zu beschleunigen. Zuerst implementieren wir Branch-and-Check, das die Benders decomposition in einen Branch-and-Bound Baum einbettet. Dadurch muss das MP nicht in jeder Iteration vollständig gelöst werden. Der zweite und dritte Ansatz versuchen das Lösen des MP zu beschleunigen, indem es nicht bis zur Optimalität gelöst wird. Einerseits wird das erreicht indem bei der Lösungssoftware für MILPs eine Gap erlaubt wird. Andererseits entwerfen wir eine auf General Variable Neighborhood Search basierende Heuristik für das MP, die mittels Delta-Evaluation die Nachbarschaften durchsucht. Wir haben außerdem eine Network Flow Formulierung entworfen, um das SP zu lösen. Wie sich allerdings herausstellt ist der Network Simplex Algorithmus mit der Network Flow Formulierung nicht schneller als eine Lösungssoftware für Linear Programs mit dem entsprechenden Linear Program. Daher untersuchen wir diesen Ansatz nicht im Detail.

Es stellt sich heraus, dass das direkte Lösen des MILPs für kleine und mittelgroße Instanzen besser funktioniert als der Ansatz mit der Benders decomposition. Bei großen

Instanzen kann die Lösungssoftware für MILPs allerdings keine vernünftigen Lösungen finden. Hier kann die Benders decomposition mit guten Lösungen punkten. Insbesondere der Ansatz mit der Heuristik findet gute Lösungen für große Instanzen, allerdings findet diese Variante bei vielen dieser Instanzen keine duale Schranke.

Contents

Abstract **xi**

Kurzfassung **xiii**

Contents **xv**

1 Introduction **1**
1.1 Overview 1
1.2 Outline 3

2 Methodological Approaches **5**
2.1 Mixed Integer Linear Programming 5
2.2 Benders Decomposition 7
2.3 Generalized Variable Neighborhood Search 7

3 Problem Specification **11**

4 Related Work **15**

5 NP-Hardness of the FSP **19**

6 Strengthening Constraints for Conflicting Reserverations **21**

7 Network flow formulation **23**

8 Benders Decomposition **25**
8.1 Basic Benders Decomposition 25
8.2 Extension of the MP 29
8.3 Solving the MP 32

9 Experiments and Results **35**
9.1 Benchmark instances 35
9.2 Implementation and Computing Environment 36
9.3 Comparison of the MILP and the Benders Decomposition 36
9.4 Impacts of Individual Improvements 49

9.5 Impact of Scaling E^{sur} 55
9.6 Impact of Reducing E^{res} 59

10 Conclusion **63**

List of Figures **67**

List of Tables **69**

List of Algorithms **71**

Bibliography **73**

CHAPTER 1

Introduction

1.1 Overview

Driving with a privately owned car has been a major mean of human transportation in western areas over the last decades. Due to the expansion of public transport in urban areas, private cars are getting less attractive for urban residents. Still urban residents occasionally want to use a car to travel outside the urban area. A possibility is the use of carsharing. Due to the demand, carsharing is being implemented in over 1100 cities world-wide [Web19].

At the same time the overall costs per kilometer of electric cars dropped below the costs of cars with combustion engine [Web19]. The reason are the much lower running costs that compensate the currently still higher purchase costs if the car is used extensively. Since carsharing systems reach a high utilization compared to privately used cars, electric cars are well suited for carsharing and lead to lower costs there.

By additionally generating the energy to charge the vehicles on-site it is possible to reduce the costs further. The return of investment for photovoltaic plants can be as low as three years, if providing the energy to the grid [YLTA15]. Using the generated energy to charge the vehicles can reduce the return of investment even further.

However the long charging times pose a challenge for the scheduling of vehicles within a carsharing system that relies on electric vehicles (EVs). On-site energy generation and time-of-use tariffs, where the energy price is time dependent, make the scheduling even harder. Therefore sophisticated algorithms are necessary to schedule charging and usage of a fleet of EVs to maximize cost efficiency and provide the availability of EVs when needed.

We formulate and investigate the Fleet Scheduling Problem (FSP) that arises in the carsharing context with EVs and optional on-site energy generation. For the problem we

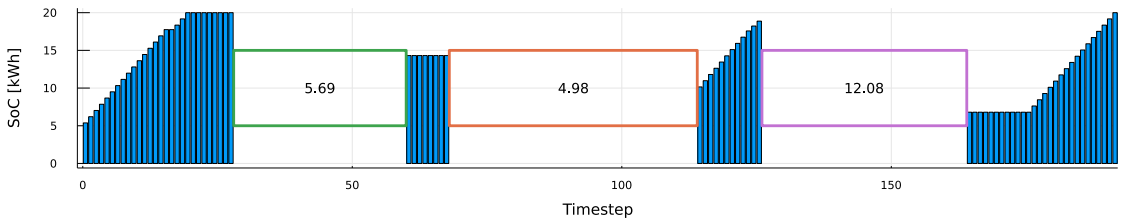


Figure 1.1: Example for the charging plan of a single EV together with the reservations that have been assigned to that vehicle.

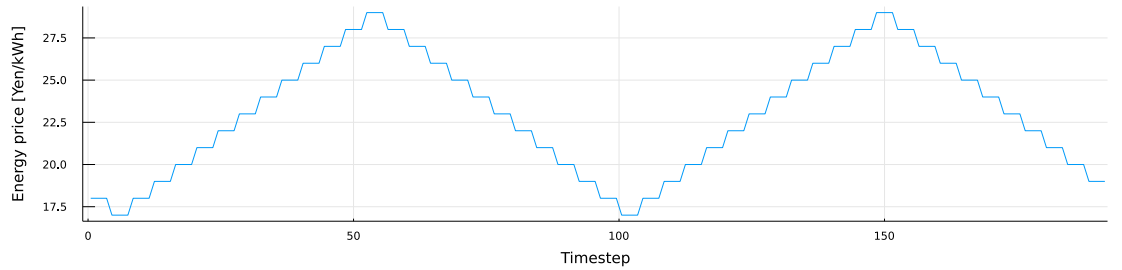


Figure 1.2: Example for the energy price over time.

consider a discrete and finite time horizon. Given is a fleet of identical EVs and a set of reservations, each having a departure and an arrival time and an estimated energy demand. Aim is to assign reservations to vehicles and to create a charging plan for each vehicle, while minimizing a cost function. It is also possible to leave reservations unassigned, however this is associated with a cost, e.g. for filling this demand with a conventional vehicle with combustion engine. Figure 1.1 shows an example of a charging plan for a single EV. It plots the state of charge of the EV over time. Time intervals in which the vehicle is used for a reservation are represented by boxes and the number within the box denotes the energy demand of the reservation. Note that vehicles cannot be charged when used for a reservation.

Charging can be performed with energy from the grid or surplus energy from the on-site energy generation. For grid energy a time-of-use tariff is assumed. Figure 1.2 shows an example of the time dependence of the energy price. Contrary to grid energy, surplus energy is assumed to not cause any costs as it is the case e.g. with a photovoltaic system. However only a time-dependent limited amount of surplus energy is available. Figure 1.3 shows the available surplus energy for an example instance. Additionally the charging energy that is used for an optimized solution is plotted. It can be seen that most of the surplus energy is used for charging, since it is cheaper than grid energy.

For the FSP we assume that the amount of surplus energy and the energy price are known in advance. Energy prices in time-of-use tariffs are often made public beforehand. If using a photovoltaic plant, the surplus energy can be forecasted using the weather prediction. We also assume that all reservations are known in advance as well, i.e., only

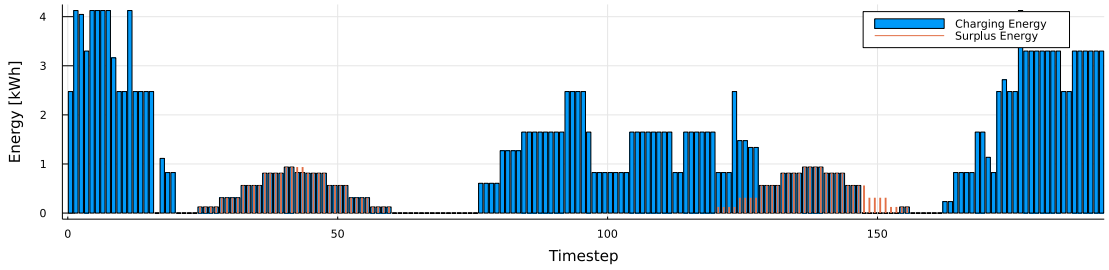


Figure 1.3: Example of the available surplus energy of an instance and the used charging energy for an optimized solution.

the given reservations are considered for the optimization.

We formulate the FSP as Mixed Integer Linear Program (MILP) and derive a Benders decomposition from that formulation. Benders decomposition decomposes a MILP into a master problem (MP) and a subproblem (SP). In a more classical approach the MP contains all integer variables and all other variables are handled in the SP. The Benders decomposition of the FSP is then adapted in different ways to see which measures lead to improvements. These measures include

- adding strenghtening constraints,
- altering the distribution of variables to the MP and the SP,
- embedding the decomposition in the Branch-and-Bound procedure of the MILP solver and
- not solving the MP in an exact manner for the first iterations, either with a heuristic or with the MILP solver and an allowed gap.

Furthermore we investigate how a scaled surplus energy and a reduced energy for reservations influences the practical solvability of the problem.

1.2 Outline

In Chapter 2 we begin with introducing the methodology used in the thesis. In particular the chapter covers the basics of MILPs as well as the solution process of their solvers, the basics of Benders decomposition and a description of the metaheuristic Generalized Variable Neighborhood Search.

Afterwards, Chapter 3 formally defines the FSP and formulates it as MILP. Some techniques are discussed to make the formulation smaller and stronger.

Chapter 4 discusses related work. Besides covering problems that arise in the context of E-carsharing, the chapter summarizes two papers that are directly related.

In Chapter 5 the FSP is proven to be NP-hard. To do so, a reduction from the FSP to the Partition Problem is given. The idea of the reduction is borrowed from the work of Sassi and Oulamara [SO17].

Chapter 6 derives a new class of strengthening constraints for the FSP. Since the number of such strenghtening constraints can be huge, the separation problem for adding the constraints lazily in a cutting plane fashion is given. Unfortunately the separation problem turns out to be NP-hard too. Two inexact approaches to deal with the problem in practice are proposed.

Afterwards, Chapter 7 presents a network flow formulation that can be used to solve to subproblem.

Subsequently, Chapter 8 derives the Benders decomposition of the given MILP in two variants. First it is derived in a more classical way with only the integer variables in the MP. Then some of the variables of the SP are moved to the MP and the derivation is repeated. Possibilities to solve the MP are discussed. These include a heuristic that is based on General Variable Neighborhood Search.

Chapter 9 discusses and compares the experimental results of the different approaches for randomly generated benchmark instances. As it turns out, the MILP formulation with a state-of-the-art solver is superior to the Benders decompositions for small to medium sized instances. For large instances, where the MILP solver is not able to directly find reasonable solutions in a reasonable amount of time anymore, the Benders decomposition, especially the variant with the heuristic, is able to find much better solutions.

Finally, Chapter 10 concludes everything and gives an outlook on possible future work.

CHAPTER 2

Methodological Approaches

This chapter discusses the the methods that have been used to solve the considered problem. These are in particular Mixed Integer Linear Programming, Benders decomposition and the metaheuristic General Variable Neighborhood Search.

2.1 Mixed Integer Linear Programming

Our problem will be formulated as Mixed Integer Linear Program (MILP). MILPs are the instances of a very general NP-complete optimization problem which many problems in NP can be reduced to easily. There are powerful solvers that are able to find optimal solutions to MILPs such as Gurobi[1] [Gur21] and CPLEX[2]. This section introduces MILPs and describes the basics of the solution process of MILP solvers to the extend that is needed to understand this work. For a thorough introduction to Mixed Integer Linear Programming refer to the book of Wolsey [Wol98].

A MILP is formulated as

$$\begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} && (2.1)\\ \text{subject to} \quad & A\boldsymbol{x} + B\boldsymbol{y} \geq \boldsymbol{b} && (2.2)\\ \text{and} \quad & \boldsymbol{x} \in \mathbb{Z}^{n_1} && (2.3)\\ & \boldsymbol{y} \in \mathbb{R}^{n_2} && (2.4) \end{aligned}$$

with $\boldsymbol{c} \in \mathbb{R}^{n_1}$, $\boldsymbol{d} \in \mathbb{R}^{n_2}$, $A \in \mathbb{R}^{n_1 \times m}$, $B \in \mathbb{R}^{n_2 \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$. The expression in Equation 2.1 is the *objective function* and the inequalities in Equation 2.2 are called *constraints*. Aim is to find vectors $\boldsymbol{x} \in \mathbb{Z}^{n_1}$ and $\boldsymbol{y} \in \mathbb{R}^{n_2}$ that minimize the objective function or to determine that no such vectors exist. Note that it is possible to model an equality constraint with two inequality constraints.

[1] https://www.gurobi.com

[2] https://www.ibm.com/products/ilog-cplex-optimization-studio

The *linear program relaxation* of a MILP is obtained by enlarging the domain of $\boldsymbol{x}$ to $\mathbb{R}^{n_1}$ which results in

$$\begin{aligned} \min \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\boldsymbol{y} && (2.5)\\ \text{subject to} \quad & A\boldsymbol{x} + B\boldsymbol{y} \geq \boldsymbol{b} && (2.6)\\ \text{and} \quad & \boldsymbol{x} \in \mathbb{R}^{n_1} && (2.7)\\ & \boldsymbol{y} \in \mathbb{R}^{n_2} && (2.8) \end{aligned}$$

The solution space of the resulting *linear program* (LP) is a superset of the solution space of the MILP. Therefore the optimal solution of the LP is a *lower bound* or *dual bound* to the optimal solution of the MILP. Linear programs can be solved in polynomial time by e.g. interior point methods [Gon12] and there are algorithms like the simplex algorithm that work well in practice.

By adding constraints that remove solutions of the LP but not of the MILP, the bound can be improved. Such constraints are called *strengthening contraints* or *valid inequalities* and there are two kinds of them. On the one side state-of-the-art MILP solvers derive several classes of more general valid inequalities from other constraints. On the other side the special structure of a problem that is formulated as a MILP may allow problem-specific valid inequalities.

Each LP has a *dual LP*. For the above LP it is formulated as

$$\begin{aligned} \max \quad & \boldsymbol{b}^T\boldsymbol{z} && (2.9)\\ \text{subject to} \quad & A^T\boldsymbol{z} = \boldsymbol{c} && (2.10)\\ & B^T\boldsymbol{z} = \boldsymbol{d} && (2.11)\\ \text{and} \quad & \boldsymbol{z} \in \mathbb{R}^{n_1}, \boldsymbol{z} \geq 0 && (2.12) \end{aligned}$$

Note that constraints become variables of the dual LP and variables become constraints of the dual LP. By the strong duality theorem the optimal objective value of the dual LP is the same as the optimal objective value of the primal LP. That implies that the objective value of each solution to the dual LP is lower than the objective value of each solution to the primal LP.

Branch-and-Bound is a fundamental algorithm following the concept of divide and conquer that is used by most MILP solvers as underlying framework to find the optimal solution to a MILP. It proceeds by repeatedly dividing the solution space into smaller subspaces. Using the *incumbent solution*, i.e. the currently best one, and dual bounds the algorithm can remove some of the subspaces without losing optimality. In particular if the dual bound of a subspace is worse than the objective value of the incumbent solution, the subspace cannot contain the optimal solution.

To improve the incumbent solution heuristics are invoked repeatedly. MILP solvers often provide callback functions that are called on finding solutions by the heuristic. Such

callback functions can add constraints. These so called *lazy constraints* or *cuts* are often used, if the number of a particular type of constraint for a problem is too huge. The callback function then determines for a solution, if it violates a constraint that has not been added so far. If so it adds at least one constraint that is violated to the MILP to cut off the respective part of the search space. In general not all constraints have to be added to arrive at the optimal solution and therefore using lazy constraints for a huge number of constraints often leads to a speedup.

2.2 Benders Decomposition

Benders decomposition is a general approach to tackle MILPs of a certain structure and was proposed almost six decades ago by Benders [Ben62]. It works by splitting up the problem into a master problem (MP) and a subproblem (SP). For the classical version, the MP contains the integer variables and all constraints that only contain these integer variables. The SP contains the remaining variables and constraints and is formulated for a fixed solution of the MP. First the MP is solved and the values of the solution are fixed, which leads to the SP. Extreme rays and points of the dual SP are then used to add feasibility respectively optimality cuts to the MP. That way MP and SP are solved in an alternating fashion, both forwarding information to the other problem. An optimal solution to the MP that yields a feasible SP provides an upper and a lower bound, which can be used to determine, whether the found solution is also optimal for the overall problem. A disadvantage of this procedure is that the MP has to be solved from scratch in each iteration. To overcome this one can embed the decomposition in a single branch-and-bound tree. Feasibility and optimality cuts are then added as lazy constraints in the callback function. Branch-and-bound with Benders decomposition embedded is called *Branch-and-check*. In their literature review, Rahmaniani et al. [RCGR17] discuss the Benders decomposition and variants of it in detail and investigate different speedup strategies and extensions. Benders decomposition is useful, if a MILP formulation has almost a diagonal block structure. If done right, the SP can then be decomposed into multiple smaller problems that can be solved independently. That normally leads to a speedup. Problems of such a structure occur e.g. in Stochastic Programming [VSW69].

2.3 Generalized Variable Neighborhood Search

Generalized Variable Neighborhood Search (GVNS) is a metaheuristic based on local search. Local search starts with a solution and repeatedly improves the solution by searching its neighborhood until there is no better solution in the neighborhood. If there is no better solution in the neighborhood of a solution, it is also called a *local optimum*. A *neighborhood structure* is a function that maps from solutions to sets of solutions. The image of a solution for a given neighborhood structure is the *neighborhood* of that solution. Neighborhood structures are often obtained via moves. A *move* applies a normally small change to a solution. A neighborhood of a solution is then defined to be the set of all solutions that can be obtained by applying a move of special structure to that solution.

Variable Neighborhood Descend (VND) generalizes local search to multiple neighborhoods $N_1, \ldots, N_k$ and finds a solution that is a local optimum for all neighborhoods. To achieve that, all neighborhoods are searched consecutively, restarting at the first neighborhood upon finding a better solution. The pseudo code is given in Algorithm 2.1.

Algorithm 2.1: VND with the neighborhoods $N_1, \ldots, N_k$

```
1  Function VND_{N_1,...,N_k}(Initial solution s_0)
2      s ← s_0 ;
3      l ← 1 ;
4      repeat
5          s' ← minarg_{n∈N(s)} obj(n) ;
6          if s' better than s then
7              s ← s' ;
8              l ← 1 ;
9          else
10             l ← l + 1 ;
11         end
12     until l > k;
13     return s ;
14 end
```

Local search and VND stop at the local optimum that is reached first. This local optimum can have a bad objective value. To be able to escape local optima, General Variable Neighborhood Search (GVNS) extends VND with shaking. *Shaking* chooses a random neighbor from a given neighborhood, no matter if the neighbor has a better or worse objective value. The pseudo code of GVNS is given in Algorithm 2.2.

Algorithm 2.2: GVNS with the neighborhoods $N_1, \ldots, N_k$ for VND and $\tilde{N}_1, \ldots, \tilde{N}_m$ for shaking

```
1  Function GVNS_{N_1,...,N_k;Ñ_1,...,Ñ_m} (Initial solution s_0)
2      s ← s_0 ;
3      repeat
4          l ← 1 ;
5          repeat
6              s' ← random element in Ñ_l(s) ;
7              s'' ← VND_{N_1,...,N_k}(s') ;
8              if s'' better than s then
9                  s ← s'' ;
10                 l ← 1 ;
11             else
12                 l ← l + 1 ;
13             end
14         until l > m;
15     until stopping criteria;
16     return s ;
17 end
```

CHAPTER 3

Problem Specification

In the Fleet Scheduling Problem (FSP) we consider in this work, we are given

- a discretized planning time horizon $T = \{1, \ldots, t_{\max}\}$ with each time step having length Δt,
- a set $V = \{1, \ldots, n\}$ of n uniform EVs with
  - maximal possible charging power $P^{\max} > 0$ and
  - battery capacity $E^{\text{cap}} > 0$,

  and for each vehicle $v \in V$
  - the subset of time steps $T_v^{\text{avail}} \subseteq T$ at which the vehicle is available and
  - the initial energy level $E_{v,0} \geq 0$,
- a set $R = \{1, \ldots, r_{\max}\}$ of reservations of vehicles, for each $r \in R$
  - the continuous set of time steps $T_r^{\text{res}} = \{t_r^{\text{start}}, \ldots, t_r^{\text{end}}\} \subseteq T$ in which a vehicle is needed and
  - the energy $E_r^{\text{res}} \geq 0$ that will be consumed during the reservation,
- the grid's costs for electricity c_t per unit of energy during each time step $t \in T$; note that these can also be negative to encourage the charging of vehicles beyond the needs for the reservations,
- an assumed maximum surplus energy $E_t^{\text{surmax}} \geq 0$ available during each time step $t \in T$, for example from a photovoltaic system, and
- costs $c^{\text{uncov}} > 0$ for covering each unit of energy required by unassigned reservations, for example by cars with combustion engines.

A solution comprises

- a partial assignment of reservations to vehicles, modeled by variables $x_{r,v} \in \{0,1\}$ for $r \in R,\ v \in V$, indicating with value one that reservation r is to be fulfilled by vehicle v, and
- a charging plan, modeled by variables $p_{v,t} \geq 0$ indicating the power by which vehicle $v \in V$ is charged during time step $t \in T$.

The objective is to minimize the costs spent for grid energy and uncovered reservations and to maximize the used surplus energy. These two objectives are combined in a linear fashion in which the used surplus energy is weighted by a factor α.

