NTNU

Norwegian University of
Science and Technology

# A Distributed Algorithm for Large-scale Stochastic Optimization Problems

Lars Liahagen
Haakon Hals Rød

Industrial Economics and Technology Management
Submission date: June 2016
Supervisor: Asgeir Tomasgard, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

# Problem description

Mixed-integer nonlinear programming problems occur naturally in several applications, such as in the energy industry. The problem size often explodes when uncertainty for different input parameters are considered, making the problems intractable. However, the problems often exhibit an underlying structure suitable for decomposition. An extension of the generalized Benders decomposition method for non-convex mixed-integer nonlinear programming problems, called the non-convex generalized Benders decomposition method, is presented and its potential in terms of parallelization is explored through an implementation in a distributed computing environment.

# Preface

We submit this master thesis in fulfillment of the requirements for the Master of Science degree in Industrial Economics and Technology Management at the Norwegian University of Science and Technology (NTNU). The thesis was written during the spring of 2016 at NTNU in Trondheim, the Federal University of Santa Catarina (UFSC) in Florianópolis and at the Institute of Pure and Applied Mathematics (IMPA) in Rio de Janeiro. It is a continuation of a research project we undertook in the fall of 2015.

Several people deserve special recognition for their contributions towards the completion of this thesis. First and foremost we would like to thank our advisor, Professor Asgeir Tomasgard of NTNU, for providing us with an interesting research topic, invaluable guidance and for helpful hints throughout the year. We would also like to thank him for putting us in contact with Professor Erlon Finardi of the Federal University of Santa Catarina (UFSC) in Florianopolis and Claudia Sagastizabal of the Institute of Pure and Applied Mathematics (IMPA) in Rio de Janeiro. Professor Finardi provided us with data, guidance and access to his knowledge about optimization in the energy industry which we could not have managed without. Claudia Sagastizábal helped us resolve several algorithmic issues, suggested several improvements to our implementation and shared from her wealth of knowledge on optimization. We would also like to thank them for making our stay in Brazil very enjoyable.

A special thanks is also due to Rafael Lobato of the University of Campinas (UNICAMP) for invaluable help with implementational issues. We would also like to give a special mention to the COIN-OR project and the Gurobi team for their indispensable software.

# Abstract

This thesis presents a parallel algorithm for non-convex large-scale stochastic optimization problems, specifically scenario-based two-stage stochastic mixed-integer nonlinear programming problems. The method is called the Non-convex Generalized Benders Decomposition method, which is presented together with the Generalized Benders Decomposition method of which it is an extension. A parallel version of the algorithm is presented and a distributed implementation is developed and presented. We test the method on a case study based on a stochastic unit commitment problem based on the Brazilian electrical system.

There are two main goals of this thesis: (1) to show that the non-convex generalized Benders decomposition method is parallelizable and that it scales well in a distributed computing environment and (2) to solve the stochastic unit commitment problem formulated as a non-convex two-stage mixed integer nonlinear programming problem. Three sets of results are presented: one for a reduced convex version of the case study to test scalability, another for the full convex version of the case study and the last set for the non-convex version of the case study. The results suggests that the method has potential in terms of parallelization, but that it is essential to keep the parallelizable portion of the algorithm large.

# Sammendrag

Vi presenterer en parallell algoritme for ikke-konvekse, storskala, stokastiske optimeringsproblemer formulert som scenario-baserte to-stegs stokastiske ikke-lineære blandede heltallprogrammeringproblemer. Metoden som benyttes er den ikke-konvekse generaliserte Benders dekomponeringsmetode som presenteres sammen med den generaliserte Benders dekomponeringsmetoden som den utvider. En parallel versjon av metoden presenteres og en distribuert implementasjon utvikles. Implementasjonen testes på et case study basert på et stokastisk unit commitment-problem som igjen er basert på det brasilianske kraftsstemet.

Denne oppgaven har to hovedmål: (1) å vise at den ikke-konvekse generaliserte Benders dekomponeringsmetoden er parallelliserbar og at den skalerer bra i en distribuert dataklynge og (2) løse det stokastiske unit commitment-problemet formulert som et ikke-konvekst blandet heltallsprogrammeringsproblem. Tre sett med resultater presenteres: ett for et redusert konvekst problem får å vise at metoden skalerer, ett annet for hele det konvekse problemet og det siste for den ikke-konvekse versjonen av problemet. Resultatene viser at metoden har potensial i forhold til parallelisering, men at det er essensielt å sørge for at den parallelliserbare delen av problemet er stor.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Modelling is an essential part of mathematical programming. A good model must capture the most important features of the problem to give a realistic description. On the other hand it must avoid redundancy and features that unnecessarily complicate the model. If a model manages to do this it can be an important tool for decision-support and for describing and explaining reality. A classic example of a model that captures the essential features while avoiding unnecessary complexity is that of predicting the trajectory of a projectile, such as a cannon ball. The only features considered in the model is gravity, initial velocity, angle and height while air resistance, irregularities of the ball and other non-essential factors are neglected. This model provides accurate answers and describes reality well. If the effects of air resistance and irregularities of the ball were taken into account, the answers would not become much more accurate, even though the model would become much more complex. To capture the essential features while avoiding the unimportant ones is the art of modeling. However, even though the model is an essential part of mathematical programming, so are the solution methods. If a model captures reality well but is also intractable, it is useless. On the other hand, if a model is easily solved but too simple, the results will not provide any guidance. This trade-off between complexity and tractability is at the heart of mathematical programming.

One common simplification made when modeling problems is to assume that the world is deterministic. Does a newspaper boy know how much demand he will face each day? The answer is clearly no. There are several ways to take uncertainty into consideration. One is to make a deterministic model, but to keep that assumption in mind when interpreting the results. Another is to incorporate the uncertainty into the model through a framework such as stochastic programming. However, when uncertainty is taken into account this way the problem size and complexity grows quickly and it can potentially become intractable for standard solution methods.

In this thesis we present an algorithm and a decomposition method called the non-convex Generalized Benders Decomposition (NGBD) method (Li et al., 2012b) and a parallel, distributed version of it. We also show that it is suitable for two-stage stochastic nonlinear mixed-integer problems (MINLP) where the nonlinearities can be either convex or non-convex. The deterministic equivalent program (DEP) of this kind of problem can be formulated generally as follows:

$$\begin{aligned} \underset{x,y_1,y_2,\ldots,y_s}{\text{minimize}} \ & \sum_h w_h(c_h^T x + f_h(y_h)) && \text{(P)} \\ & B_h x + g_h(y_h) \leq 0 && \forall h \in S \\ & Ax \leq d \\ & p_h(y_h) \leq 0 && \forall h \in S \\ & x \in \mathbb{Z}_+^n \\ & y_h \geq 0 && \forall h \in S \end{aligned}$$

where $x$ are assumed to be integer variables and $y \in \mathbb{R}^m$ are assumed to be continuous variables. $w_h$ are the probabilities for each scenario. The problem is non-convex if at least one of the functions $g_h$, $p_h$ or $f_h$ is non-convex. If not, the problem is convex. The DEP naturally decomposes into a first-stage and several second-stage problems, one for each scenario considered:

$$\begin{aligned} \text{minimize} \quad & c^T x \ + \ \mathbb{E}[Q_h(x)] \\ \text{subject to} \quad & Ax \leq d \\ & x \in \mathbb{Z}_+^n \end{aligned} \qquad \text{(P-1)}$$

where

$$\begin{aligned} Q_h(x) = \ \text{minimize} \quad & f_h(x,y) \\ \text{subject to} \quad & B_h x + g_h(y_h) \leq 0 \\ & p_h(y_h) \leq 0 \\ & y \geq 0 \end{aligned} \qquad \text{(P-2)}$$

The world is not a linear place and while simpler, linearized models often can be helpful in some cases they will not be sufficiently realistic in others. The same applies when modeling non-convexities with convex or piecewise linear approximations, which might lead to models of unsatisfactory accuracy. This is often the the case in hydropower production planning (Finardi and Da Silva, 2006). The stochastic unit commitment problem has been chosen as our case study as it displays several of the characteristics that makes it a suitable test case for the NGBD method. It is also an interesting problem in its own right and a highly relevant one.

In a warming world, renewable and clean energy sources are of enormous importance. They are already becoming an important part of the electrical systems around the world, providing many new challenges for the energy industry in terms of planning (Sagastizábal, 2012). Many renewables function very differently from traditional energy sources. With the notable exception of reservoir-based hydropower plant, most renewables are volatile and cannot store energy in the same way that fossil energy sources and water reservoirs can. A wind farm is a good example of such a renewable resource. It has very low marginal cost and the opportunity cost is non-existent because the energy cannot be stored and used at another time so one would usually try to produce as much as possible whenever possible. The wind farm is also very volatile and it is very hard to predict the wind and consequently how much power it will generate. This uncertainty must be taken into account, making the models used for short-term production planning more complex. Even so, the uncertainty certainly does not provide any more time for the planning process. New solution methods for these models are therefore needed, which is why the unit commitment problem has been chosen as the case study for our implementation. Mixed-integer problems occur naturally in many other applications as well, especially problems where either/or decisions must be made such as in investment planning problems and scheduling problems.

Advances in stochastic programming and integer programming have also translated to advances in stochastic mixed-integer programming which is an active area of research. The NGBD method is a novel decomposition method, which has shown promising results so exploring its potential further and to try and take advantage of its potential for parallelization is a natural increment of the research into the method and mixed-integer nonlinear programming. Furthermore, the parallel implementation developed for this algorithm could possibly be transformed to be used in other scenario-based decomposition methods.

