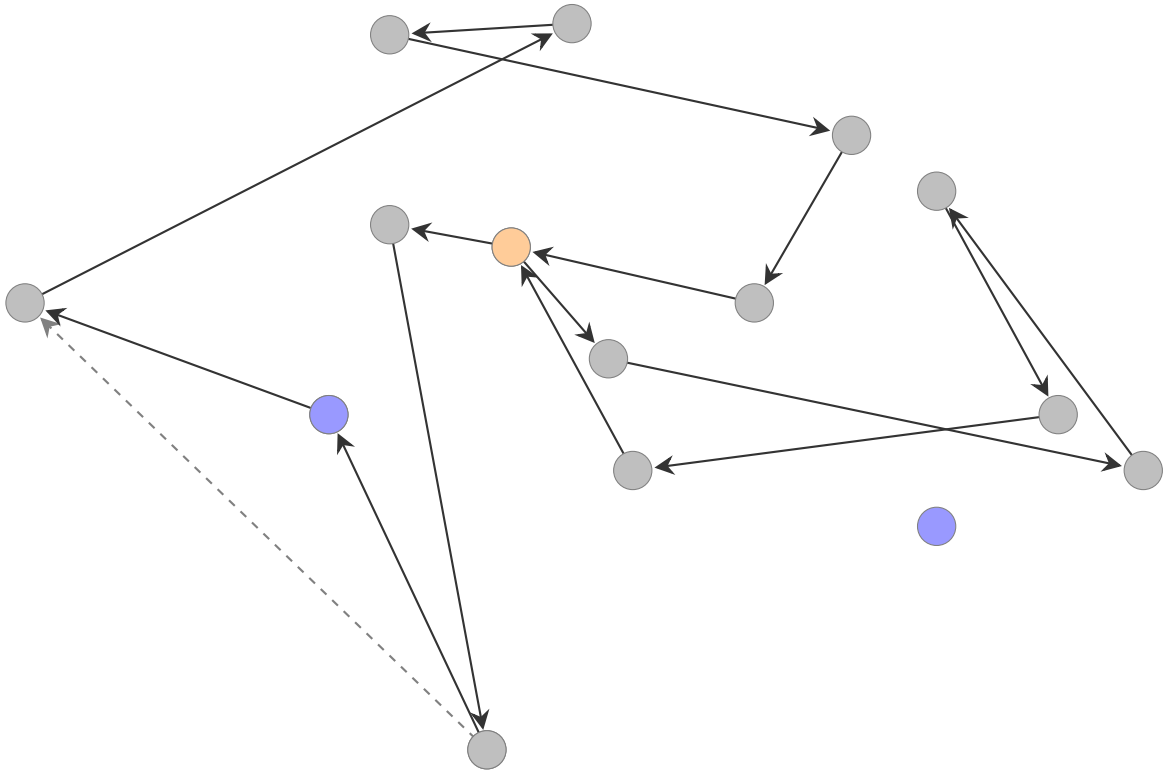


Battery Dimensioning for Hybrid Vehicles in a Routing Application

Generalised Duality and Logic-Based Benders Decomposition

Master's thesis in Engineering Mathematics and Computational Science

JONAS KINDSTRAND & LINUS NORDGREN

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's Thesis 2018

Battery Dimensioning for Hybrid Vehicles in a Routing Application

Generalised Duality and Logic-Based Benders Decomposition

Authors:
Jonas Kindstrand
Linus Nordgren

Supervisor:
Ann-Brith Strömberg

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Battery Dimensioning for Hybrid Vehicles in a Routing Application
An Application of Logic-Based Benders Decomposition
Jonas Kindstrand & Linus Nordgren

Supervisor & examiner: Ann-Brith Strömberg

Master's thesis 2018
Department of Mathematical Sciences
Chalmers University of Technology and
University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31-772 1000

Cover: *An illustration of a solution to a hybrid vehicle routing problem with two vehicles. The depot node is orange and the dashed arrow indicates the freedom to not take the detour through the blue recharge station node.*

Typeset in LaTeX
Gothenburg, Sweden 2018

Battery Dimensioning for Hybrid Vehicles in a Routing Application
An Application of Logic-Based Benders Decomposition
Jonas Kindstrand & Linus Nordgren

Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

The Vehicle Routing Problem (VRP), which is defined as to find optimal routes for a fleet of delivery vehicles to various customers, constitute an important class of combinatorial optimisation problems of both practical and theoretical interest. Among the various flavours of VRP, this report specifically focuses on a case with hybrid vehicles with two fuel types, with the goal of finding the optimal battery sizes which minimises the total cost.

We present an exact solution method using a generalised Benders decomposition method, known as logic-based Benders decomposition. In this method, the subproblems are generalised to mixed integer linear optimisation problems. The master problem is a simple routing problem, while the subproblems concern resource constraints and battery types.

The mixed integer master problem is solved by branch-and-bound, and lower bounds are generated from the solution tree. Only small instances of up to 14 customers are solved to optimality, and the performance of our algorithm is compared with more direct solution methods. As it is, the method is slower than solving the full problem directly, and further work is needed to make it competitive.

Keywords: Vehicle routing problem (VRP), hybrid vehicles, battery capacity, logic-based Benders decomposition (LBBD), branch-and-bound

Contents

1

Introduction

Vehicle Routing Problems, abbreviated throughout this text as VRPs, constitute a well-studied class of problems in combinatorial optimisation and operational research. Generally, the problem consists of a road network, a set of customers with demands interspersed in the road network, and a vehicle fleet to service those demands. The goal is typically to minimise the total distance that the vehicles travel. Multiple variations of this problem have been investigated; usually some additional constraints are added to the problem to more accurately model real-life routing applications. Some well-studied variations add limited load capacities to vehicles [32], time windows for deliveries [46], synchronisation constraints [23], or the need to visit some customers in a particular order [6].

Mathematically, a VRP is represented as a weighted graph; customers, the depot, and charging stations are represented as nodes, and roads are represented as arcs. A route in the road network corresponds to a path in the graph, and road lengths are represented by the arc weights. The problem is then to find routes emerging from the depot node such that all customer nodes are included in at least one route, and to minimise the total length of the combined routes while doing so. As alluded to earlier, additional constraints may be put on this minimisation problem to represent the amount of goods to be picked up at each customer along with vehicle load capacities, time constrained nodes that must be visited during certain time windows, and so on.

From an environmental point of view, minimising the total fuel use of the fleet is of great interest. Normally this is not meaningfully different mathematically from minimising total distance travelled, since each arc is assigned a constant cost in both cases. But if instead the problem contains two different fuel types and the total combined fuel cost is minimised, this is qualitatively a different problem compared to the single fuel case. For example, this is the case with hybrid vehicles which can use both conventional fuel and electricity. Hybrid vehicles are unable to drive long distances on electricity before recharging. Any VRP model containing hybrid vehicles needs to consider how this changes the problem. Normally fuel stations can be left out of the model since the time to refuel is negligible, but this is not the case for electric vehicles. Driving on electricity is typically cheaper, which means that it might be beneficial for a vehicle to take a longer route if this allows a stop at a charging station and less use of conventional fuel. All of these complications must be reflected in a solution method.

Solution methods are an important aspect of VRP type problems, since the problems are often computationally heavy due to the large number of constraints and variables needed to model them. In practice, both exact and heuristic methods are used. Branch-and-bound and variants thereof are some of the most common exact solution methods. A search of literature revealed few studies which solve VRPs with hybrid vehicles, and especially few which employed exact methods.

For more complex VRP models, branch-and-bound might not be enough, due to the fact that the problems become too large. Column generation (originally introduced in [28]) is a type of decomposition method which has been found the be effective at solving these types of problems. Another one is Benders decomposition which was first proposed by Benders [5] in

1962. It is a method for decomposing certain kinds of complicated optimisation problems. Since the original suggestion of the method by Benders, it has been generalised to work for a larger class of problems. One such generalisation is the logic-based Benders decomposition algorithm, abbreviated as LBBD, presented by Hooker and Dawande [22] which is based on ideas from computational logic. One advantage of these kinds of decomposition methods is that by splitting the problem into several parts, solution methods can be tailored to each part of the problem.

1.1 Problem Formulation & Project Aim

The problem studied in this project is a modification of the VRP with load capacity and time window constraints for hybrid vehicles. Multiples vehicles have to be routed out of one depot in such a way that all customers are visited by exactly one vehicle. Each customer has a time window in which they can be serviced, and requires a certain amount of cargo. All pairs of customers are connected to each other and to the depot by road links. For simplicity, the vehicles are modelled to have the same maximum cargo capacity.

The vehicles use two different fuel types: Conventional fuel and electricity. It is assumed that the conventional fuel capacities are unlimited, since either refuelling takes a negligible amount of time or the distances are sufficiently small such that refuelling is not needed. Recharging batteries, however, takes a significant amount of time. Therefore maximum battery capacities are included in the model together with the location of recharge stations and recharge times. Moreover, it is assumed that the cost for each unit of distance is significantly higher when using conventional fuel than when using electricity, and that recharge stations are connected to every customer and to the depot. In addition, it is assumed that the vehicles always wait for a full recharge before leaving the recharge stations.

Customers, the depot, and the recharge stations are modelled as nodes in a graph, with arcs representing routes between nodes. Combined with the constraints mentioned above, the problem can then be written as a minimisation problem. A so called three-index vehicle flow formulation [18] is used as the model, and the different resource constraints are easily added to this model. The full mathematical model with all constraints is presented in Chapter 4.

This model is solved using a Benders decomposition algorithm, which splits the problem into two layers. These are referred to as the master problem and the subproblem. The master problem is a simplified routing problem without many of the constraints from the full formulation. Cargo capacity, time windows, and battery constraints are included only in the subproblem, which turns out be decomposable into three corresponding parts. A consequence of this particular split is that the subproblem is not of the kind typically considered in the classical Benders decomposition algorithm, since it includes binary variables. This necessitates a different approach, and a so called logic-based Benders decomposition algorithm will be used instead, which has the following requirements:

i. A proof scheme has to be devised for the solution to the subproblem.

ii. A method for generating sensitivity information from the solution to the subproblem has to be established.

The aim of this project is to formulate and implement a logic-based Benders decomposition algorithm, and in particular the two requirements above, for the problem outlined in this section. Limitations to this problem are discussed in the following section.

1.2 Limitations

To simplify the model, it is assumed that vehicles are instantaneously recharged at recharge stations. The technical reason for this is so that the variables that model time and charge should not be directly dependent on each other. Further, it is possible to consider a problem with several depot nodes, but only one depot will be used in the model.

The master problem from the logic-based Benders decomposition algorithm is computationally quite difficult to solve, but this is not the primary focus of this project. Instead, the focus is on the theoretical aspect of how to solve and generate sensitivity information from the subproblem, which is a difficult problem in its own right. The subproblem consists of three parts, and the focus is on the battery capacity part. The time window and cargo capacity parts are simplified variants of the battery part, and are thus not as interesting in the context of the focus chosen.

Finally, there is a natural limit imposed by the size of problem instances a computer can feasibly solve in a reasonable time frame with the available hardware. For this project only instances with no more than 15 customers have been fully solved.

1.3 Outline of Thesis

Following this introductory chapter, Chapter 2 contains a literature review, focusing on the vehicle routing problem and Benders decomposition. Chapter 3 is a presentation of the necessary theoretical concepts used later in the report: basic mixed-integer linear programming (MILP), classical Benders decomposition for linear programs (LP), and logic-based Benders decomposition for MILP. In Chapter 4, we present the mathematical formulation of our VRP as one large MILP. Chapter 5 describes how this model is partitioned in order for a logic-based Benders decomposition scheme to be applied, and discusses the master problem and subproblems which arise. The chapter also contains a brief description of our computer implementation of the resulting algorithm. Chapter 6 contains our results, along with a discussion about the efficacy of our method. Finally, in Chapter 7 we discuss concrete directions for further research, both in terms of performance tuning and in broader changes to our method. The report ends with our conclusions in Section 7.3.

2
Literature study

In the following section a brief overview of existing literature on different VRP problems is presented together with some common methods used for solving them. Section 2.2 continues by focussing on the Benders decomposition algorithm, and extensions to it.

2.1 Overview of Vehicle Routing Problems

The vehicle routing problem was originally introduced as 'The Truck Dispatching Problem' by Dantzig and Ramser [11] as a generalisation of the so called Travelling Salesman Problem (TSP), which is the problem of finding the shortest path that visits all the nodes in a graph. Their VRP was distinct from a multiple TSP by the introduction of carrying capacities. Although the concept of an NP problem (we will not be needing concepts from complexity theory, but an overview can be found in [1]) had at this point not yet been named or even formalised, the computational intractability of the travelling salesman problem for even relatively small problem instances was well recognised. As a generalisation, the VRP inherits all the difficulties of the TSP, and hence the method of Dantzig and Ramser was heuristic and could only find near-best solutions in some scenarios. An example of a VRP route is shown in Figure 2.1.

An early exact method for the VRP appeared in Christofides and Eilon [7], in the form of a so called branch-and-bound algorithm. Exact methods based on so called cutting planes had however been described for the TSP much earlier [12]. These two approaches, branching and cutting planes, are the core of most exact methods. A combination of the two, branch-and-cut, forms the basis for several general purpose integer optimisation solvers, including IBM ILOG CPLEX Optimization Studio [27], Gurobi [19], and COIN-OR [47].

A survey of different exact methods was done by Laporte and Nobert [34], who present the state of the art up until the late eighties. Methods discussed included direct tree search, dynamic programming, and methods using integer linear programming (ILP) formulations. Most of the ILP was based on vehicle flow formulations, where binary variables were used to indicate if a vehicle travels between two nodes. Two different types were used: three-index formulations and two-indexed formulations. The three-index formulation has one binary variable for each combination of vehicle, origin, and destination, whereas the two-indexed formulation has integer variables counting how many vehicles travel from each origin to each destination. The three-indexed formulation was first introduced by Golden, Magnanti, and Nguyen [18], and the two-indexed formulation was presented by Laporte and Nobert [32] together with a commodity flow model. Letchford and Salazar-González [37] give a survey of formulations for the capacitated VRP, including the three-index formulations. A set partitioning formulation of the capacitated vehicle routing problem together with an exact algorithm was presented in [2]. More recent surveys of exact methods are Toth and Vigo [49] and Laporte [33].

Solomon [46] presented a variant of the VRP with time window constraints, together with

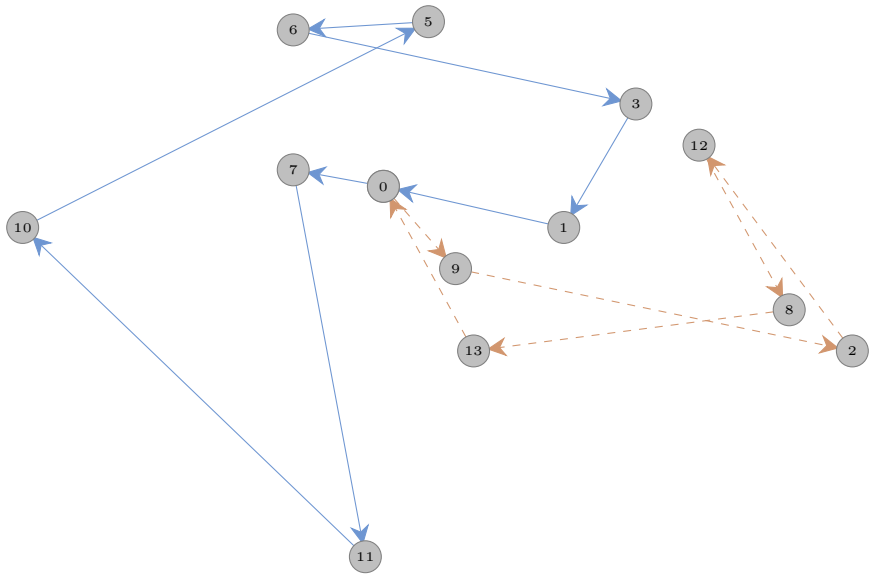


Figure 2.1: Example of route for a VRP with two vehicles (red and blue) and one depot (0). Every node except for the depot is a customer that must be visited by at least one vehicle. All vehicles start and end in the depot.

heuristic solution methods and data sets of test problem instances. Another variant of the VRP is the electrical vehicle routing problem in which the vehicles have a limited range before they have to recharge at a recharge station. Several different models for this problem have been proposed. Lin, Zhou, and Wolfson [38] presented a version where battery consumption was affected by vehicle loads. They found that the effect of vehicle load on routing strategy cannot be ignored. A branch-and-price-and-cut algorithm was presented in [14] as a method for solving the electrical VRP with time windows, where a labelling algorithm was used to solve the subproblems.

An extension to the electrical VRP is to consider vehicles that can use both conventional fuel and electrical charge. A 'Hybrid Vehicle Routing problem' was introduced by Mancini [39], in which a large neighbourhood search is used to solve a VRP where a vehicle switches to the more expensive conventional fuel once the battery is empty.

2.2 Benders Decomposition and Applications to VRP

The (classical) Benders decomposition algorithm was introduced by Benders [5] in 1962. In broad terms, the method partitions an optimisation problem into a master problem and one or more LP subproblems. The master problem is solved and yields trial values which are fed to the subproblems. The subproblems are solved, and using LP duality theory, constraints (commonly called cuts) are generated and fed back to the master problem, which is then re-solved, leading to new trial values for the subproblem. This back-and-forth continues until a solution to the original problem has been found and verified. More explicitly, classical Benders cuts are obtained by

solving the dual of the subproblem, from which a bound on the optimal value of the subproblem is deduced. If the subproblem is an ILP or MILP, the LP dual is however not defined. An LP can still be obtained by relaxing the integrality constraints, and the bound from this relaxed problem is indeed valid for the original problem. Even so, if the duality gap is large, this bound will not be very strong [43], and results in slow convergence if used in the Benders decomposition method. Geoffrion [17] extended the Benders algorithm to non-linear convex subproblems.

Another extension is presented by Hooker [20], where an inference dual is defined for very general optimisation problems. Solving this dual amounts to examining the results of a primal method. The inference dual is used in [22] to develop a generalised Benders decomposition method, resulting in a new method named logic-based Benders decomposition. This method requires a proof schema, and that a method of generating Benders cuts be devised for each class of optimisation problems. Hooker and Dawande [22] and Hooker and Ottosson [24] applied this to binary programming problems by interpreting the branch-and-bound tree as a proof of optimality. This was then used to perform sensitivity analysis of the problem. The analysis was extended to branch-and-bound methods which included cutting planes.

Hooker [21] studied a planning and scheduling problem where the problem naturally decomposes into an assignment portion and a scheduling portion. The assignment problem was solved as an MILP model, and the scheduling problem was solved as a constraint programming model. A logic-based Benders decomposition algorithm was used to link these problems, and it was found that this gave a speedup over other state of the art methods. Coban and Hooker [9] showed that for pure scheduling problems, a logic-based Benders decomposition algorithm performed well when the problem size scaled up. Hooker [26] contains several examples of inference rules for different classes of problems, together with a so called branching dual which obtains dual information by inspection of a solution tree formed by branching on the primal problem.

Kloimüllner and Raidl [30] applied a logic-based Benders decomposition algorithm to a 'Balancing Bike Sharing Systems problem'. The problem was to balance bikes in a public bike sharing network to prevent too many rental stations running full or empty. They found that using a logic-based Benders decomposition algorithm was an improvement over previous exact methods.

A branch-and-check method was introduced in Thorsteinsson [48], which they describe as essentially a branch-and-cut method in which a Benders subproblem is used as what they call the separation subproblem. This combines the ideas from the logic-based Benders decomposition algorithm with a branching scheme, so that subproblems of almost any form can be considered. They present a branch-and-check method for solving a capacitated VRP with time windows. Lam and Hentenryck [31] considers a branch-and-price-and-check method for solving a VRP with location congestion. Branch-and-price is used to solve the underlying VRP, and constraint programming to check the feasibility of the location congestion subproblem. It contains a nice overview of different hybrid methods for solving VRPs.

Riazi, Seatzu, Wigström, and Lennartson [42] proposed a logic-based Benders decomposition algorithm for solving the heterogeneous multi-vehicle routing problem, a extension of the multi-TSP. The nodes are assigned to different vehicles in a master problem and each subproblem consisted of solving a TSP for that vehicle. These subproblems were solved using a standard TSP.

Fischetti, Ljubić, and Sinnl [15, 16] describes the basic steps for designing a Benders decomposition approach to be embedded in a modern MILP solver.

3

Theory

In this chapter, we first introduce some preliminary theory of linear programming (LP) and integer linear programming (ILP) in Section 3.1. This theory forms a necessary basis for understanding the methods presented in Sections 3.2, 3.3, and 3.5. Section 3.2 introduces the concept of inference duals. This is then used in Section 3.3 to generalise the classical Benders decomposition algorithm. Section 3.4 introduces notation for logical clauses and the resolution algorithm. Finally, in Section 3.5, we fill in the details necessary to formulate a logic-based Benders decomposition algorithm for a problem with mixed-integer linear programming (MILP) subproblems.

We adopt the convention that some vectors are in the dual (i.e. row) vector space, and denote this as $v \in V^\top$. This eliminates the need to write explicit transposes of vectors.

3.1 Linear and Integer Linear Programming Preliminaries

The theory of (continuous) LP forms the basis for the usual treatment of exact methods for ILP and MILP. This section begins with a short review of this theory and describes its application to the branch-and-bound method for MILP. The section ends with a description of the Benders decomposition method for LP.

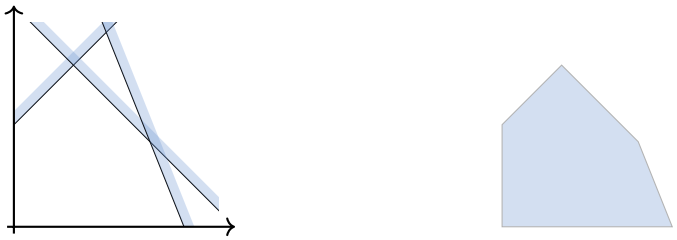
3.1.1 Linear Programming, Duality and Sensitivity Analysis

Consider a general LP,

$$\min_x \quad ax, \tag{3.1.1a}$$
$$\text{s.t.} \quad Ax \geq c, \tag{3.1.1b}$$
$$x \geq 0, \tag{3.1.1c}$$

where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $a \in (\mathbb{R}^n)^\top$ and $c \in \mathbb{R}^m$. The feasible set defined by the constraint (3.1.1b) together with the domain defined by constraint (3.1.1c) define a convex polyhedron. An example of a polyhedron defined in this way is shown in Figure 3.1. The shape of the region together with the fact that the objective function is linear guarantees that an optimal solution can always be found in a vertex, of which there are a finite number, provided that the polyhedron is not unbounded. Since a solution can always be found in one of these vertices, one solution method might be to check the objective function value for each vertex. One of the most commonly used algorithms for solving LP is the simplex method. The simplex method is a refinement of the idea of checking vertices which draws conclusions from how the objective function value changes when the values of the variables change.