The FSP can be expressed by the following mixed integer linear programming model.

$$\min \quad \sum_{t \in T} c_t E_t^{\text{grid}} + c^{\text{uncov}} \sum_{r \in R} E_r^{\text{res}} y_r - \alpha \sum_{t \in T} E_t^{\text{sur}} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{v \in V} x_{r,v} + y_r = 1 \qquad r \in R \tag{3.2}$$

$$x_{r,v} = 0 \qquad v \in V,\ r \in R \mid T_r^{\text{res}} \not\subseteq T_v^{\text{avail}} \tag{3.3}$$

$$p_{v,t} = 0 \qquad v \in V,\ t \in T \setminus T_v^{\text{avail}} \tag{3.4}$$

$$p_{v,t} \leq P^{\max} \cdot (1 - \sum_{r \in R | t \in T_r^{\text{res}}} x_{r,v}) \qquad v \in V,\ t \in T \tag{3.5}$$

$$E_{v,0} + \Delta t \sum_{k=1}^{t} p_{v,k} - \sum_{r \in R | t_r^{\text{end}} \leq t-1} E_r^{\text{res}} x_{r,v} \leq E^{\text{cap}} \qquad v \in V,\ t \in T^{\text{ends}} \cup \{t_{\max}\} \tag{3.6}$$

$$E_{v,0} + \Delta t \sum_{k=1}^{t-1} p_{v,k} - \sum_{r \in R | t_r^{\text{start}} \leq t} E_r^{\text{res}} x_{r,v} \geq 0 \qquad v \in V,\ t \in T^{\text{starts}} \tag{3.7}$$

$$E_t^{\text{grid}} + E_t^{\text{sur}} = \sum_{v \in V} \Delta t\, p_{v,t} \qquad t \in T \tag{3.8}$$

$$E_t^{\text{grid}} \geq 0 \qquad t \in T \tag{3.9}$$

$$0 \leq E_t^{\text{sur}} \leq E_t^{\text{surmax}} \qquad t \in T \tag{3.10}$$

$$x_{r,v} \in \{0,1\} \qquad v \in V,\ r \in R \tag{3.11}$$

$$0 \leq y_r \leq 1 \qquad r \in R \tag{3.12}$$

$$0 \leq p_{v,t} \leq P^{\max} \qquad v \in V,\ t \in T \tag{3.13}$$

Additionally used variables are:

- $y_r,\ r \in R$: one if reservation r is uncovered and zero else; note that these variables can be kept continuous in the model as they will automatically get an integral value due to equalities (3.2).

- $E_t^{\text{grid}},\ t \in T$: total amount of energy consumed from the grid during time step t.
- $E_t^{\text{sur}},\ t \in T$: total amount of surplus energy consumed during time step t.

By T^{starts} and T^{ends} we denote the sets of time steps in which reservations start and end, respectively, i.e.,

$$T^{\text{starts}} := \left\{ t_r^{\text{start}} \mid r \in R \right\} \quad \text{and} \quad T^{\text{ends}} := \left\{ t_r^{\text{end}} \mid r \in R \right\}. \tag{3.14}$$

The meaning of the constraints is as follows.

- Equation 3.2 prohibits the assignment of multiple EVs to a reservation r. Furthermore it forces y_r to one if no EV is assigned to reservation r.
- Equation 3.3 prevents assigning EVs when they are not available.
- Equation 3.4 prevents charging EVs when they are not available.
- Equation 3.5 prevents charging EVs when they are used in a reservation. Furthermore it prevents EVs to be assigned to multiple reservations at the same time. Note that splitting up that constraint into two would lead to a weaker formulation.
- Equation 3.6 and Equation 3.7 prevent the batteries of vehicles from getting overcharged or exhausted. The following details have been applied to strengthen the formulation and to reduce the number of inequalities:
  - If for some time step the SoC does not exceed the battery capacity and no reservation starts at that time step, then the SoC does not exceed the battery capacity for the previous time step. Therefore it is sufficient to limit the SoC only in time steps before a reservation and at the last time step.
  - The SoC can only drop below zero when the energy of a reservation is subtracted. Therefore, Equation 3.7 is only needed for such time steps.
  - It is irrelevant for the correctness of the formulation whether the energy of a reservation is subtracted at the start or at the end of the reservation interval or somewhere in between. In order to make the inequalities stronger, we use the end for Equation 3.6 and the start for Equation 3.7.
  - For Equation 3.6, $p_{v,t}$ can be added, making the constraint stronger.
  - Analogously $p_{v,t}$ can be omitted for Equation 3.7.
- Equation 3.8 links the power by which EVs are charged at each time step with the variables for the energy used from the grid and the used surplus energy.
- Equation 3.9 to Equation 3.13 define the domains of all variables.

CHAPTER 4

Related Work

Brandstätter et al. classify problems that arise in the context of E-carsharing systems into two categories [BGL+16]. The first category includes problems that decide on the structure of an E-carsharing system. In particular determining the locations of stations for station-based systems and distributing vehicles to existing stations fall into this category. The second category involves problems that occur on a regular basis when operating an E-carsharing system. This includes the relocation of vehicles where vehicles are moved between stations to satisfy user demands. It also includes Electric Vehicle Routing Problems.

Vehicle Routing Problems aim to assign vehicles to routes that serve geographically distributed customers while minimizing costs [KP12]. Normally besides feasibility constraints like a vehicle load capacity or customer time windows are imposed. For Elecric Vehicle Routing Problems, constraints for the limited range and charging are needed additionally [EC19]. These problems have been studied extensively in the literature and many algorithms for solving them have been proposed. Erdelić and Carić [EC19] discuss different variants of Electric Vehicle Routing Problems and summarize the proposed algorithms.

A very similar class of problems addresses the scheduling of vehicles [BK09]. Here a set of trips, each having a location and time for departure and arrival, is given. Aim is to find a feasible assignment from trips to vehicles that minimizes costs. Vehicles start and end at depots. Problem variants either consider a single depot or multiple depots. Recent advances also consider electric or mixed fleets [MO19, RPDV20].

A different kind of problem regarding EVs concentrates on the charging process. Regarding smart grids these problems can be classified into three categories [WLDK16]. First smart grid oriented problems address frequency regulation, voltage regulation or load flattening, i.e. avoiding too high loads. While for smart grid oriented problems there is a central controller, at aggregator oriented problems the smart grid instead has multiple aggregators,

each controlling its own set of EVs. Customer oriented problems are from the point of view of a customer. Here a customer possessing a single EV normally wants to charge the EV while minimizing costs. Often the customer has to consider uncertainties like the uncertainty of the electricity price.

Like Vehicle Routing Problems and Vehicle Scheduling Problems the FSP assigns reservations to vehicles. However the FSP does not consider different locations. For all reservations the EV is picked up and returned at the location, where the charging of the EVs is performed. Regarding smart grids the FSP can be seen as aggregator oriented problem, since multiple EVs are operated by a single entity, which is connected to the grid. Altough the FSP has overlaps with many problems that fall into above mentioned classes, there are only two papers that are strongly related. One of them is by Sassi and Oulamara [SO17] and the other one by Betz et al. [BWL16]. The following paragraphs give an overview on these two papers.

Sassi and Oulamara [SO17]. The main difference of the problem Sassi and Oulamara are considering is the optimization objective. First, they maximize the distance that is travelled with EVs. Then the charging costs are minimized, s.t. the distance stays constant on its optimal value. Another difference is that the number of alternative vehicles that can be used for uncovered reservations, e.g., petrol cars, is limited. Moreover, there are also some differences regarding the charging process: Surplus energy is not considered, i.e., all the energy is taken from the grid. However the grid is limited in the total power it can deliver, so the charging processes of different EVs must also be optimized together. Additionally to this maximum power of the grid and the maximum charging power for each EV, there also exists a lower limit for the charging power. Thus, an EV is either charged within its lower and upper power limits or not charged. Sassi and Oulamara propose two heuristics to solve their problem, which they name Sequential Heuristic (SH) and Global Heuristic (GH). The SH repeats the following steps for each EV:

- Select a subset of reservations that are non-conflicting s.t. the driven distance is as large as possible and assign them to the EV.
- Find a charging schedule for the EV that minimizes charging costs.

Afterwards the uncovered reservations are assigned to the petrol cars. The GH works basically the same but calculates a charging schedule for all vehicles at once after all reservations have been assigned to vehicles. These two heuristics were compared to a MILP formulation on random instances. Both, GH and SH, outperformed the MILP formulation on these instances in terms of the quality of heuristic solutions. Compared to SH, GH in general found feasible solutions on more instances and better solutions on the instances where both heuristics found solutions. However, SH was significantly faster.

Betz et al. [BWL16] differ in their problem variant from our FSP by limiting the number of available chargers. These chargers can have different charging powers. Betz et

al. also formulate their problem as a MILP and solved it using the MATLAB integrated MILP solver. The hardest instance that could be solved within two hours included 30 trips and eight vehicles. The authors also investigated the effect on operational costs and environmental impact when using a fleet of EVs compared to a fleet of vehicles with combustion engines.

CHAPTER 5

NP-Hardness of the FSP

Sassi and Oulamara [SO17] tried to prove the NP-hardness of their fleet scheduling problem variant called Electric Vehicle Scheduling and Charging Problem (EVSCP). The FSP is similar to the EVSCP and indeed the idea of the proof can be adopted for the FSP. However, we remark that the original proof in [SO17] contains a severe error because the reduction from the used partition problem may result in an exponentially large EVSCP instance as the size of the time horizon T depends on an input value of the partition problem. In the following we present an adapted and corrected proof for our FSP.

We reduce the partition problem, which is known to be NP-complete [Kar72], to the decision variant of the FSP. The decision variant of the FSP asks whether or not there is a solution that has an objective value less than or equal to a given $c_{\max}$. The partition problem can be stated as follows.

Partition Problem
Input: A multiset $P = \{a_1, \ldots, a_m\}$ of positive integers that sum up to $2s$.
Question: Is there a partition $\{M_1, M_2\}$ of $\{1, \ldots, m\}$, s.t. $\sum_{i \in M_1} a_i = s = \sum_{i \in M_2} a_i$?

Theorem 1. *The FSP is NP-hard.*

Proof. Given an instance of the partition problem, an instance of the decision variant of the FSP is constructed as follows.

- For each integer in P there is a time step: $T = \{1, \ldots, m\}$
- There are two EVs: $V = \{1, 2\}$
- Both EVs have

  - a battery capacity and an initial energy level of s: $E^{\text{cap}} = s, E_{1,0} = E_{2,0} = s$
  - no charging capabilities: $P^{\max} = 0$
  - an unlimited availability: $T_1^{\text{avail}} = T_2^{\text{avail}} = T$
- For each integer in P there is a reservation: $R = \{1, \ldots, m\}$
  - All reservations are non-overlapping: $T_r^{\text{res}} = \{r\}$, $r \in R$
  - The energy consumption of a reservation is: $E_r^{\text{res}} = a_r$
- There are only costs for uncovered reservations: $c^{\text{uncov}} = 1$, $c_t = 0$, $t \in T$, $\alpha = 0$
- The objective value has to be less than or equal to zero: $c_{\max} \leq 0$
- No surplus energy is available: $E_t^{\text{surmax}} = 0$, $t \in T$

Assume we have a solution for this problem instance. This solution cannot have uncovered reservations as they would increase the objective value to a positive value. Therefore, the two EVs have to cover all reservations. The total energy consumption of all reservations is $2s$. Both vehicles have an initial energy level of s and no recharging possibilities. Therefore the energy level of both vehicles is zero after time step m, and the energy consumptions of the reservations served by a single vehicle sum up to s. This gives a solution for the corresponding partition problem:

$$M_v := \{i \mid x_{v,i} = 1,\ i = 1,\ \ldots, m\} \qquad v \in V \tag{5.1}$$

Given a solution to the partition problem, a corresponding solution to the FSP is constructed by assigning each vehicle $v \in V$ to the reservations corresponding to M_v. Due to a similar reasoning as above, the solution to the FSP has to be feasible.

Overall, there is a solution to the partition problem, iff there is a solution to the corresponding FSP. As all the described transformations can be done in polynomial time, it follows that the FSP is NP-hard. □

CHAPTER 6

Strengthening Constraints for Conflicting Reserverations

Let $R' \subseteq R$ be a set of non-overlapping reservations to which vehicle $v \in V$ is assigned to. The maximal achievable energy level $E^{\max}_{v,t}(R')$ of vehicle v for $t \in T \cup \{0\}$ in dependence of R' can be calculated by

$$E^{\max}_{v,t}(R') = \begin{cases} E_{v,0} & \text{if } t = 0 \\ \min(E^{\max}_{v,t-1}(R') + P^{\max}\Delta t, E^{\text{cap}}) & \text{if } t \in T^{\text{avail}}_v \setminus \bigcup_{r \in R'} T^{\text{res}}_r \\ E^{\max}_{v,t-1}(R') - \sum_{r \in R' | t = t^{\text{start}}_r} E^{\text{res}}_r & \text{otherwise.} \end{cases} \tag{6.1}$$

If $E^{\max}_{v,t}(R')$ would be negative at a time step, not all reservations of R' can be assigned together to vehicle v. In other words at least one reservation in R' may not be assigned to vehicle v in this case, which yields the inequality

$$\sum_{r \in R'} x_{r,v} \leq |R'| - 1. \tag{6.2}$$

This kind of constraint in general strengthens the MILP formulation, i.e., there are solutions to the LP relaxation of the model that do not fulfill Equation 6.2. Take for example the FSP instance with one vehicle of capacity and initial SoC 3 and two reservations 1 and 2 having a reservation energy of 2 and reservation intervals $T^{\text{res}}_1 = \{1\}$ and $T^{\text{res}}_2 = \{2\}$. Then the LP relaxation of the MILP formulation in Chapter 3 has a solution with $x_{1,1} = x_{2,1} = 0.75$, which does not fulfill Equation 6.2 for $R' = \{1, 2\}$. Still $E^{\max}_{1,2}(\{1, 2\}) = -1$ is less than zero and therefore the constraint can be added. A problem is that the number of these inequalities can get exponentially large in $|R|$. Therefore we consider the dynamic separation of such inequalities in a branch-and-cut approach. The respective separation problem can be stated for a specific vehicle $v \in V$ as follows.

Separation Problem

Input: For each $r \in R$: $x_{r,v} \in [0,1]$, $E_r^{\text{res}} \geq 0$, $T_r^{\text{res}} \subseteq T$. Furthermore $P^{\max}$, Δt, E^{cap} and for each $v \in V$: $E_{v,0}$ and T_v^{avail}.

Question: Does there exist a non-overlapping set of reservations $R' \subseteq R$ with $T_r^{\text{res}} \subseteq T_v^{\text{avail}}$ s.t. $E_{v,t}^{\max}(R') < 0$ for some $t \in T \setminus \{1\}$ and $\sum_{r \in R'} x_{r,v} > |R'| - 1$?

For the special case of $P^{\max} = 0$, E^{cap} sufficiently large, and unlimited availabilities ($T_v^{\text{avail}} = T$), the maximal achievable energy can be calculated in an easy way by

$$E_{v,t}^{\max}(R') = E_{v,0} - \sum_{r \in R' | t_r^{\text{start}} \leq t} E_r^{\text{res}}. \tag{6.3}$$

The condition $E_{v,t}^{\max}(R') < 0$ then becomes $\sum_{r \in R'} E_r^{\text{res}} > E_{v,0}$ as it is sufficient to check for $t = t_{\max}$. If no reservations in R overlap with each other, the separation problem corresponds to (the decision version of) the 0–1 knapsack problem with item weights $1 - x_{r,v}$, capacity one, and item values E_r^{res}. Therefore the separation problem is NP-hard.

One way to deal with the separation problem in practice is to discretize all values $x_{r,v}$ and to apply dynamic programming over $\sum_{r \in R'}(1 - x_{r,v})$ with the reservations ordered by their start times, maximizing $E_{v,R'}^{\max}(t)$ for each intermediate result. Alternatively, one could discretize the energy and minimize $\sum_{r \in R'}(1 - x_{r,v})$.

In practice it seems reasonable to apply a fast greedy heuristic to possibly identify a violated inequality and the more complex dynamic programming approach only when the heuristic does not succeed. We implemented such a heuristic and tested it with the MILP formulation solved by Gurobi. However that did not lead to a speedup and therefore we did not implement the more elaborate dynamic programming algorithm.

One can also initially check all pairs of non-overlapping reservations and provide respective valid inequalities already from the beginning instead of later separating all of them dynamically. We tried that approach for the Benders decomposition, but again it did not lead to a speedup. More on that result in Chapter 9.

CHAPTER 7

Network flow formulation

As already pointed out in [SO17] certain variants of the charging scheduling problem can be modeled as minimum cost flow problems. Minimum cost flows in capacitated networks can be efficiently determined by dedicated algorithms like, for example, network simplex, which can be more efficient than a more general linear programming solver.

For now, let us assume we have known values for the assignment variables x, like for example for the SP of a Benders decomposition of the problem. We are then able to calculate an optimal charging plan p by determining a minimum cost flow in the network depicted in Figure 7.1. Edge capacities are given in green, edge costs in red. The flow flowing from s to t has to be sufficiently large, e.g. $\sum_{r \in R} E_r^{\text{res}} + nE^{\text{cap}}$. Nodes are arranged in a grid. For each vehicle the grid has a row and for each time step it has a column. Given a vehicle v, a time step t and the corresponding node, the flow entering that node from the left represents the state of charge of vehicle v at time step $t-1$. The state of charge of a vehicle may never exceed E^{cap}, therefore E^{cap} is used as capacity for these edges. M is a constant that is sufficiently large to force a flow of $E_{v,0}$ into the node of vehicle v and time step 1. Charging happens by adding flow from the nodes above, which can take E_t^{surmax} flow from the source node at cost $-\alpha$ and arbitrary much flow at cost c_t. Available vehicles can be charged with $E^{\max} := \Delta t P^{\max}$ flow, non-available vehicles cannot be charged. For each reservation the reservation energy has to be taken away. That is achieved with arcs of cost $-M$ and capacity E_r^{res} that go dirctly to the target node.

The network can be used to determine the optimal solution to the SP of the more classical Benders decomposition of the FSP, which will be explained in Section 8.1. An efficient algorithm that solves the Minimum Cost Flow Problem is network simplex. However we found that the formulation with the network simplex algorithm does not perform better than Gurobi 9.1[1] [Gur21] with the linear program. Therefore we will use an LP solver

[1] https://www.gurobi.com

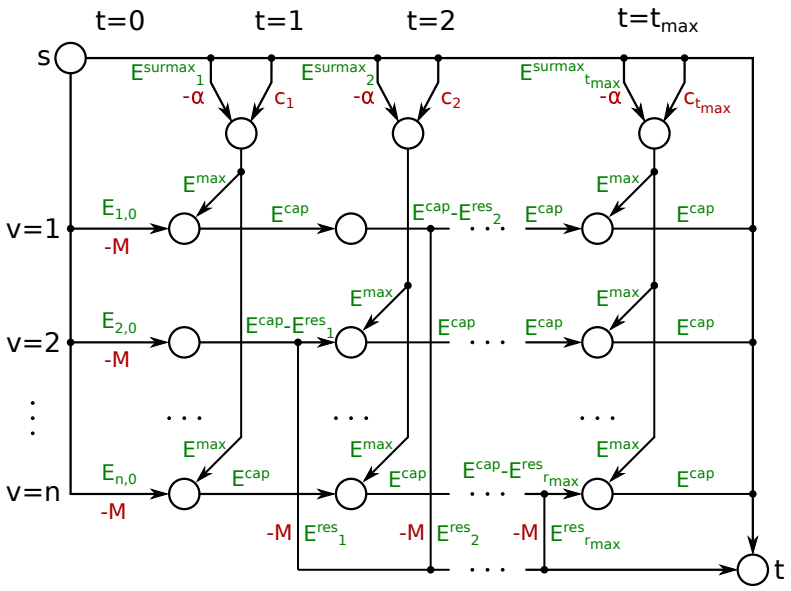


Figure 7.1: Minimum cost flow network for determining a charging plan for given assignments x. Edge capacities are given in green, edge costs in red. No given edge capacity means a capacity of ∞, no given edge cost means a cost of 0. $E^{\max} := \Delta t P^{\max}$.

for solving the SP.

CHAPTER 8

Benders Decomposition

8.1 Basic Benders Decomposition

We now apply the Benders decomposition to our MILP from the previous section, coming up with the respective master and subproblems as well as the dual problem of the latter, which is needed for deriving the Benders cuts.

8.1.1 Master- and Subproblems

The following master problem (MP) only considers the assignment variables $x_{r,v},\ r \in R,\ v \in V$ and $y_r, r \in R$ and abstracts from the actual charging details. The latter will only be considered in the subproblem.

$$\min \quad c^{\text{uncov}} \sum_{r \in R} E_r^{\text{res}} y_r + \mu \tag{8.1}$$

$$\text{s.t.} \quad \sum_{v \in V} x_{r,v} + y_r = 1 \qquad r \in R \tag{8.2}$$

$$x_{r,v} = 0 \qquad v \in V,\ r \in R \mid T_r^{\text{res}} \nsubseteq T_v^{\text{avail}} \tag{8.3}$$

$$\sum_{r \in R \mid t \in T_r^{\text{res}}} x_{r,v} \leq 1 \qquad v \in V,\ t \in T \tag{8.4}$$

$$\text{Feasibility cuts} \tag{8.5}$$

$$\text{Optimality cuts} \tag{8.6}$$

$$x_{r,v} \in \{0, 1\} \tag{8.7}$$

$$0 \leq y_r \leq 1 \tag{8.8}$$

$$\mu \geq \sum_{t \in T \mid c_t < 0} c_t \, n \, \Delta t \, P^{\max} - \alpha \sum_{t \in T} E_t^{\text{surmax}} \tag{8.9}$$

Variable μ represents additional costs that are to be contributed by the optimality cuts.

Such cuts will be iteratively derived and added by solving the following subproblem $\mathrm{SP}(\bar{x})$ for specific reservation assignments $\bar{x} = (\bar{x}_{r,v})_{r\in R, v\in V}$ in a current optimal solution to the master problem.