Another reason for our approach is that while computing power is still getting cheaper and faster, it is not happening at the same rate as before and in a different way. As sequential processing power is closing in on its physical limits, parallelization provides an alternative route towards better performance. But to take advantage of these resources an algorithm must be parallel. Additionally, the amount of speed-up achieved by parallelization varies widely between different algorithms. Scenario-based stochastic programs are well-suited for massive parallelism as they can often be parallelized in terms of scenarios as the different scenarios often are independent of each other. This means that adding scenarios for more accurate modeling of uncertainty does not necessarily translate into longer runtimes.

The main goal of this thesis is to explore the NGBD method and its potential, especially in terms of its potential as a parallel algorithm for decomposable problems such as problem (P) and its scalability when the number of scenarios increase. We develop a distributed implementation of the NGBD method and test it on a stochastic unit commitment problem whose efficient solution is our secondary goal. The stochastic unit commitment problem is a problem that arises in production planning in electrical systems where the goal is to find the optimal generation schedule.

We will now review some relevant literature on the unit commitment problem and optimization in the energy sector and on the NGBD method.

## 1.1 Relation to existing literature

A similar decomposition method as the one focused on in this thesis is presented in Li et al. (2012a) and applied on a stochastic pooling problem. The pooling problem is an optimization problem that arises in different industrial settings such as natural gas production, water treatment and other industries where some kind of blending in a network occurs. In the the problem formulation the only non-convexities occur in the form of bi-linear terms that can be treated differently than general non-convex functions, so the decomposition method treats a special case of non-convex MINLPs. The computational study provided shows that there is a considerable computational advantage of this method over state-of-the-art global optimizers such as BARON (Sahinidis, 2014).

In Li (2013) strategies for parallelization of the NGBD method is presented, including its application on a natural gas network planning problem. The presented strategies are called naive scenario parallelization, adaptive scenario parallelization, and adaptive scenario and bounding parallelization. The first strategy solves subproblems for different scenarios in parallel. The second uses the information obtained from the already solved subproblems and adapts to the information given. The bounding parallelization refers to using available resources to update the bounds whenever there are available resources. This blurs the iterations so that subproblems corresponding to different iterations can be solved at the same time. The parallelization in the article is mainly concerned with utilizing multiple cores on multi-core processors. The author suggests testing the NGBD method in a distributed computing environment as future research.

Pagès-Bernaus et al. (2015) presents a parallel distributed implementation of a Branch and Fix Coordination algorithm for large-scale multi-stage stochastic mixed-integer problems where all the integer variables are binary. They consider the

deterministic equivalent program of a multi-stage stochastic program, which will exhibit a block structure, but with block structure that is connected through non-anticipativity constraints. It is a branch-and-bound algorithm where the coordination part of the algorithm refers to ensuring that non-anticipativity is maintained while providing feasible solutions for the unconnected variables. The parallelization occurs in the different sub-problems that correspond to a sub-tree in the scenario tree describing the problem. The decomposition and the parallel nature of the algorithm allows for otherwise intractable problems to be solved in a reasonable amount of time as their computational experience shows. Different branching strategies are also tested, such as best, breadth and depth first.

Frank et al. (2012) presents a survey of different solution strategies for the optimal power flow problem, which is a nonlinear optimization problem concerned with the optimal electric power generation, transmission and distribution in power networks subject to different system constraints and control limits. The problem is very similar to the unit commitment problem, but focuses more on the power flow in the network. It is generally regarded as harder due to the nature of electrical transmission networks and the properties of a realistic model. The solution strategies discussed consist of several exact methods such as various interior point methods and decomposition methods such as the generalized Benders decomposition method. Heuristic methods are also discussed at length.

Saravanan et al. (2013) gives an overview of different approaches to the unit commitment problem and a brief background of the problem. The challenging aspects of the problem when it is made more realistic is discussed. The different market environments the unit commitment problem occurs in, mainly regulated or deregulated environments, and how they affect the problem is discussed. The authors divide the solution approaches into conventional, non-conventional and hybrid methods. Conventional methods cover different decomposition methods, relaxation methods and mathematical programming approaches. Non-conventional methods range from methods inspired by AI such as genetic algorithms and neural networks to expert systems. Hybrid methods try to borrow the best of both worlds and several of the discussed methods are successful in that regard. The NGBD method presented in this paper is a purely conventional method based on exact methods, but could possibly be improved with heuristics tailored to the unit commitment problem of the case study.

Finardi and Da Silva (2006) presents a unit commitment problem on a hydro-thermal electrical system similar to the one we present in our case study. Their solution approach combines Lagrangian decomposition, sequential quadratic programming

and bundle methods. The challenges faced in the hydro unit commitment problem where time stages are connected through water usage and the complexities in modeling hydro-generation units are discussed. The production functions for hydro power units are high order non-convex polynomials, which we also use in our case study. They discuss the problems with simplified versions of these functions and their consequences on the applicability of results obtained from models with simplified production functions. In Takigawa et al. (2012) a similar and improved method is applied to a hydro-thermal scheduling problem with network constraints meaning that the transportation of power between generating source and demand is considered.

Sagastizábal (2012) gives an overview of the challenges faced by the energy industry when renewable energy sources and market liberalisation introduces more uncertainty into electrical systems and how this affects optimization problems that arise in these systems. As mentioned in the introduction, this is one of the reasons the unit commitment is an interesting case study for the NGBD method which is aimed at large-scale decomposable problems. Sagastizábal (2012) presents several decomposition methods for market equilibrium, planning and scheduling problems. Among them is a Benders-like decomposition method along time stages for both two-stage and multi-stage stochastic problems. Bundle methods and their application to energy optimization problem is also discussed. Bundle methods will be presented later in chapter 3.

Tahanan et al. (2015) also gives an overview of different approaches to the unit commitment problem for large-scale versions of it. They discuss several of the conditions surrounding the use of the unit commitment problem such as it needing to be solved in a short amount of time and well ahead of when the schedule will be followed due to rules and regulations in several power markets. They present different approaches to cope with uncertainty introduced by renewable energy sources, mainly chance-constrained programming, robust optimization and the scenario-based stochastic programming approach that we consider in this work.

## 1.2 Outline of the thesis

The rest of the thesis is structured as follows. Chapter 2 gives an overview of relevant theory and a review of some important concepts that are referred to throughout the report. The chapter reviews stochastic two-stage programs and some of their properties and important results in nonlinear duality that the NGBD method builds upon. An overview of parallel and distributed computing is also given.

In chapter 3 the NGBD method is presented together with the generalized Benders method (GBD) which the NGBD method builds upon and extends. Their application on two-stage stochastic mixed-integer programs are shown. The chapter also includes an introduction to bundle methods and their use in Benders decompositions. Chapter 4 presents the distributed versions of the algorithms and the distributed architecture of the implementations. Essential software such as the optimization solvers used are presented.

Chapter 5 presents the unit commitment problem and its characteristics. A specific case study based on the Brazilian power system and a model of the problem is formulated. The properties of the specific problem and the data provided are discussed. Implementation choices are explained and discussed.

The results are presented and discussed in chapter 6 followed concluding remarks in the final chapter. Our conclusion is presented together with our thoughts and ideas for further improvement of our work and for further research.

The appendices contain a link to a repository with the source code and details on the parameters and options used in the different software. Tables containing results are given in appendix C.

# Chapter 2

# Theory

This chapter is divided into three parts. The first section covers important results from nonlinear duality theory that lays the foundations for the GBD method and its non-convex extension, the NGBD decomposition method. The second section reviews the basics of stochastic programming and scenario-based stochastic optimization. The third section gives a brief introduction to distributed computing, its advantages and its limits.

## 2.1 Nonlinear Duality Theory

This section is mainly based on the paper by Geoffrion (1971) on duality in nonlinear programming.

Duality is one of the most important concepts in optimization and all methods based on some kind of bounding, such as Branch-and-Bound methods, use dual reasoning to establish bounds. The most important theorems and results that are made use of in this thesis are presented in this section. Consider the following nonlinear optimization problem:

$$\begin{aligned} z = \underset{x \in X}{\text{minimize}}\ & f(x) \\ \text{subject to}\ & g_i(x) \leq 0, \quad \forall i = 1, ..., m \end{aligned} \tag{P}$$

where both $f(x)$ and $g_i(x)$ are assumed to be convex on $X$. Its dual can be formulated as:

$$\begin{aligned} w = \underset{\lambda \geq 0}{\text{maximize}}\ & L(\lambda) \\ \text{where}\ & L(\lambda, x) = \underset{x \in X}{\text{minimize}}\ f(x) + \sum_i \lambda_i g_i(x) \end{aligned} \tag{D}$$

(D) is sometimes referred to as the Lagrange dual, but will in this thesis just be referred to as the dual.

The duality gap of a problem is the difference between the optimal objective values of the problem and its dual. The weak duality theorem states that the duality gap is always greater than or equal to zero.

**Theorem 2.1.1 (Weak Duality)** *If $\bar{x}$ is a feasible point for (P) and $\bar{\lambda}$ is a feasible point for (D), then*

$$f(\bar{x}) \geq L(\bar{\lambda})$$

*In particular,*

$$z^* \geq w^*$$

*or equivalently*

$$z^* - w^* \geq 0$$

Weak duality is easily shown through the following inequalities:

$$\operatorname*{infimum}_{x \in X} f(x) + \sum_i \bar{\lambda}_i^T g_i(x) \leq f(\bar{x}) + \sum_i \bar{\lambda}_i^T g_i(\bar{x}) \leq f(\bar{x})$$

where $\bar{\lambda}$ is some feasible dual vector and $\bar{x}$ is some feasible point. It is clear that the infimal value of the problem above will always be less than or equal to another feasible point $\bar{x} \in X$. $\bar{\lambda}$ is non-negative since it is a feasible dual vector and $\bar{x}$ is a feasible point so $g(\bar{x}) \leq 0$. This means that the last inequality holds.