Consider now a vector of non-negative multipliers $u \in (\mathbb{R}^m)^\top$, where each multiplier is associated with one row of the constraint matrix A. Since $u \geq 0$, multiplying u with both sides



(a) (b)

Figure 3.1: The feasible set together with the domain define a convex polyhedron. Figure 3.1a shows the feasible set defined by a linear inequalities and the domain $\mathbb{R}^2_{\geq 0}$. Figure 3.1b shows the convex polyhedron.

of (3.1.1b) implies a new inequality $uAx \geq uc$. Assuming that $uA \leq a$ yields the bound $uc \leq ax$ on the optimal solution of the model (3.1.1). Since this bound is valid for all $u \geq 0$ satisfying the inequality $uA \leq a$, it is also valid for the u maximising uc. This new maximisation problem is referred to as the dual problem, and the original problem as the primal problem. We write it as

$$\max_{u} \quad uc, \tag{3.1.2a}$$
$$\text{s.t.} \quad uA \leq a, \tag{3.1.2b}$$
$$u \geq 0. \tag{3.1.2c}$$

From the construction of the dual problem, it is apparent that the optimal value of the primal problem is bounded from below by the optimal value of the dual problem. This property is referred to as *weak duality*. In fact, an even stronger statement holds for LP. This is the content of the following theorem due to von Neumann, which is presented here without proof. The crucial property that is used to prove Theorem 3.1.1 is some version of a hyperplane separation theorem, typically Farkas' lemma. Several proofs can be found in [13].

Theorem 3.1.1 (Strong Duality)**.** *If the primal problem* (3.1.1) *has an optimal solution, then so does the dual problem* (3.1.2)*, and their optimal objective function values coincide.*

The LP dual can be used to perform sensitivity analysis on the primal LP model (3.1.1). Let u^* be the optimal solution to the dual problem (3.1.2). If the vector c is perturbed to $c + \Delta c$, u^* remains dual feasible, since the feasible set has not changed for the dual problem. This gives a dual objective function value $u^*(c + \Delta c)$. Since u^* is dual feasible, the optimal dual value is bounded from below by $u^*(c + \Delta c)$. From Theorem 3.1.1 it follows that the analogue is true for the optimal value of the primal problem. This gives us a lower bound $ax \geq u^*(c + \Delta c)$ on the optimal value of the perturbed primal problem.

3.1.2 Mixed Integer Linear Programming and Branch-and-Bound

A general MILP can be stated as

$$\begin{aligned} \min \quad & cx, \\ \text{s.t.} \quad & Ax \geq a, \\ & x \geq 0, \\ & x \in \mathbb{Z}^k \times \mathbb{R}^{n-k}, \end{aligned}$$

with $c \in (\mathbb{R}^n)^\top$, $A \in \mathbb{R}^{m \times n}$, and $a \in \mathbb{R}^n$. This class of optimisation problems contains LP, binary linear programs, and ILP as special cases. Moreover, since general MILP are known to be NP-hard, any NP-complete problem can in fact be reformulated as an MILP. (For general complexity theory, see, e.g., [1].) As a consequence, any exact method for general MILP must be able to handle a very broad class of problems. In practice, this means that it is typically undesirable to use an entirely general method, and consequently more problem specific methods are usually developed. This section is concerned with branch-and-bound—one of the most prevalent design schemes for such methods.

The branch-and-bound paradigm of algorithms is the backbone of many exact methods for MILP. At the core of the method is the idea of implicit enumeration of the feasible set (also referred to as the search space in the pure integer case). Consider for a moment a pure binary program in n variables $x_1, \ldots, x_n$ with objective function f. A divide-and-conquer approach to this problem might be to choose one of the variables, say x_1, and to split the search space into two halves corresponding to the trial assignments $x_1 = 0$ and $x_1 = 1$. This process is called branching. If either of these smaller search spaces represents an infeasible problem, that branch need not be considered further. However, for each half that was feasible, choose a new variable, say x_2, and again make trial assignments $x_2 = 0$ and $x_2 = 1$. A search tree is built by iterating this procedure. Iteration continues until all assignment options have been explicitly considered or deemed infeasible. After the process terminates, all feasible points can be found as leaf nodes of the search tree, along with the leaf nodes that were deemed infeasible. The optimal variable assignment $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ is then the one with the lowest objective function value, $f(x^*)$, among all these feasible leaf nodes. In general, a branching procedure like this can be performed by partitioning the variables and forming one new branch per set in the partition.

Branching alone only amounts to a method for explicit enumeration of the search space; the objective function is evaluated at every feasible point. To improve on this, the method is augmented with the help of bounds on the objective function value. Every feasible point found by branching gives an upper bound on the objective function. By relaxing the condition $x \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ to $x \in \mathbb{R}^n$ in the node problems, LP are obtained. Solving this LP relaxation of a node problem gives a lower bound on the objective function for that branch. If at any point such a relaxed node problem is solved and the optimal solution is found to exceed the best known upper bound, then that branch need not be considered, and can be discarded from the problem. This way, parts of the search space are not considered explicitly and instead removed by bounding, and as such branch-and-bound is said to perform *implicit enumeration* of the search space. The general MILP case, where x may contain both discrete and continuous parts, does not represent a problem for branch-and-bound. Branching need only be performed on the discrete part of x, and since LP relaxations are solved, the continuous part is automatically handled.

3.1.3 A Branch-and-Bound Example

In this section, consider a simple branch-and-bound application to the ILP

$$\begin{aligned} \min \quad & -5x_1 - 8x_2, && \text{(3.1.4a)} \\ \text{s.t.} \quad & x_1 + x_2 \leq 6, && \text{(3.1.4b)} \\ & 5x_1 + 9x_2 \leq 45, && \text{(3.1.4c)} \\ & x_1, x_2 \in \mathbb{Z}_+. && \text{(3.1.4d)} \end{aligned}$$

We will build the solution tree in a series of iterations, solve an LP relaxation in each one, and decide whether to branch or not in each individual node using the information from the LP relaxation.

Iteration 0 (L_0): Let L_0 be the LP relaxation of problem (3.1.4). The relaxation has an optimal solution $x^0 = (2.25, 3.75)$ with objective function value $z^0 = f(x^0) = -41.25$. Both x_1^0 and x_2^0 are fractional, so we pick (arbitrarily) x_1^0 to branch on. We form one new LP L_1 by adding the constraint $x_1 \leq 2$ to L_0, and another new LP L_2 by instead adding the constraint $x_1 \geq 3$ to L_0. These two nodes constitute the new branches in the branches in the branching tree.

Iteration 1 ($L_1 = L_0 \wedge (x_1 \leq 2)$): The solution to the LP L_1 is $x^1 = (2, 3.\overline{888})$ with the objective function value $z^1 = -41.\overline{111}$. Only x_2^1 is fractional, so it is the branching variable. We form the problem L_3 by adding the constraint $x_2 \geq 3$, and the problem L_4 by adding the constraint $x_2 \geq 4$.

Iteration 2 ($L_3 = L_1 \wedge (x_2 \leq 3)$): The solution to the LP L_3 is $x^3 = (2, 3)$ with the objective function value $z^3 = -34$. Since x^3 is feasible, there's no reason to branch, and $z^* \leq z^3$ is our current best upper bound on the optimal value of the objective function.

Iteration 3 ($L_4 = L_1 \wedge (x_2 \geq 4)$): The solution to the LP L_3 is $x^3 = (1.8, 4)$ with the objective function value $z^3 = -41$. Only x_1^3 is fractional, so it is the branching variable. We form the problem L_5 by adding the constraint $x_1 \leq 1$ and the problem L_6 by adding the constraint $x_1 \geq 2$.

Iteration 4 ($L_5 = L_4 \wedge (x_1 \leq 1)$): The solution to the LP L_5 is $x^5 = (1, 4.\overline{444})$ with the objective function value $z^5 = -40.\overline{555}$. Only x_2^5 is fractional, so it is the branching variable. Form the problem L_7 by adding the constraint $x_2 \leq 4$ and the problem L_8 by adding the constraint $x_2 \geq 5$.

Iteration 5 ($L_7 = L_5 \wedge (x_2 \leq 4)$): The solution to the LP L_7 is $x^7 = (1, 4)$ with the objective function value $z^7 = -37$. Since x^7 is feasible, there is no need to branch, and since $z^7 < z^3$, we obtain a new best upper bound $z^* < z^7$ on the optimal objective function value.

Iteration 6 ($L_8 = L_5 \wedge (x_2 \geq 5)$): The solution to the LP L_8 is $x^8 = (0, 5)$ with the objective function value $z^8 = -40$. Since x^8 is feasible, there's no need to branch, and since $z^8 < z^7$, we obtain a new best upper bound $z^* < z^8$ on the optimal objective function value.

Iteration 7 ($L_6 = L_4 \wedge (x_1 \geq 2)$): The LP is infeasible.

Iteration 8 ($L_2 = L_1 \wedge (x_2 \geq 4)$): The solution to the LP L_2 is $x^2 = (3, 3)$ with the objective function value $z^2 = -39$. Since $z^2 > z^8$, there is no need to branch.

Figure 3.2 depicts the branch-and-bound tree for this example graphically. The tree proves that the optimal solution is $z^* = -40$, and that it is attained at $x^* = (0, 5)$. Only nine points are explicitly considered by the algorithm, while a complete enumeration would have to consider at least the 25 points that make up the feasible set. Remaining points are only implicitly considered by means of the bounding process, which is why the method is often described as an implicit enumeration.

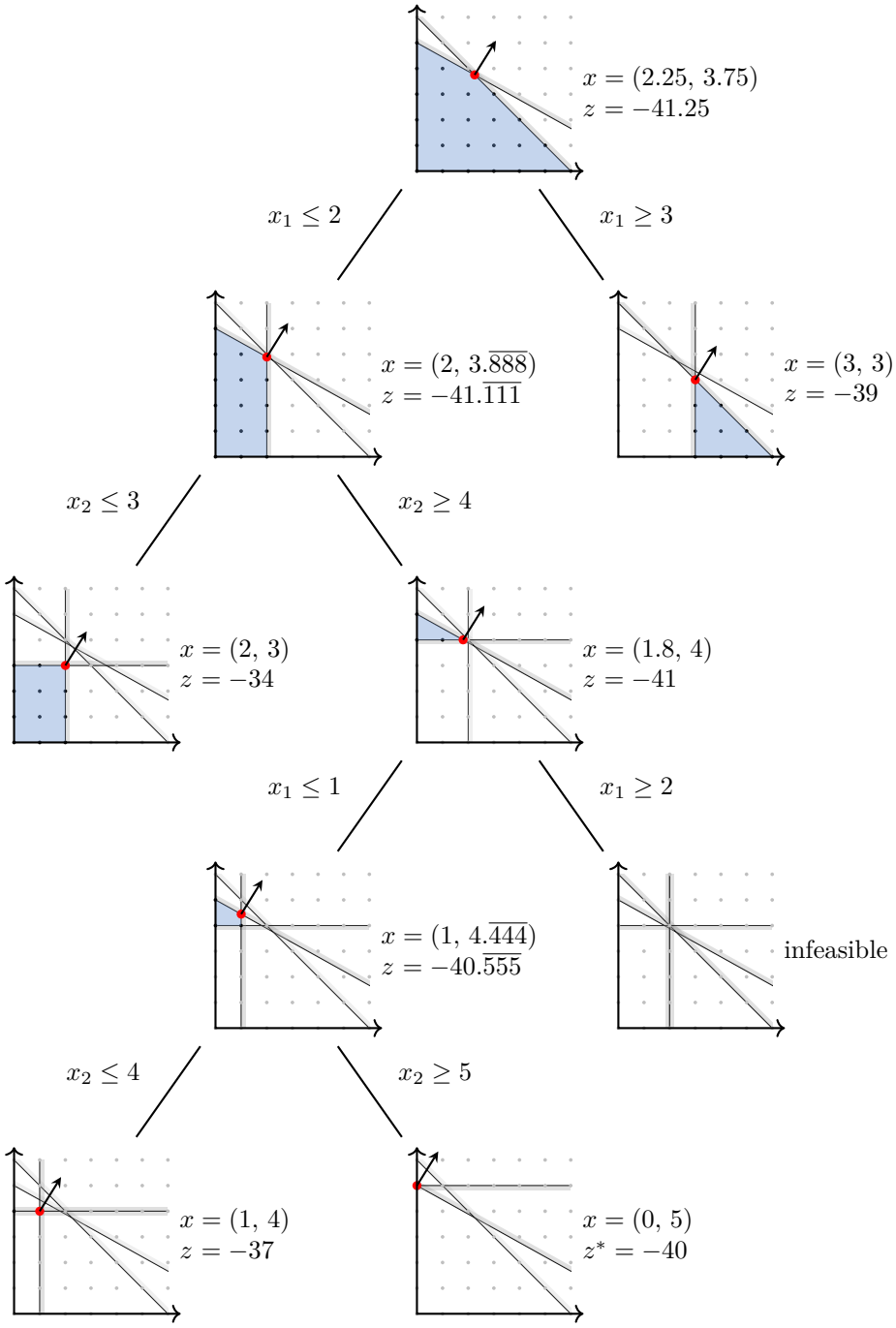


Figure 3.2: A branch-and-bound tree for the the solution to the example problem (3.1.4). Each node contains the solution to the relaxed problem and its optimal value and a visualisation of the relaxed problem in that node. The shaded regions indicate the feasible areas, the arrows represent the objective function gradient, and the optimal solutions are marked with dots. For clarity, only the relevant branch cuts are shown in each node.

3.1.4 The Benders Decomposition Algorithm

The classical Benders decomposition algorithm introduced in [5] is a method for solving a certain class of optimisation problems. The problem is partitioned into a master programming problem (which can be continuously linear, discrete, or nonlinear) and an LP subproblem. In other words, it is required that the subproblem can be written as an LP.

For example, an MILP can be solved using this method by decomposition into a pure ILP master problem and a pure LP subproblem. These problems are then solved iteratively to arrive at the optimal solution to the full problem. Explicitly, one first solves the master problem, fixes the integer variables, and then sends them to the linear subproblem as constants. After solving the subproblem with these variables fixed, new constraints, so called cuts, are generated to feed back into the master problem, which is then re-solved. The hope is that solving these smaller problems will be considerably simpler than solving the full problem.

The Benders decomposition algorithm can also be applied to an LP problem, and the goal is then to split a very large model into smaller LP that are quicker to solve. The master variables are the complicating variables, and if after fixing them the subproblem separates further into multiple smaller problems, the result is many smaller problems which are simpler to solve.

Consider the problem

$$\min_{x,y} \; z \coloneqq ax + by, \tag{3.1.5a}$$

$$\text{s.t.} \;\; Ax + By \geq c, \tag{3.1.5b}$$

$$x, y \geq 0. \tag{3.1.5c}$$

Any LP of the type (3.1.1) can be written in this form by partitioning the variables into two groups. Let z^* be the optimal value of this problem. The variables x are in this case continuous, but they can for example also be in $\{0, 1\}$ or $\mathbb{Z}_{\geq 0}$. As previously stated the variables y are required to be linear. Fixing x to $\bar{x}$ yields the subproblem

$$\min_{y} \; z(\bar{x}) \coloneqq by + a\bar{x}, \tag{3.1.6a}$$

$$\text{s.t.} \;\; By \geq c - A\bar{x}, \tag{3.1.6b}$$

$$y \geq 0. \tag{3.1.6c}$$

The terms involving x are renamed $\bar{x}$ and moved to the right hand side to indicate that they are considered constants in the problem (3.1.6). If the values $\bar{x}$ are feasible in the original problem, the optimal value of the full problem z must be at least as small as the optimal value of (3.1.6a):

$$z^* \leq z^*(\bar{x})$$

By associating the constraint (3.1.6b) with a dual variable u, the dual of the subproblem can be written as

$$\begin{aligned} \min_{u} \; & u(c - A\bar{x}) + a\bar{x}, \\ \text{s.t.} \;\; & uB \leq b, \\ & y \geq 0. \end{aligned}$$

Let $u(\bar{x})$ be the optimal solution to this dual problem. Weak duality gives the bound

$$z(\bar{x}) \geq u(\bar{x}) \cdot (c - A\bar{x}) + a\bar{x} \tag{3.1.8}$$

on the value of $z(\bar{x})$, and since $u(\bar{x})$ is feasible for any choice of $\bar{x}$, the inequality (3.1.8) remains valid if we replace $\bar{x}$ by x. Here, it is assumed that there exists a feasible solution u to the

subproblem dual, since if the subproblem is infeasible, the original problem is also infeasible for that choice of $\bar{x}$. In summary, we get the bound

$$z \geq u(\bar{x}) \cdot (c - Ax) + ax \tag{3.1.9}$$

for the subproblem; this is the classical Benders cut. This procedure assumes that (3.1.6) has a feasible solution. If instead the subproblem is infeasible or unbounded, it is possible to obtain a cut

$$u(\bar{x}) \cdot (c - Ax) \leq 0, \tag{3.1.10}$$

where $u(\bar{x})$ is the ray that solves

$$\begin{aligned} \max_{v} \ & v(c - A\bar{x}), \\ \text{s.t.} \ & vB \leq 0, \\ & v \geq 0. \end{aligned}$$

The ray defined by $u(\bar{x})$ is an unbounded direction and so long as inequality (3.1.10) is true it will remain an unbounded direction.

Let S denote the set of points x such that the system of inequalities (3.1.6b) and (3.1.6c) are consistent. With this notation, the original problem (3.1.5) can be written as

$$\begin{aligned} \min_{x} \ & z^*(x), \\ \text{s.t.} \ & x \in S, \\ & x \geq 0. \end{aligned}$$

This problem can be linearised by introducing a new variable ξ such that $\xi \geq z$. Using the cuts (3.1.9) and (3.1.10), the problem (3.1.5) can be written as

$$\begin{aligned} \min_{x,\xi} \ & \xi, \\ \text{s.t.} \ & \xi \geq ax + u_i(c - Ax), && i \in U, \\ & 0 \geq v_i(c - Ax), && i \in V, \\ & x \geq 0, \end{aligned}$$

where U is the set of extreme points and V is the set of extreme rays of the subproblem polytope.

Instead of enumerating the full sets U and V, the Benders decomposition algorithm constructs these iteratively by generating the cuts (3.1.9) and (3.1.10) in the subproblem. These are then added to the master problem, which is solved again, and new trial values $\bar{x}_i$ are sent to the subproblem. This procedure continues until the objective value of the subproblem is no longer greater than the objective value obtained from the last solution of the master problem. The general structure of the Benders decomposition algorithm is summarised in Figure 3.3.

There are two main reasons why the Benders decomposition algorithm might be a good choice. First by partitioning the variables in a way that exploits the program structure, the resulting subproblem might be in a class of problems which are known to be simple to solve. Second, the subproblem might decouple into several smaller subproblems. These properties are interesting for more classes of optimisation problems than can be treated by the classical Benders decomposition algorithm. However, the Benders decomposition algorithm relies on the ability to obtain a bound on the optimal value of the full problem from the LP dual. If after decomposing the problem, the subproblem is not an LP, this dual does not exist. In Section 3.2, a new type of dual is introduced that can be written for a general optimisation problem. This is the first step to a generalised Benders decomposition algorithm.

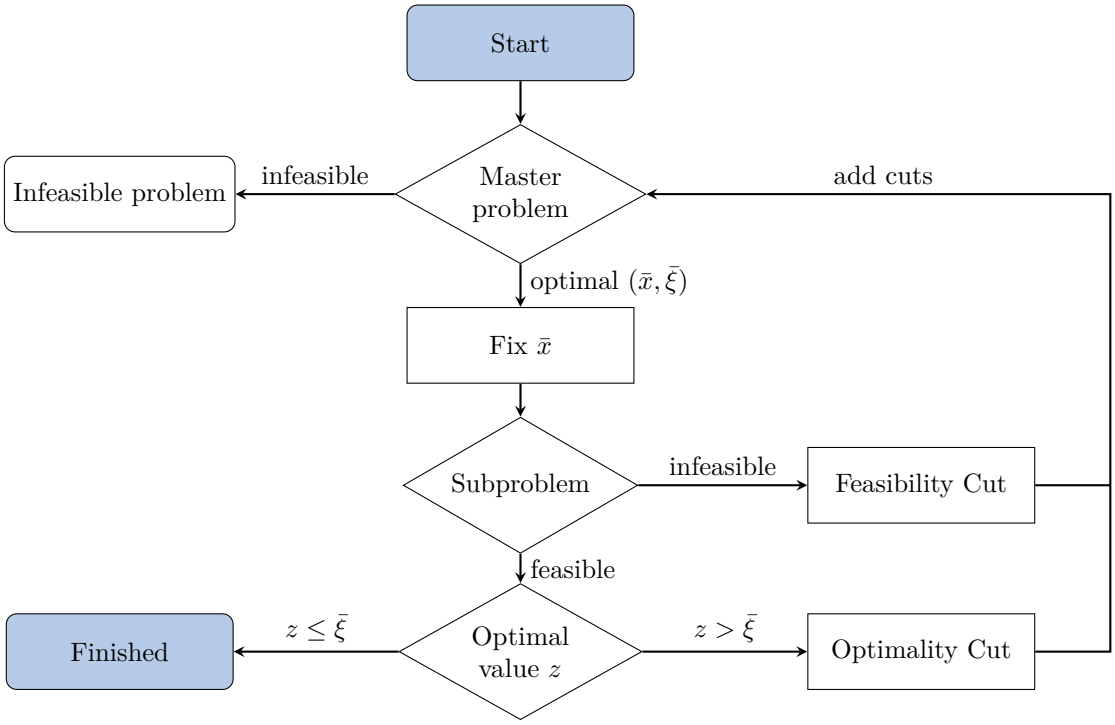


Figure 3.3: Flowchart of the general structure of the Benders decomposition algorithm.