We define

- for each vehicle $v \in V$ the set of time steps when the vehicle is available and not reserved
$$T_v^{\text{home}} := T_v^{\text{avail}} \setminus \bigcup_{r\in R|\bar{x}_{r,v}=1} T_r^{\text{res}} \tag{8.10}$$
- and for each vehicle $v \in V$ the energy difference up to time step $t \in T$ caused by assigned reservations. The energy of a reservation may be subtracted at the start or the end of the reservation interval, leading to the two definitions
$$E_{v,t}^{\text{res,start}} := \sum_{r\in R|t\le t_r^{\text{start}}\wedge\bar{x}_{r,v}=1} E_r^{\text{res}} \quad \text{and} \quad E_{v,t}^{\text{res,end}} := \sum_{r\in R|t\le t_r^{\text{end}}\wedge\bar{x}_{r,v}=1} E_r^{\text{res}}. \tag{8.11}$$

The primal subproblem $\mathrm{SP}(\bar{x})$ can now be formulated as follows.

$$(\mathrm{SP}(\bar{x}))\ \min \quad \sum_{t\in T} c_t E_t^{\text{grid}} - \alpha \sum_{t\in T} E_t^{\text{sur}} \tag{8.12}$$

$$\text{s.t.} \quad p_{v,t} = 0 \qquad (\lambda_{v,t}^{\text{p=0}}) \qquad v \in V,\ t \in T \setminus T_v^{\text{home}} \tag{8.13}$$

$$E_{v,0} + \Delta t \sum_{k=1}^{t} p_{v,k} - E_{v,t-1}^{\text{res,end}} \le E^{\text{cap}} \quad (\lambda_{v,t}^{\text{ub}})\ v \in V, t \in T^{\text{ends}} \cup \{t_{\max}\} \tag{8.14}$$

$$E_{v,0} + \Delta t \sum_{k=1}^{t-1} p_{v,k} - E_{v,t}^{\text{res,start}} \ge 0 \qquad (\lambda_{v,t}^{\text{lb}}) \qquad v \in V, t \in T^{\text{starts}} \tag{8.15}$$

$$E_t^{\text{grid}} + E_t^{\text{sur}} = \sum_{v\in V} \Delta t\, p_{v,t} \qquad (\lambda_t^{\text{E}}) \qquad t \in T \tag{8.16}$$

$$E_t^{\text{grid}} \ge 0 \qquad t \in T \tag{8.17}$$

$$0 \le E_t^{\text{sur}} \le E_t^{\text{surmax}} \qquad (\lambda_t^{\text{sur}}) \qquad t \in T \tag{8.18}$$

$$0 \le p_{v,t} \le P^{\max} \qquad (\lambda_{v,t}^{\text{p}}) \qquad v \in V,\ t \in T \tag{8.19}$$

To get feasibility and optimality cuts, we have to derive the dual of this subproblem. In the above primal problem, the λ-variables in parentheses denote the dual variables that

correspond to each constraint. The dual subproblem is then:

$$\begin{aligned}
\max \quad & \sum_{v \in V} \sum_{t \in T^{\text{ends}} \cup \{t_{\max}\}} \left(E^{\text{cap}} + E^{\text{res,end}}_{v,t-1} - E_{v,0} \right) \lambda^{\text{ub}}_{v,t} + \\
& \sum_{v \in V} \sum_{t \in T^{\text{starts}}} \left(E^{\text{res,start}}_{v,t} - E_{v,0} \right) \lambda^{\text{lb}}_{v,t} + \\
& \sum_{t \in T} E^{\text{surmax}}_{t} \lambda^{\text{sur}}_{t} + \\
& \sum_{v \in V} \sum_{t \in T} P^{\max} \lambda^{\text{p}}_{v,t} && (8.20)
\end{aligned}$$

$$\begin{aligned}
\text{s.t.} \quad & \lambda^{\text{E}}_{t} \leq c_t && (E^{\text{grid}}_{t}) && t \in T && (8.21) \\
& \lambda^{\text{E}}_{t} + \lambda^{\text{sur}}_{t} \leq -\alpha && (E^{\text{sur}}_{t}) && t \in T && (8.22) \\
& \lambda^{\text{p}}_{v,t} - \Delta t \lambda^{\text{E}}_{t} + \Delta t \sum_{k \in T^{\text{ends}} \cup \{t_{\max}\} | k \geq t} \lambda^{\text{ub}}_{v,k} + \Delta t \sum_{k \in T^{\text{starts}} | k \geq t+1} \lambda^{\text{lb}}_{v,k} \leq 0 \\
& && (p_{v,t}) && v \in V,\ t \in T^{\text{home}}_{v} && (8.23) \\
& \lambda^{\text{p=0}}_{v,t} + \lambda^{\text{p}}_{v,t} - \Delta t \lambda^{\text{E}}_{t} + \Delta t \sum_{k \in T^{\text{ends}} \cup \{t_{\max}\} | k \geq t} \lambda^{\text{ub}}_{v,k} + \Delta t \sum_{k \in T^{\text{starts}} | k \geq t+1} \lambda^{\text{lb}}_{v,k} \leq 0 \\
& && (p_{v,t}) && v \in V,\ t \in T \setminus T^{\text{home}}_{v} && (8.24) \\
& \lambda^{\text{p=0}}_{v,t} \in \mathbb{R} && && v \in V,\ t \in T && (8.25) \\
& \lambda^{\text{E}}_{t} \in \mathbb{R} && && t \in T && (8.26) \\
& \lambda^{\text{sur}}_{t} \leq 0 && && t \in T && (8.27) \\
& \lambda^{\text{p}}_{v,t} \leq 0 && && v \in V,\ t \in T && (8.28) \\
& \lambda^{\text{ub}}_{v,t} \leq 0 && && v \in V,\ t \in T^{\text{ends}} \cup \{t_{\max}\} && (8.29) \\
& \lambda^{\text{lb}}_{v,t} \geq 0 && && v \in V,\ t \in T^{\text{starts}} && (8.30)
\end{aligned}$$

This dual subproblem can be simplified in multiple regards as follows.

- Variable $\lambda^{\text{p=0}}_{v,t}$ appears only once in Equation 8.24 and is unbounded. Therefore, it can always be chosen in a way that the inequality is fulfilled. Thus, $\lambda^{\text{p=0}}_{v,t}$ as well as Equation 8.24 can be removed.

- Variables λ^{sur}_{t} and $\lambda^{\text{p}}_{v,t}$ appear in the objective function with positive coefficients and in two constraints that only give upper limits. The value of such a variable will always be the smaller upper limit in an optimal solution.

This leads to the following equivalent formulation.

$$
\begin{aligned}
(\mathrm{DSP}(\bar{x})) \quad \max \quad & \sum_{v\in V}\sum_{t\in T^{\mathrm{ends}}\cup\{t_{\max}\}} \left(E^{\mathrm{cap}} + E^{\mathrm{res,end}}_{v,t-1} - E_{v,0}\right)\lambda^{\mathrm{ub}}_{v,t} + \\
& \sum_{v\in V}\sum_{t\in T^{\mathrm{starts}}} \left(E^{\mathrm{res,start}}_{v,t} - E_{v,0}\right)\lambda^{\mathrm{lb}}_{v,t} + \\
& \sum_{t\in T} E^{\mathrm{surmax}}_t \min\left(0,\ -\alpha - \lambda^{\mathrm{E}}_t\right) + \\
& \sum_{v\in V}\sum_{t\in T^{\mathrm{home}}_v} \Delta t P^{\max} \min\left(0,\ \lambda^{\mathrm{E}}_t - \sum_{k\in T^{\mathrm{ends}}\cup\{t_{\max}\}|k\geq t} \lambda^{\mathrm{ub}}_{v,k} - \right. \\
& \left. \sum_{k\in T^{\mathrm{starts}}|k\geq t+1} \lambda^{\mathrm{lb}}_{v,k}\right) \qquad (8.31) \\
\text{s.t.} \quad & \lambda^{\mathrm{E}}_t \leq c_t \qquad t\in T \qquad (8.32) \\
& \lambda^{\mathrm{ub}}_{v,t} \leq 0 \qquad v\in V,\ t\in T^{\mathrm{ends}}\cup\{t_{\max}\} \qquad (8.33) \\
& \lambda^{\mathrm{lb}}_{v,t} \geq 0 \qquad v\in V,\ t\in T^{\mathrm{starts}} \qquad (8.34)
\end{aligned}
$$

For reasons of clarity we define for a given solution to $\mathrm{DSP}(\bar{x})$, denoted by $\bar{\lambda}^{\mathrm{E}}_t$, $\bar{\lambda}^{\mathrm{lb}}_{v,t}$ and $\bar{\lambda}^{\mathrm{ub}}_{v,t}$,

$$
C_{v,t} := \min\left(0, \bar{\lambda}^{\mathrm{E}}_t - \sum_{k\in T^{\mathrm{ends}}\cup\{t_{\max}\}|k\geq t} \bar{\lambda}^{\mathrm{ub}}_{v,k} - \sum_{k\in T^{\mathrm{starts}}|k\geq t+1} \bar{\lambda}^{\mathrm{lb}}_{v,k}\right). \qquad (8.35)
$$

8.1.2 Feasibility Cuts

Feasibility cuts are inferred from extreme rays of the dual subproblem in case this problem is unbounded. Let $\bar{\lambda}^{\mathrm{lb}}_{v,t}$, $\bar{\lambda}^{\mathrm{ub}}_{v,t}$, and $\bar{\lambda}^{\mathrm{E}}_t$ represent such an extreme ray. We then have to have $\bar{\lambda}^{\mathrm{E}}_t \leq 0$ and the objective function without constants (i.e., without α) has to be positive. The third term of the objective function is zero in this case. Furthermore, $\bar{\lambda}^{\mathrm{E}}_t = 0$ due to a similar argument as before. The formulation can now be split into different parts for the different vehicles, which are independent from each other. As the total objective value is positive, some of these parts also have to provide positive contributions and therefore

also form rays. Using these rays, we get the following feasibility cuts for vehicles $v \in V$.

$$\begin{aligned}
&\sum_{t \in T^{\text{ends}} \cup \{t_{\max}\}} \left(E^{\text{cap}} + E^{\text{res,end}}_{v,t-1} - E_{v,0}\right) \bar{\lambda}^{\text{ub}}_{v,t} + \\
&\quad \sum_{t \in T^{\text{starts}}} \left(E^{\text{res,start}}_{v,t} - E_{v,0}\right) \bar{\lambda}^{\text{lb}}_{v,t} + \\
&\quad \sum_{t \in T^{\text{home}}_v} \Delta t P^{\max} C_{v,t} = \\
&\sum_{t \in T^{\text{ends}} \cup \{t_{\max}\}} \left(E^{\text{cap}} + \sum_{r \in R | t^{\text{end}}_r \leq t-1} E^{\text{res}}_r x_{r,v} - E_{v,0}\right) \bar{\lambda}^{\text{ub}}_{v,t} + \\
&\quad \sum_{t \in T^{\text{starts}}} \left(\sum_{r \in R | t^{\text{start}}_r \leq t} E^{\text{res}}_r x_{r,v} - E_{v,0}\right) \bar{\lambda}^{\text{lb}}_{v,t} + \\
&\quad \sum_{t \in T^{\text{avail}}_v} \Delta t P^{\max} C_{v,t} \left(1 - \sum_{r \in R | t \in T^{\text{res}}_r} x_{r,v}\right) \leq 0
\end{aligned} \tag{8.36}$$

8.1.3 Optimality Cuts

Optimality cuts are inferred from extreme points of the dual subproblem. Let $\bar{\lambda}^{\Delta \text{E}}_{v,t}$ and $\bar{\lambda}^{\text{E}}_t$ be such an extreme point. Unlike for extreme rays, we cannot discard constants so the above simplifications do not work here. The resulting cuts are

$$\begin{aligned}
&\sum_{v \in V} \sum_{t \in T^{\text{ends}} \cup \{t_{\max}\}} \left(E^{\text{cap}} + \sum_{r \in R | t^{\text{end}}_r \leq t-1} E^{\text{res}}_r x_{r,v} - E_{v,0}\right) \bar{\lambda}^{\text{ub}}_{v,t} + \\
&\quad \sum_{v \in V} \sum_{t \in T^{\text{starts}}} \left(\sum_{r \in R | t^{\text{start}}_r \leq t} E^{\text{res}}_r x_{r,v} - E_{v,0}\right) \bar{\lambda}^{\text{lb}}_{v,t} + \\
&\quad \sum_{t \in T} E^{\text{surmax}}_t \cdot \min\left(0, -\alpha - \bar{\lambda}^{\text{E}}_t\right) + \\
&\quad \sum_{v \in V} \sum_{t \in T^{\text{avail}}_v} \Delta t P^{\max} C_{v,t} \left(1 - \sum_{r \in R | t \in T^{\text{res}}_r} x_{r,v}\right) \leq \mu.
\end{aligned} \tag{8.37}$$

8.2 Extension of the MP

By adding information about the SP to the MP from the beginning, we may expect to reduce the number of iterations and thus the overall runtime. We do this in the following two ways.

1. The strengthening inequalities from chapter 6 can be added. As there may be exponentially many such constraints, we only add those constraints considering two reservations.

2. We can move the variables E_t^{res} and E_t^{sur} from the SP into the MP. The number of these variables is low enough ($O(|T|)$) to still keep the MP reasonably small, but now it becomes possible to add further strengthening inequalities to the MP that make use of E_t^{res} and E_t^{sur}.

For the rest of this subsection we will concentrate on this second kind of strengthening inequalities.

The MP is updated as follows.

$$\min \quad \sum_{t\in T} c_t E_t^{\text{grid}} + c^{\text{uncov}} \sum_{r\in R} E_r^{\text{res}} y_r - \alpha \sum_{t\in T} E_t^{\text{sur}} \tag{8.38}$$

$$\text{s.t.} \quad \sum_{v\in V} x_{r,v} + y_r = 1 \qquad r \in R \tag{8.39}$$

$$x_{r,v} = 0 \qquad v \in V, r \in R \mid T_r^{\text{res}} \nsubseteq T_v^{\text{avail}} \tag{8.40}$$

$$\sum_{r\in R | t\in T_r^{\text{res}}} x_{r,v} \leq 1 \qquad v \in V, t \in T \tag{8.41}$$

Conflicting Reservations (8.42)

$$E_t^{\text{grid}} + E_t^{\text{sur}} \leq \sum_{v\in V} \Delta t P^{\max} \left(1 - \sum_{r\in R | t\in T_r^{\text{res}}} x_{r,v}\right) \qquad t \in T \tag{8.43}$$

$$\sum_{k=1}^{t} \left(E_k^{\text{grid}} + E_k^{\text{sur}}\right) + \sum_{v\in V} \left(E_{v,0} - \sum_{r\in R | t_r^{\text{end}} \leq t-1} E_r^{\text{res}} x_{r,v}\right) \leq n E^{\text{cap}}$$

$$t \in T^{\text{ends}} \cup \{t_{\max}\} \tag{8.44}$$

$$\sum_{k=1}^{t-1} \left(E_k^{\text{grid}} + E_k^{\text{sur}}\right) + \sum_{v\in V} \left(E_{v,0} - \sum_{r\in R | t_r^{\text{start}} \leq t} E_r^{\text{res}} x_{r,v}\right) \geq 0 \qquad t \in T^{\text{starts}} \tag{8.45}$$

Feasibility cuts (8.46)

$$E_t^{\text{grid}} \geq 0 \qquad t \in T \tag{8.47}$$

$$0 \leq E_t^{\text{sur}} \leq E_t^{\text{surmax}} \qquad t \in T \tag{8.48}$$

$$x_{r,v} \in \{0, 1\} \tag{8.49}$$

$$0 \leq y_r \leq 1 \tag{8.50}$$

(8.51)

- As already mentioned E_t^{grid} and E_t^{sur} are added.
- The parts of the overall objective function that contain E_t^{sur} and E_t^{grid} are added to the objective function of the MP.
- Optimality cuts are therefore not necessary any more, as the objective function of the MP already comprises all parts of the original objective.

- Strengthening inequalities (8.43) to (8.45) are added.

Also the dual subproblem changes in some regards.

- The term with λ_t^{sur} in the objective drops because E_t^{sur} is no longer part of the subproblem.

- E_t^{sur} and E_t^{grid} turn into constants, leading to a new term in the objective function.

- The constraints corresponding to E_t^{sur} and E_t^{grid} are dropped.

After simplification the dual of the updated subproblem can be stated as follows.

$$
\begin{aligned}
&(\text{DSP}(\bar{x}, \bar{E}^{\text{grid}}, \bar{E}^{\text{sur}})) \\
&\max \quad \sum_{v \in V} \sum_{t \in T^{\text{ends}} \cup \{t_{\max}\}} \left(E^{\text{cap}} + E_{v,t-1}^{\text{res,end}} - E_{v,0} \right) \lambda_{v,t}^{\text{ub}} + \\
&\qquad \sum_{v \in V} \sum_{t \in T^{\text{starts}}} \left(E_{v,t}^{\text{res,start}} - E_{v,0} \right) \lambda_{v,t}^{\text{lb}} + \\
&\qquad \sum_{t \in T} \left(\bar{E}_t^{\text{grid}} + \bar{E}_t^{\text{sur}} \right) \lambda_t^{\text{E}} + \\
&\qquad \sum_{v \in V} \sum_{t \in T_v^{\text{home}}} \Delta t P^{\max} \min \left(0, -\lambda_t^{\text{E}} - \sum_{k \in T^{\text{ends}} \cup \{t_{\max}\} | k \geq t} \lambda_{v,k}^{\text{ub}} - \right. \\
&\qquad\qquad \left. \sum_{k \in T^{\text{starts}} | k \geq t+1} \lambda_{v,k}^{\text{lb}} \right) \qquad (8.52) \\
&\text{s.t.} \quad \lambda_t^{\text{E}} \in \mathbb{R} \qquad t \in T \qquad (8.53) \\
&\qquad \lambda_{v,t}^{\text{ub}} \leq 0 \qquad v \in V,\ t \in T^{\text{ends}} \cup \{t_{\max}\} \qquad (8.54) \\
&\qquad \lambda_{v,t}^{\text{lb}} \geq 0 \qquad v \in V,\ t \in T^{\text{starts}} \qquad (8.55)
\end{aligned}
$$

8.2.1 Feasibility Cuts

Let $\bar{\lambda}^{\mathrm{p}}_{v,t}$, $\bar{\lambda}^{\mathrm{ub}}_{v,t}$ and $\bar{\lambda}^{\mathrm{lb}}_{v,t}$ be a solution to the above dual subproblem that leads to a positive objective value. The feasibility cuts then result in

$$\begin{aligned}
&\sum_{v\in V}\sum_{t\in T^{\mathrm{ends}}\cup\{t_{\max}\}}\left(E^{\mathrm{cap}}+\sum_{r\in R|t_r^{\mathrm{end}}\leq t-1}E_r^{\mathrm{res}}x_{r,v}-E_{v,0}\right)\bar{\lambda}^{\mathrm{ub}}_{v,t}+\\
&\quad\sum_{v\in V}\sum_{t\in T^{\mathrm{starts}}}\left(\sum_{r\in R|t_r^{\mathrm{start}}\leq t}E_r^{\mathrm{res}}x_{r,v}-E_{v,0}\right)\bar{\lambda}^{\mathrm{lb}}_{v,t}+\\
&\quad\sum_{t\in T}\left(E_t^{\mathrm{grid}}+E_t^{\mathrm{sur}}\right)\lambda^{\mathrm{E}}_t+\\
&\quad\sum_{v\in V}\sum_{t\in T_v^{\mathrm{avail}}}\Delta t P^{\max}\left(1-\sum_{r\in R|t\in T_r^{\mathrm{res}}}x_{r,v}\right)\cdot\\
&\qquad\qquad\min\left(-\bar{\lambda}^{\mathrm{E}}_t-\sum_{k\in T^{\mathrm{ends}}\cup\{t_{\max}\}|k\geq t}\bar{\lambda}^{\mathrm{ub}}_{v,k}-\sum_{k\in T^{\mathrm{starts}}|k\geq t+1}\bar{\lambda}^{\mathrm{lb}}_{v,k}\right)\leq 0. \quad (8.56)
\end{aligned}$$

Note that there are no optimality cuts as the variables from the SP do not occur in the objective function. Also note that it is still possible to add the feasibility cuts from subsection 8.1.2. We did that in our experiments.

8.3 Solving the MP

This section discusses the use of a MILP solver and a GVNS heuristic to solve the MP.

8.3.1 MILP

In the classical Benders decomposition the MP is solved with a MILP solver. The whole Benders decomposition approach is then an exact solution approach. This means the process of iteratively solving the MP, the dual SP, and adding derived Benders cuts to the MP eventually stops with a proven optimal solution when no further cuts can be derived, given sufficient time and memory. The dual bound derived from the solution of the MP is only valid if the MP was solved to optimality. Solving the MP always to proven optimality is in general time consuming and unnecessary, if afterwards a cut is added, especially as a large portion of the time is required to prove the optimality of a found solution. We therefore solve the MP only up to a gap of 2% in the beginning. As soon as the added cuts neither invalidate nor worsen the solution to the MP, we solve the MP in the remaining iterations to proven optimality. That way we hope to improve performance while preserving exactness.

8.3.2 GVNS Heuristic

Even solving to a gap of 2% with the MILP solver is time consuming for larger instances. Therefore we also implemented a heuristic for the MP to be used in the first iterations instead of the MILP solver with a gap of 2%. The heuristic is a GVNS and its parts are described in the paragraphs below. Note that the heuristic is dedicated only to the Benders decomposition whose MP contains the variables x and y. That is because only then it is possible to speed up searching the neighborhood by using delta evaluation.

An instance of the MP is represented by

- the instance of the FSP as well as
- a set of feasibility cuts and
- a set of optimality cuts.