The concept of strong duality entails that the optimal solution of both the primal (P) and its dual (D) are equal:

$$f(x^*) = g(x^*)$$

Unlike in linear programming where strong duality always holds, strong duality cannot be generally established for convex programs. Strong duality is important because it provides an alternative way of solving a problem, through its dual, and also because it provides tight bounds for how far away some solution is to the optimum. This is much more difficult to do when the duality gap is nonzero. There are however several conditions that when satisfied are sufficient for strong duality to hold for a convex program. A necessary and sufficient condition for strong duality relies on the notion of stability.

**Definition 2.1.1 (Stability)** *The problem $P$ is said to be stable if $v(0)$ is finite and there exists a scalar $M$ ¿ 0 such that:*

$$\frac{v(0) - v(y)}{\|y\|} \leq M \quad \forall y \neq 0$$

*where the function $v(y)$ is the perturbation function associated with problem $P$. The norm $\|\cdot\|$ used for $y$ is not important.*

**Definition 2.1.2 (Perturbation function)** *A perturbation function $v(\cdot)$ can be defined on $\mathbb{R}^m$ as:*

$$v(y) = \underset{x \in X}{\textit{infimum}} \{f(x) \textit{ subject to } g_i(x) \leq y_i \textit{ for } i = 1, ..., m\}$$

*where $y = (y_1, y_2, ..., y_m)$ is an $m$-dimensional vector.*

Intuitively a problem is stable if $v(0)$ is finite and if $v(y)$ doesn't decrease infinitely steeply in any perturbation direction. As mentioned above if a problem is stable strong duality holds and all other sufficient conditions will of course be satisfied.

These conditions are sometimes referred to as constraint qualifications. Other constraint qualifications than stability are often easier to work with, even though they can be less encompassing. One such constraint qualification is Slater's condition (Slater, 2014). Given problem (P), Slater's condition is satisfied if there exists a Slater point.

**Definition 2.1.3 (Slater point)** *Given a convex program such as (P) a Slater point is a point $x$ such that:*

$$g_i(\bar{x}) < 0 \textit{ for all convex constraints}$$
$$g_i(\bar{x}) = 0 \textit{ for all linear constraints}$$

In other words, a Slater point is a feasible point which is not on the edge of the feasible region. Slater's condition is a sufficient condition for strong duality to hold and states that if there exists a Slater point, then there is no duality gap and strong duality holds. Informally, this means that the duality gap is zero if a feasible interior point exists. This is not a very strong condition and it will be satisfied for many convex optimization problems.

An alternative condition which rests on the notion of an optimal multiplier vector $\lambda$ (also called Lagrange multipliers or dual variables) is also useful.

**Definition 2.1.4 (Optimal multiplier vector)** *An optimal multiplier vector is any non-negative vector $\lambda$ such that $(x, \lambda)$ satisfies the following conditions.*

1. *$x$ minimizes $f(x) + \sum_{i=1}^{m} \bar{\lambda}_i g_i(\bar{x})$ over $X$.*
2. *$\sum_{i=1}^{m} \bar{\lambda}_i g_i(\bar{x}) = 0$*
3. *$\lambda \geq 0$*
4. *$g(x) \leq 0$*

(2) is the familiar complementary slackness requirement of an optimal solution and (1) is the dual objective function. $x$ and $\lambda$ must be feasible. From the definition of an optimal multiplier vector one can see that it is not necessarily one vector, but possibly a set of vectors associated with the optimal solution $x$. The existence of an optimal multiplier vector requires that a feasible solution $x$ exists.

**Theorem 2.1.2** *Assume that (P) has an optimal solution. Then an optimal multiplier vector exists if and only if the dual (D) has an optimal solution and the optimal values of the primal (P) and its dual (D) are equal.*

The existence of an optimal multiplier vector and the existence of a Slater point are both sufficient conditions for strong duality to hold in the convex case. If Slater's condition is satisfied this also implies the existence of the optimal multiplier vector by 2.1.2.

The following theorem summarizes this section.

**Theorem 2.1.3 (Strong duality)** *If (P) is stable, then*

1. *(D) has an optimal solution*
2. *The optimal values of (P) and (D) are equal*
3. *Every optimal solution $\lambda*$ of (D) characterizes the set of all optimal solutions of (P) as the minimizers of $f(x)+(\lambda*)^T g(x)$ over $X$ that also satisfy the feasibility condition $g_i(x) \leq 0$ for all $i = 1, ..., m$ and the complementary slackness condition (2)*

The proof of the strong duality theorem using the concept of stability can found in the paper by Geoffrion (1971) that this section builds upon.

## 2.2 Optimization under uncertainty

There are several different frameworks for optimization under uncertainty, but the three most well-known are robust optimization, chance-constrained optimization and stochastic programming. This thesis considers scenario-based two-stage stochastic programs with recourse which belongs in the stochastic programming framework. Scenario-based stochastic problems needs data so that it is possible to generate realistic scenarios. If garbage is put into the model garbage comes out, a concept also known as GIGO. This section reviews important concepts and terminology that will be used throughout the thesis.

A scenario-based two-stage stochastic program with recourse can be formulated, as shown in the introduction, as follows

$$\begin{aligned} \text{minimize} \quad & c^T x \ + \ \mathbb{E}[Q_h(x)] \\ \text{subject to} \quad & Ax \leq d \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

where

$$\begin{aligned} Q_h(x) = \ \text{minimize} \quad & f_h(x, y) \\ \text{subject to} \quad & B_h x + g_h(y_h) \leq 0 \\ & p_h(y_h) \leq 0 \\ & y \geq 0 \end{aligned}$$

$\mathbb{E}[Q_h(x)]$ is the recourse function. The index $h$ refers to the scenario. Scenarios can be obtained or generated in several ways and is a rich subject in itself, but not inside the scope of this thesis; it will only be discussed briefly in chapter 5. Every scenario is assumed to have some probability of occurring, reflected in its weight in the expected value function. The recourse is the action taken in the second stage where there no longer is uncertainty and one makes the best action possible given the decisions taken in the first stage, when uncertainty was present. The deterministic equivalent program (DEP) can be formulated by combining the problems for each stage into one together with so-called non-anticipativity constraints that can be either implicit or explicit. With implicit non-anticipativity constraints the DEP becomes

$$
\begin{aligned}
\underset{x,y_1,y_2,\ldots,y_s}{\text{minimize}}\ & c^T x + \sum_h w_h f_h(y_h) && && \text{(P)}\\
& B_h x + g_h(y_h) \leq 0 && \forall h \in S\\
& Ax \leq d\\
& p_h(y_h) \leq 0 && \forall h \in S\\
& x \in \mathbb{Z}_+^n\\
& y_h \geq 0 && \forall h \in S
\end{aligned}
$$

which is the same problem shown in the introduction and the type of problem that will be considered in this thesis. In an explicit formulation one would replace the first stage variables $x$ with a set of variables $x_h$ for each scenario and add a set of constraints that force them to be equal so that one would have a set of constraints binding together the problem rather than a set of variables.

### 2.2.1 Different types of recourse

A stochastic problem with recourse can have several different types of recourse with regards to feasibility. Only two types will be considered here, relatively complete and complete recourse. One says that a two-stage stochastic program has complete recourse if every first stage solution has a feasible second stage. It has relatively complete recourse if every feasible first stage solution leads to a feasible second stage. Another type of recourse is called fixed recourse and occurs if the recourse doesn't change with the scenario, i.e. the effects of the decisions one can take in the second stage doesn't change, only the environment. In terms of the (P) this means that coefficients connected to the $y_h$ doesn't change with the scenario. $B_h$ and terms in $g_h(y_h)$ not associated with $y_h$ can change.

## 2.3 Distributed and parallel computing

Parallel computing means doing computations in parallel. The basic principle is that large problems often can be decomposed into several smaller problems. If these problems also are independent they can be solved completely in parallel, but often there is something that connects the smaller problems. This is often the challenging part of parallel computation as one has to understand the dependencies between the different parts of a large problem. In optimization this often occurs, which have lead to the development of several decomposition methods such as Dantzig-Wolfe decomposition and Benders decomposition where some complicating factors connect

the problem. If this is not the case one often hears the term *embarrassingly parallel*, which refers to problems where there is little to no effort involved in parallelizing it. One classic example of this is matrix multiplication where every element of the product matrix can be computed independently of the others.

Amdahl's law (Amdahl, 1967) gives a theoretical upper bound on the effects of parallelization for a fixed amount of work. A task $T$ of a fixed amount of work can be divided into its parallelizable part $pT$ and its sequential part. Then the parallelizable part can be sped-up by some factor $s$, e.g. by dividing $pT$ into $n$ parts and then run each part on a separate processor for a maximum speed-up of $\frac{pT}{n}$. More formally, this becomes:

$$S(s) = \frac{1}{1 - p - \frac{p}{s}} \qquad \text{(Amdahl's law)}$$

where $p$ is the part of the program that can be parallelized and $s$ is the speed-up it achieves, e.g. the number of extra processors if that is the bottleneck. One can see that:

$$S(s) \leq \frac{1}{1-p} \text{ as } \lim_{s \to \infty} \frac{p}{s} = 0$$

So the upper bound of possible speed-up is given by the part of problem that cannot be parallelized due to sequential dependencies. Imagine a program where 90% can be parallelized, then one can never achieve more than ten times the speed-up due to the last 10% that cannot be parallelized, i.e. $S(s) < \frac{1}{1-0.9} = 10$. There is also an alternative model for parallel called Gustafson's law (Gustafson, 1988), which criticizes the assumption of a fixed size of work when available computing resources increase. However, the assumption of fixed work holds for the subject matter of this report.