3.2 Inference Duality

Consider the general minimisation problem

$$\min_x \ f(x), \tag{3.2.1a}$$
$$\text{s.t.} \ \ x \in S, \tag{3.2.1b}$$
$$x \in D, \tag{3.2.1c}$$

with domain D, the feasible set S, and objective function $f : \Omega \to \mathbb{R}$ for some superset Ω of D and S. It need not be true that S is a subset of D, nor D of S. For example, a general LP can be described by the objective function $f(x) = cx$, the feasible set $S = \{x \in \mathbb{R}^n \mid Ax \geq b\}$, and the domain $D = \mathbb{R}^n$. Choosing instead the domain $D = \mathbb{Z}^k \times \mathbb{R}^{n-k}$, for some $k \leq n$, yields a general MILP. Other classes of optimisation problems can similarly be represented by appropriate choices of f, S, and D. We will in this section outline a generalisation of LP duality which is applicable in this more general setting, following the description by Hooker and Ottosson [24].

Some notation is needed before the inference dual can be stated. Let P and Q be two propositions whose truth or falsehood are functions of x. This gives the following definition:

Definition 3.2.1. *P implies Q with respect to D* (notated $P \xrightarrow{D} Q$) if Q is true for any $x \in D$ for which P is true.

The inference dual is defined as a maximisation problem over a proof family, where we want to find the proof which proves for x in the domain the largest lower bound of $f(x)$ over the feasible

set. Mathematically, the inference dual is written as

$$\max_{\beta} \ \beta, \tag{3.2.2a}$$

$$\text{s.t.} \quad x \in S \xrightarrow{D} f(x) \geq \beta, \tag{3.2.2b}$$

where β can be seen as the largest lower bound on the function $f(x)$ for any $x \in D$ that can be inferred from $x \in S$. For convenience we assume that the problem has an optimal value, where we allow $\beta^* = -\infty$ as a possible value if the problem is unbounded, and $\beta^* = \infty$ if it is infeasible. With these conventions we get a form of strong duality, i.e. the problem (3.2.1) has the same value as the dual problem (3.2.2), which is true by definition.

A solution to the inference dual is a proof that the inequality $f(x) \geq \beta$ holds. Hooker and Ottosson [24] give two steps for solving the inference dual:

i. Identify inference rules that are complete for the type of constraints in the problem, i.e. they can be used to infer any valid implication of the form $f(x) \geq z$.

ii. Use the rules to prove optimality.

The exact procedure differs for each class of problem. In Section 3.2.1 this procedure is demonstrated for a LP, where we see that this yields the standard LP dual. In Section 3.5 this is done for a MILP by inspecting a branch-and-bound tree.

3.2.1 The Inference Dual of a Linear Program

The classical LP dual is a special case of the inference dual, where the inference rule is to take a linear combination of the constraints. The proof that this is a complete inference rule is essentially the same as the classical separation lemmas for polyhedra [20].

Consider a linear minimisation problem:

$$\begin{aligned} \min_{x} \ & cx, \\ \text{s.t.} \ & Ax \geq a, \\ & x \geq 0, \end{aligned}$$

and the corresponding inference dual

$$\max_{z} \ z, \tag{3.2.4a}$$

$$\text{s.t.} \quad (Ax \geq a, x \geq 0) \xrightarrow{\mathbb{R}^n} cx \geq z, \tag{3.2.4b}$$

where $Ax \geq a$ and $x \geq 0$ defines the feasible set and $\mathbb{R}^n$ is the domain of x. This constraint set can be rewritten by the following observation.

Theorem 3.2.1 (Linear implications)**.** *$Ax \geq a$ implies $cx \geq z$ if and only if $Ax \geq a$ is infeasible or there is a real vector $u \geq 0$ for which $uA \leq c$ and $ua \geq z$.*

Theorem 3.2.1 is essentially just a reformulation of the usual separation lemmas for convex polyhedra. Using Theorem 3.2.1, we get that (3.2.4) is equivalent to finding a non-negative linear combination of $Ax \geq a$ that dominates $cx \geq z$, i.e., $uA \leq c$ and $ua \geq z$, and maximises z. This is the same as the classical dual for an LP:

$$\begin{aligned} \max \ & ua, \\ \text{s.t.} \ & uA \leq c, \\ & u \geq 0. \end{aligned}$$

The vector u can be seen as encoding a proof of optimality, since using it we can deduce the optimal value of z.

3.3 The Logic-Based Benders Decomposition Algorithm

The basic idea behind the logic-based Benders decomposition algorithm [24] is the same as in the classical Benders decomposition algorithm, but in a more abstract setting. Variables in the problem are partitioned into two vectors, x and y. A subproblem containing only the variables y is obtained by fixing the variables x to some trial values $x := \bar{x}$. If the solution to the subproblem reveals that the trial assignments are either infeasible or result in a non-optimal objective value for the full problem, the reason why is identified using the constraint set. New trial values for x are then generated using this information. One continues iteratively in this way until an optimal solution is found, or until the problem is found to be infeasible.

The classical Benders decomposition algorithm uses the LP dual to get this information. In the logic-based Benders decomposition algorithm an inference dual (3.2.1) is instead used, which is the problem of inferring a strongest possible bound from the constraint set. The solution to the dual is a proof that the bound is valid for $x = \bar{x}$. Identifying for which other values of x the same reasoning holds, we get a valid bound on the objective value as a function of x. Since an inference dual is used instead of an LP dual, the logic-based Benders decomposition algorithm can be used to solve more general optimisation problems.

Chu and Xia [8] define a valid Benders cut by the following:

i. The cut must exclude the current solution to the master problem if it is not globally feasible.

ii. The cut must not exclude any globally feasible solutions.

The first property guarantees finite convergence, since in the worst case scenario it results in a complete enumeration of the master variables. If the master problem is finite, the enumeration terminates in a finite number of steps. The second property guarantees optimality since no feasible solutions are removed from the problem. This means that the optimal solution remains in the problem. Consider a general optimisation problem of the form

$$\begin{aligned} \min_{x,y} \ & f(x,y), \\ \text{s.t.} \ & (x,y) \in S, \\ & x \in D_x, \\ & y \in D_y. \end{aligned}$$

The logic-based Benders decomposition algorithm goes as follows: Fix x to some trial value $\bar{x} \in D_x$, resulting in the subproblem

$$\begin{aligned} \min_{y} \ & f(\bar{x},y), \\ \text{s.t.} \ & (\bar{x},y) \in S, \\ & y \in D_y. \end{aligned}$$

The inference dual, see (3.2.2), to the above subproblem,

$$\max_{\beta} \ \beta, \tag{3.3.3a}$$

$$\text{s.t.} \ \ (\bar{x},y) \in S \xrightarrow{D_y} f(\bar{x},y) \geq \beta, \tag{3.3.3b}$$

is finding the tightest bound β on $f(\bar{x}, y)$ that can be inferred from $(\bar{x}, y) \in S$. The next step is to somehow derive a function $\beta(x)$ that gives a valid bound on the value of the subproblem for any value $\bar{x} \in D_x$. How this bound is generated differs between classes of problems, but the general idea is the following: Let β^* be the optimal value of (3.3.3), whose solution is a proof of the fact that β^* is a lower bound on the optimal value when $x = \bar{x}$. Using *this same line of argument* for other values of x yields a valid lower bound as a function of x. This bounding function $\beta_{\bar{x}}(\cdot)$ yields a Benders cut $z \geq \beta_{\bar{x}}(x)$ generated from the trial assignment $x = \bar{x}$. The algorithm continues in the same way as the classical Benders decomposition. In each iteration the master problem has the form

$$\begin{aligned} \min_{z,x} \quad & \bar{z} := z, \\ \text{s.t.} \quad & z \geq \beta_{\bar{x}^i}(x),\ i = 1, \ldots, K, \\ & x \in D_x, \end{aligned}$$

where $\bar{x}^1, \ldots, \bar{x}^K$ are the trial values obtained from earlier iterations and $\beta_{\bar{x}^i}(x)$ are the corresponding bounding functions. A new trial value $(\bar{z}^{K+1}, \bar{x}^{K+1})$ is obtained by solving the subproblem for this trial value. The algorithm terminates when the optimal value of the subproblem β^* is equal to $\bar{z}$.

3.4 Logical Clauses and Resolution

This section will give a brief introduction to propositional logic so that it can be used to define the logic-based Benders decomposition algorithm for a mixed binary linear program (MBLP). For general MILP, propositional logic is not applicable, instead see [22].

Propositional logic consists of formulae containing *atomic propositions* x_j which can be either true or false. These can then be connected to create formulae with for example 'and', 'or', and 'not'. These are written in the following way: the conjunction 'x_1 *and* x_2' is written as $x_1 \wedge x_2$, the disjunction 'x_1 *or* x_2' is written as $x_1 \vee x_2$, and the negation '*not* x_1' is written as $\neg x_1$. A formula F_1 is said to imply another formula F_2 if for any x_j that make F_1 true, x_j also make F_2 true. A *literal* is an atomic proposition or its negation, and a logical clause is a disjunction of literals such that the clause is true if any of the literals it contains are true. The empty clause is the clause containing no literals; it is false by necessity. We will need the following lemma from Hooker and Dawande [22]:

Lemma 3.4.1. *A clause C_1 implies clause C_2 if and only if all the literals in C_1 occur in C_2. C_1 is then said to* absorb *C_2.*

A method for deriving all implications of a given set of clauses is the *resolution* algorithm introduced by Quine [40, 41]. A brief overview of the resolution algorithm is given below.

Algorithm 1 A brief overview of the resolution algorithm.

A resolution rule is used to generate a new clause from two other clauses which contain exactly one atomic proposition x_j that occurs positively in one clause (x_j) and negatively in the other clause ($\neg x_j$). This new clause is called the *resolvent*, and consists of the disjunction of all literals from either clause except for x_j and $\neg x_j$ (this is the resolution rule). Quine showed that the resolution algorithm is complete, i.e. if it begins with a set S of clauses and terminates with S', then any clause implied by S is absorbed by some clause in S'. Then S is false if and only if S' contains the empty clause.

If after performing a resolution step, as described in Algorithm 1, on a set S we receive the empty set, this proves that there is no solution to the original set. As stated above the empty set is always false, so any set of clauses that imply the empty set must also be false.

3.4.1 Resolution Example

Consider three atomic propositions x_1, x_2, and x_3. Write down a set of clauses containing these propositions. The resolution method can then be used to prove that there is no solution to this set. As an example consider the set

$$\begin{aligned} \neg x_1 \vee \neg x_2 \vee \neg x_3, &\quad \text{(a)} \\ x_3, &\quad \text{(b)} \\ x_2, &\quad \text{(c)} \\ \neg x_2 \vee \neg x_3, &\quad \text{(d)} \\ x_1 \vee x_3, &\quad \text{(e)} \\ x_1 \vee x_2. &\quad \text{(f)} \end{aligned}$$

A resolution proof that the set (a)–(f) of clauses is false can be performed as follows.

Step 1. The resolution of clause (a) and (b) is (g): $\neg x_1 \vee \neg x_2$.

Step 2. The resolution of clause (d) and (e) is (h): $x_1 \vee \neg x_2$.

Step 3. The resolution of clause (g) and (c) is (i): $\neg x_1$.

Step 4. The resolution of clause (h) and (f) is (j): x_1.

Step 5. The resolution of clause (i) and (j) is (k): $\emptyset$.

Since we get the empty clause, this proves falsehood for the set of clauses (a)–(f). This procedure is visualised in Figure 3.4.

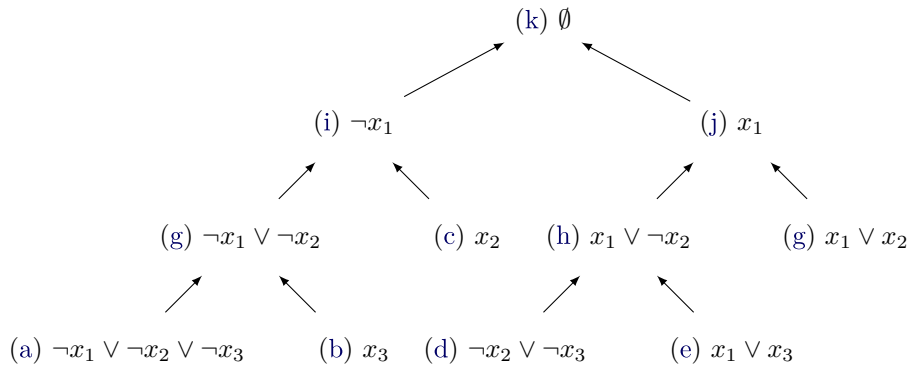


Figure 3.4: Visualisation of the resolution proof performed in this section. The clauses are associated with leaf nodes, and the resolvents with non-leaf nodes. The arrows show which two clauses are used to produce the corresponding resolvent.

3.5 An LBBD Method for Mixed binary Linear Programming

This section broadly follows the work in Hooker and Dawande [22] and Hooker [25]. We consider an MBLP with n variables x_j, where the binary variables correspond to $j = 1, \ldots, k$ and the continuous variables correspond to $j = k+1, \ldots, n$:

$$\min\ z \coloneqq cx \tag{3.5.1a}$$

$$\text{s.t.}\quad Ax \geq a \tag{3.5.1b}$$

$$-x \geq -b \tag{3.5.1c}$$

$$x \geq 0 \tag{3.5.1d}$$

$$x_j \in \{0,1\}\ j = 1, \ldots, k \tag{3.5.1e}$$

$$x_j \in \mathbb{R} \qquad j = k+1, \ldots, n \tag{3.5.1f}$$

Inequality (3.5.1b) contains the constraints of the problem and the b_j is an upper bound on x_j, which is taken to be 1 for the binary variables. In each node of the branch-and-bound tree a relaxed problem of the following form is solved:

$$\min\ z = cx \tag{3.5.2a}$$

$$\text{s.t.}\quad Ax \geq a \tag{3.5.2b}$$

$$Hx \geq h \tag{3.5.2c}$$

$$-x \geq -b \tag{3.5.2d}$$

$$x \geq 0 \tag{3.5.2e}$$

The constraints $Hx \geq h$ correspond to the active branch cuts, which have the form of $x_j \leq 0$ or $x_j \geq 1$. Describing the variable bounds explicitly with b and the branch cuts with H and h in this way will be convenient for the sensitivity analysis performed in this section.

The sensitivity analysis consists of two parts, where the first part is to recover a proof scheme that the optimal value of z is z^*, i.e. $z \geq z^*$, from the solution to the primal problem. The second part of the analysis consists of fixing this proof and and investigating under which perturbations of the problem this proof remains valid.

We will present two different proof schemes, which both infer bounds from the branching tree used to solve the primal problem. The first proof scheme is more easily motivated, but results in very complicated cuts. Lower bounds from the relaxed node problems are valid for the original problem. For leaf nodes this is the best value we can infer, while for a non-leaf node we get that the smallest bound from its children is a valid bound.

Consider a node $i \in I$, where I are all the non-leaf nodes. Then let J_i be the children of node i and z_i^* be optimal solution in node i. For a leaf node j let $\text{LB}_j = z_j^*$, and for a non-leaf node $i \in I$ let the inferred bound be

$$\text{LB}_i = \max\left\{z_i^*, \min\left\{\text{LB}_j \mid j \in J_i\right\}\right\}.$$

By starting at the bottom and going up through the tree we can recursively get a valid bound for the root node which is valid for the original MBLP, and it can be used to prove optimality after a perturbation of the right-hand side. The bounding function is then a piecewise linear function, containing nested min and max functions. For large trees, this function can be difficult to interpret, since it contains information of all the nodes. A full description of this proof scheme is presented in [26].

The second proof scheme only uses information from leaf nodes, which gives a more practical way of generating the bounding function. Violated inequalities are associated with each leaf node. In feasible nodes we start with $z < z^*$, and in infeasible nodes with one of the violated constraints. From these we derive surrogate inequalities—see Section 3.5.1—which we associate with the respective nodes. These surrogate inequalities are violated in their associated leaf nodes. Logical clauses are inferred from the surrogate inequalities, and these clauses are proven to be inconsistent using a resolution proof. This is used as a refutation proof that the inequality z^* is the optimal value, i.e. a proof that $z < z^*$ is false.

This proof remains valid if the violated inequalities at each leaf node continue to imply the logical clauses used in the resolution proof. More details are presented in Section 3.5.2. In the context of logic-based Benders decomposition algorithm, this is used to generate the bounding functions $\beta_{\bar{x}}(\cdot)$. Due to the fact that strong duality holds for inference duals, this is a strong dual, yielding a strong cut when used in a Benders decomposition scheme.

The resulting Benders cuts from using this method are in general nonlinear. However this is not a problem if the master problem is solved by branching; Hooker and Dawande [22] contains a description on how this is done. The cuts can also be linearised; see [25] for how this can be done.

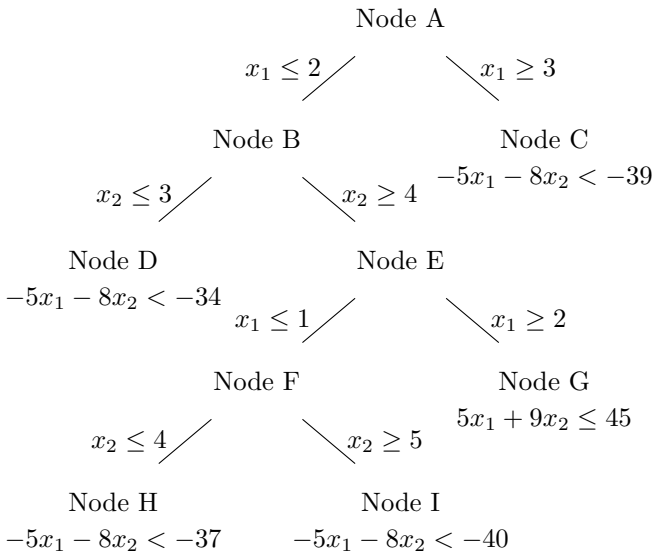


Figure 3.5: The branching tree resulting from the solution of (3.1.4). Each leaf node (C, D, G, H, I) is associated with an inequality, which violates the fixed variables for that node. Each feasible leaf node (C, D, H, I) is associated with the inequality that the value of z is lower than the objective value for that node. Each infeasible leaf node (G) is associated with one of the violated constraints.

If the branching tree is viewed from the bottom it represents a proof of optimality, and a solution to a suitably defined inference dual. Figure 3.5 shows a proof of optimality for the ILP (3.1.4). The branch cuts in each leaf node violate the inequality associated with that node. The resolution algorithm described in Section 3.4 then proves that these inequalities are inconsistent. This means that a feasible solution must violate at least one of the inequalities associated with feasible nodes. This is only possible if the optimal value from the branch-and-bound tree is a lower bound on the objective function for any feasible values of x_1 and x_2. Solving a MIP by branching

method, as in Section 3.1.2, solves an inference dual simultaneously. This is an analogue of the simplex method solving the primal and dual problems simultaneously [26].

3.5.1 Surrogate Inequalities

Consider the tree obtained by solving (3.5.1) using a branch-and-bound, and let T be the set of leaf nodes. For a leaf node $t \in T$, define a *partial assignment* as disjoint subsets J_1^t, J_0^t of $J = 1, \ldots, k$, such that J_1^t is the set containing all j for which x_j are fixed to 1 and J_0^t, analogously, those x_j which are fixed to 0. If $x_j = 1$ then the literal x_j is said to be true and analogously if $x_j = 0$ the negation $\neg x_j$ is true. This partial assignment can be uniquely associated with the weakest clause that it falsifies, i.e. ,

$$C_t = \bigvee_{j \in J_0^t} x_j \vee \bigvee_{j \in J_1^t} \neg x_j,$$

that one of the values of x_j differ from the fixed values in the node.

The set of clauses C_t for $t \in T$ implies that there is an x which is not contained in any of the leaf nodes, i.e. it differs from the fixed values for all nodes. This statement is proven to be false by the resolution algorithm described in 3.4. Branching on x_j creates two clauses that contain exactly one literal which is positive in one clause (x_j) and negative in the other clause ($\neg x_j$). By starting at the bottom of the tree we can associate with the parent node the resolvent of the clauses of its children. The resolvent will contain neither x_j nor $\neg x_j$, and by continuing in this way all x_j are successively removed, and the empty clause will be associated with the root node. Since the resolution algorithm is complete, it proves that the clauses C_t for all leaf nodes are inconsistent.

The missing piece is how to get inequalities for each leaf node t that imply C_t. Since the set of these clauses for the leaf nodes can be proven to be inconsistent, a system of such inequalities must also be inconsistent. These inequalities are constructed such that if they are inconsistent they prove that z^* is the optimal value of the problem (3.5.1).

Consider the relaxed node problem in (3.5.2). Let $\bar{z}$ be the best objective value found from previous nodes, and possibly $\bar{z} = \infty$ if no solution has been found. In each node there are three cases:

Case 1. (3.5.2) is feasible and its value $\hat{z}$ is not better than the current bound $\bar{z}$, i.e. $\hat{z} \geq \bar{z}$.

Case 2. (3.5.2) is feasible and its value $\hat{z}$ is better than the current bound $\bar{z}$, i.e. $\hat{z} < \bar{z}$.

Case 3. (3.5.2) is infeasible.

For each of these cases we associate the node with a surrogate inequality which is violated by the fixed variables, so that they imply C_t. These inequalities are constructed from the dual solutions in each node. Associating dual variables to the relaxation (3.5.2) as λ corresponding to (3.5.2b), μ corresponding to (3.5.2c), and ν corresponding to (3.5.2d), results in the following dual LP:

$$\max_{\lambda,\mu,v} \lambda a + \mu h - \nu b, \tag{3.5.3a}$$

$$\text{s.t. } \lambda A + \mu H - \nu \leq c, \tag{3.5.3b}$$

$$\lambda, \mu, \nu \geq 0. \tag{3.5.3c}$$

Each of the Cases 1–3 are considered separately and results in different surrogate inequalities.