For each optimality cut the factors β of x and a constant β' are stored, s.t. the cut results in

$$\sum_{r \in R} \sum_{v \in V} \beta_{r,v} x_{r,v} + \beta' \leq \mu. \tag{8.57}$$

The same is done for feasibility cuts, with the difference that each cut is restricted to a single vehicle v and therefore also the vehicle has to be stored. With the factors γ and the constant γ' such a cut results in

$$\sum_{r \in R} \gamma_r x_{r,v} + \gamma' \leq 0. \tag{8.58}$$

Solutions are represented by the assignment from reservations to vehicles. To increase performance additional data is stored. In particular

- the objective value that corresponds to the assignment as well as
- for each optimality and each feasibility cut the value to which the left hand side of the inequality results in under the assignment and
- for each vehicle a bitset of time steps in which the vehicle is reserved

are stored.

Storing the left hand side of the inequalities enables to perform delta evaluation when searching a neighborhood for a better solution. To see that, assume the reservation r, which is first assigned to vehicle v_1, gets assigned to vehicle v_2. Furthermore, assume a feasibility cut for vehicle v with the factors γ whose former left hand side value is b. If $v = v_1$ the new left hand side value calculates to $b - \gamma_r$ and if $v = v_2$ the new left

hand side value calculates to $b + \gamma_r$. Otherwise it does not change. If this increases the value above zero, the feasibility cut is violated and the new solution is not valid. The same can be done with optimality cuts. Again assume that reservation r is moved from vehicle v_1 to vehicle v_2 and furthermore assume the optimality cut with the factors β and former left hand side value of b is given. Then the new left hand side value calculates to $b - \beta_{r,v_1} + \beta_{r,v_2}$. The new objective value can be calculated from the old objective value by subtracting the former maximum over all left hand side values of optimality cuts and adding the new maximum.

Besides the validity of feasibility cuts there is another condition that has to be satisfied for the new solution to be valid. The reassigned reservation may not overlap with any reservations of the vehicle it is assigned to. That can be checked efficiently by intersecting the reservation interval with the bitset of time steps in which the vehicle is reserved. If the resulting bitset is not empty there is an overlap and the new solution is not valid. Bitset is a datastructure in Julia that represents sets of integers. In a bitset an integer has a corresponding bit, which determines whether the integer is contained in the set. By using bitwise operations it is possible to efficiently implement set operations like intersection, union or the set difference.

Neighborhood structures. We implemented three neighborhood structures, based on the following moves.

1. Assign an unassinged reservation or delete the assignment of an assigned reservation.
2. Rotate the reservations $r_1, \ldots, r_k$ for $k \geq 2$. I.e. r_1 gets assigned to the vehicle r_2 is currently assigned to, r_2 gets assigned to the vehicle r_3 is assigned to and so on. Finally r_k gets assigned to the vehicle r_1 was assigned to. To make the neighborhood smaller, all reservations have to be assigned to different vehicles and the reservation intervals of two consecutive reservations have to overlap. It is also allowed for one reservation to be unassigned.
3. Swap the vehicles of two reservations. Note that this is the special case of rotation with $k = 2$. However optimizations can be done when being restricted to $k = 2$ and therefore a dedicated implementation for this case is reasonable and was implemented.

For all of these moves delta evaluation can be done as described above.

For shaking the assignments of some reservations are deleted, allowing the following VNS to converge to a different solution. Which assignments are deleted is determined in two different ways. For the first shaking method a fixed number of reservations are selected randomly and for the second shaking method a fixed number of vehicles are selected randomly and all assignments to these vehicles are deleted.

CHAPTER 9

Experiments and Results

We first describe how we created artificial benchmark instances as real world instances were not available to us. Then, the results of the MILP as well as the different Benders decomposition approaches are presented.

9.1 Benchmark instances

The benchmark instances were created randomly. We consider the following combinations of $t_{\max}$ and n:

- $t_{\max} = 32$ and $n \in \{1, 2, 5\}$
- $t_{\max} = 192$ and $n \in \{5, 10, 20\}$
- $t_{\max} = 768$ and $n \in \{20, 50, 100\}$

For each of these combinations we consider $r_{\max} = 4n$, $r_{\max} = 8n$ and $r_{\max} = 16n$. The other input values are set as follows.

- $\Delta t = 15$min, i.e., T corresponds to 8 hours, 2 days or 8 days
- $P^{\max} = 3.3$kW, $E^{\text{cap}} = 20$kWh
- For each vehicle $v \in V$
  - $T_v^{\text{avail}} = T$
  - $E_{v,0}$ is selected uniformly at random from $[0, E^{\text{cap}}]$
- For each reservation $r \in R$

  - Δt_r is selected uniformly at random from $[0, \frac{t_{\max}}{4} - 1]$
  - t_r^{start} is selected uniformly at random from $[1, t_{\max} - \Delta t_r]$
  - $t_r^{\text{end}} = t_r^{\text{start}} + \Delta t_r$
  - E_r^{res} is selected uniformly at random from $[0, E^{\text{cap}}]$ uniformly random
- Energy prices and surplus energies come from real world data, using a randomly selected time span. Surplus is are scaled according to the number of vehicles.
- $c^{\text{uncov}} = 75$, $\alpha = 75$

Thirty instances were created for each combination of $t_{\max}$, n, and $r_{\max}$. Some of the more specific tests were performed only on a subset of these instances to reduce the requirements of used computing power. To this end we selected ten instances of each instances size; we will refer to this subset as the reduced testset. Moreover we created two further testsets by altering the instances of the reduced testset in two ways. For one of these testsets we multiplied the surplus energies by 0.5. The other testset was obtained by multiplying the reservation energies by 0.15.

Figure 9.1 shows the optimal solution to such a randomly generated instance. It can for example be observed that almost all reservations have been assigned. The only two remaining reservations cannot be assigned to any of the vehicles since they would overlap with other reservations.

9.2 Implementation and Computing Environment

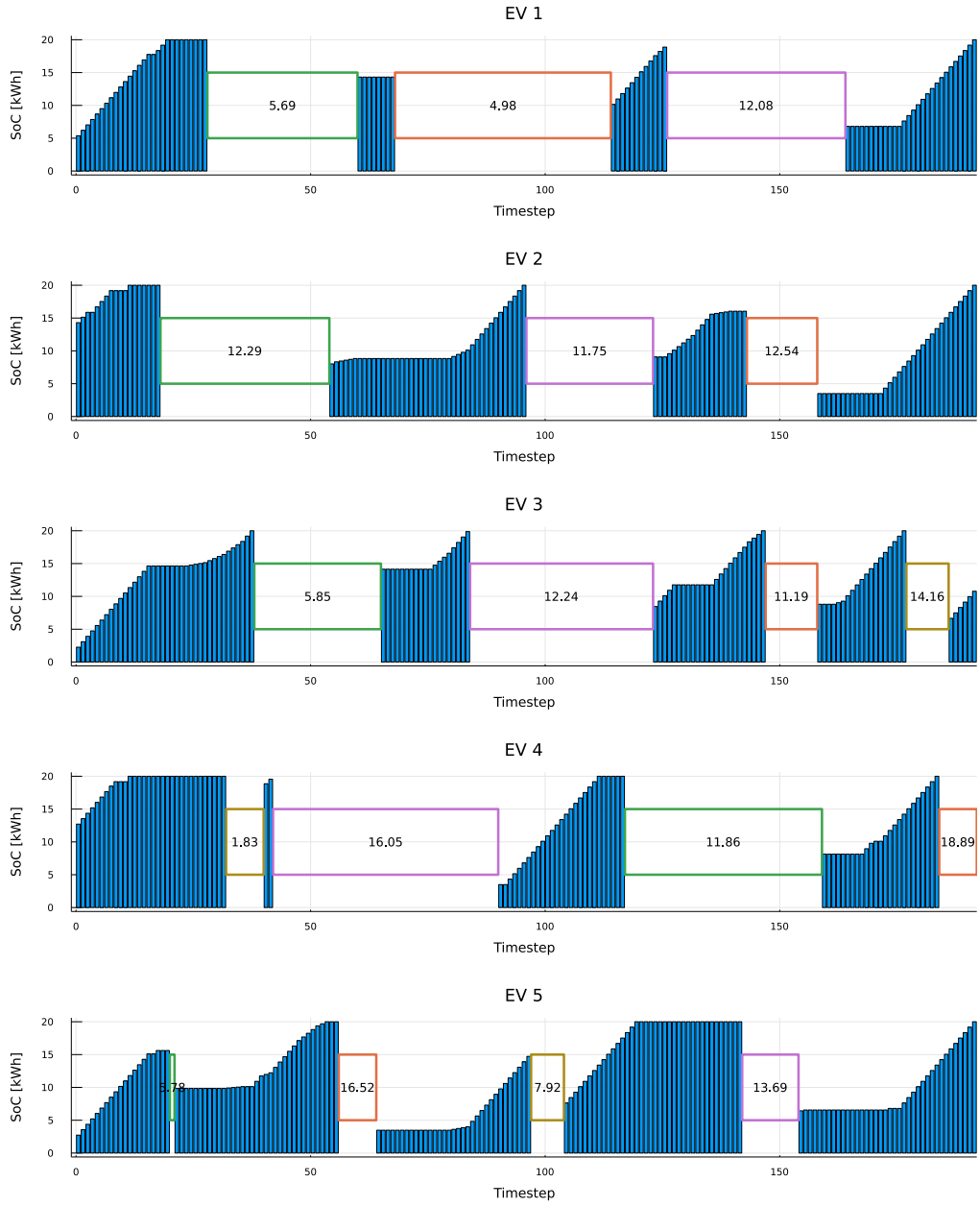
We used the MILP solvers Gurobi 9.1[1] [Gur21], which is a leading commercial product, and the open source solver SCIP 7.0.2[2] [GAB+20]. The models as well as the Benders decomposition approach were implemented in Julia 1.6[3] [BEKS17]. JuMP was used as interface to the solvers [DHL17]. Each experiment was performed on a single core of an Intel Xeon E5-2640 v4 with a time limit of one hour provided to the MILP solvers. We furthermore imposed a memory limit of 36 gigabytes.

We solved the test instances with the presented approaches in different configurations. These configurations and the results are presented in the following.

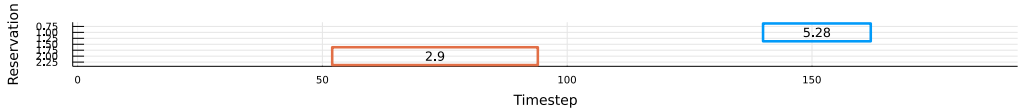
9.3 Comparison of the MILP and the Benders Decomposition

We evaluated the whole benchmark instance set with the MILP formulation and the Benders decomposition, both with Gurobi and SCIP as solver. For the Benders decomposition, the variant with the extended MP that allows a gap for the MP for the first

[1] https://www.gurobi.com
[2] https://scip.zib.de
[3] https://julialang.org



(a) Charging plans and assigned reservations for the different vehicles.



(b) Uncovered reservations

Figure 9.1: A solution of a randomly generated instance with $t_{\max} = 192$, $n = 5$ and $r_{\max} = 20$.

$t_{\max}$	n	$r_{\max}$	Solved	Runtime [s]	Gap [%]	σ_{Gap} [%]	BnB nodes
32	1	4	30 / 30	0.5	0.0	0.0	0.6
32	1	8	30 / 30	0.5	0.0	0.0	1.4
32	1	16	30 / 30	0.6	0.0	0.0	1.1
32	2	8	30 / 30	0.5	0.0	0.0	1.4
32	2	16	30 / 30	0.6	0.0	0.0	3.4
32	2	32	30 / 30	0.8	0.0	0.0	32.7
32	5	20	30 / 30	1.3	0.0	0.0	148.1
32	5	40	30 / 30	4.2	0.0	0.0	3152.7
32	5	80	30 / 30	15.6	0.0	0.0	5870.0
192	5	20	30 / 30	15.0	0.0	0.0	4136.5
192	5	40	30 / 30	146.7	0.0	0.0	25277.8
192	5	80	30 / 16	3090.9	3.5	4.7	89129.0
192	10	40	30 / 4	3611.0	0.4	0.7	40013.5
192	10	80	30 / 0	3611.3	9.8	4.0	11141.9
192	10	160	30 / 0	3612.1	31.4	26.5	10667.8
192	20	80	30 / 1	3612.5	1.0	1.4	5624.9
192	20	160	30 / 0	3613.6	44.4	61.1	689.0
192	20	320	30 / 0	3611.4	69.1	16.1	289.7
768	20	80	30 / 15	3428.0	0.1	0.2	3464.7
768	20	160	30 / 0	3618.9	467.3	591.6	30.7
768	20	320	30 / 0	3625.7	381.9	48.4	0.0
768	50	200	30 / 0	3641.3	33.0	108.0	143.9
768	50	400	30 / 0	3664.2	682.3	18.6	0.0
768	50	800	30 / 0	3715.1	385.4	6.5	0.0
768	100	400	30 / 0	3724.8	421.6	424.5	0.0
768	100	800	30 / 0	3827.9	675.1	12.9	0.0
768	100	1600	29 / 0	4051.5	389.6	4.9	0.0

Table 9.1: Results of the MILP solved with Gurobi.

iterations, was used. Furthermore we evaluated the benchmark instance set with the Benders decomposition that uses the heuristic for the MP with Gurobi as underlying solver. The results are shown in Table 9.1 to Table 9.5. For the largest instances with 100 vehicles SCIP exceeded our memory limit. We therefore decided to exclude those instances from the evaluation with SCIP. The tables for the MILP show the following columns.

- "$t_{\max}$", "n", "$r_{\max}$": The size of the instances.
- "Solved": Number of instances for which feasible solutions were found and the number of instances solved to proven optimality.

$t_{\max}$	n	$r_{\max}$	Solved	Runtime [s]	Gap [%]	σ_{Gap} [%]	BnB nodes
32	1	4	30 / 30	0.3	0.0	0.0	1.0
32	1	8	30 / 30	0.3	0.0	0.0	1.0
32	1	16	30 / 30	0.4	0.0	0.0	1.0
32	2	8	30 / 30	0.3	0.0	0.0	1.0
32	2	16	30 / 30	0.6	0.0	0.0	1.3
32	2	32	30 / 30	1.1	0.0	0.0	18.1
32	5	20	30 / 30	3.8	0.0	0.0	95.5
32	5	40	30 / 30	13.3	0.0	0.0	5426.3
32	5	80	30 / 30	21.5	0.0	0.0	27921.2
192	5	20	29 / 29	91.0	0.0	0.0	6902.2
192	5	40	30 / 23	1196.3	2.2	5.6	76609.2
192	5	80	30 / 0	3600.5	17.8	11.9	121867.2
192	10	40	30 / 1	3600.5	2.0	1.9	9221.2
192	10	80	30 / 0	3600.6	30.6	12.0	2697.3
192	10	160	30 / 0	3601.1	62.9	67.6	1306.2
192	20	80	30 / 0	3601.7	13.9	40.4	292.1
192	20	160	30 / 0	3602.7	209.1	605.8	7.9
192	20	320	30 / 0	3605.5	55.3	15.5	8.1
768	20	80	30 / 0	3608.7	37.9	71.2	91.8
768	20	160	30 / 0	3612.0	4618.7	607.8	1.0
768	20	320	30 / 0	3619.2	2744.6	531.1	1.0
768	50	200	29 / 0	3634.9	24516.6	12781.3	1.0
768	50	400	30 / 0	3659.4	8319.9	2746.2	1.0
768	50	800	30 / 0	3713.7	5563.5	40.4	1.0

Table 9.2: Results of the MILP solved with SCIP.

- "Runtime": Median solving time in seconds.
- "Gap": Mean optimality gaps over the instances where feasible solutions were found. For a primal bound c_{P} and a dual bound c_{D} the gap is calculated as

$$\left| \frac{c_{\text{P}} - c_{\text{D}}}{c_{\text{P}}} \right|. \tag{9.1}$$

- "σ_{Gap}": The standard deviation of the gap.
- "BnB nodes": Number of branch-and-bound nodes of the solver.

The tables for the Benders decomposition have additionally the following columns.

- "Runtime": The runtime is split up into the time spent in (re-)solving the MP and in solving the subproblem instances. All stated times are median values.

$t_{\max}$	n	$r_{\max}$	Solved	Runtime [s] Total	Master	Sub	Gap [%]	σ_{Gap} [%]	Iterations	Cuts
32	1	4	30 / 30	2.3	0.1	1.8	0.0	0.0	1.2	0.0
32	1	8	30 / 30	2.3	0.1	1.8	0.0	0.0	1.5	0.0
32	1	16	30 / 30	4.2	2.0	1.8	0.0	0.0	2.8	0.0
32	2	8	30 / 30	4.9	2.5	1.9	0.0	0.0	8.0	11.5
32	2	16	30 / 30	5.2	2.9	2.0	0.0	0.0	9.2	14.2
32	2	32	30 / 30	6.3	3.9	2.1	0.0	0.0	11.7	19.8
32	5	20	30 / 30	11.5	8.5	2.7	0.0	0.0	19.8	50.9
32	5	40	30 / 30	36.7	33.0	3.5	0.0	0.0	26.9	74.7
32	5	80	30 / 30	52.7	48.2	3.6	0.0	0.0	28.6	86.0
192	5	20	30 / 0	3605.4	3479.6	61.8	2.3	2.5	417.7	445.1
192	5	40	30 / 1	3605.7	3517.9	47.8	6.1	4.0	227.9	311.7
192	5	80	21 / 0	3607.1	3586.2	19.7	18.1	13.9	56.3	201.0
192	10	40	30 / 0	3609.7	3419.2	106.5	3.8	2.4	265.1	435.5
192	10	80	8 / 0	3610.9	3589.5	18.8	10.6	2.2	43.2	285.2
192	10	160	0 / 0	3623.6	3592.5	31.1	–	–	31.1	308.4
192	20	80	29 / 0	3626.6	3456.2	146.7	4.7	2.0	136.5	656.4
192	20	160	0 / 0	3635.3	3595.9	38.0	–	–	26.1	443.6
192	20	320	0 / 0	3679.3	3617.5	67.7	–	–	28.4	538.5
768	20	80	30 / 0	3641.6	3074.1	340.7	1.1	0.7	82.4	256.6
768	20	160	4 / 0	3677.4	3501.1	164.9	186.0	6.2	36.5	472.2
768	20	320	0 / 0	3691.8	3624.8	66.0	–	–	10.5	215.4
768	50	200	3 / 0	3687.7	3150.6	543.2	248.2	3.8	26.2	295.7
768	50	400	13 / 0	3832.8	3674.8	144.9	190.7	2.9	6.0	243.5
768	50	800	20 / 0	4305.1	3809.7	176.5	128.7	1.9	3.4	139.5
768	100	400	9 / 0	3678.2	3062.9	648.8	249.8	5.6	8.7	173.9
768	100	800	29 / 0	4965.5	3935.0	297.5	191.3	2.2	1.8	71.9
768	100	1600	28 / 0	5855.4	4149.8	310.1	140.4	49.1	1.1	14.0

Table 9.3: Results of the Benders decomposition with Gurobi.

- "Iterations": Mean number of iterations.
- "Cuts": Mean number of added cuts.

Table 9.6 brings together the columns "Solved", "Runtime" and "Gap" for an easy direct comparison of the MILP formulation with the Benders decomposition that does not use the heuristic. Table 9.7 compares the Benders decomposition with heuristic with the MILP formulation and the Benders decomposition without heuristic. Since the Benders decomposition with heuristic was not able to find dual bounds for many of the larger instances, the mean objective value and dual bound are shown instead of the gap.

$t_{\max}$	n	$r_{\max}$	Solved	Runtime [s] Total	Master	Sub	Gap [%]	σ_{Gap} [%]	Iterations	Cuts
32	1	4	30 / 30	0.7	0.2	0.1	0.0	0.0	1.4	0.0
32	1	8	30 / 30	0.8	0.3	0.1	0.0	0.0	1.9	0.0
32	1	16	30 / 30	2.9	2.2	0.2	0.0	0.0	2.9	0.0
32	2	8	30 / 29	4.4	3.2	0.6	0.1	0.7	126.5	129.7
32	2	16	30 / 30	6.1	4.9	0.6	0.0	0.0	9.3	13.9
32	2	32	29 / 28	8.7	7.6	0.7	0.1	0.4	9.9	15.7
32	5	20	29 / 29	41.1	37.9	2.5	0.0	0.0	18.6	46.0
32	5	40	30 / 30	158.6	153.7	4.2	0.0	0.0	24.3	63.9
32	5	80	30 / 29	163.2	158.2	3.9	0.0	0.2	22.4	67.9
192	5	20	11 / 0	3601.5	3590.6	6.1	110.7	225.1	17.6	31.8
192	5	40	8 / 0	3601.4	3595.5	5.2	161.4	215.5	9.4	43.7
192	5	80	0 / 0	3602.4	3593.6	8.5	–	–	10.8	61.0
192	10	40	7 / 0	3601.4	3587.0	6.2	47.0	48.2	13.2	63.4
192	10	80	0 / 0	3603.8	3593.2	11.0	–	–	9.3	90.3
192	10	160	0 / 0	3608.1	3587.4	20.2	–	–	9.6	101.5
192	20	80	6 / 0	3607.5	3563.9	45.6	86.2	75.2	24.5	202.7
192	20	160	0 / 0	3612.7	3588.4	25.6	–	–	7.1	135.2
192	20	320	1 / 0	3631.9	3559.7	64.8	92.0	0.0	8.6	170.9
768	20	80	2 / 0	3625.5	3585.7	41.4	115.2	114.4	2.2	9.9
768	20	160	3 / 0	3628.1	3588.8	45.4	128.3	125.2	3.0	45.5
768	20	320	0 / 0	3647.6	3589.6	65.5	–	–	2.1	43.1
768	50	200	1 / 0	3748.3	3523.3	215.0	589.4	0.0	2.2	26.4
768	50	400	1 / 0	3845.1	3560.4	290.7	223.2	0.0	2.0	65.6
768	50	800	2 / 0	4032.6	3705.6	322.5	134.8	0.3	1.2	54.1

Table 9.4: Results of the Benders decomposition with SCIP.