There is no clear boundary between the terms parallel and distributed computing, but parallel computing can refer to utilizing all the cores of a processor or using multiple processors. The term distributed computing on the other hand is more often used when considering multiple processors connected through communication links and communicating by messages through these links and can as such be considered a subset of parallel computing. A computing system that isn't physically located at the same geographical location would be inherently distributed.

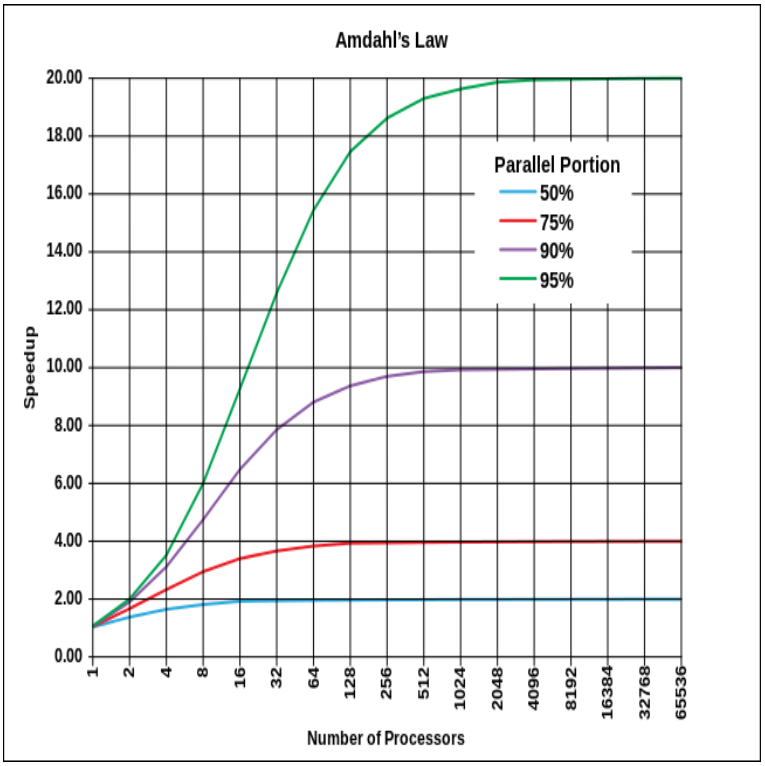



Figure 2.1: Illustration of Amdahl's law for different types of programs where $s$ is the number of processors (Wikimedia, 2008)

### 2.3.1 Message Passing Interface

The message passing interface is a standardized and portable message passing system for distributed computing. MPI is the de-facto standard for parallel computing and is a specification of syntax and semantics of several core functions and routines for effective message passing, but it is not a specific library or program. OpenMPI is an open-source implementation of MPI that is used in this project.

# Chapter 3

# The non-convex generalized Benders decomposition method

This chapter presents the method that is the main focus in this thesis: the NGBD method which is an extension of the generalized Benders decomposition (GBD) method that can handle non-convexities in mixed-integer nonlinear programming problems. The first section presents the GBD method and how it applies to two-stage stochastic MINLPs, while the second section covers the NGBD method. The third section introduces bundle methods and their potential use in Benders decomposition methods.

## 3.1 The Generalized Benders Decomposition method

This section is mainly based the work by Geoffrion (1962) and presents the GBD method, which is, as the name suggests, a more general version of Benders decomposition (Benders, 1962) as it can handle NLPs and MINLPs if some additional conditions are satisfied. The method itself is more or less the same, but some new properties and assumptions are given to show that it can converge for nonlinear programs as well. The GBD method can solve certain types of NLP and MINLP problems to optimality, as opposed to only MILP and LP problems with the original Benders decomposition method. The decomposition method exploits the structure of problems that have a set of complicating variables, variables that when fixed or removed makes the remaining problem easy to solve. This can be due to rendering the remaining problem convex, giving it some well-known structure for which efficient solution procedures exist or that it falls apart into several smaller, independent problems.

A MINLP problem can be formulated by adding integer variables to (P) from the previous section:

$$\begin{aligned} &\text{minimize } f(x,y) \\ &\text{subject to } G(x,y) \leq 0 \\ &x \in X \\ &y \geq 0 \end{aligned} \tag{MINLP}$$

where $X$ a discrete finite set making $x$ the so-called complicating variables due to their integrality requirement while $y$ are nice, continuous variables. $G(x,y)$ is the vector of $g_i(x,y)$ functions, now with integer variables in the mix, that makes up $m$ constraints. Some assumptions on the sets $Y$ and $X$ are necessary.

**Assumption 3.1.1** *$Y$ is a non-empty convex set and $G(x,y)$ is convex on $Y$ for every fixed $x$.*

It it is also assumed that the remaining problem for when $x$ is fixed is considerably easier to solve, which is essential for the GBD method to be useful.

Two of the key ideas behind the GBD method is the concepts of projection and dualization that are used to make an outer approximation of the convex functions and the feasible set (Geoffrion, 1970). By projecting the problem MINLP onto $X$ it can be viewed as a problem in $x$-space:

$$\text{minimize } v(x) \text{ subject to } x \in X \cap V \tag{3.1}$$

where

$$v(x) := \text{ infimum}^1 \ f(x,y) \text{ subject to } G(x,y) \leq 0, y \in Y \tag{3.2}$$

and

$$V := \{x : G(x,y) \leq 0 \text{ for some } x \in X\} \tag{3.3}$$

$V$ is the set of values of $x$ for which there exists at least one $y$ that gives a feasible solution. $v(x)$ is the projected problem for values of $x$ that are feasible in the original problem (MINLP). The following theorem states the equivalence of (MINLP) and (3.1) and establishes projection as a route to solving the original problem.

[1]The infimum of a (partially ordered) set is informally the largest lower bound of the set. Infimum is similar to minimum, but is more general as the the minimum needs to be part of the set, while the infimum does not. If the minimum exists the minimum and the infimum are equivalent.

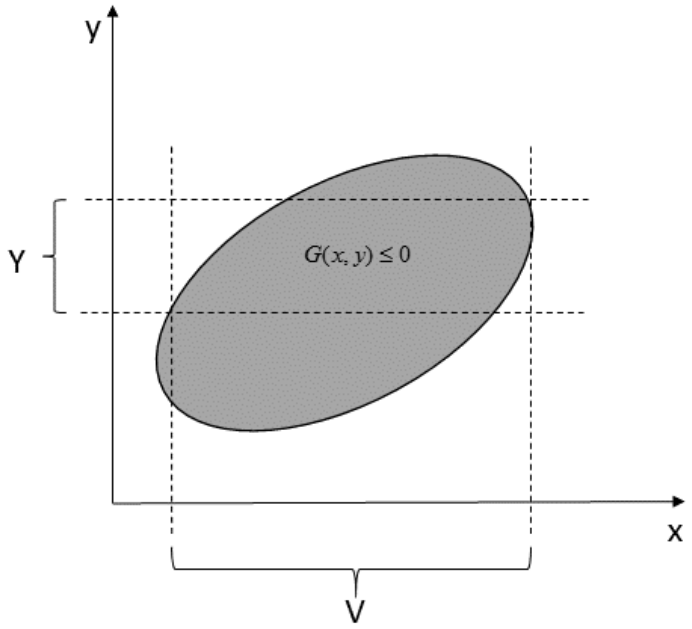



Figure 3.1: Illustration of the $V$ set

**Theorem 3.1.1 (Projection)** *The original problem (MINLP) is infeasible or has unbounded optimal value if and only if the same is true of the projected problem (3.1). If $(x^*, y^*)$ is optimal in (MINLP), then $x^*$ is optimal in (3.1). If $x^*$ is optimal in (3.1) and $y^*$ achieves the infimum in (3.2) with $x = x^*$, then $(x^*, y^*)$ is optimal in (MINLP). If $\bar{y}$ is $\epsilon_1$-optimal in (3.1) and $\bar{x}$ is $\epsilon_2$-optimal in (3.2) with $y = \bar{y}$, then $(\bar{x}, \bar{y})$ is $(\epsilon_1 + \epsilon_2)$-optimal in (MINLP).*

Since every $v(x)$ gives the optimal value for that value of $x$, an optimal $x^*$ must give $(x^*, y^*)$, which is the optimal solution for the original problem. And by the assumption that the remaining problem for fixed $x$ is considerably easier to solve, $v(x)$ as defined above is easy to solve. However, neither (3.2) or (3.3) are known explicitly. This difficulty is dealt with through a cutting-plane procedure that iteratively builds up a representation of $v(x)$ and $V$ based on the following theorems.

**Theorem 3.1.2 (V-representation)** *Given (MINLP) assume that $Y$ is a non-empty convex set and that $G(x, y)$ is convex on $Y$ for every fixed $x \in X$. Assume further that the set $Z_x \equiv \{z \in \mathbb{R}^m | G(x, y) \leq z$ for some $y \in Y\}$ is closed for each*

*fixed $x \in X$. Then a point $\bar{x} \in X$ is also in $V$ if and only if $\bar{x}$ satisfies the system:*

$$\begin{gathered} \left\{ \underset{y \in Y}{\textit{infimum}}\ \lambda^T G(x,y) \right\} \leq 0, \forall \lambda \in \Lambda \\ \textit{where } \Lambda \equiv \left\{ \lambda \in \mathbb{R}^m | \lambda \geq 0, \sum_{i=1}^{m} \lambda_i = 1 \right\} \end{gathered} \tag{3.4}$$

If a point $\bar{x}$ is in $V$ then one can verify directly that (3.4) is satisfied. However, the inverse establishes a way of approximating $V$ as it can be represented in terms of the intersection of different regions that all contain it. Assume that $\bar{x}$ satisfies (3.4); then the supremum of (3.4) is zero since $\lambda = 0$ is allowed. The supremum of (3.4) can be thought of as the dual of the program:

$$\text{minimize } 0^T y \text{ subject to } G(\bar{x}, y) \tag{3.5}$$

which is

$$\underset{\lambda \geq 0}{\text{maximize}} \{ \underset{y \in Y}{\text{minimize}}\ 0^T y + \lambda G(\bar{x}, y) \}$$

That is, a program where every feasible solution is optimal due to the objective being zero. It clearly has optimal value zero, which is the same as the optimal value of its dual with respect to the $G(x,y)$ constraints. Under the assumption that $Z_x \equiv \{z \in \mathbb{R}^m | G(x,y) \leq z \text{ for some } y \in Y\}$ is closed and non-empty, then by 3.1.3 the primal (3.5) is feasible and in fact any problem constrained by $G(\bar{x}, y)$ will be feasible. This again means that $\bar{x}$ is in $V$ and that 3.1.2 gives a way of approximating $V$.