(Case 1.) $\bar{z}$ is the minimum value of the objective function. By adding $cx < \bar{z}$ to the original constraint set, the resulting constraint set becomes infeasible. This can be shown by writing down the following system:

$$-cx \geq \epsilon - \bar{z}, \tag{3.5.4a}$$
$$Ax \geq a, \tag{3.5.4b}$$
$$Hx \geq h, \tag{3.5.4c}$$
$$-x \geq -b, \tag{3.5.4d}$$

where $\epsilon > 0$ is added to remove the strict inequality. Consider a linear positive combination of inequalities (3.5.4a–d) with multipliers $(1, \lambda, \mu, \nu)$. If the above system has a feasible solution, it implies that such a combination also has a solution. The proof of inconsistency is the content of Lemma 3.5.1.

Lemma 3.5.1. *A linear positive combination of* inequalities (3.5.4a–d) *with specified multipliers* $(1, \lambda, \mu, \nu)$ *has no solution if* (λ, μ, ν) *is a feasible solution to the dual problem* (3.5.3).

Proof. Assume there is a feasible solution to the inequality

$$1(-cx) + \lambda(Ax) + \mu(Hx) + \nu(-x) \geq 1(\epsilon - \bar{z}) + \lambda(a) + \mu(h) + \nu(-b).$$

By reordering, we obtain

$$(\lambda A + \mu H - \nu - c)x \geq \epsilon + (\lambda a + \mu h - \nu b - \bar{z}), \tag{3.5.5}$$

and since the dual solution (λ, μ, ν) satisfies the inequalities

$$\lambda A + \mu H - \nu - c \leq 0,$$
$$\lambda a + \mu h - \nu b - \bar{z} \geq 0,$$

the surrogate inequality (3.5.5) implies

$$0 \geq (\lambda A + \mu H - \nu - c)x \geq \epsilon + (\lambda a + \mu h - \nu b - \bar{z}) \geq \epsilon > 0,$$

which is false. The initial assumption must then be false, which concludes the proof. □

If a positive linear combination of inequalities has no solution it is implied that the system formed by these inequalities also has no solution. Combining the results above, we obtain an infeasible system:

$$(\lambda A - c)x \geq \lambda a + \epsilon - \bar{z} \tag{3.5.6a}$$
$$Hx \geq h \tag{3.5.6b}$$
$$-x \geq -b \tag{3.5.6c}$$

In branch-and-bound we only consider values of x such that $Hx \geq h$ and $-x \geq -b$, so the inequality (3.5.6a) violates the fixed variables, i.e. it implies C_t. In other words, inequality (3.5.6a) is our surrogate inequality.

(Case 2.) The analysis is analogous as in Case 1, with $\bar{z}$ replaced by $\hat{z}$. The surrogate inequality thus obtained is

$$(\lambda A - c)x \geq \lambda a + \epsilon - \hat{z}. \tag{3.5.7}$$

(Case 3.) Since the primal problem (3.5.2) is infeasible, there exists a non-negative vector of dual variables (λ, μ, ν) such that these dual variables prove infeasibility, that is

$$\begin{aligned} \lambda A + \mu B - \nu &\leq 0, \\ \lambda a + \mu b - \nu h &> 0. \end{aligned}$$

Like in the previous cases

$$\begin{aligned} \lambda A x &\geq \lambda a, \\ H x &\geq h, \\ -x &\geq -b. \end{aligned}$$

is infeasible. Hence the surrogate inequality is given by

$$\lambda A x \geq \lambda a. \tag{3.5.8}$$

The surrogate inequalities (3.5.6a), (3.5.7), and (3.5.8) all imply C_t in their respective leaf nodes.

3.5.2 Sensitivity Analysis

For the logic-based Benders decomposition algorithm, we want to investigate for which perturbations the inequalities (3.5.6a), (3.5.7), and (3.5.8) are still valid for the refutation proof that z^* is the optimal value, i.e. when they still imply C_t. In this algorithm only the right hand side a is perturbed. We will only consider MBLP and not general MILP. For a complete description of the general integer case, see [22]. Consider thus an MBLP which has been decomposed so that the variables are split into two groups:

$$\begin{aligned} &\min\ z = cx + dy, \\ &\text{s.t.} \\ &\quad Ax + By \geq a, \\ &\quad\quad -x \geq -b, \\ &\quad\quad x \geq 0, \\ &\quad\quad x_j \in \{0,1\}, \quad j = 1, \ldots, k, \\ &\quad\quad y \in \{0,1\}^m. \end{aligned}$$

Here, $x \in \mathbb{Z}^k \times \mathbb{R}^{n-k}$ are the subproblem variables, and y are the master variables. After fixing the master variables to $y = \bar{y}$, they can be moved to the right hand side and treated as constants:

$$\begin{aligned} &\min\ z = cx + d\bar{y}, && \text{(3.5.10a)} \\ &\text{s.t.} \\ &\quad Ax \geq a - B\bar{y}, && \text{(3.5.10b)} \\ &\quad -x \geq -b, && \text{(3.5.10c)} \\ &\quad x \geq 0, && \text{(3.5.10d)} \\ &\quad x_j \in \{0,1\}, \quad j = 1, \ldots, k. && \text{(3.5.10e)} \end{aligned}$$

We obtain our surrogate inequalities by applying inequalities (3.5.6a), (3.5.7), and (3.5.8) to our perturbed constraint (3.5.10b):

$$\text{(Case 1.)} \quad (\lambda A - c)x \geq \lambda(a - B\bar{y}) - \bar{z} + \epsilon \tag{3.5.11a}$$

$$\text{(Case 2.)} \quad (\lambda A - c)x \geq \lambda(a - B\bar{y}) - \hat{z} + \epsilon \tag{3.5.11b}$$

$$\text{(Case 3.)} \quad \lambda A x \geq \lambda(a - B\bar{y}) \tag{3.5.11c}$$

The next step is to see for which other values of y these inequalities continue to imply C_t. A necessary and sufficient condition for when an inequality implies a clause of the form C_t is the content of the following lemma[1].

Lemma 3.5.2. *Consider $x \in \{0,1\}^k \times \mathbb{R}^{n-k}$, $a \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$. Let $J = \{1, \ldots, n\}$ and J_0, J_1 be disjoint subsets of $\{1, \ldots, k\}$. Let $a_j^+ = \max\{0, a_j\}$, $j \in J$. Moreover, assume that the variables x are bounded as $0 \leq x_j \leq h_j$ for all $j \in J$, where $h_j = 1$ for $j \in \{1, \ldots, k\}$. Then, the inequality*

$$\sum_{j \in J} a_j x_j \geq \alpha$$

implies the clause

$$C_t = \bigvee_{j \in J_0} x_j \vee \bigvee_{j \in J_1} \neg x_j$$

if and only if it holds that

$$\sum_{j \in J_1} a_j + \sum_{j \notin J_0 \bigcup J_1} a_j^+ h_j < \alpha. \tag{3.5.12}$$

Proof. We will first rewrite the first statement to show that it is equivalent to (3.5.12). By contraposition

$$\sum_{j \in J} a_j x_j \geq \alpha \longrightarrow C_t \iff \neg C_t \longrightarrow \sum_{j \in J} a_j x_j < \alpha.$$

This is further equivalent to the inequalities $\sum_{j \in J} a_j x_j < \alpha$, for all x such that C_t is false, which is true if and only if

$$\max \left\{ \sum_{j \in J} a_j x_j \mid \neg C_t \right\} < \alpha.$$

The next step is to show that the left-hand side (3.5.13) is equal to the left-hand side of (3.5.12). Fix x to $\bar{x}$ where $\bar{x}_j = 0$ for $j \in J_0$ and $\bar{x}_j = 1$ for $j \in J_1$. Then the negation of C_t, i.e. ,

$$\neg C_t = \bigwedge_{j \in J_0} \neg x_j \wedge \bigwedge_{j \in J_1} x_j$$

is always true for $x = \bar{x}$. Further, it holds that

$$\sum_{j \in J} a_j \bar{x}_j = \sum_{j \in J_1} a_j + \sum_{j \notin J_0 \bigcup J_1} a_j \bar{x}_j.$$

The result then follows from the fact that the maximum of $a_j x_j$ is $a_j^+ h_j$ for all $j \in J$. □

[1] An alternate proof of this lemma can be found in Appendix C.

If the surrogate inequalities still imply C_t for a new trial assignment of y, the proof of optimality outlined in the previous section still holds. From Lemma 3.5.2 it follows that the surrogate inequalities imply C_t for different values of y if and only if the following inequalities are satisfied:

$$\text{(Case 1.)} \quad \sum_{j \in J_1} (\lambda A_j - c_j) + \sum_{j \notin J_0 \bigcup J_1} (\lambda A_j - c_j)^+ h_j \leq \lambda(a - By) - \bar{z} \tag{3.5.13a}$$

$$\text{(Case 2.)} \quad \sum_{j \in J_1} (\lambda A_j - c_j) + \sum_{j \notin J_0 \bigcup J_1} (\lambda A_j - c_j)^+ h_j \leq \lambda(a - By) - \hat{z} \tag{3.5.13b}$$

$$\text{(Case 3.)} \quad \sum_{j \in J_1} \lambda A_j + \sum_{j \notin J_0 \bigcup J_1} (\lambda A_j)^+ h_j < \lambda(a - By) \tag{3.5.13c}$$

After inequalities for each node in the solution tree have been generated, they are combined to obtain a Benders cut. If all of the above inequalities hold in each leaf node for y it is possible to prove that $z(\bar{y})$ is still a lower bound on the value of the subproblem. For each node $t \in T$, depending on which case arose, let I_t denote the corresponding surrogate inequality (3.5.13a–c). Let ξ be the variable introduced to the master problem by the decomposition. The Benders cut is then given by

$$\xi \geq f(y) + C_{\min} + (z(\bar{y}) - C_{\min}) \mathbb{1}_{\{\bigwedge_{t \in T} I_t\}}(y),$$

where T is the set of leaf nodes, $C_{\min}$ is the smallest possible value that the subproblem can attain for any fixed x and provides a default bound. This cut is then used in the logic-based Benders decomposition algorithm as described in Section 3.3.

4

Mathematical Formulation of the Problem

The problem is formulated with a three-index vehicle flow formulation, first introduced by Golden, Magnanti, and Nguyen [18]. As a starting point, we used the model presented by Ruffieux [44], which we extended to include variable battery capacities. Definitions for the different sets, parameters, and variables used to describe the model are presented in Table 4.1. The full model is presented in Section 4.1. This model is then rewritten in Section 4.2 to facilitate our decomposition.

Table 4.1: Definitions of sets, parameters, and variables used to model the problem.

Notation	*Description*
Sets	
$\mathcal{V}_\mathrm{c}$	The set of all nodes containing customers
$\mathcal{V}_\mathrm{r}$	The set of all nodes containing recharge stations
$\{0\}$	The depot node
$\mathcal{V} = \{0\} \cup \mathcal{V}_\mathrm{c} \cup \mathcal{V}_\mathrm{r}$	The set of all nodes in the graph
$\mathcal{K} = \{1, \ldots, K\}$	The set of all vehicles
$\mathcal{A} \subseteq \mathcal{V} \times \mathcal{V}$	The set of all arcs in the graph
$\mathcal{N}$	The set of all battery sizes
Parameters	
K	Size of vehicle fleet (number of vehicles)
T	Latest time to return to the depot [h]
U	Cargo storage capacity in each vehicle [kg]
Q_n	Battery capacity for battery type $n \in \mathcal{N}$ [kWh]
$c^\mathrm{fu}/c^\mathrm{el}$	Cost of fuel/electricity [€/l]/[€/kWh]
$r^\mathrm{fu}/r^\mathrm{el}$	Consumption rate of fuel/electricity [l/h]/[kWh/h]
c^w_n	Cost of choosing battery type $n \in \mathcal{N}$ [€]
t_{ij}	Travelling time over arc $(i, j) \in \mathcal{A}$ [h]
s_i	Service time at node $i \in \mathcal{V}$ [h]
p_i	Demand of cargo in node $i \in \mathcal{V}$ [kg]
d_{ij}	Length of arc $(i, j) \in \mathcal{A}$ [h]
e_i/l_i	Earliest/latest time at which the service in node $i \in \mathcal{V}$ can start [h]
M_T	A large enough time ($M_\mathrm{T} = 2T$ suffices) [h]

Continued on next page

Notation	*Description*
M_{U}	A large enough cargo capacity ($M_{\mathrm{U}} = U$ suffices) [kg]
M_{Q}	A large enough battery capacity ($M_{\mathrm{Q}} = 2\max_{n\in\mathcal{N}} Q_n$ suffices) [kWh]
Variables	
x_{ij}^k	$= 1$ if arc $(i,\, j) \in \mathcal{A}$ is used by vehicle $k \in \mathcal{K}$; $= 0$ otherwise
τ_i^k	Arrival time at node $i \in \mathcal{V}\setminus\{0\}$ for vehicle $k \in \mathcal{K}$ [h]
u_i^k	Amount of cargo in vehicle $k \in \mathcal{K}$ at arrival in node $i \in \mathcal{V}$ [kg]
q_i^k	Battery level for vehicle $k \in \mathcal{K}$ upon arrival in node $i \in \mathcal{V}$ [kWh]
$z_{ij}^{\mathrm{fu},k}/z_{ij}^{\mathrm{el},k}$	Fuel/electricity used on arc $(i,\, j) \in \mathcal{A}$ by vehicle $k \in \mathcal{K}$ [l]/[kWh]
w_n^k	$= 1$ if battery type $n \in \mathcal{N}$ is used by vehicle $k \in \mathcal{K}$; $= 0$ otherwise

4.1 A Mixed Binary Linear Optimisation Model

The three-indexed vehicle flow formulation of our VRP is given by the following:

$$\min_{x,\tau,u,q,z^{\mathrm{fu},},z^{\mathrm{el},},w} \quad \sum_{k\in\mathcal{K}}\sum_{(i,\,j)\in\mathcal{A}}\left(c^{\mathrm{fu}}z_{ij}^{\mathrm{fu},k} + c^{\mathrm{el}}z_{ij}^{\mathrm{el},k}\right) + \sum_{k\in\mathcal{K}}\sum_{n\in\mathcal{N}} c_n^{\mathrm{w}} w_n^k, \tag{4.1a}$$

s. t.

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k = 1, \qquad i \in \mathcal{V}_{\mathrm{c}}, \tag{4.1b}$$

$$\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k \le 1, \qquad k \in \mathcal{K}, i \in \mathcal{V}, \tag{4.1c}$$

$$\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k = \sum_{j\in\mathcal{V}:(j,\,i)\in\mathcal{A}} x_{ji}^k, \qquad k \in \mathcal{K}, i \in \mathcal{V}, \tag{4.1d}$$

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}} x_{0j}^k \le K, \tag{4.1e}$$

$$e_i \le \tau_i^k \le l_i, \qquad k \in \mathcal{K}, i \in \mathcal{V}, \tag{4.1f}$$

$$\tau_i^k \le T - (s_i + t_{i0}), \qquad k \in \mathcal{K}, i \in \mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}, \tag{4.1g}$$

$$\tau_j^k - \tau_i^k \ge (s_i + t_{ij}) - M_{\mathrm{T}}\left(1 - x_{ij}^k\right), \qquad k \in \mathcal{K}, i \in \mathcal{V}\setminus\{j\}, j \in \mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}, \tag{4.1h}$$

$$u_0^k \le U, \qquad k \in \mathcal{K}, \tag{4.1i}$$

$$u_i^k - u_j^k \ge p_i x_{ij}^k - M_{\mathrm{U}}\left(1 - x_{ij}^k\right), \qquad k \in \mathcal{K}, i \in \mathcal{V}\setminus\{j\}, j \in \mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}, \tag{4.1j}$$

$$z_{ij}^{\mathrm{fu},k}/r^{\mathrm{fu}} + z_{ij}^{\mathrm{el},k}/r^{\mathrm{el}} = d_{ij}x_{ij}^k, \qquad k \in \mathcal{K},\ (i,\, j) \in \mathcal{A}, \tag{4.1k}$$

$$q_i^k - q_j^k \ge z_{ij}^{\mathrm{el},k} - M_{\mathrm{Q}}\left(1 - x_{ij}^k\right), \qquad k \in \mathcal{K}, j \in \mathcal{V}\setminus\{i\}, i \in \mathcal{V}_{\mathrm{c}}, \tag{4.1l}$$

$$\sum_{n\in\mathcal{N}} Q_n w_n^k - q_j^k \ge z_{ij}^{\mathrm{el},k} - M_{\mathrm{Q}}\left(1 - x_{ij}^k\right), \qquad k \in \mathcal{K}, j \in \mathcal{V}\setminus\{i\}, i \in \mathcal{V}_{\mathrm{r}}\cup\{0\}, \tag{4.1m}$$

$$\sum_{n\in\mathcal{N}} w_n^k = 1, \qquad k \in \mathcal{K}, \tag{4.1n}$$

$$x_{ij}^k \in \{0,1\}, \qquad k \in \mathcal{K},\ (i,\, j) \in \mathcal{A}, \tag{4.1o}$$

$$w_n^k \in \{0,1\}, \qquad k \in \mathcal{K}, n \in \mathcal{N}, \tag{4.1p}$$

$$z_{ij}^{\mathrm{fu},k}, z_{ij}^{\mathrm{el},k} \ge 0, \qquad k \in \mathcal{K},\ (i,\, j) \in \mathcal{A}, \tag{4.1q}$$

$$\tau_i^k, u_i^k, q_i^k \ge 0, \qquad k \in \mathcal{K}, i \in \mathcal{V}. \tag{4.1r}$$

The constraints (4.1b) ensure that at most one vehicle visits each customer, the constraints (4.1c) that at most one vehicle visits each node, the constraints (4.1d) state that if a vehicle enters a node then it also must leave it, the constraint (4.1e) ensures that at most K vehicles leave the depot (node 0). Furthermore, the constraints (4.1f) guarantees that the time windows are followed, while the constraints (4.1g) and (4.1h) model the travel times between consecutive nodes in the routes. The cargo constraints are modelled by the constraints (4.1i) and (4.1j) which say that a vehicle can not carry more cargo than its storage capacity, and that the cargo in a vehicle decreases by the demand of each visited node. The constraints (4.1k) define the distance travelled on each used arc as the sum of the distances travelled using fuel and electricity, respectively. The constraints (4.1l) keep track of the vehicles' battery charge levels, where constraints (4.1m) resets the charge to max each time it leaves the depot or visits a recharge station. The constraints (4.1n) ensure that each vehicle chooses exactly one battery type. There are no specific subtour elimination constraints, i.e. that each vehicle must follow a continuous route containing the depot, but these are implied by the time and cargo constraints (4.1g–j).

The variables $z_{ij}^{\mathrm{el},k}$ appear in the objective function with a positive coefficient, and since problem (4.1) is a minimisation problem it would at first seem like that it would be profitable to set $z_{ij}^{\mathrm{el},k}$ to a small as possible value. The constraints (4.1l) would in that case not guarantee that in (4.1l) the battery use $z_{ij}^{\mathrm{el},k}$ is equal to the difference in battery levels $q_i^k - q_j^k$. This is in fact not the case since a reasonable assumption is that $c^{\mathrm{el}} r^{\mathrm{el}} < c^{\mathrm{fu}} r^{\mathrm{fu}}$, and using more electricity would always be cheaper since this uses less fuel. This forces $z_{ij}^{\mathrm{el},k}$ to be as large as possible to minimise the use of conventional fuel, and we get equality between battery use and the difference between battery levels in (4.1l).

Of note is that we can divide the variables and constraints in four different groups: One group with the variables x_{ij}^k, one group with the time variables τ_i^k, one group with the cargo variables u_i^k, and one group with the variables $z_{ij}^{\mathrm{fu},k}$, $z_{ij}^{\mathrm{el},k}$, and w_n^k. The only shared variables are x_{ij}^k, which are in fact present in all four groups. This will be used when we formulate our decomposition of the model. Before we get to the decomposition, however, we will need to slightly rewrite the model, which is the contents of the next section.

4.2 A Reformulation of the Mixed Binary Linear Model

The model in the previous section does not have a explicit cost related to how the route looks. Instead there are the requirement that the fuel and electricity use covers all the routes. When decomposing the problem in the next chapter, we would like there to be some information about the objective function in the master problem. This would give the master problem a reason for choosing shorter routes, instead of unnecessary long routes. The shortest route is not necessarily the optimal one, but it seems like a good place to start.

Further, it is not necessary to keep track of how much fuel is used, i.e. the variables $z_{ij}^{\mathrm{fu},k}$, since we can assume that fuel is used for all arcs x_{ij}^k that is not driven on by electricity. It is not clear that changing the model in this way does not affect the speed at we can solve the problem, but it makes it simpler when we later decompose the problem.

The model is reformulated so that it no longer contains $z_{ij}^{\mathrm{fu},k}$. This is achieved by using the constraints (4.1k) to remove the variables $z_{ij}^{\mathrm{fu},k}$ from the model. The variables $z_{ij}^{\mathrm{fu},k}$ is only present in the objective, and the new objective now instead contains x_{ij}^k and $z_{ij}^{\mathrm{el},k}$. This results in a explicit cost on the variables x_{ij}^k which is used when solving the problem using the logic-based Benders decomposition algorithm. Renaming $z_{ij}^{\mathrm{el},k} = z_{ij}^k$, results in the following reformulated

model:

$$\min_{x,\tau,u,q,z,w} \quad \sum_{k\in\mathcal{K}}\sum_{(i,\,j)\in\mathcal{A}} c^{\mathrm{fu}} r^{\mathrm{fu}} d_{ij} x_{ij}^k + \sum_{k\in\mathcal{K}}\sum_{(i,\,j)\in\mathcal{A}} \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right) z_{ij}^k + \sum_{k\in\mathcal{K}}\sum_{n\in\mathcal{N}} c_n^{\mathrm{w}} w_n^k \tag{4.2a}$$

s. t.