Additionally to the tables, Figure 9.2 to Figure 9.5 present the distributions of runtimes and gaps as boxplots over the different instance sizes. Note that the axes are scaled logarithmically. As a logarithmic axis does not include the value zero, all gaps less than 10^{-4} are represented by the value 10^{-4}.

For the larger instances the time limit of one hour is sometimes exceeded. The reason for the MILP formulation is the overhead for setting up the model in Julia. For the Benders decomposition the last iteration of the main loop including the solving of the subproblem is always completed even when the time limit has been reached.

The MILP approach is mostly superior to the Benders decomposition without heuristic in terms of solved instances, needed times, and optimality gap. The exception is the gap on large instances. For these the solver has difficulties in finding any non-trivial primal solution with the MILP. The Benders decomposition approach finds here slightly

$t_{\max}$	n	$r_{\max}$	Solved	Runtime [s] Total	Master	Sub	Gap [%]	σ_{Gap} [%]	Iterations	Cuts
32	1	4	30 / 30	2.0	1.5	0.1	0.0	0.0	4.0	1.5
32	1	8	30 / 30	4.5	3.8	0.1	0.0	0.0	6.3	2.9
32	1	16	30 / 30	6.9	6.4	0.2	0.0	0.0	9.0	5.5
32	2	8	30 / 30	6.4	5.8	0.2	0.0	0.0	8.2	6.3
32	2	16	30 / 30	8.1	7.3	0.3	0.0	0.0	10.8	10.7
32	2	32	30 / 30	11.8	11.1	0.4	0.0	0.0	14.3	16.7
32	5	20	30 / 30	14.8	13.5	0.8	0.0	0.0	20.3	33.9
32	5	40	30 / 30	35.3	33.0	1.6	0.0	0.0	27.5	56.5
32	5	80	30 / 29	53.7	51.5	2.1	0.2	0.9	27.7	70.3
192	5	20	30 / 1	3593.1	3532.0	57.2	0.8	0.6	321.6	31.8
192	5	40	30 / 1	3601.5	3563.6	36.6	5.2	4.4	149.9	90.9
192	5	80	30 / 0	3602.5	3585.5	15.2	41.8	31.9	66.6	165.0
192	10	40	30 / 0	3602.7	3581.3	22.3	13.9	5.9	83.1	146.8
192	10	80	30 / 0	3607.9	3594.3	13.4	–	–	39.7	198.7
192	10	160	30 / 0	3634.8	3606.7	24.3	–	–	39.7	230.9
192	20	80	30 / 0	3610.4	3592.3	16.2	44.6	14.9	29.0	177.5
192	20	160	30 / 0	3637.2	3600.3	32.7	–	–	30.9	307.4
192	20	320	30 / 0	3638.4	3557.6	74.7	–	–	33.9	391.2
768	20	80	30 / 0	3612.0	3449.1	154.2	14.8	5.7	69.4	192.4
768	20	160	30 / 0	3636.1	3560.2	61.2	–	–	25.2	150.5
768	20	320	30 / 0	3636.3	3463.8	152.0	–	–	31.3	272.1
768	50	200	30 / 0	3654.9	3433.0	178.3	30.6	8.6	18.4	183.5
768	50	400	30 / 0	3687.6	3270.2	340.9	–	–	26.1	373.0
768	50	800	30 / 0	3758.8	2557.2	1050.3	–	–	30.0	830.1
768	100	400	29 / 0	3767.5	2850.5	758.5	45.5	15.8	25.0	376.0
768	100	800	19 / 0	3922.5	2023.7	1586.0	–	–	29.6	1028.4
768	100	1600	0 / 0	3655.7	2017.1	1620.6	–	–	9.0	809.0

Table 9.5: Results of the Benders decomposition with heuristic and Gurobi.

better primal solutions and much better dual bounds. Hence the Benders decomposition achieves better gaps in these cases.

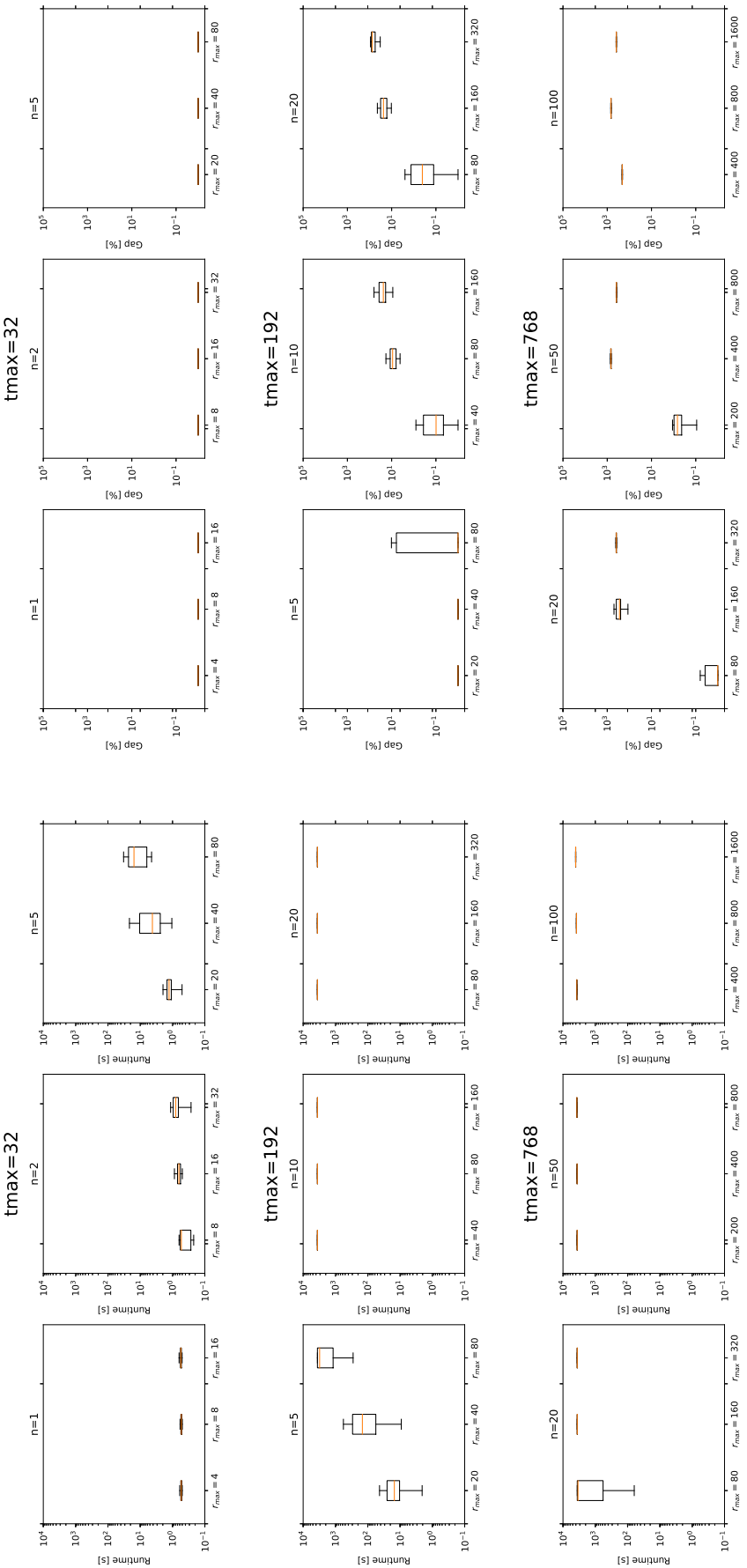
Naturally the difficulty of the instances increases with n and $r_{\max}$. But also $t_{\max}$ increases the difficulty, although $t_{\max}$ does not influence the MP directly. However, increasing $t_{\max}$ leads to a higher number of iterations. This is especially pronounced for $n = 5$ and the transition from $t_{\max} = 32$ to $t_{\max} = 192$. We also want to point out that the fraction between $r_{\max}$ and n, which indicates whether the carsharing system is overbooked, has a high influence on the runtime and gap. This can be seen particularly well for the Benders decomposition with Gurobi for the instance sizes with $n = 10$ and $n = 20$. Here solutions to almost all instances with $r_{\max} = 4n$ could be found, but not many solutions with

			Solved				Runtime [s]				Gap [%]			
			MILP		Benders		MILP		Benders		MILP		Benders	
$t_{\max}$	n	$r_{\max}$	Gurobi	SCIP	Gurobi	SCIP	Gurobi	SCIP	Gurobi	SCIP	Gurobi	SCIP	Gurobi	SCIP
32	1	4	30 / 30	30 / 30	30 / 30	30 / 30	0.5	0.3	2.3	0.7	0.0	0.0	0.0	0.0
32	1	8	30 / 30	30 / 30	30 / 30	30 / 30	0.5	0.3	2.3	0.8	0.0	0.0	0.0	0.0
32	1	16	30 / 30	30 / 30	30 / 30	30 / 30	0.6	0.4	4.2	2.9	0.0	0.0	0.0	0.0
32	2	8	30 / 30	30 / 30	30 / 30	30 / 29	0.5	0.3	4.9	4.4	0.0	0.0	0.0	0.1
32	2	16	30 / 30	30 / 30	30 / 30	30 / 30	0.6	0.6	5.2	6.1	0.0	0.0	0.0	0.0
32	2	32	30 / 30	30 / 30	30 / 30	29 / 28	0.8	1.1	6.3	8.7	0.0	0.0	0.0	0.1
32	5	20	30 / 30	30 / 30	30 / 30	29 / 29	1.3	3.8	11.5	41.1	0.0	0.0	0.0	0.0
32	5	40	30 / 30	30 / 30	30 / 30	30 / 30	4.2	13.3	36.7	158.6	0.0	0.0	0.0	0.0
32	5	80	30 / 30	30 / 30	30 / 30	30 / 29	15.6	21.5	52.7	163.2	0.0	0.0	0.0	0.0
192	5	20	30 / 30	29 / 29	30 / 0	11 / 0	15.0	91.0	3605.4	3601.5	0.0	0.0	2.3	110.7
192	5	40	30 / 30	30 / 23	30 / 1	8 / 0	146.7	1196.3	3605.7	3601.4	0.0	2.2	6.1	161.4
192	5	80	30 / 16	30 / 0	21 / 0	0 / 0	3090.9	3600.5	3607.1	3602.4	3.5	17.8	18.1	–
192	10	40	30 / 4	30 / 1	30 / 0	7 / 0	3611.0	3600.5	3609.7	3601.4	0.4	2.0	3.8	47.0
192	10	80	30 / 0	30 / 0	8 / 0	0 / 0	3611.3	3600.6	3610.9	3603.8	9.8	30.6	10.6	–
192	10	160	30 / 0	30 / 0	0 / 0	0 / 0	3612.1	3601.1	3623.6	3608.1	31.4	62.9	–	–
192	20	80	30 / 1	30 / 0	29 / 0	6 / 0	3612.5	3601.7	3626.6	3607.5	1.0	13.9	4.7	86.2
192	20	160	30 / 0	30 / 0	0 / 0	0 / 0	3613.6	3602.7	3635.3	3612.7	44.4	209.1	–	–
192	20	320	30 / 0	30 / 0	0 / 0	1 / 0	3611.4	3605.5	3679.3	3631.9	69.1	55.3	–	92.0
768	20	80	30 / 15	30 / 0	30 / 0	2 / 0	3428.0	3608.7	3641.6	3625.5	0.1	37.9	1.1	115.2
768	20	160	30 / 0	30 / 0	4 / 0	3 / 0	3618.9	3612.0	3677.4	3628.1	467.3	4618.7	186.0	128.3
768	20	320	30 / 0	30 / 0	0 / 0	0 / 0	3625.7	3619.2	3691.8	3647.6	381.9	2744.6	–	–
768	50	200	30 / 0	29 / 0	3 / 0	1 / 0	3641.3	3634.9	3687.7	3748.3	33.0	24516.6	248.2	589.4
768	50	400	30 / 0	30 / 0	13 / 0	1 / 0	3664.2	3659.4	3832.8	3845.1	682.3	8319.9	190.7	223.2
768	50	800	30 / 0	30 / 0	20 / 0	2 / 0	3715.1	3713.7	4305.1	4032.6	385.4	5563.5	128.7	134.8
768	100	400	30 / 0	0 / 0	9 / 0	0 / 0	3724.8	–	3678.2	–	421.6	–	249.8	–
768	100	800	30 / 0	0 / 0	29 / 0	0 / 0	3827.9	–	4965.5	–	675.1	–	191.3	–
768	100	1600	29 / 0	0 / 0	28 / 0	0 / 0	4051.5	–	5855.4	–	389.6	–	140.4	–

Table 9.6: Comparison of the MILP and Benders approach with Gurobi and SCIP as underlying MILP solver.

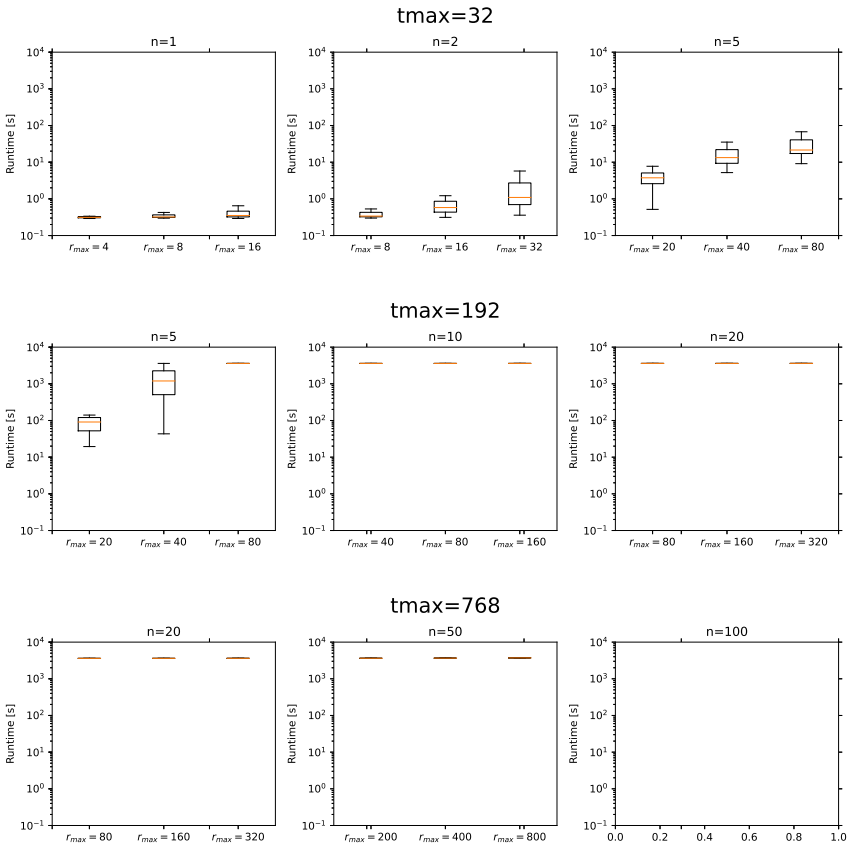
			Solved			Objective			Dual Bound		
				Benders			Benders			Benders	
$t_{\max}$	n	$r_{\max}$	MILP	Normal	Heur	MILP	Normal	Heur	MILP	Normal	Heur
32	1	4	30 / 30	30 / 30	30 / 30	902.9	902.9	902.9	902.9	902.9	902.9
32	1	8	30 / 30	30 / 30	30 / 30	3155.1	3155.1	3155.1	3155.1	3155.1	3155.1
32	1	16	30 / 30	30 / 30	30 / 30	8672.5	8672.5	8672.5	8672.5	8672.5	8672.5
32	2	8	30 / 30	30 / 30	30 / 30	993.5	993.5	993.5	993.5	993.5	993.5
32	2	16	30 / 30	30 / 30	30 / 30	6339.4	6339.4	6339.4	6339.4	6339.4	6339.4
32	2	32	30 / 30	30 / 30	30 / 30	18672.1	18672.1	18672.1	18672.0	18672.0	18672.1
32	5	20	30 / 30	30 / 30	30 / 30	129.5	129.5	129.5	129.5	129.5	129.5
32	5	40	30 / 30	30 / 30	30 / 30	13700.1	13700.1	13700.1	13699.6	13699.8	13699.8
32	5	80	30 / 30	30 / 30	30 / 29	42601.0	42601.0	42634.7	42597.8	42598.8	42555.8
192	5	20	30 / 30	30 / 0	30 / 1	-11992.4	-11991.6	-11990.7	-11993.3	-12258.3	-12090.9
192	5	40	30 / 30	30 / 1	30 / 1	-9146.7	-9092.6	-9128.6	-9147.3	-9596.6	-9572.3
192	5	80	30 / 16	21 / 0	30 / 0	6939.1	7854.2	7810.7	6753.7	6645.9	5087.1
192	10	40	30 / 4	30 / 0	30 / 0	-26219.5	-26084.9	-25843.7	-26325.6	-27076.3	-29442.0
192	10	80	30 / 0	8 / 0	30 / 0	-22372.0	-21122.4	-19046.0	-24465.2	-23287.1	-34630.2
192	10	160	30 / 0	0 / 0	30 / 0	14687.5	–	25627.1	11000.9	–	1402.9
192	20	80	30 / 1	29 / 0	30 / 0	-54185.2	-53745.9	-52201.0	-54726.8	-56245.2	-75246.6
192	20	160	30 / 0	0 / 0	30 / 0	-36967.7	–	-40475.3	-53829.0	–	–
192	20	320	30 / 0	0 / 0	30 / 0	72716.0	–	47410.0	17334.9	–	–
768	20	80	30 / 15	30 / 0	30 / 0	-65048.9	-64900.2	-64559.9	-65116.8	-65596.4	-74111.7
768	20	160	30 / 0	4 / 0	30 / 0	23516.1	109664.3	-83376.5	-92294.4	-94076.9	-116818.6
768	20	320	30 / 0	0 / 0	30 / 0	240765.6	–	-35795.4	-678548.8	–	–
768	50	200	30 / 0	3 / 0	30 / 0	-148418.9	104985.3	-158456.3	-161688.8	-155522.1	-206772.6
768	50	400	30 / 0	13 / 0	30 / 0	298170.1	258738.0	-214365.2	-1734478.1	-234573.9	–
768	50	800	30 / 0	20 / 0	30 / 0	607800.0	570137.8	-90132.9	-1734026.8	-163596.4	–
768	100	400	30 / 0	9 / 0	29 / 0	298917.9	218878.4	-318291.1	-955325.0	-327358.2	-462961.6
768	100	800	30 / 0	29 / 0	19 / 0	603159.2	529445.7	-444636.5	-3467469.7	-483324.0	–
768	100	1600	29 / 0	28 / 0	0 / 0	1199898.3	1124455.1	–	-3474518.3	-457390.1	–

Table 9.7: Comparison of the Benders decomposition with heuristic with the MILP formulation and the Benders decomposition with extended MP and allowed gap for the MP in the first iterations.

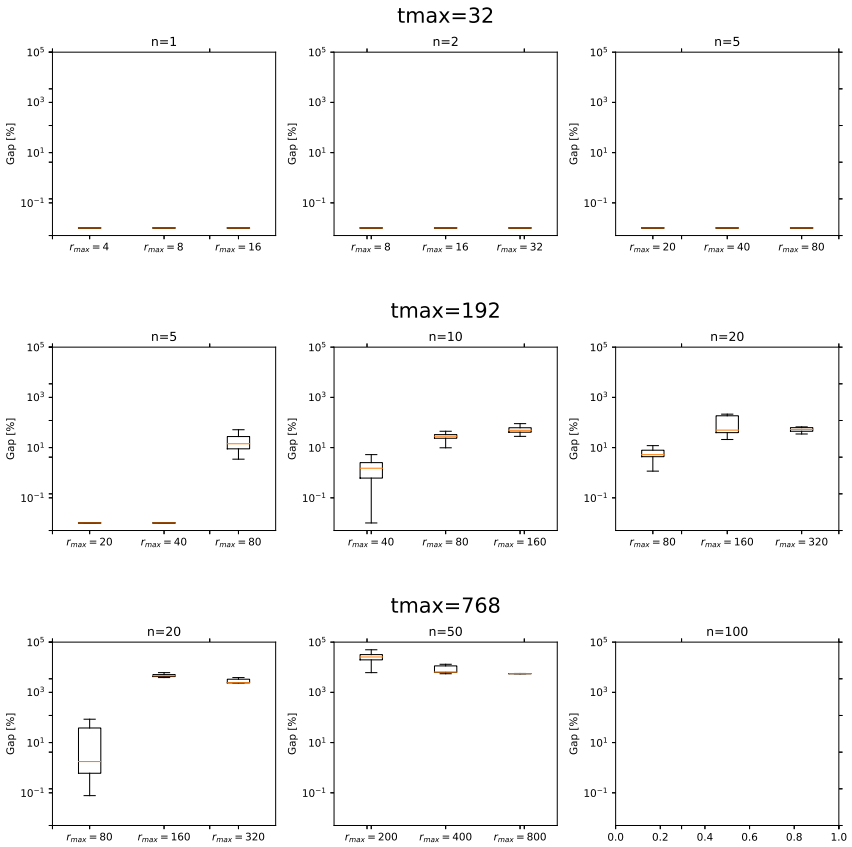


(a) Runtimes (b) Gaps

Figure 9.2: Runtimes and gaps for the MILP solved by Gurobi.