**Theorem 3.1.3** *If $Z_x \equiv \{z \in \mathbb{R}^m | G(x,y) \leq z \text{ for some } y \in Y\}$ is closed and the optimal value of the dual is finite, then the primal is feasible.*

**Theorem 3.1.4 (v-representation)** *Assume that $Y$ is a non-empty convex set and that $f(x,y)$ and $G(x,y)$ is convex on $Y$ for each fixed $x \in X$. Assume further that, for each fixed $\bar{x} \in X \cap V$, at least one of the following three conditions hold:*

*(a) $v(\bar{x})$ is finite and (MINLP) for $x = \bar{x}$ possesses an optimal multiplier vector.*

*(b) $v(\bar{x})$ is finite, $G(\bar{x}, y)$ and $f(\bar{x}, y)$ are continuous on $Y$, $Y$ is closed and the $\epsilon$-optimal solution set of (MINLP) for $x = \bar{x}$ is non-empty and bounded for some $\epsilon \geq 0$.*

*(c)* $v(\bar{x}) = -\infty$

*Then, the optimal value of (MINLP) for a fixed* $x$ *equals the optimal value of its dual on* $X \cap V$*.*

$$v(x) = \underset{\lambda \geq 0}{supremum}^{2} \left\{ \underset{y \in Y}{infimum}\ f(x,y) + \lambda^T G(x,y) \right\}, \quad \forall x \in X \cap V \tag{3.6}$$

(3.6) is the Lagrange dual of (MINLP) with respect to the $G(x,y)$ constraints. Condition (a) will often hold, for example when Slater's condition holds, which is a sufficient condition for strong duality. Then an optimal multiplier vector will also exist as there is no duality gap, as theorem 2.1.2 states. A finite optimal solution also follows from this. Condition (b) will also very often be justified as it basically says that there exists a non-empty set of $\epsilon$-optimal solutions. Condition (c) just says that the problem is unbounded.

Under the assumptions in 3.1.2 and 3.1.4 that $f(x,y)$ and $G(x,y)$ are convex on $Y$ and that $Y$ is a convex set, the original problem (MINLP) can be transformed to an equivalent *master problem*. This transformation is done through projection and invoking the dual representations of $v(x)$ and $V$, resulting in the following master problem:

$$\begin{aligned} &\underset{x \in X}{\text{minimize}} \left\{ \underset{\lambda \geq 0}{\text{supremum}} \left\{ \underset{y \in Y}{\text{infimum}} \left\{ f(x,y) + \lambda^T G(x,y) \right\} \right\} \right\} \\ &\text{s.t. } (3.4) \end{aligned} \tag{3.7}$$

where the problem has been projected onto the $x$-space and $v(x)$ has been dualized. 3.1.1 and 3.1.4 establishes this as a route to solving the original problem. An equivalent formulation and the one that will be used from here on is:

$$\underset{x \in X}{\text{minimize}}\ \theta \tag{3.8}$$

$$\text{subject to } \theta \geq \underset{y \in Y}{\text{infimum}}\ \left\{ f(x,y) + \lambda^T G(x,y) \right\}, \quad \forall \lambda \geq 0 \tag{3.9}$$

$$0 \geq \underset{y \in Y}{\text{infimum}}\ \left\{ \sigma^T G(x,y) \right\}, \quad \forall \sigma \in S \tag{3.10}$$

[2]The supremum of a (partially ordered) set is informally the least upper bound of the set. Supremum is similar to maximum, but is more general since the maximum needs to be part of the set while the supremum does not. If the maximum exists the maximum and the supremum are equivalent.

This is a semi-infinite program, which means that there is an infinite number of constraints and a finite number of variables or vice versa. The program above is an instance of the first case as it has an infinite number of constraints and a finite number of variables. Relaxation of the problem is therefore a natural solution strategy by relaxing all except for a few of the constraints (3.9) and (3.10). Constraints are then iteratively added to the relaxed master problem until a solution of satisfactory accuracy has been found. This is sometimes referred to as row generation. A consequence of this is that the solutions found in the master problem are non-decreasing as it is restricted more and more by optimality and feasibility cuts.

The *subproblem* responsible for generating feasibility and optimality cuts is the remaining problem with $x$ fixed to some $\bar{x}$:

$$\underset{y \in Y}{\text{minimize}} f(\bar{x}, y) \text{ subject to } G(\bar{x}, y) \leq 0$$

For simplicity when discussing the GBD method, the functions $L^*(x, \lambda)$ and $L_*(x, \sigma)$ are defined:

$$L^*(x, \lambda) \equiv \underset{y \in Y}{\text{infimum}} \left\{ f(x, y) + \lambda^T G(x, y) \right\}, \quad x \in X, \quad \lambda \geq 0 \tag{3.11}$$

$$L_*(x, \sigma) \equiv \underset{y \in Y}{\text{infimum}} \left\{ \sigma^T G(x, y) \right\}, \quad x \in X, \quad \sigma \geq 0 \tag{3.12}$$

Note that $L^*(x, \lambda)$ and $L_*(x, \sigma)$ are found in the right hand sides in constraints 3.9 and 3.10 in the master problem, which are the optimality and feasibility cuts, respectively.

For the GBD method to converge the problem must satisfy **Property (P)**. This property says that for every $u \geq 0$, the infimum over $Y$ of $f(x, y) + u^T G(x, y)$ can be found essentially independently of $x$. This means that the function $L^*(\cdot, u)$ for any $x$ can be obtained explicitly with little or no more effort than is required to evaluate it at a single value of $x$. Similarly, for every $\lambda \in \Lambda$, the infimum over $Y$ of $\lambda^T G(x, y)$ can be taken essentially independently of $x$, so that the function $L_*(\cdot, \lambda)$ on $X$ can be obtained explicitly with little or no more effort than is required to evaluate it at a single value of $x$. There are several problem classes for which Property (P) holds, such as problems that are linearly separable in $x$ and $y$:

$$f(x, y) = f(x) + f(y)$$
$$G(x, y) = G(x) + G(y)$$

Note that linear separability is not a necessary condition for Property (P) to hold; it represents only one case in which it holds. Another case in which $L^*(\cdot, y)$ and $L_*(\cdot, y)$ can be obtained explicitly is when the globally optimal solution of (MINLP) for a fixed $x$, $y^*$ is also the solution of the minimization problems defined by (3.11) and (3.12). This is called **Property (P')** (Bagajewicz and Manousiouthakis, 1991) and is satisfied when, among other cases, $f(x, y)$ and $G(x, y)$ are convex and separable in $y$.

### The Cutting-Plane Procedure

The cutting-plane procedure can be stated as follows:

**Algorithm 1** Generalized Benders Decomposition procedure

**procedure** Generalized Benders Decomposition Procedure
  Let a point $\bar{x} \in X \cap V$ be known
  Solve the subproblem (3.1) and obtain an optimal multiplier vector $\bar{\lambda}$ and the function $L^*(x, \bar{\lambda})$
  $p \leftarrow 1$, $q \leftarrow 0$, $\lambda^1 = \bar{\lambda}$, $UBD = v(\bar{x})$, $LBD = -\infty$
  Set some error tolerance $\epsilon$
  **while** $UBD - LBD \geq \epsilon$ **do**
    Solve the current relaxed master problem given by:

$$\underset{x \in X}{\text{minimize}}\ \theta \text{ subject to } \theta \geq L^*(x, \lambda^j), \quad j = 1, ..., p$$
$$0 \geq L_*(x, \sigma^j), \quad j = 1, ..., q$$

    Let $(\hat{x}, \hat{\theta})$ be an optimal solution and set
    $LBD \leftarrow min(LBD, \hat{\theta})$.
    Solve the new subproblem given by $\hat{x}$
    **if** $v(\hat{x})$ is finite **then**
      Obtain an optimal multiplier vector $\bar{\lambda}$ and the function $L^*(x, \bar{\lambda})$,
      $p \leftarrow p + 1$ and $u^p = \bar{u}$
      Update $UBD = max(UBD, v(\hat{x}))$
    **else**
      Determine an $\hat{\sigma} \in S$ and the function $L_*(x, \sigma)$,
      $q \leftarrow q + 1$ and $\sigma^q = \hat{\sigma}$
  $(x^*, y^*)$ corresponding to UBD is the optimal (or $\epsilon$-optimal) solution to (MINLP)

The following theorems states that 1 converges finitely.

**Theorem 3.1.5 (Finite Convergence)** *Assume that $X$ is a finite discrete set, that the assumptions of theorems 3.1.2 and 3.1.4 holds, omitting (b). Then the generalized Benders decomposition procedure terminates in a finite number of steps.*

Theorem 3.1.5 follows as a direct consequence of $X$ being a finite discrete set and that the GBD procedure prevents an $\bar{x} \notin V$ from ever being again being feasible due to the feasibility cut added for that $\bar{x}$. For $\bar{x} \in V$ an optimality cut is added and if the same $\bar{x}$ would occur again in some future iteration this would imply optimality.