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k = 1, \qquad i\in\mathcal{V}_{\mathrm{c}}, \tag{4.2b}$$

$$\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k \le 1, \qquad k\in\mathcal{K}, i\in\mathcal{V}, \tag{4.2c}$$

$$\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k = \sum_{j\in\mathcal{V}:(j,\,i)\in\mathcal{A}} x_{ji}^k, \qquad k\in\mathcal{K}, i\in\mathcal{V}, \tag{4.2d}$$

$$\sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}} x_{0j}^k \le K, \tag{4.2e}$$

$$e_i \le \tau_i^k \le l_i, \qquad k\in\mathcal{K}, i\in\mathcal{V}, \tag{4.2f}$$

$$\tau_i^k \le T - (s_i + t_{i0}), \qquad k\in\mathcal{K}, i\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}, \tag{4.2g}$$

$$\tau_j^k - \tau_i^k \ge (s_i + t_{ij}) - M_{\mathrm{T}}\left(1 - x_{ij}^k\right), \qquad k\in\mathcal{K}, i\in\mathcal{V}\setminus\{j\}, j\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}, \tag{4.2h}$$

$$u_0^k \le U, \qquad k\in\mathcal{K}, \tag{4.2i}$$

$$u_i^k - u_j^k \ge p_i x_{ij}^k - M_{\mathrm{U}}\left(1 - x_{ij}^k\right), \qquad k\in\mathcal{K}, i\in\mathcal{V}\setminus\{j\}, j\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}, \tag{4.2j}$$

$$z_{ij}^k \le r^{\mathrm{el}} d_{ij} x_{ij}^k, \qquad k\in\mathcal{K}, (i,\,j)\in\mathcal{A}, \tag{4.2k}$$

$$q_i^k - q_j^k \ge z_{ij}^k - M_{\mathrm{Q}}\left(1 - x_{ij}^k\right), \qquad k\in\mathcal{K}, j\in\mathcal{V}\setminus\{i\}, i\in\mathcal{V}_{\mathrm{c}}, \tag{4.2l}$$

$$\sum_{n\in\mathcal{N}} Q_n w_n^k - q_j^k \ge z_{ij}^k - M_{\mathrm{Q}}\left(1 - x_{ij}^k\right), \qquad k\in\mathcal{K}, j\in\mathcal{V}\setminus\{i\}, i\in\mathcal{V}_{\mathrm{r}}\cup\{0\}, \tag{4.2m}$$

$$\sum_{n\in\mathcal{N}} w_n^k = 1, \qquad k\in\mathcal{K}, \tag{4.2n}$$

$$x_{ij}^k \in \{0,1\}, \qquad k\in\mathcal{K}, (i,\,j)\in\mathcal{A}, \tag{4.2o}$$

$$w_n^k \in \{0,1\}, \qquad k\in\mathcal{K}, n\in\mathcal{N}, \tag{4.2p}$$

$$z_{ij}^k \ge 0, \qquad k\in\mathcal{K}, (i,\,j)\in\mathcal{A}, \tag{4.2q}$$

$$\tau_i^k, u_i^k, q_i^k \ge 0, \qquad k\in\mathcal{K}, i\in\mathcal{V}. \tag{4.2r}$$

The cost $\left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right)$ in front of z_{ij}^k represents the costs incurred by using electricity instead of fuel over a arc. A reasonable assumption is that $c^{\mathrm{el}} r^{\mathrm{el}} < c^{\mathrm{fu}} r^{\mathrm{fu}}$. Hence, this cost will be negative, and it will be beneficial to use electricity over fuel whenever possible. If instead $c^{\mathrm{el}} r^{\mathrm{el}} \ge c^{\mathrm{fu}} r^{\mathrm{fu}}$, the optimal value is attained by $z_{ij}^k = 0$ and the problem is the same as a normal VRP. The constraints (4.1k) also gives a bound for z_{ij}^k which is included in the model as constraints (4.2k).

5
Method

A logic-based Benders decomposition algorithm was used to solve the problem defined in Chapter 4. The method works by splitting the optimisation model (4.2) into a master problem and a subproblem. This separation was done in such a way that the variables x_{ij}^k occur in the master problem and the other variables occur in the subproblem. Constraints containing only the variables x_{ij}^k are the only ones left in the master problem. Considering only these parts, the following problem is obtained

$$\min_x \sum_{k\in\mathcal{K}} \sum_{(i,\,j)\in\mathcal{A}} c^{\mathrm{fu}} r^{\mathrm{fu}} d_{ij} x_{ij}^k, \tag{5.0.1a}$$

s. t.

$$\sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k = 1, \qquad i \in \mathcal{V}_{\mathrm{c}}, \tag{5.0.1b}$$

$$\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k \leq 1, \qquad k \in \mathcal{K},\ i \in \mathcal{V}, \tag{5.0.1c}$$

$$\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k = \sum_{j\in\mathcal{V}:(j,\,i)\in\mathcal{A}} x_{ji}^k, \quad k \in \mathcal{K},\ i \in \mathcal{V}, \tag{5.0.1d}$$

$$\sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}} x_{0j}^k \leq K. \tag{5.0.1e}$$

The problem (5.0.1) will constitute the basis for the master problem, where the full master problems contains additional constraints in the form of cuts. These cuts are generated from the subproblem, which consists of three smaller subproblems. After fixing the values of the variables x_{ij}^k in (4.2), the model separates into three parts: one problem containing the variables τ_i^k that verify that the time window constraints are obeyed, one problem containing the variables u_i^k that verify that the cargo capacity constraints are obeyed, and one problem containing the variables q_i^k, z_{ij}^k, and w_n^k which distributes where the vehicles use battery charge instead of conventional fuel. The first two are LPs, so these cuts can be generated using the LP dual. These two problems are similar in that they posses resource constraints. They differ in that cargo amount is decreasing and time is increasing, and also in that the variables representing cargo all have the same bounds whereas the time variables all have different bounds. The third subproblem is to distribute the battery charge, and is a MBLP, so it does not posses an LP dual. Instead an inference dual is used, and the solution method used for this is a branch-and-bound algorithm.

Two types of Benders cuts have been formulated for the problem: classical Benders cuts from the LP subproblems, and a special cut using bounds inferred from the constraint set of the MBLP subproblem. This inference from the constraint set is achieved by inspecting the branch-and-bound tree used to solve the subproblem. A overview of the algorithm employed can

be found in Section 5.5. The models for the different subproblems are presented in Sections 5.1 and 5.2 together with corresponding bounds. These bounds are then combined in Section 5.4 to create the cut that is added to the master problem.

5.1 Linear Programming Subproblems

The time subproblem and the cargo subproblem are both LP problems, which means that the same method can be used to generate cuts from both. The cuts are normal Benders feasibility cuts which are constructed by solving the LP dual. The LP dual and the corresponding feasibility cuts are defined in the following sections. Only feasibility cuts are generated since neither subproblem contains an objective function.

5.1.1 The Time Window Subproblem

The time window constraints are enforced in the model by the τ_i^k variables, which appears in the constraints (4.1f–h). By isolating these constraints, a time window subproblem can be formulated as

$$\min_{\tau} \; 0, \tag{5.1.1a}$$

s. t.

$$e_i \leq \tau_i^k \leq l_i, \qquad k \in \mathcal{K},\, i \in \mathcal{V}, \tag{5.1.1b}$$

$$\tau_i^k \leq T - (s_i + t_{i0}), \qquad k \in \mathcal{K},\, i \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}, \tag{5.1.1c}$$

$$\tau_j^k - \tau_i^k \geq (s_i + t_{ij}) - M_\mathrm{T}\left(1 - \bar{x}_{ij}^k\right), \quad k \in \mathcal{K},\, i \in \mathcal{V} \setminus \{j\},\, j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}, \tag{5.1.1d}$$

$$\tau_i^k \geq 0, \qquad k \in \mathcal{K},\, i \in \mathcal{V}. \tag{5.1.1e}$$

Since the variables τ_i^k do not appear in the objective function (4.1a), the subproblem (5.1.1) is a feasibility problem, and since the variables x_{ij}^k are fixed, the subproblem is an LP. This problem separates even further since each vehicle is independent of the others, resulting in $|\mathcal{K}|$ subproblems. For each vehicle $k \in \mathcal{K}$, we write down the LP dual to the corresponding subproblem:

$$\begin{aligned}\max_{\alpha,\beta,\gamma,\delta} \; & \sum_{i \in \mathcal{V}} \left(\alpha_i e_i - \beta_i l_i\right) + \sum_{i \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}} \gamma_i \left(t_{i0} + s_i - T\right) \\ & + \sum_{(i,j) \in \mathcal{A} : j \notin \{0\}} \delta_{ij} \left(\left(t_{ij} + s_i\right) - M_\mathrm{T}\left(1 - \bar{x}_{ij}^k\right)\right),\end{aligned} \tag{5.1.2a}$$

s. t.

$$\alpha_0 - \beta_0 - \sum_{j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}} \gamma_j \leq 0, \tag{5.1.2b}$$

$$\alpha_i - \beta_i - \gamma_i + \sum_{\substack{j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}: \\ (i,j) \in \mathcal{A}}} \delta_{ij} - \sum_{j:(j,i) \in \mathcal{A}} \delta_{ji} \leq 0, \quad i \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}, \tag{5.1.2c}$$

$$\alpha_i, \beta_i, \gamma_i \geq 0, \quad i \in \mathcal{V}, \tag{5.1.2d}$$

$$\gamma_i \geq 0, \quad i \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}, \tag{5.1.2e}$$

$$\delta_{ij} \geq 0, \quad j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c} \setminus \{i\},\, i \in \mathcal{V}. \tag{5.1.2f}$$

The dual variables α_i, β_i are associated with the constraints (5.1.1b), γ_i with the constraint (5.1.1c), δ_{ij} with the constraint (5.1.1d). Since the primal problem is a feasibility problem, the dual is either feasible or unbounded. If it is feasible the primal problem is also feasible, and no cut is generated.

Let $(\bar{\alpha_i}, \bar{\beta_i}, \bar{\gamma_i}, \bar{\delta_{ij}})$ be the direction of an unbounded ray to the dual problem so that the objective value is increasing along it. The feasibility cut, see the inequality (3.1.10), from this subproblem is then given by the inequality

$$\sum_{i \in \mathcal{V}} \left(\bar{\alpha_i} e_i - \bar{\beta_i} l_i\right) + \sum_{i \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}} \bar{\gamma_i} \left(t_{i0} + s_i - T\right) + \sum_{(i,j) \in \mathcal{A} : j \notin \{0\}} \bar{\delta_{ij}} \left((t_{ij} + s_i) - M_T \left(1 - x_{ij}^k\right)\right) \leq 0. \tag{5.1.3}$$

The inequality (5.1.3) is then added to the master problem (5.0.1).

5.1.2 The Cargo Subproblem

The cargo capacity subproblem is a simpler variant of the time window subproblem as described in the previous section. It separates analogously, one subproblem for each $k \in \mathcal{K}$, which are formulated as

$$\min_{u} \ 0, \tag{5.1.4a}$$

s. t.

$$u_0^k \leq U, \tag{5.1.4b}$$

$$u_i^k - u_j^k \geq p_i \bar{x}_{ij}^k - M_\mathrm{U} \left(1 - \bar{x}_{ij}^k\right), \quad i \in \mathcal{V} \setminus \{j\},\ j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}, \tag{5.1.4c}$$

$$u_i^k \geq 0, \qquad i \in \mathcal{V}. \tag{5.1.4d}$$

It is also a feasibility problem due to the lack of objective function. There are two main differences between (5.1.4) and the time window problem (5.1.1). First, all variable bounds (5.1.4b) are equal in the cargo subproblem, and second, the cargo variable is counting down between nodes instead of counting up as in the time window problem. These differences do, however, not change how the feasibility cuts are obtained. We start by writing down the LP dual of (5.1.4) as

$$\max_{n,m} \ -nU + \sum_{(i,j) \in \mathcal{A} : j \notin \{0\}} m_{ij} \left(p_j \bar{x}_{ij}^k - M_\mathrm{U} \left(1 - \bar{x}_{ij}^k\right)\right), \tag{5.1.5a}$$

s. t.

$$-n + \sum_{j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}} m_{0j} \leq 0, \tag{5.1.5b}$$

$$\sum_{j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c} : (i,j) \in \mathcal{A}} m_{ij} - \sum_{j : (j,i) \in \mathcal{A}} m_{ji} \leq 0, \quad i \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c}, \tag{5.1.5c}$$

$$n \geq 0, \tag{5.1.5d}$$

$$m_{ij} \geq 0, \quad j \in \mathcal{V}_\mathrm{r} \cup \mathcal{V}_\mathrm{c} \setminus \{i\}, i \in \mathcal{V}. \tag{5.1.5e}$$

Here n corresponds to the constraint (5.1.4b) and m_{ij} correspond to the constraints (5.1.4c). By the analogous argument as for the time subproblem (5.1.2), the feasibility cut corresponding to vehicle k is given by the inequality

$$-\bar{n}U + \sum_{(i,j) \in \mathcal{A} : j \notin \{0\}} \bar{m_{ij}} \left(p_j x_{ij}^k - M_U \left(1 - x_{ij}^k\right)\right) \leq 0. \tag{5.1.6}$$

The inequality (5.1.6) is then added to the master problem (5.0.1).

5.2 The Battery Charge Distribution Subproblem

The battery charge distribution subproblem is to decide which battery capacity to use for each vehicle, and to find the best way to spend the electricity charge given a specific route, i.e. for a fixed value of x. The variables w_i^k, q_i^k, and z_{ij}^k, together with their associated constraints, are contained in one subproblem for each vehicle $k \in \mathcal{K}$:

$$\min_{q,w,z} \quad \sum_{(i,j)\in\mathcal{A}} \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}} \right) z_{ij}^k + \sum_{n\in\mathcal{N}} c_n^{\mathrm{w}} w_n^k, \tag{5.2.1a}$$

s. t.

$$z_{ij}^k \le r^{\mathrm{el}} d_{ij} \bar{x}_{ij}^k, \qquad (i,j) \in \mathcal{A}, \tag{5.2.1b}$$

$$q_i^k - q_j^k - z_{ij}^k \ge -M_{\mathrm{Q}} \left(1 - \bar{x}_{ij}^k\right), \quad j \in \mathcal{V} \setminus \{i\},\ i \in \mathcal{V}_{\mathrm{c}}, \tag{5.2.1c}$$

$$\sum_{n\in\mathcal{N}} Q_n w_n^k - q_j^k - z_{ij}^k \ge -M_{\mathrm{Q}} \left(1 - \bar{x}_{ij}^k\right), \quad j \in \mathcal{V} \setminus \{i\},\ i \in \mathcal{V}_{\mathrm{r}} \cup \{0\}, \tag{5.2.1d}$$

$$\sum_{n\in\mathcal{N}} w_n^k = 1, \tag{5.2.1e}$$

$$w_n^k \in \{0,1\}, \qquad n \in \mathcal{N}, \tag{5.2.1f}$$

$$z_{ij}^k \ge 0, \qquad k \in \mathcal{K}, \quad (i,j) \in \mathcal{A}, \tag{5.2.1g}$$

$$q_i^k \ge 0 \qquad i \in \mathcal{V}. \tag{5.2.1h}$$

This subproblem is not continuous since $w_n^k \in \{0,1\}$, and therefore an inference dual is used to obtain Benders cuts; see Section 3.5. The subproblem is solved by a branch-and-bound algorithm. For a node in the branch-and-bound tree let $\mathcal{N}_0$ and $\mathcal{N}_1$ be a partial assignment corresponding to the fixed values of the variables w_n^k.

In the node the following relaxed problem is solved:

$$\min_{q,w,z} \quad \sum_{(i,j)\in\mathcal{A}} \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}} \right) z_{ij}^k + \sum_{n\in\mathcal{N}} c_n^{\mathrm{w}} w_n^k, \tag{5.2.2a}$$

s. t.

constraints (5.2.1b–d),

$$\sum_{n\in\mathcal{N}} w_n^k \ge 1, \tag{5.2.2b}$$

$$-\sum_{n\in\mathcal{N}} w_n^k \ge -1, \tag{5.2.2c}$$

$$-w_n^k \ge -1, \qquad n \in \mathcal{N}, \tag{5.2.2d}$$

$$-w_n^k \ge 0, \qquad n \in \mathcal{N}_0 \tag{5.2.2e}$$

$$w_n^k \ge 1, \qquad n \in \mathcal{N}_1 \tag{5.2.2f}$$

$$w_n^k \ge 0, \qquad n \in \mathcal{N}, \tag{5.2.2g}$$

$$z_{ij}^k \ge 0, \qquad (i,j) \in \mathcal{A}, \tag{5.2.2h}$$

$$q_i^k \ge 0, \qquad i \in \mathcal{V}. \tag{5.2.2i}$$

The relaxed subproblem (5.2.2) is feasible as long there exists a solution to the constraints (5.2.2b–c), since it is always possible to choose $q_i^k = 0, z_{ij}^k = 0$ in the solution. Branching is done

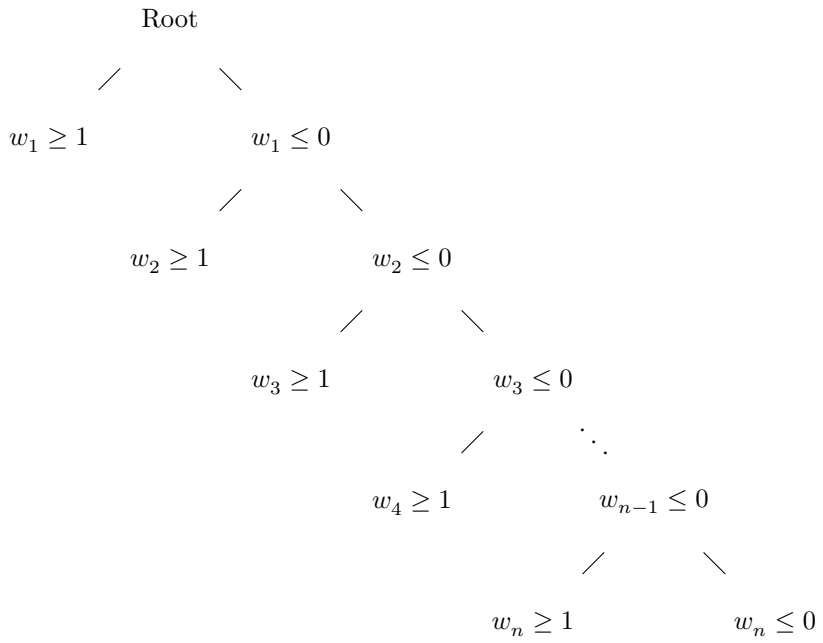


Figure 5.1: Branching tree for the battery subproblem, where the variables w_n are branched on. The structure is due to the constraint (5.2.1e), which forces one w_n to 1, and the others to be set to 0. Consequently, if any w_n is fixed to 1, this results in a feasible solution, and the branching stops. If all but one w_n are fixed to zero it is immediately guaranteed that the solution is integral, and thus the bottom two nodes are never reached.

by fixing the value of w_n, for some n, to either 0 or 1. The structure of the solution tree from the branch-and-bound procedure is simple; see Figure 5.1.

The branching procedure can never get to the point where all w_n^k have been set to zero, since a integer solution is guaranteed in the node before by the constraints (5.2.2b) and (5.2.2c). Because of this it is assured that there exists a feasible solution in each node of the branch-and-bound tree. Cuts are obtained by following the procedure outlined in Section 3.5. First the primal problem model (5.2.2) is solved. Then we construct the necessary and sufficient conditions for the optimal objective value of (5.2.2) being a valid lower bound on (5.2.1) for other trial assignments of the master variable.

In each node of the branch and bound tree, there is a relaxed node problem. For each of the node problems, write down the corresponding LP dual. From the optimal solution of the primal problem in each node we obtain a bound, and from the solution of the dual problems we obtain the sensitivity analysis. Exactly how this is done is the contents of the rest of this section. We use the theory that we developed in Section 3.5.

Let μ_{ij} be dual variables associated with the constraints (5.2.1b), π_{ij} with the constraints (5.2.1c) and (5.2.1d), σ_1 with the constraint (5.2.2b), and σ_2 with the constraint (5.2.2c). It is not necessary to explicitly write down how the dual variables corresponding to the branch cut constraints (5.2.2e–f) and the integer relaxation constraints (5.2.2b–d). Call these dual variables ψ, and as in (3.5.3), let h and H be the corresponding coefficients. The dual problem for in each

node is given by the following:

$$
\begin{aligned}
\max_{\mu_{ij},\pi_{ij},\sigma_1,\sigma_2,\psi} \quad & -\sum_{(i,j)\in\mathcal{A}} \mu_{ij} r^{\mathrm{el}} d_{ij} \bar{x}_{ij}^k - \sum_{(i,j)\in\mathcal{A}} \pi_{ij} M_{\mathrm{Q}} \left(1 - \bar{x}_{ij}^k\right) + \sigma_1 - \sigma_2 + \psi h, \\
\text{s.t.} \quad & \\
& -\mu_{ij} - \pi_{ij} - \sigma_1 + \sigma_2 \leq \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right), && (i,j) \in \mathcal{A}, \\
& \sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} (\pi_{ij} - \pi_{ji}) \leq 0, && i \in \mathcal{V}_{\mathrm{c}}, \\
& -\sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} \pi_{ji} \leq 0, && i \in \mathcal{V}_{\mathrm{r}} \cup \{0\}, \\
& \sigma_1 - \sigma_2 + Q_n \sum_{(i,j)\in\mathcal{A}:i\notin\mathcal{V}_{\mathrm{c}}} \pi_{ji} + \psi H \leq c_n^{\mathrm{w}}, && n \in \mathcal{N}, \\
& \mu_{ij} \geq 0, && (i,j) \in \mathcal{A}, \\
& \mu_{ij} \geq 0, && (i,j) \in \mathcal{A}, \\
& \sigma_1, \sigma_2 \geq 0.
\end{aligned}
$$

In (5.2.3), $\hat{\zeta}$ denotes the optimal value of the relaxed problem (5.2.2) in the node, $\mathcal{N}_0 := \{n \in \mathcal{N} \mid w_n \text{ is fixed to } 0\}$, and $\mathcal{N}_1 := \{n \in \mathcal{N} \mid w_n \text{ is fixed to } 1\}$. As in 3.5.2, define the plus operator as $a^+ = \max\{0, a\}$. The inequality, for $x = \bar{x}$, $I_{\bar{x},t}(x)$ from node t is obtained from the solution to (5.2.3), according to Section 3.5.1, as

$$
\begin{aligned}
I_{\bar{x},t}(x): \quad & \sum_{(i,j)\in\mathcal{A}} \left(-\bar{\pi}_{ij} - \bar{\mu}_{ij} - \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right)\right)^+ \cdot r^{\mathrm{el}} d_{ij} \\
& + \sum_{i\in\mathcal{V}_{\mathrm{c}}} \left(\sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} \left(\bar{\pi}_{ij} - \bar{\pi}_{ji}\right)\right)^+ \cdot Q_{\max} + \sum_{i\in\mathcal{V}\setminus\mathcal{V}_{\mathrm{c}}} \left(\sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} \left(-\bar{\pi}_{ji}\right)\right)^+ \cdot Q_{\max} \\
& + \sum_{n\in\mathcal{N}_1} \left(\bar{\sigma}^1 - \bar{\sigma}^2 + Q_n \sum_{(i,j)\in\mathcal{A}:i\notin\mathcal{V}_{\mathrm{c}}} \bar{\pi}_{ji} - c_n^{\mathrm{w}}\right) \\
& + \sum_{n\in\mathcal{N}\setminus\{\mathcal{N}_0\cup\mathcal{N}_1\}} \left(\bar{\sigma}^1 - \bar{\sigma}^2 + Q_n \sum_{(i,j)\in\mathcal{A}:i\notin\mathcal{V}_{\mathrm{c}}} \bar{\pi}_{ji} - c_n^{\mathrm{w}}\right)^+ \\
& \leq -\sum_{(i,j)\in\mathcal{A}} r^{\mathrm{el}} d_{ij} \bar{\mu}_{ij} x_{ij}^k + \sum_{(i,j)\in\mathcal{A}} M_Q \bar{\pi}_{ij} \left(x_{ij}^k - 1\right) + \bar{\sigma}^1 - \bar{\sigma}^2 - \hat{\zeta},
\end{aligned} \tag{5.2.4}
$$

If the inequality (5.2.4) $I_{\bar{x},t}(x)$ is satisfied for some other route x, the surrogate inequalities (3.5.11a–c) imply that the fixed variable values are violated. It is not necessary to write these surrogate inequalities down explicitly, since only the inequalities $I_{\bar{x},t}(x)$ are needed. A resolution proof can then be used to prove that these surrogate inequalities are false. This is then a proof that the optimal value $\hat{\zeta}$ is a lower bound for other routes that fulfil all inequalities $I_{\bar{x},t}(x)$, where t is a leaf node corresponding to the tree where $\hat{\zeta}$ was obtained.