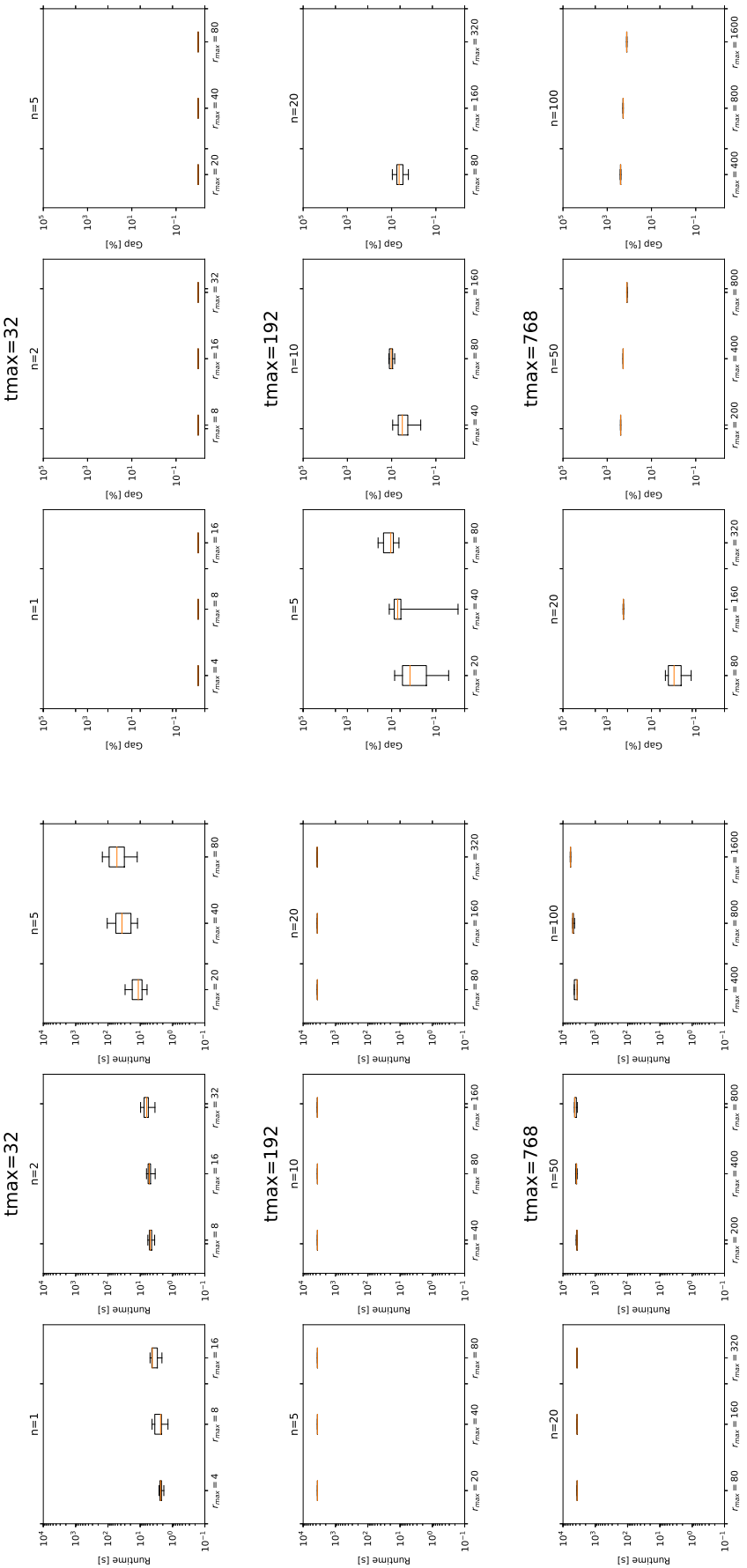


(a) Runtimes



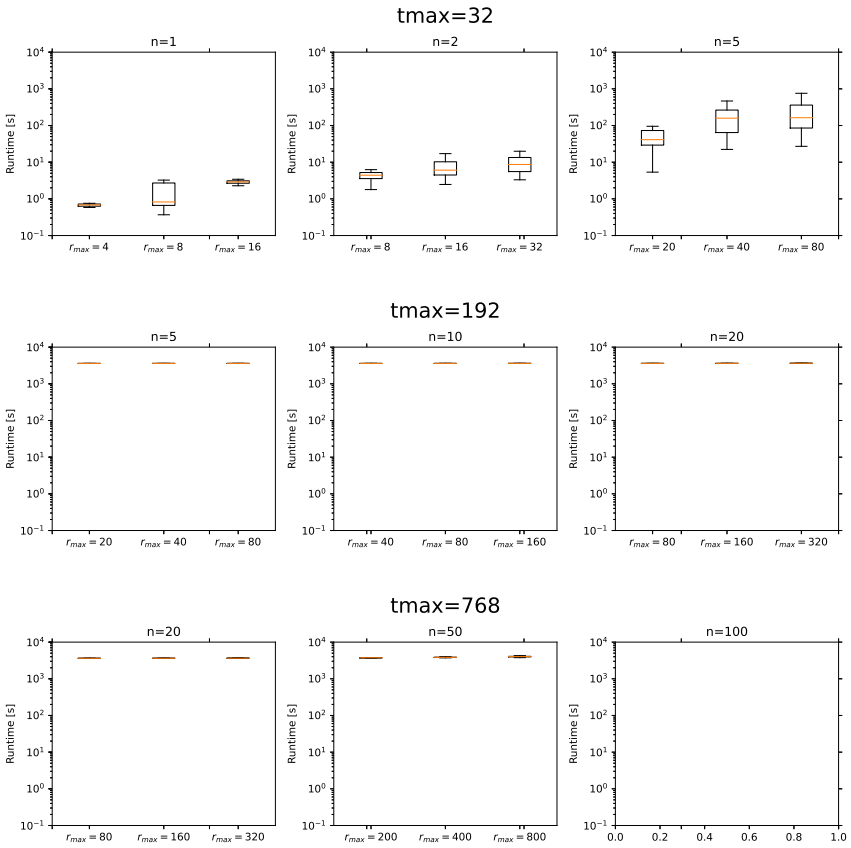
(b) Gaps

Figure 9.3: Runtimes and gaps for the MILP solved by SCIP.

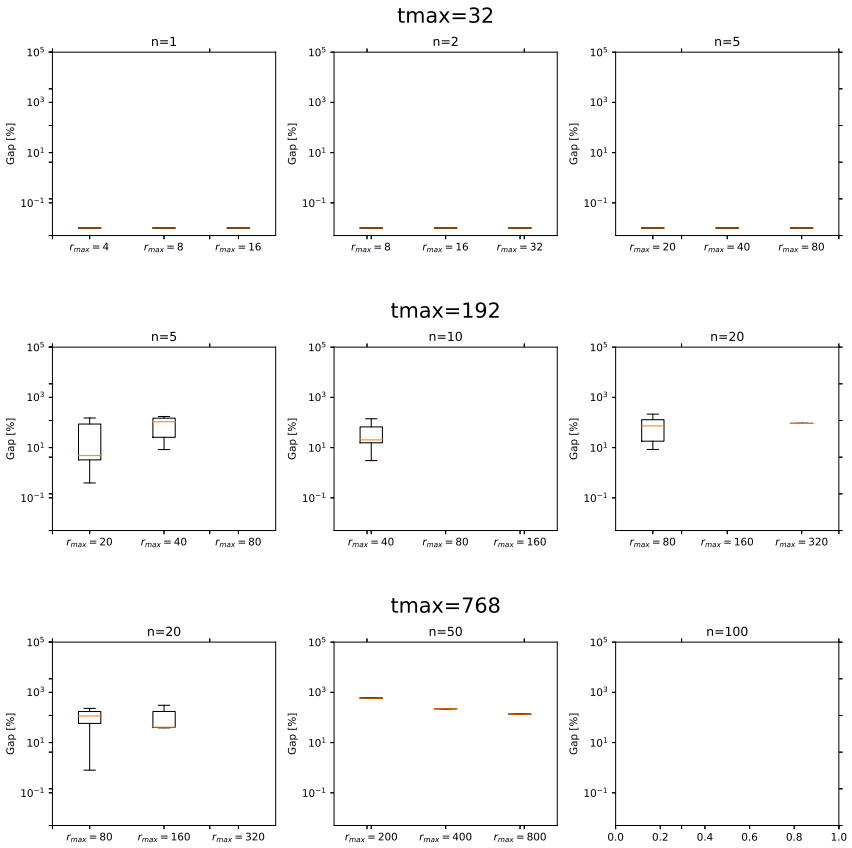


(a) Runtimes (b) Gaps

Figure 9.4: Runtimes and gaps for the Benders decomposition using Gurobi.



(a) Runtimes



(b) Gaps

Figure 9.5: Runtimes and gaps for the Benders decomposition using SCIP.

$r_{\max} = 8n$ or $r_{\max} = 16n$ have been found.

Most of the time of the Benders decomposition approach is spent in solving the MP even tough a remaining gap of 2% is allowed for the first iterations. What is more, for large instances the MP is not even solved a single time within the allowed gap. Here, the variant that uses the heuristic for the MP for the first iterations gives significantly better results. For all instance sizes with $n = 20$, $n = 50$ and $n = 100$ except $(t_{\max}, n, r_{\max}) = (192, 20, 80)$ and $(t_{\max}, n, r_{\max}) = (768, 100, 1600)$ the Benders decomposition with heuristic gives the best solutions. For many of these instance sizes it even is the only approach giving reasonable solutions.

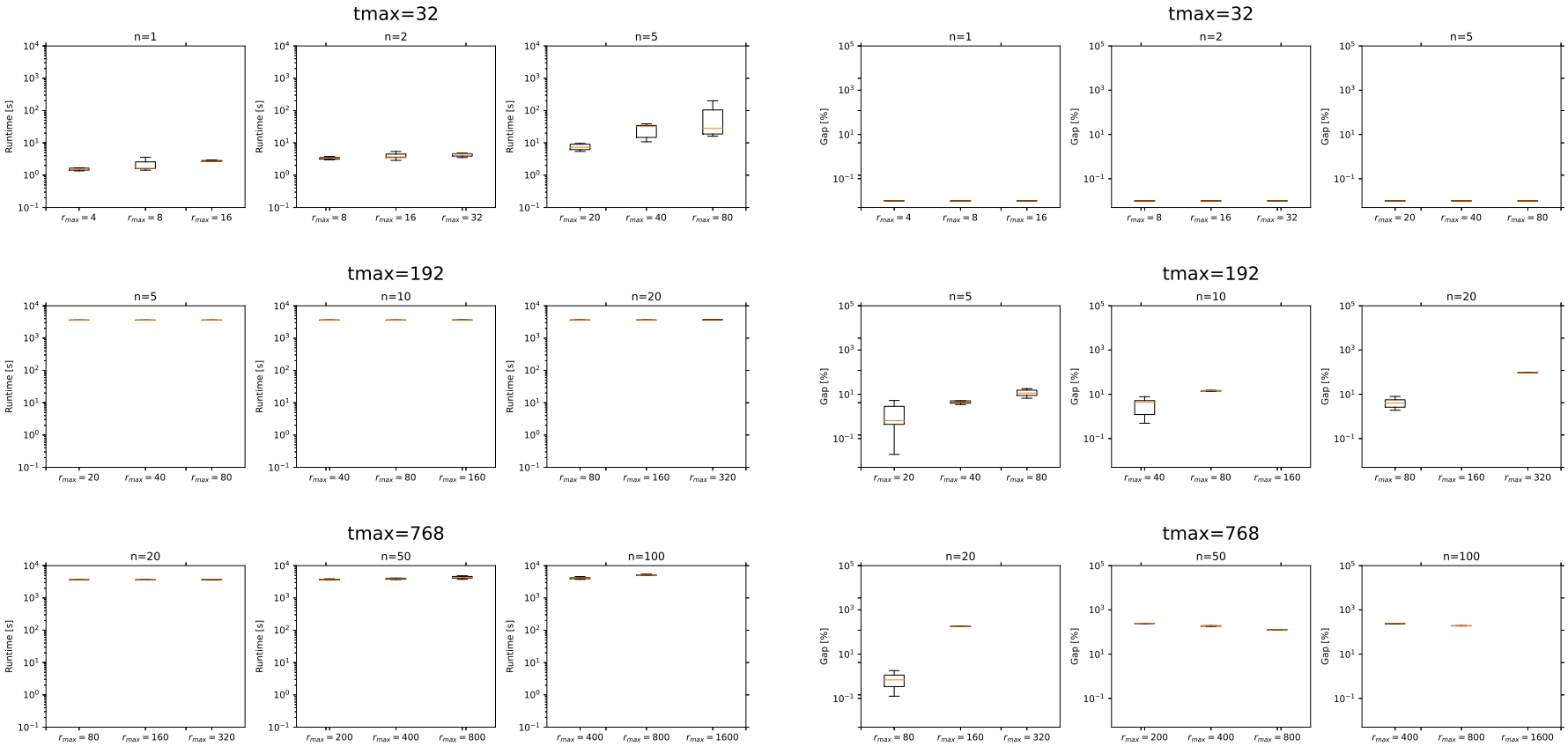
Results of SCIP are usually worse than the respective ones obtained with Gurobi, in particular in respect to running times and the number of solutions solved to optimality. In terms of running time Gurobi is up to 10 times faster than SCIP. For most instance sizes Gurobi has significantly smaller gaps. That in particular holds for the Benders decomposition on medium sized instances. In general the Benders decomposition with SCIP struggled with finding solutions for instances with $t_{\max} = 192$ and $t_{\max} = 768$.

9.4 Impacts of Individual Improvements

Figure 9.6, Figure 9.7, Figure 9.8, Figure 9.9, and Figure 9.10 show runtimes and gaps when either

1. adding inequalities from chapter 6 for pairs of vehicles already statically from the beginning to the MP,
2. solving the MP to optimality in all iterations,
3. having E^{grid} and E^{sur} in the SP instead of the MP,
4. using Branch-and-Check instead of the more classical benders approach, or
5. using the heuristic for the MP in the beginning

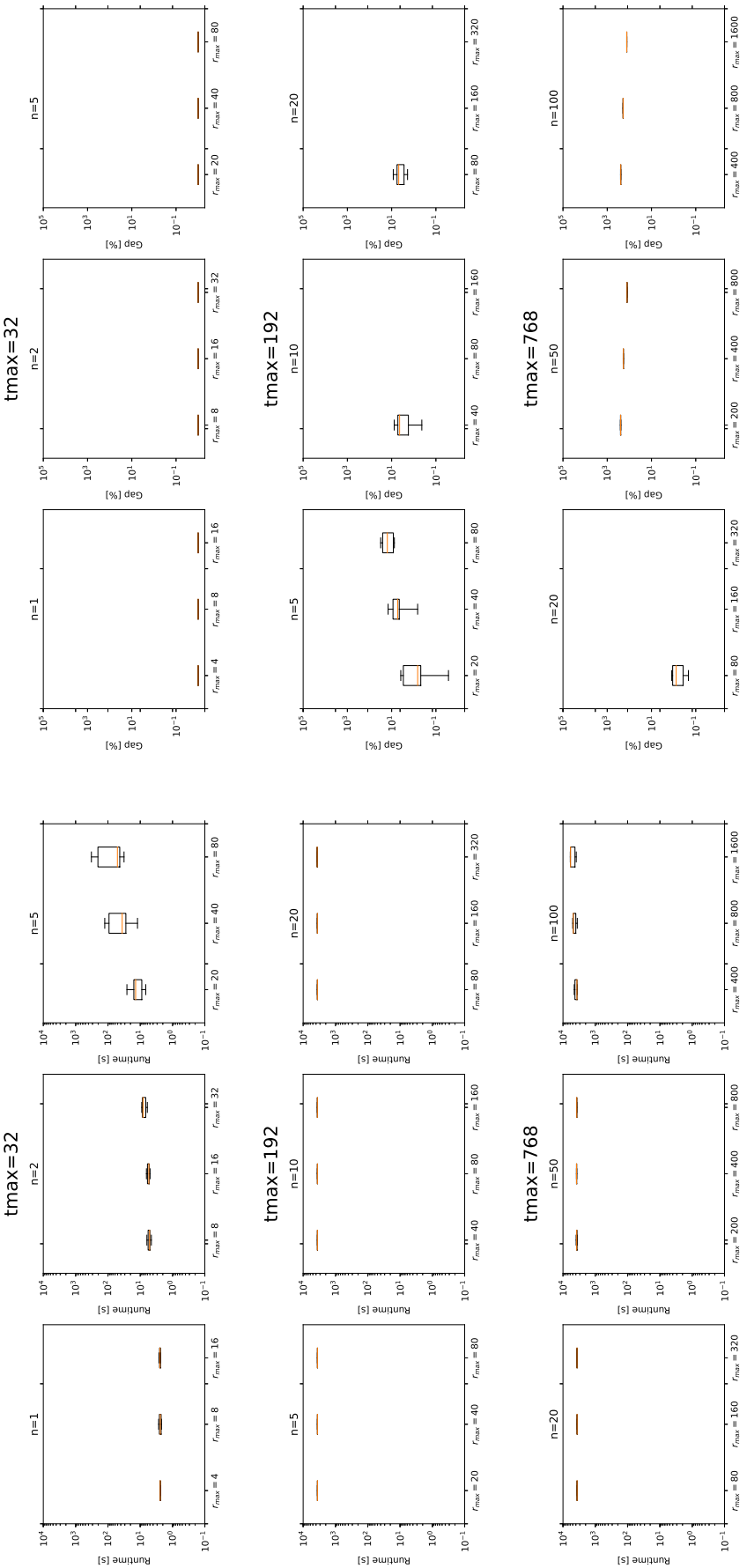
respectively. The variants of item 1 to item 4 have been evaluated with the reduced testset, variant 5 has been evaluated with the normal testset. In case 1 there is almost no difference in the runtimes or gaps. We assume that the powerful pre-solving and the generic cuts implemented in Gurobi and SCIP already substitute our problem-specific inequalities well. Solving the MP to optimality in all iterations increases the runtimes and final gaps slightly for some instance sizes and makes no difference in other cases. We conclude that a more fine-grained termination condition than our 2% gap threshold might yield higher runtime improvements. Concerning the third case, having E^{grid} and E^{sur} in the MP is again beneficial in some cases and makes no significant difference in the others. We especially point out that the gap is smaller for the large instances when E^{grid} and E^{sur} are in the MP. The reason is that for these instances only one iteration



(a) Runtimes

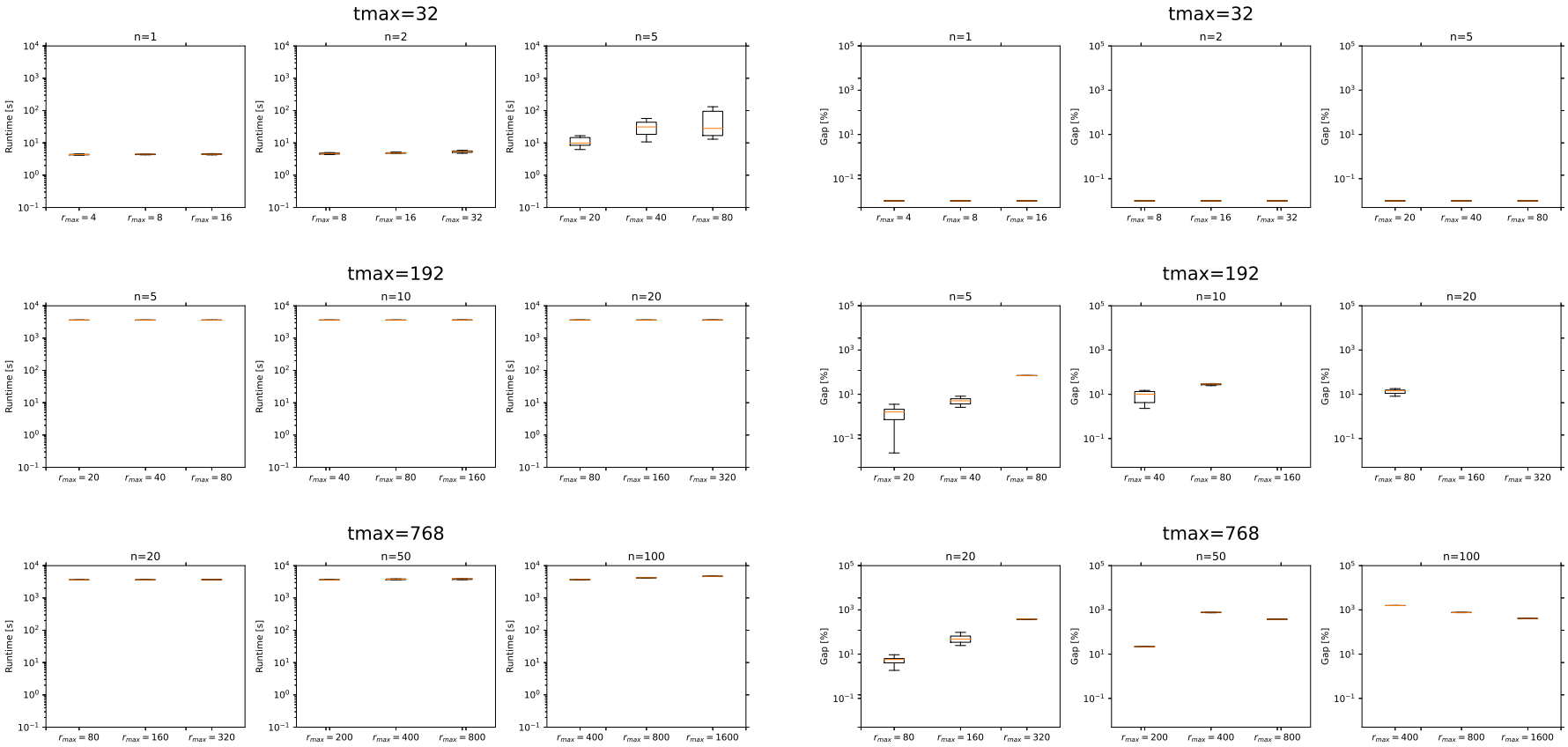
(b) Gaps

Figure 9.6: Runtimes and gaps if static cuts as discussed in chapter 6 are added to the MP.