### 3.1.1 Generalized Benders Decomposition for Two-Stage Stochastic Mixed Integer Nonlinear Programs

There are several examples of problem types where the GBD method is well-suited. Within stochastic programming the application of the Benders decomposition method is referred to as the L-shaped method (Van Slyke and Wets, 1969), which is more or less the same method. GBD is also very useful when the first stage variables are integer such as in the unit commitment problem mentioned in the introduction. This will be considered in more detail in chapter 5. These problems can be modeled as two-stage stochastic MINLP problems, which is the problem class of interest in this report. A scenario-based two-stage stochastic MINLP can be formulated, as shown in the introduction, as follows:

$$\begin{aligned} \text{minimize} \quad & c^T x \ + \ \mathbb{E}[Q_h(x)] \\ \text{subject to} \quad & Ax \leq d \\ & x \in \mathbb{Z}_+^n \end{aligned} \tag{P-1}$$

where

$$\begin{aligned} Q_h(x) = \ \text{minimize} \quad & f_h(x,y) \\ \text{subject to} \quad & B_h x + g_h(y_h) \leq 0 \\ & p_h(y_h) \leq 0 \\ & y \geq 0 \end{aligned} \tag{P-2}$$

$X \times Y \subset \mathbb{Z} \times \mathbb{R}$, $\xi$ is a random vector that represents the different scenarios. The second stage problem (3.19) can often can be quite easy to solve when given an $x \in X$ as this might give it some well-known structure, render it a convex program or make it otherwise easily decomposable. The structure of a two-stage stochastic MINLP problem of this kind lends itself naturally to the generalized Benders decomposition. The DEP of the two-stage problem above can be formulated, as shown in the introduction, as follows:

$$
\begin{aligned}
\underset{x,y_1,y_2,\ldots,y_s}{\text{minimize}} \quad & \sum_h w_h(c^T x + f_h(y_h)) && \text{(P)} \\
& B_h x + g_h(y_h) \leq 0 && \forall h \in S \\
& Ax \leq d \\
& p_h(y_h) \leq 0 && \forall h \in S \\
& x \in \mathbb{Z}_+^n \\
& y_h \geq 0 && \forall h \in S
\end{aligned}
$$

When the DEP is written out extensively with implicit non-anticipativity constraints, one can see that it exhibits a block structure such as the one shown in figure 3.2. One can see that the first stage variables connect the problem in the first set of constraints. Without them the program would fall apart into a problem for the first stage variables $x$ and problems for each scenario $h$. This is the structure that the GBD method exploits as it allows for decomposing the problem into a master problem that is the first stage with a second term that approximates the expected value of the recourse function, $\mathbb{E}[Q(x, \xi)]$.

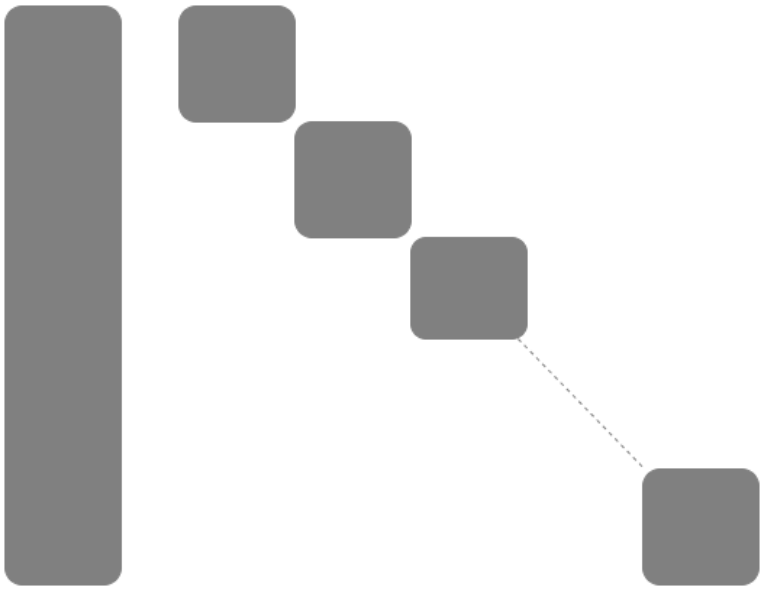


Figure 3.2: Block structure of problem (P)

For the two-stage stochastic problem (P) the master problem becomes, with the optimality cuts and feasibility cuts:

$$
\begin{aligned}
\text{minimize} \quad & c^T x + \theta \\
\text{subject to} \quad & \theta \geq \mathbb{E}[f_h(\bar{x}^i, \bar{y}_h^i) + (\lambda_h^i)^T (g_h(\bar{y}_h^i) + B_h x)] \quad \forall i \\
& 0 \geq (\sigma_h^i)^T (g_h(\bar{y}_s^i) + B_h x)] \quad \forall j, \forall h \in S \\
& x \in X \cap \mathbb{Z}
\end{aligned}
\tag{MP}
$$

where $i$ and $j$ are the numbers of the optimality and feasibility cuts added so far, respectively. The subproblems are, for each scenario $h \in S$:

$$
\begin{aligned}
\text{minimize} \quad & f_h(\bar{x}, y) \\
\text{subject to} \quad & B_h \bar{x} + g_h(y_h) \leq 0 \qquad [\lambda_s^i] \\
& p_h(y_h) \leq 0 \\
& y_h \geq 0
\end{aligned}
\tag{3.13}
$$

**Optimality cuts**

Optimality cuts are derived from the solution of the second stage problem (3.19), i.e. the subproblems. Let $\bar{y}_h^i$ be the optimal solution of the subproblem corresponding to iteration $i$ and scenario $h$, and $w_h$ the probability of scenario $h$. $\lambda_h^i$ is the optimal multiplier vector for the constraints $B_h \bar{x} + g_h(y) \leq 0$. An optimality cut is then defined by:

$$
\theta \geq \sum_{h \in S} w_h (f_h(x^i, y_h^i) + (\lambda_h^i)^T (g_h(y_h^i + B_h x)) \tag{3.14}
$$

The inner terms in the summation can be recognized as the optimal solutions of a Lagrangean relaxation of the subproblems. Another alternative to the optimality cut above is:

$$
\theta \geq f_h(\bar{x}^i, \bar{y}_s^i) + (\lambda_h^i)^T (g_h(\bar{y_h}^i + B_h x) \quad \forall h \in S \tag{3.15}
$$

The two cuts (3.14) and (3.15) refers to two different strategies for adding optimality cuts, the single-cut and multi-cut strategy, respectively. In the single-cut strategy one optimality cut is obtained by aggregating all the solutions from the subproblems while in the multi-cut strategy one optimality cut is added for each subproblem. The single-cut approach more iterations before bounds converge than the multi-cut approach. However, with the multi-cut approach the relaxed master problem also gets considerably larger in each iteration as $s$ cuts are added every iteration instead of just one. The optimal strategy depends on the problem at hand.

### Feasibility cuts

For problems without complete or relatively complete recourse, feasibility cuts might be needed if the first stage solution yields an infeasible second stage. The method for obtaining feasibility cuts is similar to the one in the L-shaped method for stochastic linear programming. It is obtained by solving the following nonlinear problem, which is the same as the subproblem except for its objective function:

$$\begin{aligned} \text{minimize} \quad & ||z_k^-|| + ||z_k^+|| \\ \text{subject to} \quad & B_h\bar{x} + g_h(y_h) - s_k = 0 \quad [\sigma_s^i] \\ & p_h(y_h) - z_k^- = 0 \\ & y_h \geq 0 \\ & z_k^- \geq 0 \\ & z_k^+ \geq 0 \end{aligned} \tag{3.16}$$

where $z_k^-$ is the slack variable for the $k$th constraint. The $-$ and $+$ refers to the sign of the slack variable. For less than or equal-constraints one uses $z^-$ and for greater than or equal-constraints one uses $z^+$. For equalities both would be used since one wouldn't know which way a potential feasibility breach would occur. $||z_k||$ is some norm that measures the distance to feasibility which is when all the slack variables are zero.

Let $\sigma_s^i$ be the optimal Lagrangean multiplier vector for the constraints $T_s\bar{x} + W_s y \leq h_s$. If the solution of its Lagrangean relaxation gives $z = 0$ for all scenarios then $\bar{x}$ from the first stage is feasible. This is because any feasible solution of (3.16) will achieve the optimal objective value of 0. If not, the infeasible solution is excluded from the feasible space of the master problem. This is done by adding the following feasibility cut:

$$0 \geq (\sigma_h^j)^T(g_h(y_s^j) + B_h x) \forall j, \forall h \in S \tag{3.17}$$

## 3.2 The Non-convex Generalized Benders Decomposition method

This section presents the NGBD method and is mainly based on the paper by Li et al. (2012b) which is an extension of the GBD method that deals with MINLPs where some of the participating functions are non-convex. When strong duality fails to hold, the convergence property of the original GBD method is affected because of the non-zero duality gap. Without strong duality the optimal multiplier vectors used to create the optimality and feasibility cuts cannot be assumed to exist either. The section is mainly based on the paper cited above.