We also use the fact that the variables q_i^k and z_{ij}^k can be bounded from above as

$$\begin{aligned} z_{ij}^k &\leq r^{\mathrm{el}} d_{ij}, && (i,\, j) \in \mathcal{A}, \\ q_i^k &\leq \max\left\{Q_n \mid n \in \mathcal{N}\right\}, && i \in \mathcal{V}. \end{aligned}$$

These bounds are independent of the choice of value for x_{ij}^k and can be added to the model in (5.2.2) without affecting the solution. Bounds are inferred from the constraints (5.2.1b) for z_{ij}^k and from the constraints (5.2.1c–d) for q_i^k.

Before the cuts for the battery charge distribution subproblems (5.2.1) can be formulated, a globally valid lower bound is needed. If the inequalities $I_{\bar{x},t}(x)$ are not fulfilled, the refutation proof that the optimal objective value is a lower bound is no longer valid. The next section formulates two different globally valid bounds and uses them to obtain the cut for the battery charge subproblem.

5.3 Default Lower Bounds

The last step is to provide a default lower bound $\zeta_{\min}$. If the proof can no longer be used to prove that $\hat{\zeta}$ is a lower bound, this globally valid bound is used instead. We present bounds from two different methods. Which method results in the tightest bound depends on the problem data. The first bound is derived from the maximum amount of charge that can be used by a single vehicle. Solving a relaxed version of the battery problem,

$$\begin{aligned} \min_{z,w} \quad & \sum_{(i,\,j)\in\mathcal{A}} \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right) z_{ij}^k + \sum_{n\in\mathcal{N}} c_n^{\mathrm{w}} w_n^k, \\ \text{s.t.} \quad & \\ & z_{ij}^k \leq r^{\mathrm{el}} d_{ij} x_{ij}^k, \quad (i,\, j) \in \mathcal{A}, \\ & z_{ij}^k \geq 0, \qquad\quad (i,\, j) \in \mathcal{A}, \\ & w_n^k \in \{0,1\}, \qquad n \in \mathcal{N}, \end{aligned}$$

provides a global bound. Due to our assumptions on the fuel costs and consumption rates the coefficient in front of z_{ij}^k is negative, so choosing each z_{ij}^k to equal its maximum is optimal in this problem. The cost for the different battery types are all positive, so choosing w_n^k all zero is optimal. Since the problem above is a relaxation of the battery subproblem the optimal value of it gives the default bound

$$\zeta_{\min} := \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right) r^{\mathrm{el}} \max\left\{ \sum_{(i,\,j)\in\mathcal{A}} d_{ij} x_{ij}^k \mid x_{ij}^k \in X \right\},$$

where X are the possible set of routes defined by constraints (5.0.1b–e). This value is of course larger than if we set all $x_{ij}^k = 1$, which yields the value

$$\zeta_{\min} := \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right) r^{\mathrm{el}} \sum_{(i,\,j)\in\mathcal{A}} d_{ij}. \tag{5.3.2}$$

A better bound can be found by finding the longest possible route. Let the length of this route be $d_{\max}$. The lower bound is then given by

$$\zeta_{\min} := \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right) r^{\mathrm{el}} d_{\max}. \tag{5.3.3}$$

Which of the two default bounds Equations (5.3.2) and (5.3.3) is the better depends on how difficult it is to find the longest route, compared to using a tighter bound. Depending on the problem data, these bounds may both be very poor since they calculate the cost as if it was possible to drive on electricity the whole way. If the battery capacities are small or if there are few recharge stations, a better bound is acquired by counting the number of recharge station and depot nodes. The amount of electricity used can not be greater than

$$\zeta_{\min} = Q_{\max}(N_{\mathrm{r}} + N_{\mathrm{d}}),$$

where N_{r} is the number of recharge stations, and N_{d} is the number of depots. Our model assumes that there is only one depot node. This gives the following valid lower bound on the objective function:

$$Q_{\max}(N_r + 1)\left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}}c^{\mathrm{fu}}\right). \tag{5.3.4}$$

Let $(\zeta^*)^k$ be the optimal objective value off the problem (5.2.1) for vehicle k obtained for the fixed value $x = \bar{x}$ and let T^k be the set of all the leaf nodes in the solution tree for that vehicle. The set of $I_{\bar{x},t}(x)$ represents the inequalities associated with these leaf nodes. The cut from the battery subproblem is then given by

$$\xi \geq \sum_{k\in\mathcal{K}} \sum_{(i,\,j)\in\mathcal{A}} \left(c^{\mathrm{fu}} r^{\mathrm{fu}} d_{ij} x_{ij}^k\right) + \sum_{k\in\mathcal{K}} \xi_k, \tag{5.3.5}$$

where, for each vehicle $k \in \mathcal{K}$, the lower bounds

$$\xi_k \geq \begin{cases} (\zeta^*)^k, & \text{if } \bigwedge_{t\in T^k} I_{\bar{x},t}(x), \\ \zeta_{\min}, & \text{otherwise,} \end{cases}$$

are those that can be proven from the branch-and-bound tree. The default lower bound $\zeta_{\min}$ can be chosen as either of the two formulated above. Which is preferable depends on the problem data and on available computation time (see further in Section 6.4).

5.4 The Master Problem

In this section we combine everything described in this chapter to formulate the master problem, together with all of the cuts. For each vehicle $k \in \mathcal{K}$, we obtained three separate subproblems that each yielded valid cuts. These were combined to form the Benders cut that was added to the master problem. Let Ω be the index set of previous iterations, and $(\beta^*)^{k,\omega}$ be the optimal objective value obtained in iteration $\omega \in \Omega$ of the battery subproblem for vehicle $k \in \mathcal{K}$. The

complete master problem with all of the cuts is then written as

$$\min_{x,\xi,\xi_1,\ldots,\xi_K} \quad \xi, \tag{5.4.1a}$$

$$\text{s. t.}$$

$$\xi \geq \sum_{k\in\mathcal{K}} \sum_{(i,\,j)\in\mathcal{A}} c^{\mathrm{fu}} r^{\mathrm{fu}} d_{ij} x_{ij}^k + \sum_{k\in\mathcal{K}} \xi_k, \tag{5.4.1b}$$

$$\xi_k \geq \begin{cases} (\zeta^*)^{\omega,k}, & \text{if } \bigwedge_{t\in T^{\omega,k}} I_{\omega,t}(x), \\ \zeta_{\min}, & \text{otherwise,} \end{cases} \qquad k\in\mathcal{K},\ \omega\in\Omega, \tag{5.4.1c}$$

$$0 \geq -\bar{n}^{\omega,k} U + \sum_{\substack{(i,\,j)\in\mathcal{A}:\\ j\notin\{0\}}} \bar{m}_{ij}^{\omega,k} \left(p_j x_{ij}^k - M_{\mathrm{U}}\left(1 - x_{ij}^k\right)\right), \qquad k\in\mathcal{K},\ \omega\in\Omega, \tag{5.4.1d}$$

$$\begin{aligned} 0 \geq \sum_{i\in\mathcal{V}} \left(\bar{\alpha}_i^{\omega,k} e_i - \bar{\beta}_i^{\omega,k} l_i\right) + \sum_{i\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}} \bar{\gamma}_i^{\omega,k} \left(t_{i0} + s_i - T\right) \\ + \sum_{\substack{(i,\,j)\in\mathcal{A}:\\ j\notin\{0\}}} \bar{\delta}_{ij}^{\omega,k} \left(\left(t_{ij} + s_i\right) - M_{\mathrm{T}}\left(1 - x_{ij}^k\right)\right), \quad k\in\mathcal{K},\ \omega\in\Omega, \end{aligned} \tag{5.4.1e}$$

$$1 = \sum_{k\in\mathcal{K}} \sum_{(i,\,j)\in\mathcal{A}} x_{ij}^k, \qquad i\in\mathcal{V}_{\mathrm{c}}, \tag{5.4.1f}$$

$$1 \geq \sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k, \qquad k\in\mathcal{K},\ i\in\mathcal{V}, \tag{5.4.1g}$$

$$\sum_{j\in\mathcal{V}:(i,\,j)\in\mathcal{A}} x_{ij}^k = \sum_{j\in\mathcal{V}:(j,\,i)\in\mathcal{A}} x_{ji}^k, \qquad k\in\mathcal{K},\ i\in\mathcal{V}, \tag{5.4.1h}$$

$$K \geq \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}_{\mathrm{r}}\cup\mathcal{V}_{\mathrm{c}}} x_{0j}^k, \tag{5.4.1i}$$

$$x_{ij}^k \in \{0,1\}, \qquad k\in\mathcal{K},\ (i,\,j)\in\mathcal{A}. \tag{5.4.1j}$$

The constraints (5.4.1b–c) contain the optimality cuts generated from the battery subproblem, where (5.4.1c) define approximate values of the subproblem objective for each vehicle. Here $T^{\omega,k}$ indexes the leaf nodes of the branch-and-bound tree from iteration ω for vehicle k, and I_t are the inequalities associated with these leaf nodes. If the route x^k for vehicle k fulfils the logical clause

$$\bigwedge_{t\in T^{k,r}} I_{\omega,t}(x^k)$$

for some iteration ω, where $I_{\omega,t}(x)$ are the inequalities (5.2.4), the proof that $(\zeta^*)^{\omega,k}$ is a lower bound remains a valid proof. If no bounds can be proven, $\zeta_{\min}$ (see (5.3.2) and (5.3.3)), is used as a default lower bound. The constraints (5.4.1d) correspond to the feasibility cuts generated from the cargo subproblems, and the constraints (5.4.1e) corresponds to the feasibility cuts from the time window subproblem. Finally, the constraints (5.4.1f–i) define feasible routes. To improve the performance of the logic-based Benders decomposition algorithm, a relaxed version, see (5.3.1), of the battery problem is added as an initial cut:

$$\xi_k \geq \sum_{(i,\,j)\in\mathcal{A}} \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right) r^{\mathrm{fu}} d_{ij} x_{ij}^k$$

5.5 Algorithm

In this section we go through more of how each iteration of our LBBD algorithm functions. The full logic-based Benders decomposition algorithm is summarised in Algorithm 2.

Algorithm 2 Logic-based Benders decomposition algorithm for the problem in Chapter 4

choose an initial route and set $\bar{x}_{ij}^k$ to the corresponding values
let $\bar{x}$ be the vector containing the values of $\bar{x}_{ij}^k$
set $\zeta^* := \infty$, $\xi := -\infty$, where ζ^*/ξ is the current upper/lower bound
    on the objective value of the complete problem (4.2)
while $\zeta^* > \xi$ **do**
    for each vehicle $k \in \mathcal{K}$ **do**
        solve the battery subproblem (5.2.1) for $\bar{x}$
        for each leaf node in the solution of the battery subproblem **do**
            let I_t be as in (5.2.4)
        end for
        solve the time window (5.1.2) and the cargo subproblem (5.1.5) for $\bar{x}$
        add Benders cuts (5.3.5), (5.1.3), and (5.1.6) to the master problem
    end for
    let ζ^* be the sum of subproblem objective values.
    solve the master problem (5.4.1)
    if master problem infeasible **then**
        original problem infeasible
    else
        set $\bar{x}$:= optimal route obtained from the master problem
        set ξ := objective value of route $\bar{x}$.
    end if
end while

Algorithm 2 starts by choosing an initial route $\bar{x}$. Let ζ^* be the current best upper bound on the objective value, which is updated each time a feasible solution of the subproblem is found. The variable ξ acts as a current best lower bound on the optimal objective value due to the fact that all cuts are valid, i.e. no feasible solutions are cut of from the master problem.

Each iteration starts by solving the subproblems. Cuts from the time window and cargo capacity subproblem are obtained as in the classical Benders algorithm, where a valid cut is constructed from the optimal solution of the respective LP duals. The battery capacity subproblem is an MBLP, and cuts are obtained by inspecting the leaf nodes of the branch-and-bound tree used to solve the problem. From each leaf node t an inequality I_t, as defined in (5.2.4), is obtained, and the objective value is a valid lower bound as long as these inequalities are all satisfied. After all cuts are generated, the master problem is solved. The optimal route is fixed and if the upper and lower bounds are not equal, the algorithm starts a new iteration by solving the subproblems again. The algorithm continues generating cuts and resolving the master problem until the optimal solution is found and verified.

6

Numerical Tests and Results

In this chapter, we present the numerical results of our logic-based Benders decomposition algorithm implementation. First, details of the implementation is outlined in Section 6.1. The data set that is used to test the algorithm is described in Section 6.2, and in Section 6.3 we go through our test setup. Then, a discussion of the performance of the implementation follows in Section 6.4. As a part of this discussion, we conclude the chapter with a discussion of the quality of our Benders cuts in Section 6.4.1.

6.1 Implementation Details

The logic-based Benders decomposition algorithm was implemented in C++98 using IBM ILOG CPLEX 12.1 to solve the master problem and the node problems. The battery subproblem was solved by a manual branch-and-bound implementation. Tests were performed on a standard PC running Linux with an Intel® Core™ i5-3470S CPU at 2.90 GHz and 16 GB of RAM.

6.2 Test Data

As a preliminary test instance, we created a six node data set which we will henceforth refer to as P6. This instance consists of four customers, one depot, and one recharge station, and it is presented in full in Appendix A. The parameters of the instance were chosen in such a way as to enforce a nontrivial solution. More specifically, we required that all resource constraints play a role in the optimal solution and that the battery size chosen was neither of the extreme cases (i.e. smallest or largest available). As long as a vehicle is able to deplete its battery and the battery charge level is zero when arriving at a recharge station, the distance that the vehicle can drive on electricity for a specific route is linearly dependent on the battery capacity. Fulfilling our requirement that a non-extreme value of the battery sizes be chosen then necessitates a nonlinear cost of increasing the battery capacities. Instance P6 allowed us to verify the validity of our implementation in the initial stages of development.

Our bigger test data sets were downsampled from a benchmark set generated by Schneider, Stenger, and Goeke [45], which in turn is based on the data sets for VRP with time windows of Solomon [46]. Their data set adds recharge stations to instances with 100 nodes so that each node can be reached from the depot using at most two recharge stations. The data set also includes time windows and demands for customers. We randomly sampled nodes from the specific 100 node test instance C101 (from [45]) in order to create our test data set, where customers and recharge stations were sampled from independently. The number of customer nodes sampled varied between four and 19, but we kept the number of recharge stations sampled fixed to two. Since the data sets from [45] were made for a pure electric VRP with time windows, we had to produce a selection of battery types for the hybrid vehicles. We adopted the parameter values

generated for instance P6, and these proved to still yield interesting solutions, i.e. they impacted the route chosen. For all versions of the downsampled problems, we used a maximum of two vehicles.

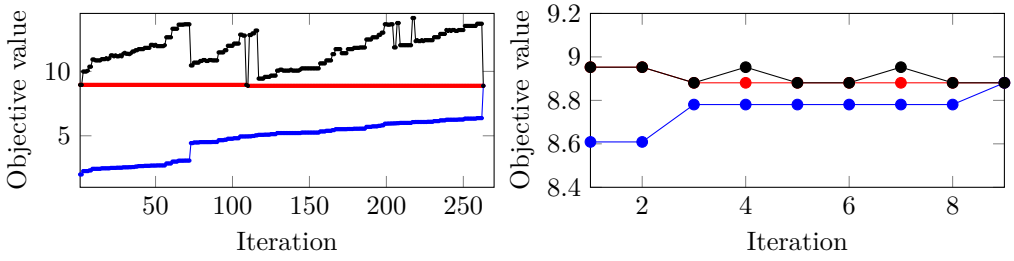
6.3 Test Setup

In its original implementation, our algorithm was too slow to test on the larger problem instances, and instead the problem instance P6 (see Appendix A) was used as a test problem to diagnose performance issues. We found that keeping time window and cargo constraints in the master problem sped up solution times while not affecting the battery subproblem, which was our main focus. All the results presented in this chapter use this decomposition.

We performed two sets of tests. Our first set of tests was done on problem instance P6. We used two different default lower bounds, (5.3.2) and (5.3.3), and compared the performance of our implementation on these. Our second set of tests was done on randomly downsampled problem instances (from C101) of different node counts. These tests were done only with the tighter default lower bound (5.3.4). For each node count, we generated and solved several problem instances. The larger problem instances were very time consuming, and we used a maximum computation time for each node count of 60/300 CPU seconds, depending on the test.

6.4 Test Results and Algorithm Performance

Bounds on the objective value for each iteration of using our algorithm on instance P6 are shown in Figure 6.1. In each iteration, a feasible solution is found from the trial route and the best value in all previous iterations provides a upper bound. The lower bounds are obtained from the objective function in the Benders master problem. When using the simple default lower bound (5.3.3), the optimal solution was found at iteration 110 for the test instance P6, but it took until iteration 263 before the algorithm concluded that this was indeed the optimal solution; see Figure 6.1a. The total calculation time was 7276 CPU seconds, and of those, 7049 CPU seconds were spent solving the master problem. The tighter bound (5.3.3) significantly improved the performance of our algorithm, and it took nine iterations before the optimal solution was found; see Figure 6.1b.



(a) Simple default lower bound

(b) Tight default lower bound

Figure 6.1: Upper (red) and lower (blue) bounds, together with the objective value of the trial routes (black), in each iteration of the logic-based Benders decomposition algorithm for instance P6 using different default lower bounds. The simple default lower bound was computed as in (5.3.3), and the tight as in (5.3.4). The optimal objective value was 8.881.

When solving the instance P6 with the simple bound, the CPU time for each iteration of the master problem was recorded, as illustrated in Figure 6.2. The initial master problem was solved in 0.12 CPU seconds, but at later iterations this took several CPU seconds. After iteration 200, each iteration took more than 30 seconds. This was due to more cuts being added to the master problem and since the cuts from the battery subproblem were non-linear, the master problem quickly became very time consuming.

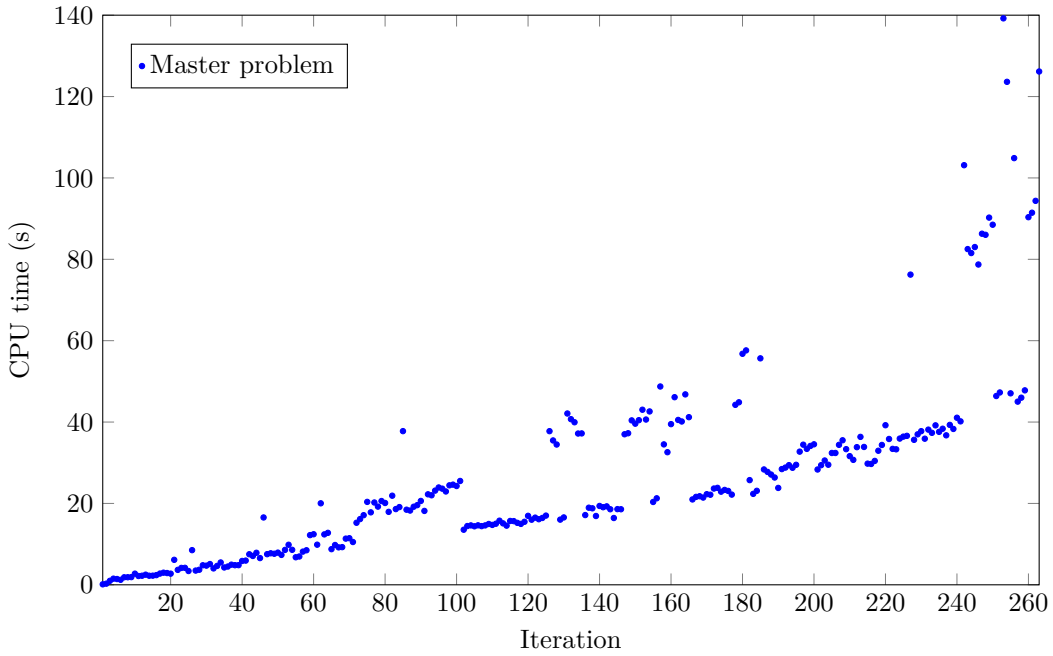


Figure 6.2: Computation times in CPU seconds for solving the master problem of the instance R6 in each iteration of the logic-based Benders decomposition algorithm.