(a) Runtimes (b) Gaps

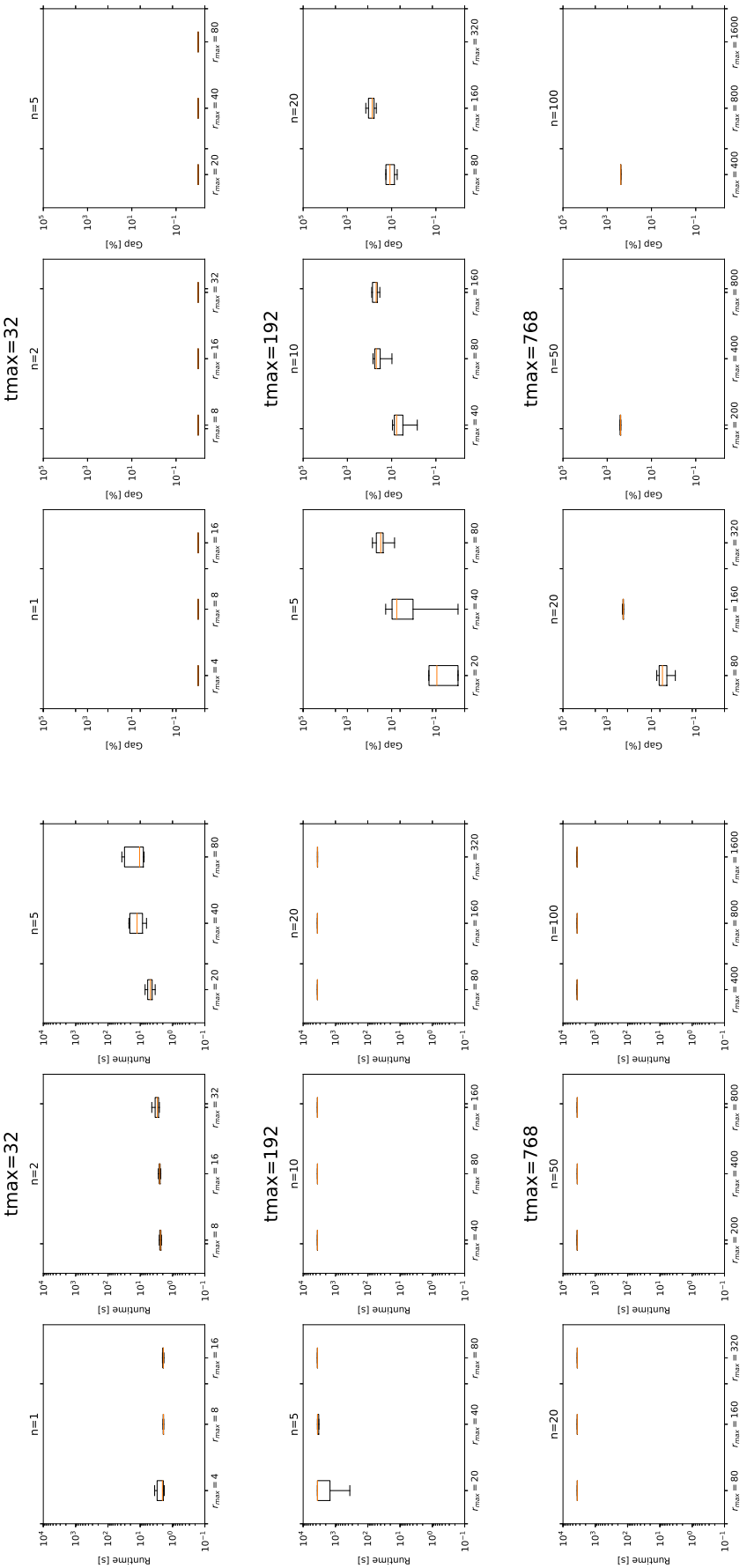
Figure 9.7: Runtimes and gaps if the MP is solved to optimality from the beginning.



(a) Runtimes

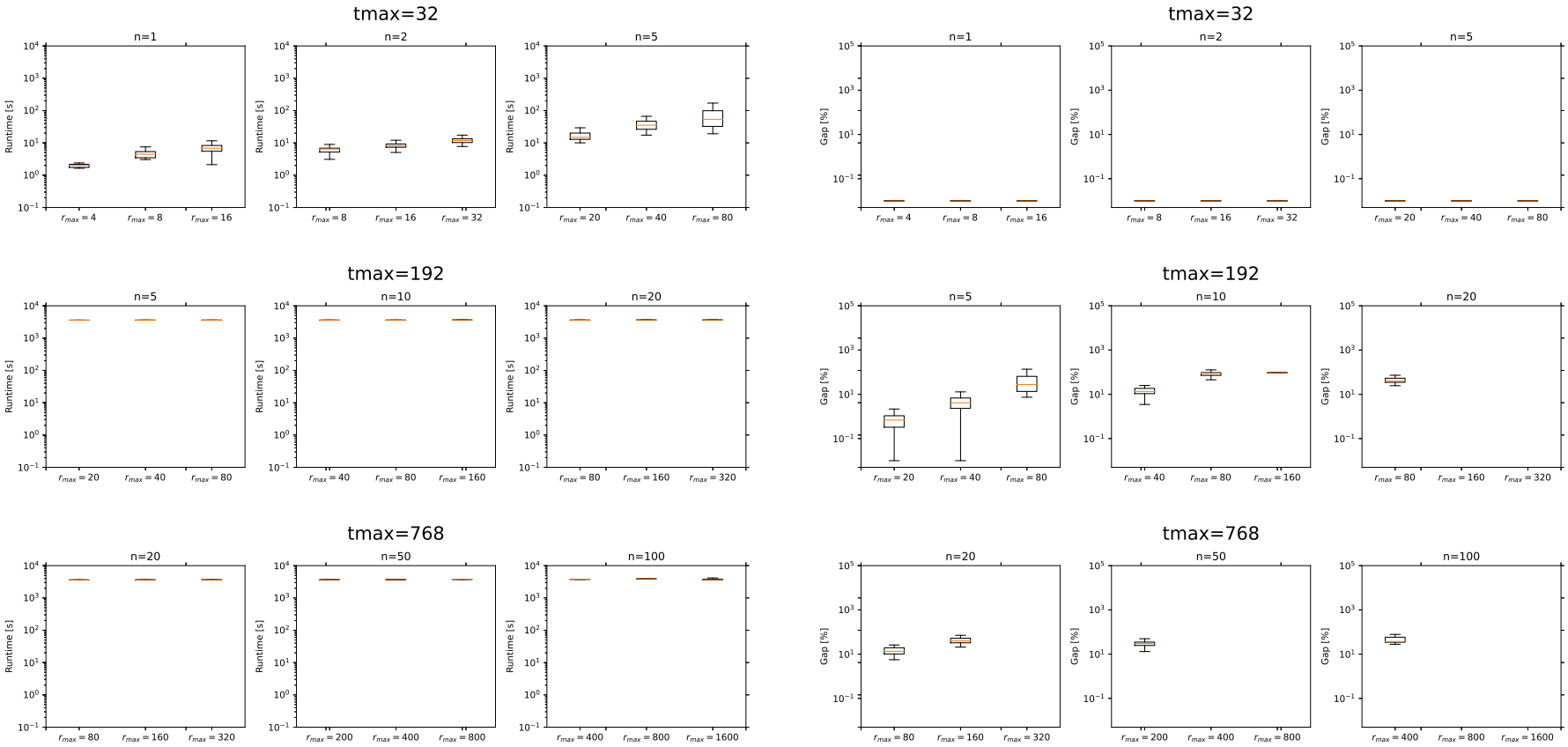
(b) Gaps

Figure 9.8: Runtimes and gaps for the Benders decomposition with E^{grid} and E^{sur} in the SP.



(a) Runtimes (b) Gaps

Figure 9.9: Runtimes and gaps for Branch-and-Check.



(a) Runtimes

(b) Gaps

Figure 9.10: Runtimes and gaps when using a heuristic in the beginning.

is performed and by moving E^{grid} and E^{sur} into the MP the MP gets strengthened by the respective inequalities. However note that having E^{grid} and E^{sur} in the MP makes it harder to solve the MP with a heuristic. Using Branch-and-Check instead of the more classical benders approach increases the runtime or final gap for most of the instance sizes. That is because Gurobi calls the callback function very frequently and often with inferior solutions. Therefore a lot of time is wasted in the subproblem to defer cuts that do not advance the solution process. Using a heuristic for the MP in the beginning worsens the gap for middle-sized instances. However this benders approach is able to find a solution to all of the middle-sized instances while the traditional benders approach does not find a solution to many of those instances. Furthermore, the solutions for large and very large instances are consistently better because the MILP solvers do not find reasonable solutions there. For very large instances the time limit is exceeded before switching to a MILP solver for the MP and therefore no dual bounds are available.

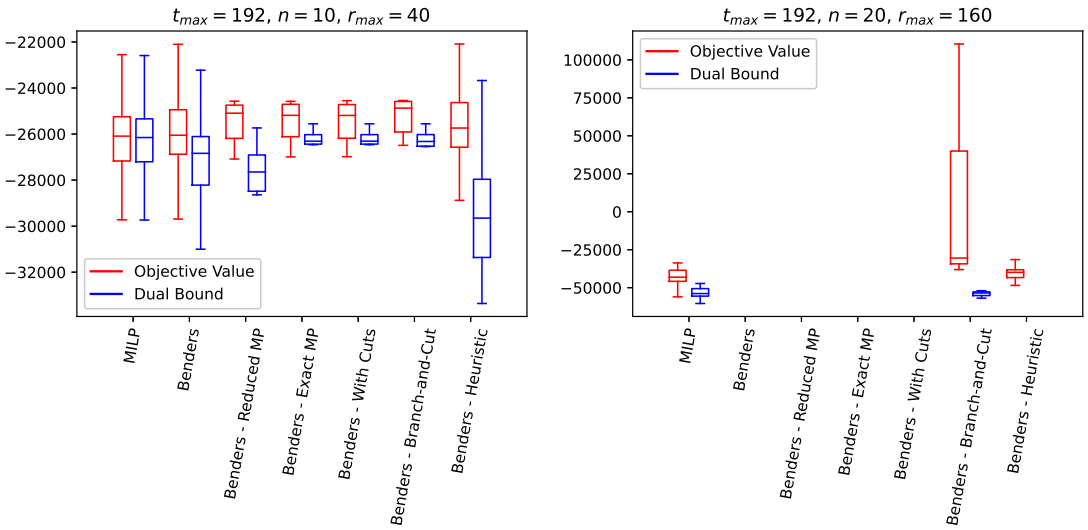
Figure 9.11 shows a comparison between the MILP formulation and the different variants of the Benders decomposition for some particular instance sizes. These plots illustrate that the Benders decomposition in general performs worse for medium sized instances, but performs better for large instances in terms of the objective value of the given solution. This can be observed in particular for the Benders decomposition with the heuristic. Both, objective value and dual bound, are here worse than those of the MILP solver for $(t_{\max}, n, r_{\max}) = (192, 10, 40)$. For $(t_{\max}, n, r_{\max}) = (192, 20, 160)$ this variant does not find a dual bound. However for the instance sizes $(t_{\max}, n, r_{\max}) = (768, 100, 400)$ and $(t_{\max}, n, r_{\max}) = (768, 100, 800)$ it is the only approach that is able to find solutions whose objective values are near the best found dual bounds. Nevertheless, the dual bound is worse for $r_{\max} = 400$ and no dual bounds were found for $r_{\max} = 800$.

9.5 Impact of Scaling E^{sur}

To learn the impact of a lower surplus energy we further consider now test runs on the reduced benchmark instance set in which E^{sur} was scaled with a factor of 0.5. We evaluate the MILP and the Benders decomposition using Gurobi as solver. Figure 9.12 and Figure 9.13 show the resulting plots.

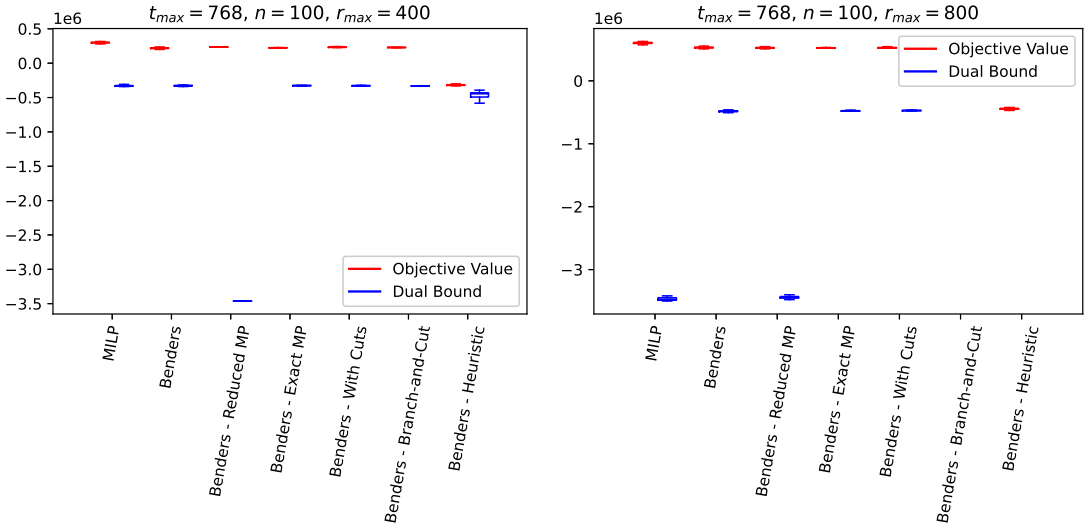
Runtimes and gaps are in both cases similar to those with for the original benchmark instance set. Only for the instance sizes $(t_{\max}, n, r_{\max}) = (192, 5, 20)$ and $(t_{\max}, n, r_{\max}) = (768, 50, 200)$ the adapted instances were harder to solve.

Moreover, Figure 9.14 shows the runtimes needed to solve instances of size $(t_{\max}, n, r_{\max}) = (192, 5, 20)$ for different scaling factors f of E^{sur}. It is apparent that for this instance size the scaling factor hardly affects the difficulty of the instance.



(a) For $t_{\max} = 192$, $n = 10$ and $r_{\max} = 40$

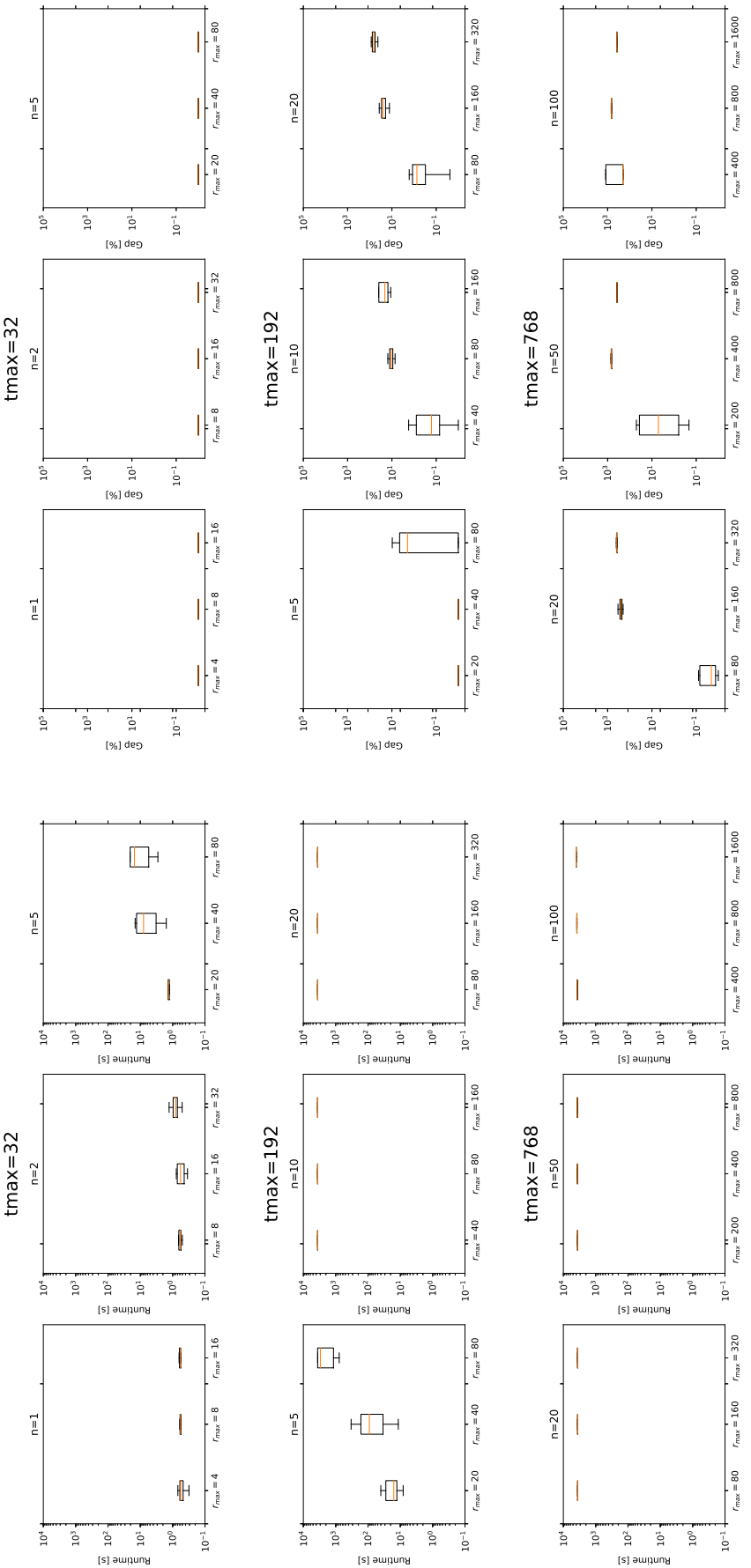
(b) For $t_{\max} = 192$, $n = 20$ and $r_{\max} = 160$



(c) For $t_{\max} = 192$, $n = 10$ and $r_{\max} = 40$

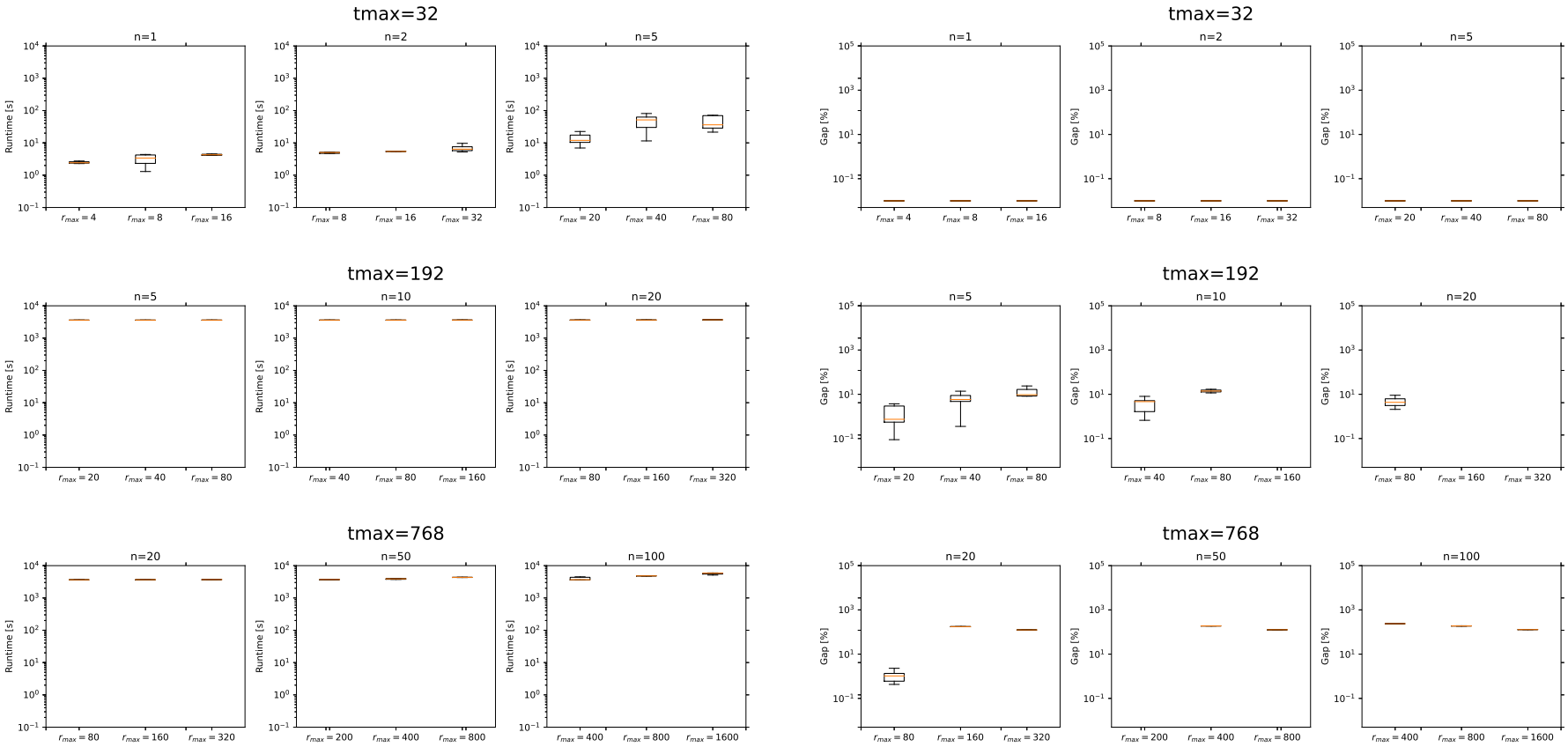
(d) For $t_{\max} = 192$, $n = 20$ and $r_{\max} = 160$

Figure 9.11: Comparison of objective values and dual bounds of the different methods for particular instance sizes.



(a) Runtimes (b) Gaps

Figure 9.12: Runtimes and gaps for the MILP when scaling E^{surmax} by a factor of 0.5.



(a) Runtimes

(b) Gaps

Figure 9.13: Runtimes and gaps for the Benders decomposition when scaling E^{surmax} by a factor of 0.5.