Consider the same standard form two-stage, stochastic MINLP with formulated as its deterministic equivalent program:

$$
\begin{aligned}
\underset{x,y_1,y_2,\ldots,y_s}{\text{minimize}} \quad & \sum_h w_h(c_h^T x + f_h(y_h)) && && \text{(P)}\\
& B_h x + g_h(y_h) \leq 0 && \forall h \in S\\
& Ax \leq d\\
& p_h(y_h) \leq 0 && \forall h \in S\\
& x \in \mathbb{Z}_+^n\\
& y_h \geq 0 && \forall h \in S
\end{aligned}
$$

where $x$ are integer variables and the $y \in \mathbb{R}^m$ are continuous variables. $w_h$ are the probabilities for each scenario. It is assumed that at least one of the functions $g_h$, $p_h$ or $f_h$ are non-convex. The same two-stage stochastic problem can also be formulated as a two-stage problem, the first stage problem and second stage problems for each scenario. This formulation is repeated here for convenience:

$$
\begin{aligned}
\text{minimize} \quad & c^T x \;+\; \mathbb{E}[Q_h(x)]\\
\text{subject to} \quad & Ax \leq d\\
& x \in \mathbb{Z}_+^n
\end{aligned}
\tag{3.18}
$$

where

$$
\begin{aligned}
Q_h(x) = \quad \text{minimize} \quad & f_h(x,y)\\
\text{subject to} \quad & B_h x + g_h(y_h) \leq 0\\
& p_h(y_h) \leq 0\\
& y \geq 0
\end{aligned}
\tag{3.19}
$$

where $x$ is taken as input in the second stage problem and $h$ represents the scenario.

Several assumptions are needed for the NGBD procedure to be guaranteed to converge, some of which are trivial.

**Assumption 3.2.1** *The problem is feasible meaning that there exists an integer solution such that every scenario is feasible.*

If the assumption above is not satisfied the problem is infeasible.

By relaxing the non-convex functions in (P) into their convex, differentiable envelopes through some convexification procedure such as McCormick relaxation (McCormick, 1976). Problem P has been formulated so that it is separable into its integer and continuous variables. By replacing the non-convex functions in P with their respective convex relaxations one obtains the lower bounding problem (LBP):

$$
\begin{aligned}
&\underset{x,y_1,\ldots,y_s,q_1,\ldots,q_s}{\text{minimize}} && \sum_{h=1}^{s} w_h \left( c_h^T x + u_{f,h}(y_h, q_h) \right) && && \text{(LBP)} \\
&\text{subject to} && B_h x + u_{g,h}(y_h, q_h) \leq 0 && \forall h \in S \\
& && Ax \leq d \\
& && u_{p,h}(y_h, q_h) \leq 0 && \forall h \in S \\
& && u_{q,h}(y_h, q_h) \leq 0 && \forall h \in S
\end{aligned}
$$

where $q_h$ denotes new variables that may be needed in the construction of differentiable relaxation. The functions $u_{q,h}$ are also added for the same reason. In the problem above all the participating functions are of course convex, due to the convexification procedure that has been performed. If all the functions in problem P are convex from the outset then the problem P and LBP are equivalent and there is no need to extend the GBD method. The NGBD method then reduces to the GBD method.

**Assumption 3.2.2** *The set* $D_h = \{(y_h, q_h) \in \mathbb{R}^m \times \mathbb{R}^{m_q} : u_{p,h}(y_h, q_h) \leq 0, u_{q,h}(y_h, q_h)\}$ *is compact for every* $h \in S$

Where $D_h$ is the set of $(y_h, q_h)$ tuples satisfying the constraints $u_{p,h} \leq 0$ and $u_{q,h} \leq 0$

**Assumption 3.2.3** *Set* $D_h$ *is nonempty and compact for any* $h \in S$

For real numbers $\mathbb{R}$ a compact set is a closed and bounded set. In a bounded set all the points are located within a finite distance of each other, i.e. there exists some number $M$ such that the distance between every pair of points is less than $M$. The distance can be measured by any norm, such as the euclidean norm. A set is closed when it includes its boundary points. The assumption implies that the feasible set is compact for every $x \in X$ so that problem LBP has a finite optimal value or is infeasible.

**Assumption 3.2.4** *Problem (LBP) satisfies Slater's condition for $x$ fixed to those elements in $X$ for which problem (LBP) is feasible.*

Strong duality must hold for problem (LBP) whenever $x$ is fixed to an element $x^{(l)}$ in $X$ which has feasible solutions in the second stage. This is sufficient for optimal multiplier vectors to exist and necessary for the dualization manipulation in the decomposition procedure.

Problem LBP can now be decomposed with the GBD method presented earlier in this section. The master problem becomes, by projecting the problem onto the integer variables:

$$
\begin{aligned}
\underset{\theta,x}{\text{minimize}}\ & \theta && && \text{(MP)}\\
\text{subject to}\ & Ax \le d\\
& u_{p,h}(y_h, q_h) \le 0 && && \forall h \in S\\
& \theta \ge \sum_{h=1}^{s} \left[ \underset{(y_h,q_h)\in D_h}{\text{infimum}}\ w_h u_{f,h}(y_h, q_h) + \lambda_h^T u_{g,h}(x_h, q_h) \right]\\
& \qquad + \left( \sum_h w_h c_h^T + \lambda_h^T B_h \right) x && && \forall \lambda_h\\
& 0 \ge \sum_{h=1}^{s} \left[ \underset{(y_h,q_h)\in D_h}{\text{infimum}}\ \mu_h^T u_{g,h}(y_h, q_h) \right] + \left( \sum_h \mu_h^T B_h \right) x && && \forall \mu\\
& y_h \ge 0 && && \forall h \in S\\
& \theta \in \mathbb{R}
\end{aligned}
$$

where $D_h = \{(x_h, q_h) \in \mathbb{R}^{n_q} \times \mathbb{R}^{n_q} : u_{p,h}(y_h, q_h) \le 0, u_{q,h}(y_h, q_h) \le 0\}$. Problem (MP) above is a semi-infinite problem as the number of constraints are infinite because of the continuous $\lambda_h$ and $\mu_h$ variables. The master problem is therefore relaxed like in the GBD method by only keeping a finite number of constraints. The master problem also contains optimization problems in the constraints, this is resolved by using the problems presented below.

$$
\begin{aligned}
\underset{\theta,x}{\text{minimize}}\ & \theta && && (\text{RMP}^{\text{k}})\\
\text{subject to}\quad & Ax \leq d && &&\\
& u_{p,h}(y_h, q_h) \leq 0 && && \forall h \in S\\
& \theta \geq obj_{PBP}(x^{(j)}) + \left(\sum_h w_h c_h^T + (\lambda_h^{(j)})^T B_h\right)(x - x^{(j)}) && && \forall j \in T^k\\
& 0 \geq obj_{FP}(x^{(i)}) + \left(\sum_h (\mu_h^{(i)})^T B_h\right)(x - x^{(i)}) && && \forall i \in I^k\\
& \sum_{r \in \{r: x_r^{(t)} = 1, r = 1, \ldots, n_x\}} x_r && &&\\
& - \sum_{r \in \{r: x_r^{(t)} = 0, r = 1, \ldots, n_x\}} x_r \leq |\{r : x_r^{(t)} = 1\}| - 1 && && \forall t \in T^k \cup I^k\\
& y_h \geq 0 && && \forall h \in S
\end{aligned}
$$

where the indexed sets $T^k$ and $I^k$ contains the the iterations when problem ($\text{PBP}^{\text{k}}$) is feasible and infeasible, respectively. This means that for any $j$ in $T^k$, $x^{(i)}$ is a feasible integer realization and for any $i$ in $I^k$, $x^{(i)}$ is an infeasible integer realization. In the first case a optimality cut is added and in the second a feasibility cut is added to the problem. In the first iteration of the algorithm it is clear that problem ($\text{RMP}^{\text{k}}$) is unbounded. This can be resolved in several ways, but here a feasible integer realization is assumed to be provided for the first iteration. This means that it is not solved in the first iteration. The optimization problems in (MP) are replaced by bounds that together provide a lower bound on the optimal value, as presented in the previous section on the GBD method. Problems ($\text{PBP}^{\text{k}}$) and ($\text{FP}^{\text{k}}{}_{\text{h}}$) are presented below. $obj_{PBP}(x^{(k)})$ provides the object value for the given integer realization and $obj_{FP}(x^{(k)}) = \sum_h obj_{FP_h}(x^{(k)})$ does the same for the feasibility problems. The integer cuts in ($\text{RMP}^{\text{k}}$) cuts away integer points that have already been visited, forcing the algorithm to find new points in subsequent iterations.

The subproblems are obtained by restricting problem (P) to a fixed integer realization $x^{(l)}$ in the set of feasible first stage solutions:

$$
\begin{aligned}
\underset{y_1, y_2, \ldots, y_s}{\text{minimize}}\ & \sum_h w_h \left(c_h^T x^{(l)} + f_h(y_h)\right) && && (\text{PP}^{\text{l}})\\
\text{subject to}\quad & B_h x^{(l)} + g_h(y_h) \leq 0 && && \forall h \in S\\
& p_h(y_h) \leq 0 && && \forall h \in S\\
& y_h \geq 0 && && \forall h \in S
\end{aligned}
$$

which again can be naturally decomposed into independent problems for each of the scenarios as follows:

$$\begin{aligned} \underset{y_h}{\text{minimize}}\ & c_h^T x^{(l)} + f_h(y_h) \\ \text{subject to}\quad & B_h x^{(l)} + g_h(y_h) \leq 0 \\ & p_h(y_h) \leq 0 \\ & y_h \geq 0 \end{aligned} \tag{PP$^{\text{l}}_{\text{h}}$}$$

The weight of the scenario is not necessary in the problem as it doesn't affect the solution.