Every cut generated in each test only cut away one solution point in the master problem, leading to potentially full enumeration of the search space of x_{ij}^k. This is elaborated upon in Section 6.4.1. The number of feasible routes of the test instance R6, ignoring the resource constraints, was found by numerical calculation to be 2680. For an arbitrary n, a general closed-form formula for the number of routes is hard to find, but a lower bound can be computed by ignoring the recharge stations. Without the recharge stations, the number of routes for k vehicles is

$$n! \cdot \binom{n+k-1}{k-1} = \frac{(n+k-1)!}{(k-1)!}.$$

The real number of feasible routes is smaller than this, since some of the routes counted might violate cargo or time window constraints. For example, instance R6 has 1026 routes that fulfil its resource constraints. For more customers, the set of feasible routes quickly becomes too large to solve with enumeration, e.g. for $n = 15$ the number of routes is larger than 10^{12}. Since the master problem is computationally expansive, see Figure 6.3, it becomes infeasible to solve larger problems in a reasonable timeframe.

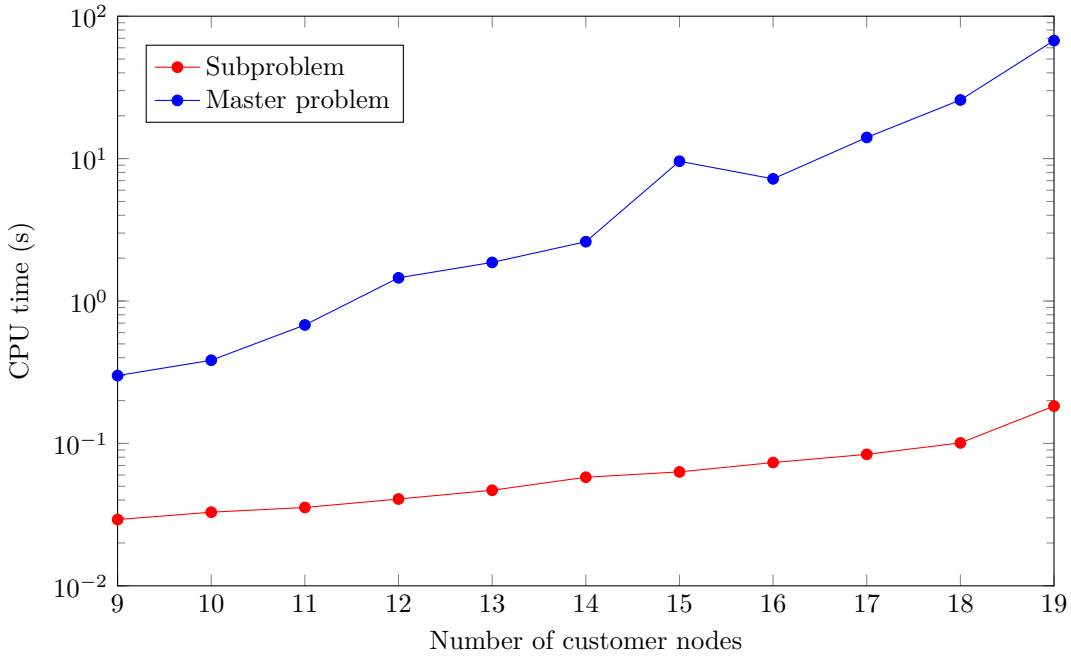


Figure 6.3: Average computation times in CPU seconds for the first iteration of the master and subproblem in the logic-based Benders decomposition algorithm. Each problem instance contains one depot and two recharge stations. For each number of customer nodes the problem was repeatedly solved for a new random sampling of a larger instance until 300 CPU seconds had elapsed.

Another issue is that the default lower bound $C_{\min}$ from (5.3.2) is much lower than the objective value of any reasonable route. For our test instance R6 we obtained $C_{\min} = -40.55$ and the value of the two subproblems in the optimal solution were $\zeta_1 = -0.017$ and $\zeta_2 = -0.089$, respectively. The bound for instance P6 using (5.3.3) was $C_{\min} = -9.655$, which is still much lower than the objective value of a normal route. This means that the first time the master problem obtains the optimal solution as a new trial value, the algorithm will not stop, and will continue checking every solution that in the master problem is only bounded by the default lower bound $C_{\min}$. If the gap between the lower bound and optimal solutions of the subproblem were smaller, the master problem would not check as many poor trial values before concluding that the optimal solution had already been found in an earlier iteration. Before iteration 73 both vehicles only used the default bound, and only after iteration 73 did one of the vehicles use bounds from earlier iterations. This can be seen in the jump of the lower bound in Figure 6.1a. In iteration 263 neither of the vehicles used the default bound, which is the cause of the sudden jump up to the optimal value.

The function (5.3.4) is a much tighter bound, giving $C_{\min} = -0.189$, and this results in much better performance of our algorithm. The cuts still only give a tight lower bound for one route, but with a better global lower bound, the algorithm does not test as many solutions in the master problem before finding the optimal solution. Let ξ^* be the objective value of the optimal solution and let $\Delta\xi$ be the global lower bound on the subproblem values. Only routes with a objective value of at most $\xi^* + \Delta\xi$ are tested. Similar jumps as for the simple bound can be seen in Figure 6.1b; see Table 6.1.

Table 6.1: The total number of iterations needed to solve test instance P6. The lower bounds in Figure 6.1 jumps at two iterates. Jumps one/two corresponds to when the algorithm has obtained a non-default bound for one/two of the vehicles, respectively.

Default lower bound	Total # iterations	Jump one	Jump two
Simple (5.3.3)	9	3	9
Tight (5.3.4)	263	73	263

We tested the performance of the algorithm on the randomly downsampled instances using the tighter default lower bound (5.3.4). Computation times using this bound is presented in Figure 6.4; CPLEX is used as comparison, and it is faster by approximately a factor of ten. The full data for the computation times for our LBBD implementation is presented in Appendix B. Of note is that some of the tests took much longer than the average.

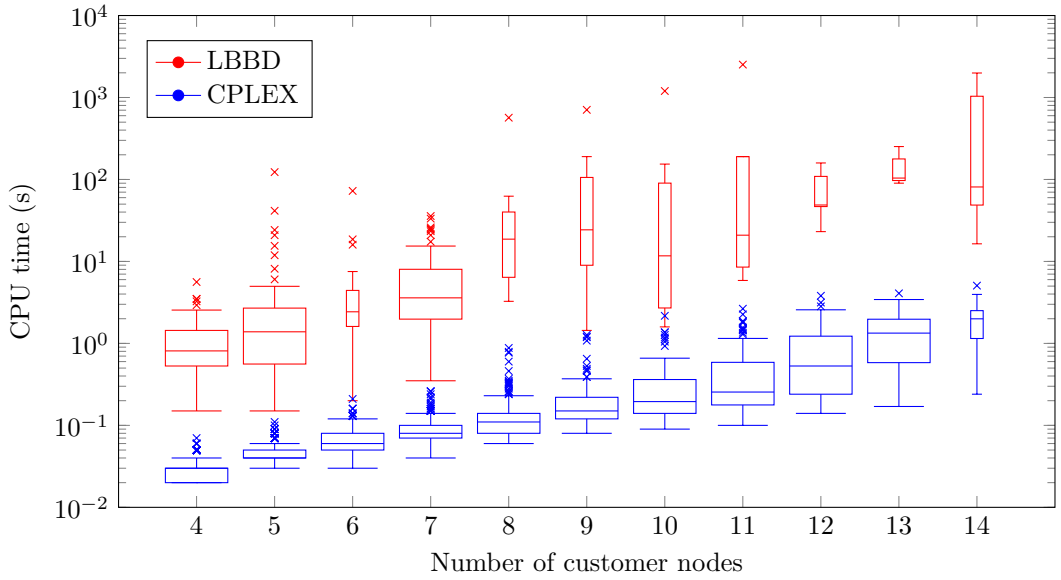


Figure 6.4: Computation times in CPU seconds for the logic-based Benders decomposition algorithm with tight default lower bound presented as a boxplot. The default lower bound was computed as in (5.3.4). As a comparison the average computation times in CPU seconds for solving the full problem with CPLEX are also shown. For each customer count, randomly downsampled instances were solved. The boxes have lines at the median, the upper quartile, and the lower quartile. Whiskers are placed at the smallest/largest data point which is larger/smaller than lower/upper quartile $\pm\, 1.5 \cdot \text{IQR}$, where IQR is the inter-quartile-range. Points outside this range are outliers (cross). The width of the boxes are scaled according to the sample size.

6.4.1 On the Strength of the Logic-Based Benders Cuts

As mentioned in Section 6.4, our cuts from the subproblem only removed one point from the master problem at a time. In other words, the cuts were equivalent to so-called no-good cuts. This is certainly not always the case for the logic-based Benders method [24], and we see no theoretical reason to believe that it will always be the case for our specific VRP and decomposition.

Moreover, even when presented with cuts for an explicit instance of our problem, it is a nonobvious conclusion to draw that only one point is being cut away. In this section we therefore describe a sufficient condition that we devised to show that a cut generated in the battery subproblem is equivalent to a no-good cut.

Consider first a relaxed node problem from the branch-and-bound solution to a general MILP (3.5.2), that is

$$\min_x \; z := cx, \tag{6.4.1a}$$
$$\text{s.t.} \quad Ax \geq a, \tag{6.4.1b}$$
$$Hx \geq h, \tag{6.4.1c}$$
$$-x \geq -b, \tag{6.4.1d}$$
$$x \geq 0, \tag{6.4.1e}$$

and let $\bar{z}$ be its optimal value. Associate the dual variable vectors λ, μ, and ν with the constraints (6.4.1b), (6.4.1c), and (6.4.1d), respectively. Combining (3.5.6a) and (3.5.7), we obtain the relation

$$(\lambda A - c)x \not> \lambda a - \bar{z}, \tag{6.4.2}$$

which holds for any optimal integer solution x to the node problem at hand. Moreover, the following system is feasible:

$$-cx \geq -\bar{z} \tag{6.4.3a}$$
$$Ax \geq a \tag{6.4.3b}$$

Multiplying the inequality (6.4.3a) by one and the inequality (6.4.3b) by $\lambda \geq 0$ and adding them yields

$$(\lambda A - c)x \geq \lambda a - \bar{z}, \tag{6.4.4}$$

which is also satisfied for any optimal solution x to the node problem. Together (6.4.4) and (6.4.2) prove equality whenever x is an integer optimal solution to the node problem., or in other words that

$$(\lambda A - c)x = \lambda a - \bar{z}. \tag{6.4.5}$$

For our battery subproblem (see Section 5.2), the dual vector λ corresponds to the variables μ_{ij}, π_{ij}, σ_1, and σ_2, and we thus obtain the following expression for λa:

$$\begin{aligned}\lambda a &= \sum_{(i,j)\in\mathcal{A}} -r^{\text{el}} d_{ij}\mu_{ij}x_{ij}^k + \sum_{(i,j)\in\mathcal{A}} -M_Q\left(1 - x_{ij}^k\right)\pi_{ij} + \sigma_1 - \sigma_2 \\ &= \sum_{(i,j)\in\mathcal{A}} \left(M_Q\pi_{ij} - r^{\text{el}} d_{ij}\mu_{ij}\right)x_{ij}^k + \left(\sigma_1 - \sigma_2 - \sum_{(i,j)\in\mathcal{A}} M_q\pi_{ij}\right)\end{aligned}$$

If we now assume that the dual solutions of π_{ij} and μ_{ij} to the relaxed node problems have the properties

$$\pi_{ij} = 0, \;\; (i,j) \in \mathcal{A}_0, \tag{6.4.6a}$$
$$\mu_{ij} > 0, \;\; (i,j) \in \mathcal{A}_0, \tag{6.4.6b}$$
$$\pi_{ij} > 0, \;\; (i,j) \in \mathcal{A}_1, \tag{6.4.6c}$$
$$\mu_{ij} = 0, \;\; (i,j) \in \mathcal{A}_1, \tag{6.4.6d}$$

it follows that λa has the structure

$$\lambda a = \sum_{(i,\,j)\in\mathcal{A}_0} -C_0 x_{ij}^k + \sum_{(i,\,j)\in\mathcal{A}_1} C_1 x_{ij}^k + C,$$

where C_0, C_1, C do not depend on x_{ij}^k, and C_0, $C_1 \geq 0$. Here $\mathcal{A}_1$ denotes the set of all indices $(i,\,j)$ such that the variables x_{ij} are fixed to one in the subproblem, and $\mathcal{A}_0$ denotes the set of indices of those that are fixed to zero, like in Section 3.5.1. It is then straightforward to see that λa attains its maximum value only if $x_{ij}^k = 0$ for all $(i,\,j) \in \mathcal{A}_0$, and $x_{ij}^k = 1$ for all $(i,\,j) \in \mathcal{A}_1$. Combining this with Equation (6.4.5), we can conclude that there is only *one* route that violates the surrogate inequality $I_{\bar{x},t}(x)$ in inequality (5.2.4). In other words, the assumptions (6.4.6) constitute a sufficient condition for the generated cut to be a no-good.

For the general battery subproblem, we do not see any reason to believe that the condition (6.4.6) on the solution values of π_{ij} and μ_{ij} will always hold. In fact, only on the following very simplified version of the dual to the relaxed node problem were we able to conclude that this must always be the case:

$$\begin{aligned}
\max_{\pi,\,\mu} \quad & \sum_{(i,\,j)\in\mathcal{A}} \left(-M_Q\left(1 - x_{ij}^k\right)\right)\pi_{ij}^k + \sum_{(i,\,j)\in\mathcal{A}} (-r^{\mathrm{el}} d_{ij} x_{ij}^k)\mu_{ij}^k, \\
\text{s. t.} \quad & \\
& -\pi_{ij}^k - \mu_{ij}^k \leq \left(c^{\mathrm{el}} - \frac{r^{\mathrm{fu}}}{r^{\mathrm{el}}} c^{\mathrm{fu}}\right), \quad (i,\,j) \in \mathcal{A}, \\
& \pi_{ij}^k, \mu_{ij}^k \geq 0, \qquad\qquad\qquad\quad (i,\,j) \in \mathcal{A}.
\end{aligned}$$

However, the advantage of the condition (6.4.6) is that it can be used to diagnose any concrete cut obtained during runtime, and in our numerical observations from actual experimental data this indeed seemed to universally be the case; in practice, all cuts we generate satisfy the condition (6.4.6).

7
Discussion

In this chapter, we discuss our method and our results from a performance perspective. To begin with, a summary of our main results and their performance is found in Section 7.1. Following that, we also describe potential improvements that could be made to our method and areas for further research in Section 7.2.

7.1 Summary of Results

The aim of this project was to formulate and implement a logic-based Benders decomposition algorithm, and as a consequence, to formulate an inference dual for our MBLP. This procedure has two main components: Generating a proof scheme to enforce strong duality, and extracting sensitivity information to be able to use the dual for estimation. As a proof scheme we used the structure of the primal branch-and-bound method. In this tree, certain inequalities are assigned to each leaf node, which facilitates a sensitivity analysis. We formulated the method for a specific VRP variant with hybrid vehicles and variable battery capacities, and implemented this method in C++. To solve the LP subproblems we used the IBM ILOG CPLEX toolkit. Nothing special was done to treat the nonlinear cuts and they were sent directly to the master problem in CPLEX.

We tested our method on instances with totally six to 16 nodes with varying degrees of success. Cuts were generated from a MILP subproblem, and we could solve these relatively small subproblem instances using our implementation of the logic-based Benders decomposition algorithm. For our test instances, our implementation was slower than a straightforward solution of the original MILP formulation using CPLEX even when we used cuts with tight default lower bounds. When cuts used the worse and more naive lower bounds, computation times increased drastically. We found that the cuts were not as good as we had expected, and the proof of each lower bound would only hold for the particular route that was used to generate the bound. This is problematic since it leads to a partial explicit enumeration of the search space. The hope was that lower bounds would be valid for more than one route, so that more information from the subproblem structure would be embodied by each cut. It is plausible that if applied to a larger instance, the cuts could be stronger and might exclude several routes from the master problem. However, since the method is too slow to scale to particularly large problems, this was not possible to verify.

The method was tested with two variations of default lower bounds in the cuts. The difference in the performance of the simple and the tighter default lower bound can be seen in Figure 6.1. This better lower bound drastically improved performance, which indicates that is important to find a tight lower bound on the subproblem objective value. If the bound is poor, the algorithm gets stuck checking solutions that are not very good during the initial iterations. Using a better lower bound speeds up the early iterations, so that the algorithm does not spend time in the beginning considering poor solutions.

Only a cursory examination was performed on the effects of battery capacities and costs. We used the same non-linear relationship between them in all tests and changed only their magnitudes. If the battery capacities were too small compared to the distances in the problem, the battery subproblem had little effect on the optimal solution. The same was true if the capacities were too large. As previously stated, we chose to mostly test the performance of the battery subproblem. The time window and cargo capacity subproblems were not of as great interest, since they are simpler variants of the battery subproblem and only need standard LP duality to produce cuts. Solving the master problem with time windows and cargo capacities was also not noticeably slower than solving the master problem without them.

7.2 Further Research

Broadly, improvements can be classified into two categories. First, there are smaller-scale improvements which can be made. These constitute the subject of Section 7.2.1. Although they are described as smaller-scale, they might potentially have a large impact on the performance of our algorithm, and due to technicalities some might be rather time consuming to implement. However, they clearly contrast with the second category: large structural changes to our method. These are broader reworkings of our method based on the same idea of generalised duality and Benders decomposition, and are discussed in Section 7.2.2.

7.2.1 Minor Improvements

Several potential improvements could be made to our Benders algorithm. As mentioned above, when we tightened the default lower bounds in our Benders cuts, we obtained a substantial increase in performance. It is possible that these default lower bounds could be tightened further, leading to better performance.

In an attempt to improve the cuts, the symmetries present in the model could be exploited. For instance, a cut generated by one vehicle is valid for all vehicles since the vehicles are identical in the master problem. We did not test if using more cuts improved the performance of the algorithm, or if the additional complexity would result in slower solution times.

There is also a potential problem in how our model avoids so called subtours (degenerate routes consisting of disjoint loops). These are prevented by the inclusion of the time variables used to enforce time window constraints in the model. A time variable is strictly increasing in each route, which prevents a route from ever forming a loop. These time variables generalise the so called Miller-Tucker-Zemlin (MTZ) constraints; see [3]. The MTZ constraints are useful because they avoid the traditional explicit subtour elimination constraints, which are exponentially many in the node count of the problem. However, the MTZ constraints are known to have poor linear relaxation, as described in [3]; the authors present several remedied MTZ constraints with better relaxation, motivated by polyhedral theory. However, it is not immediately obvious if these ideas generalise to the time variable setting. One practical option might be to add a subset of the traditional subtour elimination constraints to our master problem, preventing for example three-node and four-node loops. These constraints would be redundant in the sense that they would not change the optimal solution of the problem, but they might improve the quality of the linear relaxation of the master problem, leading to faster solution speeds.

It is possible that the flow formulation with big-M constants performs poorly when we decompose the problem in such a way that the big-M constants are included in the cuts from the subproblems. Models using big-M constants are known to have poor relaxations. These big-M constants might be the cause of the poor quality of our cuts. There are ways of avoiding big-M

constants by instead considering the disjunctions that they ultimately represent; see for instance [25, 26, 35]. None of these seem straightforward to implement, since they require a different branching scheme than what CPLEX offers.

We only tested one particular partial decomposition, but it is probable that there exists some way to split the problem which is more advantageous. A decomposition could be better if the cuts contain more information about the structure of the subproblem. It could also be better if there is a better balance between subproblem and master problem complexity. In the first case, cuts would hopefully generate valid lower bounds for more than a single route, leading the algorithm to no longer check trial values one by one. In the second case, the master problem would be simpler, and less time would be spent re-solving it. Both of these are problematic with our decomposition; our subproblem is likely too simple, so that the cuts are too weak. At the same time, our master problem is a very complex problem in its own right, and it is time consuming to solve repeatedly.

7.2.2 Structural Improvements to the Benders Algorithm

In our implementation of a logic-based Benders algorithm, we used the strategy outlined in Figure 3.3. This is how the Benders decomposition algorithm is usually formulated, but it turns out to not be a very effective way, for several reasons:

- The master problem has to be re-solved in every iteration.
- The master problem becomes more time consuming to solve as more cuts are added.
- Full convergence may take many iterations after an optimal point has been found.

Some of these problems are remedied in so called 'modern Benders', as mentioned in [15, 16]. While classical Benders is a cutting plane method, modern Benders takes this idea and reformulates it as a branch-and-cut method. Concretely, this means that instead of repeatedly solving the master problem and generating cuts, the master problem is solved only once by branching. Cuts are successively generated in the nodes of the search tree during the branching procedure in a so called cut loop.

Fischetti, Ljubić, and Sinnl [15] also propose that while it is important to produce the best possible cuts, it is even more important to use a better cut loop. Traditionally, this would often be Kelley's cut loop [29]. This cut loop is similar in structure to the Benders decomposition algorithm that we have implemented, as in Figure 3.3. A problem with Kelley's cut loop is that it may zigzag towards the optimal solution during the initial iterations, which significantly slows down convergence. Several other cut loops have been proposed to stabilise the method. Among others, these include bundle methods [36] and *in-out* methods [4].

It would be interesting to test if these methods could be extended to work with the logic-based Benders decomposition algorithm, since it seems to exhibit the same problems that classical Benders does. A potential complication is to determine whether the decomposition into master and subproblem would change, and if so, how. Such a change might then necessitate another inference dual than a branching dual for cut generation. It seems likely that a logic-based Benders decomposition algorithm could be implemented in a single search tree, like a modern Benders implementation. This would bear some similarities with the branch-and-check method [48]. However, the alternative cut loop strategies do not seem directly compatible with nonconvex problems, and it would be an additional challenge to try to overcome this.

Another difficulty is that generally, cuts generated by the logic-based Benders decomposition algorithm need not be linear. In our implementation the nonlinear cuts are modelled using extra logical variables, increasing the size of the problem, and subsequently resulting in slower computations. This can be addressed by instead solving the master problem by branching. Cuts

are then added to the master problem during the branching procedure [25]. Without the extra logical variables the master problem would be faster to solve.