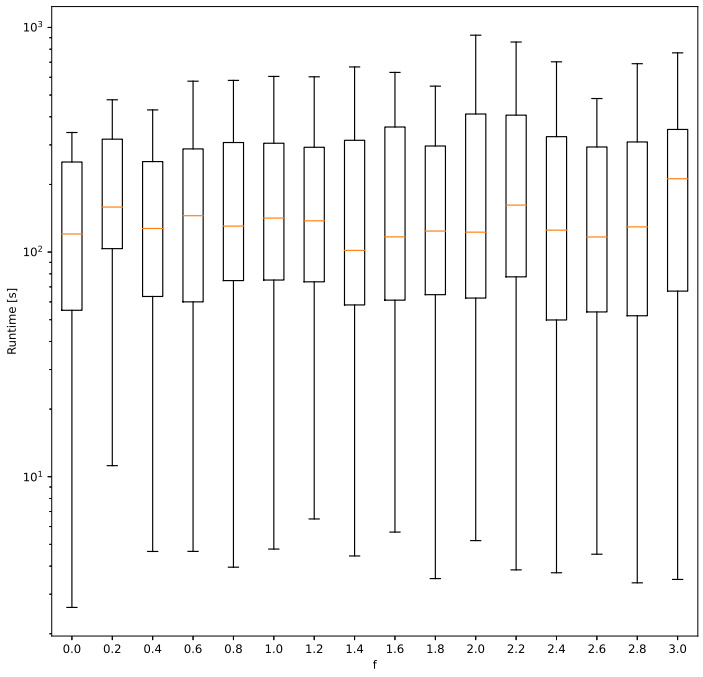
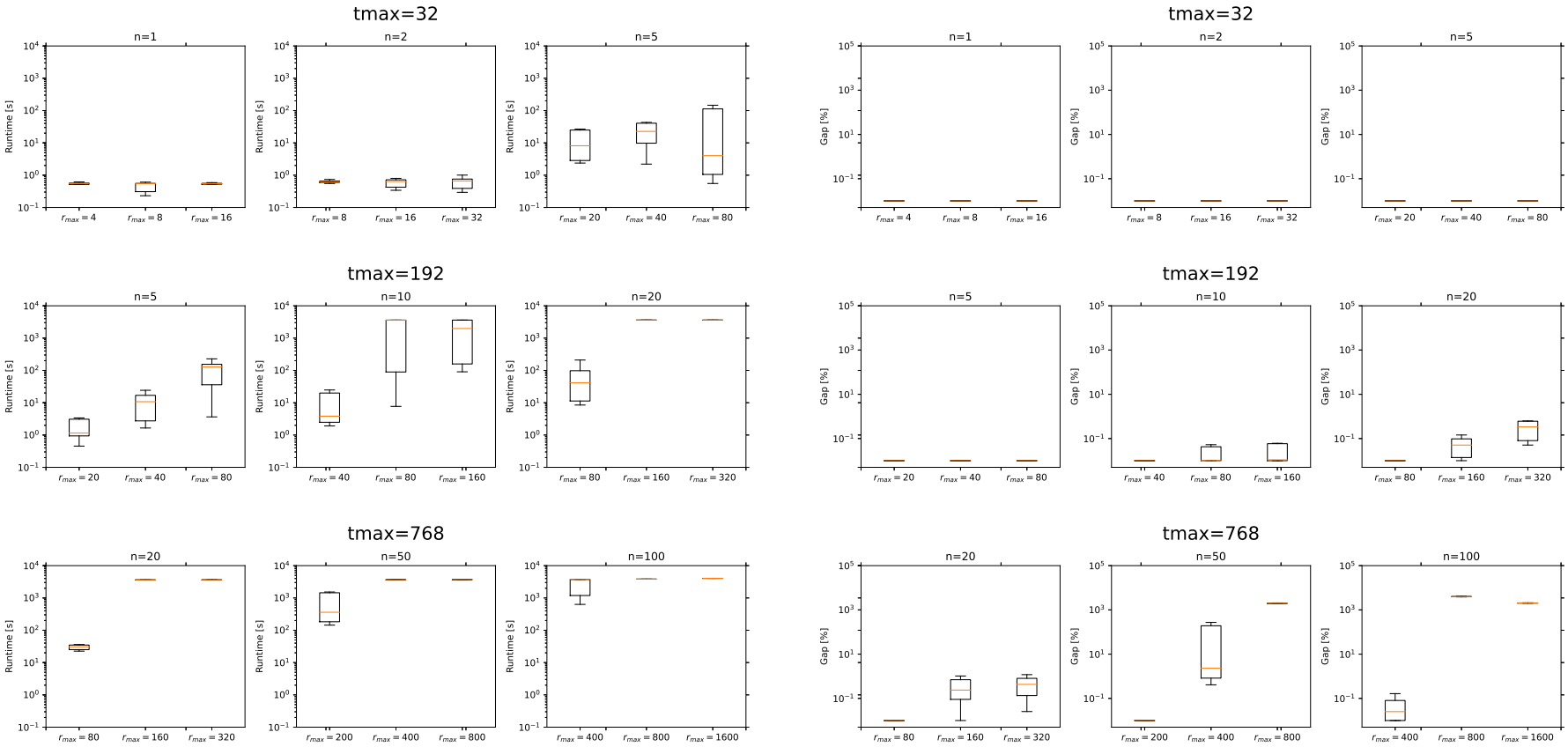


Figure 9.14: Runtimes of the MILP formulation solved with Gurobi for instances of size $t_{\max} = 192$, $n = 5$ and $r_{\max} = 40$ if the available surplus energy is scaled by a factor of f.

9.6 Impact of Reducing E^{res}

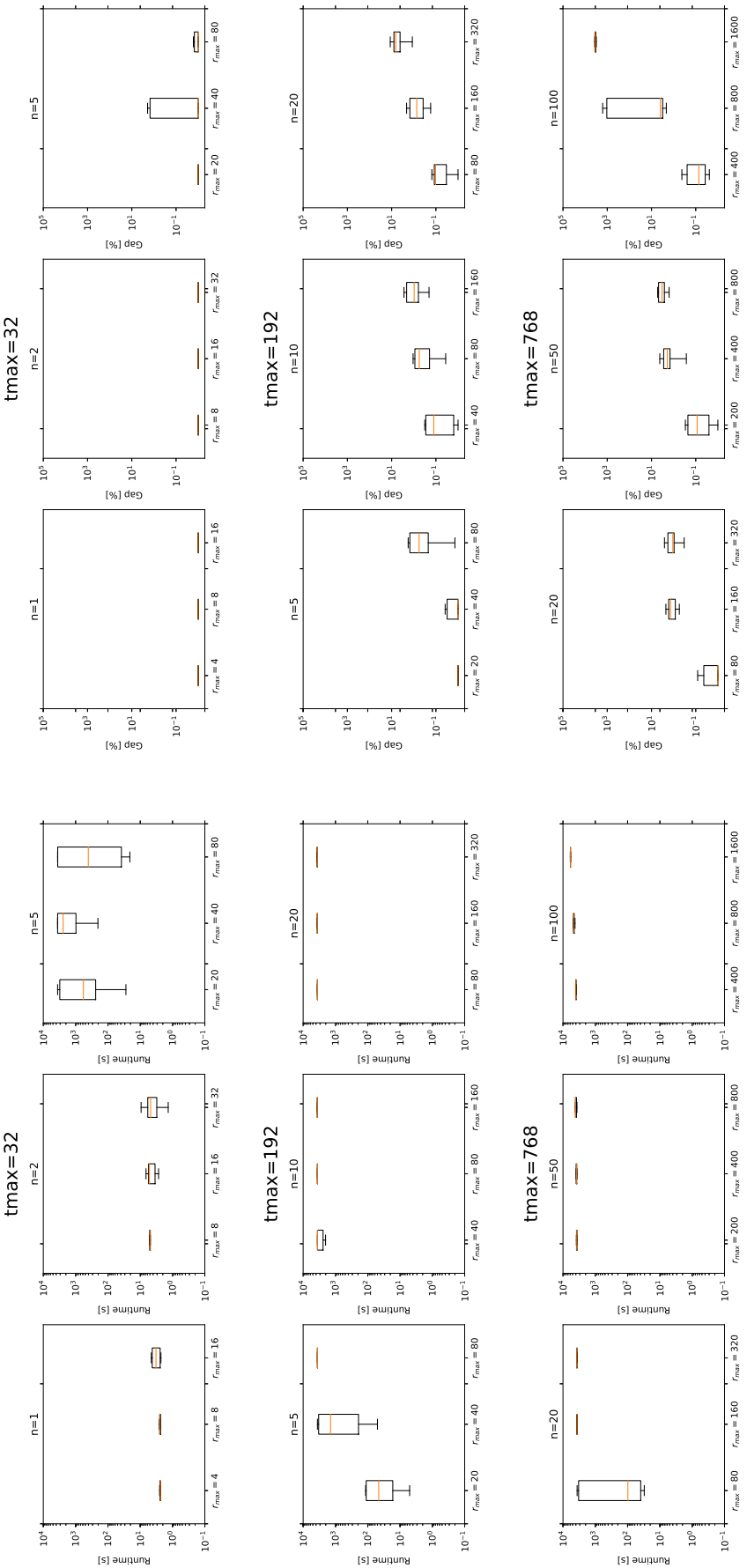
Last but not least, we investigate the impact of requiring smaller energies in the reservations. For this purpose we use the reduced testset in which E^{res} was scaled by 0.15 in comparison to the original benchmark instances. Again, we consider the MILP and the Benders decomposition using Gurobi as solver. Figure 9.15 and Figure 9.16 show the resulting plots. For almost all instance sizes the instances get significantly easier to solve, no matter if using the MILP or the Benders decomposition. The reason seems to be that before scaling some combinations of reservations cannot be assigned to some vehicles, implying additional constraints for the assignment from reservations to vehicles. These additional constraints seem to be substantially less complex in case of the smaller energy requirements in the reservations. The increased freedom seems to make it easier to solve the instance.



(a) Runtimes

(b) Gaps

Figure 9.15: Runtimes and gaps for the MILP when scaling E^{res} by a factor of 0.15.



(a) Runtimes (b) Gaps

Figure 9.16: Runtimes and gaps for the Benders decomposition when scaling E^{res} by a factor of 0.15.

CHAPTER 10

Conclusion

We considered a new variant of the fleet scheduling problem that was not addressed in the scientific literature so far. After the formal definition we proved the problem to be NP-hard. We formulated the problem as a MILP, taking care in the details to achieve a strong linear programming relaxation. Moreover, we suggest an additional class of strengthening inequalities, which prohibit the assignment of subsets of reservations to a vehicle, if that set cannot be assigned to that vehicle without violating charging constraints. Furthermore we proposed a network flow model that can be used to solve the subproblem of the more classical Benders decomposition. However the network flow model together with the network simplex method did not prove to be faster than the linear program formulation when considering a state-of-the-art LP solver like Gurobi.

As an alternative to directly solving this MILP, we proposed a Benders decomposition for the problem. The basic variant of this Benders decomposition was enhanced by five measures:

1. Variables relating to the energy consumption at each timeslot were moved from the SP to the MP, allowing to also move related inequalities from the SP to the MP. This strengthened the linear programming relaxation of the MP.

2. Strengthening inequalities concerning pairs of reservations that cannot be assigned together to vehicles have been statically added to the MP.

3. Instead of iteratively solving MP and SP, the Branch-and-Check paradigm that adds benders cuts as lazy constraints has been applied. This prohibits that the MP is solved from scratch in each iteration.

4. The MP is in the first iterations only solved up to a small remaining gap of 2%. In this way these iterations could be sped up a little without finally losing optimality.

5. The MP is in the first iterations solved by a heuristic. This heuristic is able to find reasonable solutions for the large instances of the MP much faster. By doing this the overall algorithm is able to find good solutions also for the large instances in reasonable time and if switching to a MILP solver for later iterations the algorithm still is exact.

We experimentally compared the Benders decomposition to the MILP formulation on a set of randomly generated test instances and evaluated the effect of the different measures to improve the performance of the Benders decomposition. Finally we investigated if and how the difficulty of instances change when adapting the surplus energy or reducing the energy requirements of the reservations.

For the MILP and one variant of the Benders decomposition the MILP solvers Gurobi and SCIP have been used. In comparison to SCIP the commercial solver Gurobi required in general significantly smaller runtimes and could solve more instances to proven optimality. In terms of runtime for the MILP formulation, Gurobi was up to 10 times faster. Therefore Gurobi has been used to compare the different variants of the Benders decomposition.

As it turns out the MILP formulation performs better than the Benders decomposition on small to medium sized instances. On large instances the solver with the MILP formulation fails to find good feasible solutions while for the Benders decomposition the solver finds better feasible solutions leading to lower gaps, especially when using the heuristic for the MP in the first iterations. For $r_{\max} = 4n$ the Benders decomposition with heuristic finds better solutions than the MILP solver with the MILP formulation, if n is greater or equal 50. For $r_{\max} = 8n$ and $r_{\max} = 16n$ the boundary lies at $n = 20$. Furthermore, moving variables to the MP and not solving the MP to optimality for the first iterations cause an improvement of the performance for some instances. Adding the strengthening inequalities to the MILP/MP did not affect the performance significantly; we assume that the pre-solving and the generic cuts of the considered solvers cover the aspects addressed by these inequalities already well. We advise to use a MILP solver with the MILP formulation for small to medium sized instances and the Benders decomposition with heuristic for large instances.

For the Benders decomposition the times for solving the MP turned out to be the bottleneck as this problem is NP-hard already without any Benders cuts from the subproblem. Therefore we investigated variants that concentrate on solving the MP faster. The easier measure of allowing a gap of 2% for the first iterations of the MP brings slight improvements for the gap. Another measure was to use a heuristic for the MP for the first iterations. Care was taken when choosing the Benders decomposition this variant is based on in order to enable the use of delta evaluation. The resulting variant had worse results for small and medium sized instances. However for large instances it was able to find the best solutions of all compared approaches. Another approach to make the MP faster was Branch-and-Check. Branch-and-Check aimed for avoiding recalculations in the MP by adding benders cuts as lazy constraints. The disadvantage

of this approach is that the callback function is called very regularly and therefore the SP calculates a lot of unnecessary cuts. Therefore this measure worsened the results.

We further investigated the impact on the solving time respectively the gap if scaling the surplus energy or if reducing the reservation energy. As it turns out, scaling the surplus energy hardly affects the difficulty of an instance. On the other side reducing the reservation energies make the instances significantly easier.

Future work may investigate a more finegrained approach for the amount of effort that is used to solve the MP. It seems reasonable to start with little effort, raising it whenever the derived cut does not cut off the found MP solution. Finally the algorithm will solve the MP with the MILP solver to optimality to guarantee the exactness of the approach. When using the MILP solver for the MP, one may adapt the allowed gap from higher to lower values. For the heuristic the allowed runtime may be adapted. It may even be reasonable to first start with the heuristic and switching to the MILP solver with allowed gap upon a condition before finally using the MILP solver to solve the MP to optimality.

Another interesting possibility for future work is to investigate other neighborhood structures for the heuristic search. In particular a larger neighborhood that is searched efficiently seems promising. A reduction to a problem that searches for a circle in a graph could be helpful here. Such a graph can be constructed as follows. Vertices correspond to reservations. There is an edge from reservation r_1 to reservation r_2 if r_2 can be replaced by r_1 without causing an overlap. The edge is labelled with vectors that denote the change that is caused to the left hand side values of the cuts if replacing r_2 with r_1. Cycles then correspond to moves. The feasibility and change in the objective value of a move can be derived from the edge labels. Altough we did not investigate further, we think that it is possible to search the generated neighborhood efficiently in practice either exactly or with a simple heuristic.

Furthermore it may be reasonable to add the cuts corresponding to multiple solutions in each iteration. That would also make it necessary to manage the already added cuts to keep the solver for the MP efficient. A possibility would be to partition the cuts into a set of active and a set of non-active cuts. Active cuts are passed to the heuristic, non-active are not. If a found solution violates a non-active cut, the cut is moved to the active cuts and if an active cut did not make a difference for the solver of the MP for some iterations, it is moved to the non-active cuts.

We formulated the SP as MCFP but found that the Network Simplex algorithm does not perform better than the LP formulation solved with Gurobi. It may be interesting to investigate, whether another MCFP dedicated algorithm performs better. A faster solution to the SP may also be obtained by designing an algorithm that is dedicated to the SP.

List of Figures

1.1 Example for the charging plan of a single EV together with the reservations that have been assigned to that vehicle. 2
1.2 Example for the energy price over time. 2
1.3 Example of the available surplus energy of an instance and the used charging energy for an optimized solution. 3

7.1 Minimum cost flow network for determining a charging plan for given assignments x. Edge capacities are given in green, edge costs in red. No given edge capacity means a capacity of ∞, no given edge cost means a cost of 0. $E^{\max} := \Delta t P^{\max}$. 24

9.1 A solution of a randomly generated instance with $t_{\max} = 192$, $n = 5$ and $r_{\max} = 20$. 37
9.2 Runtimes and gaps for the MILP solved by Gurobi. 45
9.3 Runtimes and gaps for the MILP solved by SCIP. 46
9.4 Runtimes and gaps for the Benders decomposition using Gurobi. 47
9.5 Runtimes and gaps for the Benders decomposition using SCIP. 48
9.6 Runtimes and gaps if static cuts as discussed in chapter 6 are added to the MP. 50
9.7 Runtimes and gaps if the MP is solved to optimality from the beginning. . 51
9.8 Runtimes and gaps for the Benders decomposition with E^{grid} and E^{sur} in the SP. 52
9.9 Runtimes and gaps for Branch-and-Check. 53
9.10 Runtimes and gaps when using a heuristic in the beginning. 54
9.11 Comparison of objective values and dual bounds of the different methods for particular instance sizes. 56
9.12 Runtimes and gaps for the MILP when scaling E^{surmax} by a factor of 0.5. 57
9.13 Runtimes and gaps for the Benders decomposition when scaling E^{surmax} by a factor of 0.5. 58
9.14 Runtimes of the MILP formulation solved with Gurobi for instances of size $t_{\max} = 192$, $n = 5$ and $r_{\max} = 40$ if the available surplus energy is scaled by a factor of f. 59
9.15 Runtimes and gaps for the MILP when scaling E^{res} by a factor of 0.15. . 60

9.16 Runtimes and gaps for the Benders decomposition when scaling E^{res} by a factor of 0.15. 61

List of Tables

9.1 Results of the MILP solved with Gurobi. 38
9.2 Results of the MILP solved with SCIP. 39
9.3 Results of the Benders decomposition with Gurobi. 40
9.4 Results of the Benders decomposition with SCIP. 41
9.5 Results of the Benders decomposition with heuristic and Gurobi. 42
9.6 Comparison of the MILP and Benders approach with Gurobi and SCIP as underlying MILP solver. 43
9.7 Comparison of the Benders decomposition with heuristic with the MILP formulation and the Benders decomposition with extended MP and allowed gap for the MP in the first iterations. 44

List of Algorithms

2.1 VND with the neighborhoods $N_1, \ldots, N_k$ 8

2.2 GVNS with the neighborhoods $N_1, \ldots, N_k$ for VND and $\tilde{N}_1, \ldots, \tilde{N}_m$ for shaking 9

Bibliography

[BEKS17] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[Ben62] Jacques F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

[BGL+16] Georg Brandstätter, Claudio Gambella, Markus Leitner, Enrico Malaguti, Filippo Masini, Jakob Puchinger, Mario Ruthmair, and Daniele Vigo. Overview of optimization problems in electric car-sharing system design and management. In Herbert Dawid, Karl F. Doerner, Gustav Feichtinger, Peter M. Kort, and Andrea Seidl, editors, *Dynamic perspectives on managerial decision making*, pages 441–471. Springer, 2016.

[BK09] Stefan Bunte and Natalia Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317, 2009.

[BWL16] Johannes Betz, Dominick Werner, and Markus Lienkamp. Fleet disposition modeling to maximize utilization of battery electric vehicles in companies with on-site energy generation. *Transportation Research Procedia*, 19:241–257, 2016.

[DHL17] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[EC19] Tomislav Erdelić and Tonči Carić. A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches. *Journal of Advanced Transportation*, 2019:5075671, 2019.

[GAB+20] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, and Katrin Halbig. The SCIP Optimization Suite 7.0. *Optimization Online, ZIB Report*, (10), 2020.

[Gon12] Jacek Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

[Gur21] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[KP12] Suresh Nanda Kumar and Ramasamy Panneerselvam. A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, 4(3):66–74, 2012.

[MO19] Bilal Messaoudi and Ammar Oulamara. Electric bus scheduling and optimal charging. In *International Conference on Computational Logistics*, pages 233–247. Springer, 2019.

[RCGR17] Ragheb Rahmaniani, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2017.

[RPDV20] Marco Rinaldi, Erika Picarelli, Andrea D'Ariano, and Francesco Viti. Mixed-fleet single-terminal bus scheduling problem: Modelling, solution scheme and potential applications. *Omega*, 96:102070, 2020.

[SO17] Ons Sassi and Ammar Oulamara. Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches. *International Journal of Production Research*, 55(2):519–535, 2017.

[VSW69] Richard M Van Slyke and Roger Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM journal on applied mathematics*, 17(4):638–663, 1969.

[Web19] Jeremy Webb. The future of transport: Literature review and overview. *Economic analysis and policy*, 61:1–6, 2019.

[WLDK16] Qinglong Wang, Xue Liu, Jian Du, and Fanxin Kong. Smart charging for electric vehicles: A survey from the algorithmic perspective. *IEEE Communications Surveys & Tutorials*, 18(2):1500–1517, 2016.

[Wol98] Laurence A. Wolsey. *Integer Programming*. Wiley, 1998.

[YLTA15] Duotong Yang, Haniph Latchman, Dave Tingling, and Anim Adrian Amarsingh. Design and return on investment analysis of residential solar photovoltaic systems. *IEEE Potentials*, 34(4):11–17, 2015.