It would also be interesting to test a different decomposition of the model with the hope that the cuts generated would be of better quality. For example Riazi, Seatzu, Wigström, and Lennartson [42] decomposed the problem into an assignment master problem and a travelling salesman subproblem, solving the former with constraint programming, and the latter with the special-purpose TSP solver Concorde [10]. For our model this decomposition would be slightly different since we have a different initial VRP model.

7.3 Conclusion

We have utilised logic-based Benders decomposition to partition a VRP into a routing master problem and a resource constraint subproblem. The VRP under consideration incorporated time windows, cargo capacities, and hybrid vehicles with variable battery capacities. The aim of this project was to devise a proof scheme for solving the resource constraint subproblem, and a method for generating sensitivity information from that solution. Since, the subproblem was an MBLP, ordinary LP duality theory could not be applied in the same way it would have been for the classical Benders decomposition. Instead, the solution tree obtained by solving the primal subproblem by branch-and-bound was viewed as proving optimality by refutation. We formulated necessary and sufficient conditions for which other routes this proof remains valid. These conditions allowed the formulation of the Benders cuts.

The logic-based Benders decomposition algorithm is a flexible method, and solution methods can be tailored individually for the master and subproblem. This flexibility comes with the difficulty of that a method for generating cuts has to be devised for each solution method since the logic-based Benders decomposition algorithm is mostly a framework. Formulating these cuts can be an arduous process, which limited the scope of the study to in regards to solution methods for the subproblem. Only one particular problem decomposition was considered.

Compared to directly using CPLEX to solve the full problem, our method is an order of magnitude slower across all test instances. Further work needs to be done to establish whether this method performs better for larger problem instances. We have investigated the reasons for the poor performance, and suggest three main areas for further research.

Firstly, this study has shown that each cut we generate using this method contain only information of a single route, which is one of the main reasons for the relatively poor performance. A natural progression of this work is to study if a different decomposition would result in better cuts, that contain information about several routes.

Secondly, the numerical results highlight the importance of finding good bounds before initiating the algorithm, as otherwise the method spends a long time at the start considering suboptimal solutions. The initial cuts presented work well for our specific data, but more general cuts would be very beneficial for improving the method. A relaxation of the subproblem might be added to the master problem in hopes of improving performance at the start of the algorithm. However, it is not immediately obvious which relaxation to use in this case.

Finally, several of the ideas from ‘modern Benders’ are of interest and could possibly be extended to work with the logic-based Benders decomposition algorithm. This would however require a much more comprehensive study.

In summary, as we chose to formulate the method for our VRP, the method results in relatively poor performance. However, logic-based Benders decomposition still seems to be an interesting framework for exact solution methods due to its flexibility, especially the possibility of solving each part of the decomposition using the most efficient method.

Bibliography

[1] S. Arora and B. Boaz. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[2] R. Baldacci, N. Christofides, and A. Mingozzi. 'An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts'. *Mathematical Programming*, 115(2):351–385, 2008. DOI: 10.1007/s10107-007-0178-5.

[3] T. Bektaş and L. Gouveia. 'Requiem for the Miller–Tucker–Zemlin subtour elimination constraints?' *European Journal of Operational Research*, 236(3):820–832, 2014. DOI: 10.1016/j.ejor.2013.07.038.

[4] W. Ben-Ameur and J. Neto. 'Acceleration of cutting-plane and column generation algorithms: applications to network design'. *Networks*, 49(1):3–17, 2007. DOI: 10.1002/net.20137.

[5] J. F. Benders. 'Partitioning procedures for solving mixed-variables programming problems'. *Numerische Mathematik*, 4(1):238–252, 1962. DOI: 10.1007/bf01386316.

[6] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. 'Static pickup and delivery problems: a classification scheme and survey'. *TOP*, 15(1):1–31, 2007. DOI: 10.1007/s11750-007-0009-0.

[7] N. Christofides and S. Eilon. 'An algorithm for the vehicle-dispatching problem'. *OR*, 20(3):309–318, 1969. DOI: 10.2307/3008733.

[8] Y. Chu and Q. Xia. 'Generating Benders cuts for a general class of integer programming problems'. In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. Ed. by J.-C. Régin and M. Rueher. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, 127–141. DOI: 10.1007/978-3-540-24664-0_9.

[9] E. Coban and J. N. Hooker. 'Single-facility scheduling by logic-based Benders decomposition'. *Annals of Operations Research*, 210(1):245–272, 2013. DOI: 10.1007/s10479-011-1031-z.

[10] W. J. Cook. *Concorde TSP Solver*. URL: http://www.math.uwaterloo.ca/tsp/concorde (visited on 2018-02-08).

[11] G. B. Dantzig and J. H. Ramser. 'The truck dispatching problem'. *Management Science*, 6(1):80–91, 1959. DOI: 10.1287/mnsc.6.1.80.

[12] G. Dantzig, R. Fulkerson, and S. Johnson. 'Solution of a large-scale traveling-salesman problem'. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954. DOI: 10.1287/opre.2.4.393.

[13] G. B. Dantzig and M. N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer US, 2003. DOI: 10.1007/b97283.

[14] G. Desaulniers, F. Errico, S. Irnich, and M. Schneider. 'Exact algorithms for electric vehicle-routing problems with time windows'. *Operations Research*, 64(6):1388–1405, 2016. DOI: 10.1287/opre.2016.1535.

[15] M. Fischetti, I. Ljubić, and M. Sinnl. 'Benders decomposition without separability: a computational study for capacitated facility location problems'. *European Journal of Operational Research*, 253(3):557–569, 2016. DOI: 10.1016/j.ejor.2016.03.002.

[16] M. Fischetti, I. Ljubić, and M. Sinnl. *Modern Benders (in a nutshell)*. 2017. URL: http://www.dei.unipd.it/~fisch/papers/slides/2017%20Lunteren%20%5BFischetti%20on%20Benders%5D.pdf.

[17] A. M. Geoffrion. 'Generalized Benders decomposition'. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972. DOI: 10.1007/BF00934810.

[18] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen. 'Implementing vehicle routing algorithms'. *Networks*, 7(2):113–148, 1977. DOI: 10.1002/net.3230070203.

[19] Gurobi Optimization. *Gurobi*. URL: http://www.gurobi.com (visited on 2018-01-26).

[20] J. N. Hooker. 'Inference duality as a basis for sensitivity analysis'. *Constraints*, 4(2):101–112, 1999. DOI: 10.1023/A:1009838725226.

[21] J. N. Hooker. 'A hybrid method for the planning and scheduling'. *Constraints*, 10(4):385–401, 2005. DOI: 10.1007/s10601-005-2812-2.

[22] J. N. Hooker and M. W. Dawande. 'Inference-based sensitivity analysis for mixed integer/linear programming'. *Operations Research*, 48(4):623–634, 2000.

[23] J. N. Hooker and N. R. Natraj. 'Solving a general routing and scheduling problem by chain decomposition and tabu search'. *Transportation Science*, 29(1):30–44, 1995. DOI: 10.1287/trsc.29.1.30.

[24] J. N. Hooker and G. Ottosson. 'Logic-based Benders decomposition'. *Mathematical Programming*, 96(1):33–60, 2003. DOI: 10.1007/s10107-003-0375-9.

[25] J. N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, Inc., 2000. DOI: 10.1002/9781118033036.

[26] J. N. Hooker. *Integrated Methods for Optimization*. Springer US, 2012. DOI: 10.1007/978-1-4614-1900-6.

[27] IBM. *ILOG CPLEX Optimization Studio*. 2018. URL: http://www.ibm.com/products/ilog-cplex-optimization-studio (visited on 2018-01-26).

[28] L. V. Kantorovich and V. F. Zalgaller. *Calculation of Rational Cutting of Stock*. Lenizdat, 1951.

[29] J. E. Kelley. 'The cutting-plane method for solving convex programs'. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[30] C. Kloimüllner and G. R. Raidl. 'Full-load route planning for balancing bike sharing systems by logic-based Benders decomposition'. *Networks*, 69(3):270–289, 2017. DOI: 10.1002/net.21736.

[31] E. Lam and P. V. Hentenryck. 'A branch-and-price-and-check model for the vehicle routing problem with location congestion'. *Constraints*, 21(3):394–412, 2016. DOI: 10.1007/s10601-016-9241-2.

[32] G. Laporte and Y. Nobert. 'A branch and bound algorithm for the capacitated vehicle routing problem'. *OR Spektrum*, 5(2):77–85, 1983. DOI: 10.1007/bf01720015.

[33] G. Laporte. 'Fifty years of vehicle routing'. *Transportation Science*, 43(4):408–416, 2009. DOI: 10.1287/trsc.1090.0301.

[34] G. Laporte and Y. Nobert. 'Exact algorithms for the vehicle routing problem'. In: *Surveys in Combinatorial Optimization*. Ed. by S. Martello, G. Laporte, M. Minoux, and C. Ribeiro. Vol. 132. North-Holland Mathematics Studies Supplement C. North-Holland, 1987, 147–184. DOI: 10.1016/S0304-0208(08)73235-3.

[35] J. Lee and S. Leyffer. *Mixed Integer Nonlinear Programming*. Springer, 2012. DOI: 10.1007/978-1-4614-1927-3.

[36] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. 'New variants of bundle methods'. *Mathematical Programming*, 69(1):111–147, 1995. DOI: 10.1007/BF01585555.

[37] A. N. Letchford and J.-J. Salazar-González. 'Projection results for vehicle routing'. *Mathematical Programming*, 105(2-3):251–274, 2005. DOI: 10.1007/s10107-005-0652-x.

[38] J. Lin, W. Zhou, and O. Wolfson. 'Electric vehicle routing problem'. *Transportation Research Procedia*, 12(Supplement C):508–521, 2016. Ninth International Conference on City Logistics 17-19 June 2015, Tenerife, Spain. DOI: 10.1016/j.trpro.2016.02.007.

[39] S. Mancini. 'The hybrid vehicle routing problem'. *Transportation Research Part C: Emerging Technologies*, 78(Supplement C):1–12, 2017. DOI: 10.1016/j.trc.2017.02.004.

[40] W. V. Quine. 'The problem of simplifying truth functions'. *The American Mathematical Monthly*, 59(8):521, 1952. DOI: 10.2307/2308219.

[41] W. V. Quine. 'A way to simplify truth functions'. *The American Mathematical Monthly*, 62(9):627, 1955. DOI: 10.2307/2307285.

[42] S. Riazi, C. Seatzu, O. Wigström, and B. Lennartson. 'Benders/gossip methods for heterogeneous multi-vehicle routing problems'. In: *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*. 2013, 1–6. DOI: 10.1109/ETFA.2013.6647983.

[43] P. Rubin. *Benders decomposition with integer subproblems*. 2013. URL: https://orinanobworld.blogspot.se/2013/07/benders-decomposition-with-integer.html (visited on 2018-01-15).

[44] J. Ruffieux. 'Optimization of routes for a fleet of plug-in hybrid vehicles Mathematical modeling and solution procedures'. MSc thesis. Chalmers University of Technology, 2017.

[45] M. Schneider, A. Stenger, and D. Goeke. 'The electric vehicle-routing problem with time windows and recharging stations'. *Transportation Science*, 48(4):500–520, 2014. DOI: 10.1287/trsc.2013.0490.

[46] M. M. Solomon. 'Algorithms for the vehicle routing and scheduling problems with time window constraints'. *Operations Research*, 35(2):254–265, 1987. DOI: 10.1287/opre.35.2.254.

[47] The COIN-OR Foundation. *COIN-OR*. URL: http://www.coin-or.org (visited on 2018-02-07).

[48] E. S. Thorsteinsson. 'Branch-and-check: a hybrid framework integrating mixed integer programming and constraint logic programming'. In: *Principles and Practice of Constraint Programming — CP 2001*. Ed. by T. Walsh. Springer Berlin Heidelberg, 2001, 16–30. DOI: 10.1007/3-540-45578-7.

[49] P. Toth and D. Vigo. 'Models, relaxations and exact approaches for the capacitated vehicle routing problem'. *Discrete Applied Mathematics*, 123(1):487–512, 2002. DOI: 10.1016/S0166-218X(01)00351-1.

A

Problem Data for Test Instance P6

Problem data for a small VRP with time windows, cargo constraints, and hybrid vehicle battery capacities. The data is presented in Tables A.1 to A.3.

Table A.1: Node data for the test problem with totally six nodes. The distance between two nodes is taken to be the Euclidean distance, where (x, y) are the geographical coordinates of a node. The average velocity is 1[km/h].

Node		x [km]	y [km]	demand [kg]	earliest time [h]	latest time [h]
Depot	A	10	20	0	0	200
Customer	B	10	10	30	0	100
”	C	15	5	30	20	50
”	D	20	30	30	0	200
”	E	20	10	30	0	200
Recharge	R	15	15	0	0	200

Table A.2: Parameter data for the test problem instance with six nodes.

Parameter	Value
Load Capacity [kg]	100
Fuel Consumption Rate [l/h]	0.1
Battery Consumption Rate [kWh/h]	1
Fuel Cost [€/l]	1.35
Battery Cost [€/kWh]	0.03
Maximum Number of Vehicles	2

Table A.3: Battery capacities and costs for ten different battery types, for the test instance with totally six nodes. Here Q_n denotes the capacity of battery type n and c_n^{w} the cost for choosing battery type n.

Type / n	1	2	3	4	5	6	7	8	9	10
Q_n [kWh]	0	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
c_n^{w} [€]	0	0.001	0.004	0.009	0.016	0.025	0.036	0.049	0.064	0.081

B

Computation Times

Table B.1: Computation times for different number of customers in CPU seconds for the logic-based Benders decomposition algorithm with the tight default lower bound (5.3.4). Each instance contains one depot and two recharge stations. The computation times are sorted by length. The larger problem instances as the smaller, since they were too time consuming. was not tested as many times. There are fewer instances solved for six node instances due to one of the six node instances taking significantly longer time to solve.

# customer nodes										
4	5	6	7	8	9	10	11	12	13	14
0.150	0.150	0.200	0.350	3.26	1.44	1.59	5.86	23.1	90.1	16.4
0.160	0.230	0.620	0.360	6.34	4.62	2.21	8.53	46.7	104	81.0
0.170	0.250	0.690	0.520	6.42	6.76	3.20	20.9	48.8	252	1990
0.180	0.280	0.720	0.610	16.8	9.72	11.7	190	109	-	-
0.180	0.290	0.990	0.720	20.6	10.0	26.5	2520	159	-	-
0.220	0.290	1.61	0.800	32.6	20.6	154	-	-	-	-
0.230	0.340	1.70	0.930	62.5	28.0	1200	-	-	-	-
0.260	0.400	1.88	1.11	567	36.1	-	-	-	-	-
0.290	0.400	2.05	1.16	-	93.8	-	-	-	-	-
0.300	0.430	2.42	1.17	-	142	-	-	-	-	-
0.310	0.440	2.43	1.43	-	190	-	-	-	-	-
0.310	0.440	2.55	1.47	-	707	-	-	-	-	-
0.320	0.440	2.76	1.50	-	-	-	-	-	-	-
0.330	0.450	2.81	1.61	-	-	-	-	-	-	-
0.330	0.460	4.36	1.70	-	-	-	-	-	-	-
0.340	0.480	4.43	1.70	-	-	-	-	-	-	-
0.350	0.500	5.12	1.79	-	-	-	-	-	-	-
0.360	0.510	7.53	1.82	-	-	-	-	-	-	-
0.390	0.510	16.0	1.86	-	-	-	-	-	-	-
0.460	0.520	18.6	1.87	-	-	-	-	-	-	-
0.470	0.530	72.5	1.88	-	-	-	-	-	-	-
0.480	0.540	-	1.91	-	-	-	-	-	-	-
0.490	0.540	-	2.00	-	-	-	-	-	-	-
0.510	0.550	-	2.04	-	-	-	-	-	-	-
0.530	0.560	-	2.06	-	-	-	-	-	-	-
0.530	0.560	-	2.06	-	-	-	-	-	-	-
0.550	0.580	-	2.13	-	-	-	-	-	-	-

Continued on next page

4	5	6	7	8	9	10	11	12	13	14
0.560	0.590	-	2.16	-	-	-	-	-	-	-
0.570	0.630	-	2.30	-	-	-	-	-	-	-
0.570	0.650	-	2.39	-	-	-	-	-	-	-
0.590	0.660	-	2.52	-	-	-	-	-	-	-
0.610	0.670	-	2.67	-	-	-	-	-	-	-
0.620	0.680	-	2.82	-	-	-	-	-	-	-
0.630	0.710	-	2.86	-	-	-	-	-	-	-
0.700	0.760	-	2.88	-	-	-	-	-	-	-
0.710	0.880	-	2.88	-	-	-	-	-	-	-
0.730	0.910	-	2.93	-	-	-	-	-	-	-
0.730	0.910	-	3.06	-	-	-	-	-	-	-
0.730	0.940	-	3.07	-	-	-	-	-	-	-
0.730	1.00	-	3.09	-	-	-	-	-	-	-
0.740	1.06	-	3.47	-	-	-	-	-	-	-
0.750	1.08	-	3.55	-	-	-	-	-	-	-
0.780	1.11	-	3.55	-	-	-	-	-	-	-
0.790	1.13	-	3.58	-	-	-	-	-	-	-
0.790	1.14	-	3.61	-	-	-	-	-	-	-
0.800	1.27	-	3.96	-	-	-	-	-	-	-
0.800	1.28	-	4.04	-	-	-	-	-	-	-
0.800	1.33	-	4.17	-	-	-	-	-	-	-
0.810	1.37	-	4.22	-	-	-	-	-	-	-
0.830	1.40	-	4.47	-	-	-	-	-	-	-
0.840	1.40	-	4.90	-	-	-	-	-	-	-
0.840	1.42	-	5.00	-	-	-	-	-	-	-
0.850	1.48	-	5.00	-	-	-	-	-	-	-
0.870	1.53	-	5.05	-	-	-	-	-	-	-
0.890	1.53	-	5.35	-	-	-	-	-	-	-
0.920	1.69	-	5.40	-	-	-	-	-	-	-
0.960	1.72	-	5.48	-	-	-	-	-	-	-
1.00	1.78	-	6.02	-	-	-	-	-	-	-
1.02	1.84	-	6.07	-	-	-	-	-	-	-
1.04	1.89	-	6.68	-	-	-	-	-	-	-
1.09	1.96	-	7.17	-	-	-	-	-	-	-
1.10	2.01	-	7.37	-	-	-	-	-	-	-
1.11	2.01	-	7.81	-	-	-	-	-	-	-
1.18	2.07	-	7.85	-	-	-	-	-	-	-
1.23	2.07	-	7.89	-	-	-	-	-	-	-
1.24	2.12	-	7.97	-	-	-	-	-	-	-
1.27	2.35	-	8.16	-	-	-	-	-	-	-
1.29	2.38	-	8.78	-	-	-	-	-	-	-
1.33	2.43	-	8.94	-	-	-	-	-	-	-
1.38	2.46	-	9.41	-	-	-	-	-	-	-
1.40	2.52	-	9.63	-	-	-	-	-	-	-
1.44	2.66	-	10.4	-	-	-	-	-	-	-
1.44	2.68	-	11.0	-	-	-	-	-	-	-

Continued on next page

4	5	6	7	8	9	10	11	12	13	14
1.50	2.70	-	11.4	-	-	-	-	-	-	-
1.52	2.79	-	11.9	-	-	-	-	-	-	-
1.53	2.79	-	12.1	-	-	-	-	-	-	-
1.57	2.80	-	12.3	-	-	-	-	-	-	-
1.57	3.03	-	13.5	-	-	-	-	-	-	-
1.62	3.07	-	14.8	-	-	-	-	-	-	-
1.66	3.14	-	15.4	-	-	-	-	-	-	-
1.69	3.31	-	17.4	-	-	-	-	-	-	-
1.71	3.58	-	20.9	-	-	-	-	-	-	-
1.72	3.86	-	23.4	-	-	-	-	-	-	-
1.77	4.21	-	24.4	-	-	-	-	-	-	-
1.84	4.46	-	25.1	-	-	-	-	-	-	-
1.88	4.53	-	26.0	-	-	-	-	-	-	-
1.90	4.68	-	33.3	-	-	-	-	-	-	-
1.97	4.83	-	35.8	-	-	-	-	-	-	-
2.16	4.97	-	-	-	-	-	-	-	-	-
2.30	4.97	-	-	-	-	-	-	-	-	-
2.33	6.04	-	-	-	-	-	-	-	-	-
2.55	8.14	-	-	-	-	-	-	-	-	-
2.93	11.9	-	-	-	-	-	-	-	-	-
3.26	15.5	-	-	-	-	-	-	-	-	-
3.48	20.9	-	-	-	-	-	-	-	-	-
3.50	24.2	-	-	-	-	-	-	-	-	-
5.61	41.5	-	-	-	-	-	-	-	-	-
-	123	-	-	-	-	-	-	-	-	-

C

Alternative Proof of Lemma 3.5.2

This chapter contains a different way to prove Lemma 3.5.2. Here, we instead consider a more general statement than in 3.5.2.

Lemma C.1. *Let $f : X \to \mathbb{R}$ be such that $\max_{x \in Y} f(x)$ exists and is attained for some $x \in Y$. Then $f(x) \geq 0$ implies $x \notin Y$ if and only if $\max_{x \in Y} f(x) < 0$.*

Proof. The general structure of the lemma is

$$(A \to B) \longleftrightarrow C.$$

By contraposition, an equivalent statement is

$$(\neg B \to \neg A) \longleftrightarrow C,$$

or, explicitly, that $x \in Y$ implies $f(x) < 0$ if and only if $\max_{x \in Y} f(x) < 0$. This statement is trivially true since the maximum of f over Y is assumed to be attained. □

Lemma 3.5.2 is now straightforward to prove.

Proof of Lemma 3.5.2. Let $X = \{0, 1\}^k \times [0, h_{k+1}] \times \cdots \times [0, h_n]$ and Y be the subset of X such that $x_j = 0$ for $j \in J_0$ and $x_j = 1$ for $j \in J_1$. In other words, $Y = \{x \in X \mid \neg C_t\}$. Let

$$f(x) = \sum_{j \in J} a_j x_j - \alpha.$$

The result then immediately follows as an application of Lemma C.1. □