The same goes for the subproblems of LBP that are created by fixing $x$ to an element in $x^{(k)}$ in $X$.

$$\begin{aligned} \underset{y_1,\ldots,y_s,q_1,..,q_s}{\text{minimize}}\ & \sum_{h=1}^{s} w_h \left( c_h^T x^{(k)} + u_{f,h}(y_h, q_h) \right) & \\ \text{subject to}\quad & B_h x^{(k)} + u_{g,h}(y_h, q_h) \leq 0 & \forall h \in S \\ & u_{p,h}(y_h, q_h) \leq 0 & \forall h \in S \\ & u_{q,h}(y_h, q_h) \leq 0 & \forall h \in S \end{aligned} \tag{PBP$^{\text{k}}$}$$

that also can be decomposed naturally into problems for each scenario $h$.

$$\begin{aligned} \underset{y_h,q_h}{\text{minimize}}\ & c_h^T x^{(l)} + u_{f,h}(y_h, q_h) \\ \text{subject to}\quad & B_h x^{k)} + u_{g,h}(y_h, q_h) \leq 0 \\ & u_{p,h}(y_h, q_h) \leq 0 \\ & u_{q,h}(y_h, q_h) \leq 0 \end{aligned} \tag{PBP$^{\text{k}}_{\text{h}}$}$$

The feasibility problems are the same as the ones in the GBD method and are solved whenever infeasibility occurs in problems (PBP$^{\text{k}}$). The feasibility problem for each scenario is:

$$
\begin{aligned}
\underset{y,q,<}{\text{minimize}} \quad & ||z_k^-|| + ||z_k^+|| \qquad (\mathrm{FP^k{}_h}) \\
\text{subject to} \quad & B_h x^{(k)} + u_{g,h}(y_h, q_h) - z_k^- = 0 \\
& u_{p,h}(y_h, q_h) - z_k^- = 0 \\
& u_{q,h}(y_h, q_h) - z_k^- = 0 \\
& z_k^- \geq 0 \\
& z_k^+ \geq 0
\end{aligned}
\tag{3.20}
$$

where $z_k^-$ and $z_k^+$ are slack variables for lesser than equal- and greater than or equal-constraints, respectively, for the $m$ constraints. Both would be used for equality constraints. The objective of problems ($\mathrm{FP^k{}_h}$) is to minimize infeasibility which here is represented in terms of slack variables that turn the constraints into equality constraints. If the objective is zero, it means that the slack variables aren't necessary and that the corresponding problem ($\mathrm{PBP^k{}_h}$) is feasible. If however the objective is non-zero the corresponding problem is infeasible and a feasibility cut must be added.

### Properties of the problems

Since problems $\mathrm{PBP^k}$ are relaxations of the corresponding problems $\mathrm{PP^l}$ the optimal value of the former is always less than or equal to the latter. This leads to the following relation; if problem $\mathrm{PP^l}$ is feasible and $obj_{PP}(y^{(l)} \leq UBD$ then:

$$
\begin{aligned}
& obj_{PP_h}(y^{(l)}) \leq UBD_h \\
& \qquad \text{where } UBD_h = UBD - \sum_{i=1}^{h-1} obj_{PP_i}(y^{(l)}) - \sum_{j=h+1}^{s} obj_{PBP_j}(y^{(l)}) \quad \forall h \in S
\end{aligned}
\tag{3.21}
$$

The above equation shows that if the objective value of the original problem fixed to a integer realization, $obj_{PP}(y^{(l)}$, can potentially be better than the current best solution found which is given by $UBD$ then $obj_{PP_h}(y^{(l)}$, which is the objective value for a given scenario, must be less than $UBD_h$. This is so since every $obj_{PBP_h}(y^{(l)})$ must be lower than its corresponding $obj_{PP_h}(y^{(l)})$. As a consequence 3.21 provides a way of cutting away integer points that cannot lead to a better upper bound without having to finish computing the actual optimal value of problem ($\mathrm{PP^l}$) for that integer point.

### 3.2.1 The algorithm

The non-convex GBD method extends the GBD method to tackle non-convex functions participating in the problem at hand. By using the GBD method on a convex relaxation of the original problem one can generate interesting integer points that is worth exploring further and cut away integer realizations that can be shown to never lead to an optimal solution. At the core it is still an enumeration method, but an intelligent one where unecessary work is avoided as much as possible so that formerly intractable problems can be solved. The method can be shown to converge to a $\epsilon$-optimal solution.

The algorithm is described below. The inner loop is the GBD-like procedure, which builds the sets containing integer points and infeasible points.

Under the assumption that the set of integer realizations is finite the algorithm terminates finitely. Theorem 3.1.5 applies for the NGBD method as well since no integer realization is ever generated twice due feasibility, optimality and integer cuts.

The NGBD algorithm has two termination criteria, so when the algorithm finishes there are two possible reasons. Because the algorithm above is a while loop the stopping criteria are the negation of the while condition. Thus the algorithm terminates when $UBDPB \geq UBD - \epsilon$ and problem (RMP$^{\text{k}}$) is infeasible or when $LBD \geq UBD - \epsilon$. *And* and *or* are used in the logical sense so if both criteria are satisfied at the same time the algorithm terminates as well. The second criterion is easy to understand because if $LBD \geq UBD - \epsilon$ a solution of satisfactory quality has been found. This is easily shown through the following inequality:

$$UBD = o\hat{bj}*_{PP} \geq o\hat{bj}_P \geq LBD \geq UBD - \epsilon$$

where $o\hat{bj}_P$ denotes the true optimal value of the original problem and $o\hat{bj}*_{PP}$ the true optimal value of problem (PP$^{\text{l}}$) for the current best integer realization $x = x^*$. The inequality clearly shows that $x = x^*$ gives a $\epsilon$-optimal solution.

The first stopping criterion is a bit more convoluted. If problem (RMP$^{\text{k}}$) is infeasible this means that all feasible integer solutions have already been visited and that integer cuts have been added to (RMP$^{\text{k}}$) so that it has become infeasible. This means that any integer realization that leads to a $\epsilon$-optimal solution must have been visited already. $UBDPB$ is always equal to the minimum value of the integer realizations for problem PBP$^{\text{k}}$ that have not yet been visited by problem PP$^{\text{l}}$. If $UBDPB \geq UBD - \epsilon$ this means that none of these values can lead to a better $UBD$ because problem PBP$^{\text{k}}$ is a relaxation of problem PP$^{\text{l}}$ meaning that those integer

**Algorithm 2** Non-convex Generalized Benders Decomposition method

Initialize iteration counters: $k \leftarrow 0$, $l \leftarrow 0$
Initialize index sets: $T^0 \leftarrow \emptyset$, $S^0 \leftarrow \emptyset$, $U^0 \leftarrow \emptyset$
Initialize bounds: $LBD \leftarrow -\infty$, $UBD \leftarrow \infty$, $UBDPB \leftarrow \infty$
First feasible integer realization, $x^{(0)}$, is given as input.
**while** (k = 0) or (($UBDPB < UBD - \epsilon$ or $RMP^k$ is feasible) and $LBD < UBD - \epsilon$) **do**
    **if** $k = 0$ or $RMP^k$ is feasible and $LBD < UBDPB$ and $LBD < UBD - \epsilon$ **then**
        **while** $LBD < UBDPB$ and ($RMP^k$) is feasible **do**
            $k \leftarrow k + 1$
            Solve $PBP_h^k$ for all scenarios
            **if** $PBP_h^k$ is feasible for all $h$ **then**
                Get optimal multiplier vectors $\lambda_h^k$, create an optimality cut and add it to $RMP$ and get $RMP^k$
                $T^k \leftarrow T^{k-1} \cup \{k\}$
                **if** $obj_{PBP}(x^k) < UBDPB$ **then**
                    $UBDPB \leftarrow obj_{PBP}(y^k)$, $x^* \leftarrow x^k$, $k^* \leftarrow k$
            **else if** Some $PBP_h^k$ is infeasible **then**
                Stop solving $PBP_h^k$ and all other $PBPs$ currently being solved.
                Solve $FP_h^k$ for all unsolved scenarios
                Get optimal multiplier vectors $\mu_h$ for all the $FP_h^k$ solved.
                Create feasibility cut and add to problem $RMP^k$
            Solve $RMP^k$
            $LBD \leftarrow obj_{RMP^k}$
            $x^{k+1} \leftarrow x$ where $x$ is the solution to $RMP^k$
    **if** $UBDPB < UBD - \epsilon$ **then**
        Solve ($PP_h(x^*)$) to $\epsilon_h$-optimality for all scenarios
        $U^l \leftarrow U^{l-1} \cup \{k^*\}$
        **if** $obj_{PP}(x^*) < UBD$ **then**
            $UBD \leftarrow obj_{PP}(y^*)$
            $x_p^* = x^*$, $y_{p,h}^* = y_h^*$ for all $h$
        **if** $T^k \setminus U^l = \emptyset$ **then**
            $UBDPB = \infty$
        **else**
            Choose $i \in T^k \setminus U^l$ such that $obj_{PBP}(x^i) = \min_{j \in T^k \setminus U^l} obj_{PBP}(x^j)$
            Update $UBDPB \leftarrow obj_{PBP}(x^i)$, $x^* \leftarrow x^i$, $k^* \leftarrow i$ and $l \leftarrow l + 1$.
Return $\epsilon$-optimal solution given by $(x_p^*, y_{p,1}^*, ..., y_{p,|S|}^*)$ or infeasibility.

realizations at best can lead to the same value which is worse than the current $UBD$.

As mentioned in the introduction, the method can be viewed as an intelligent enumeration method where the inner loop, which is the GBD-like procedure, generates

interesting integer realizations that should be explored further in the original non-convex problem while making sure that infeasible points and other uninteresting points such as integer realization that cannot lead to an optimal solution aren't explored unecessarily. This is done by using the three bounds of the algorithm and relation 3.21 that are presented in this section.

## 3.3 Relationship between the GBD method and the NGBD method

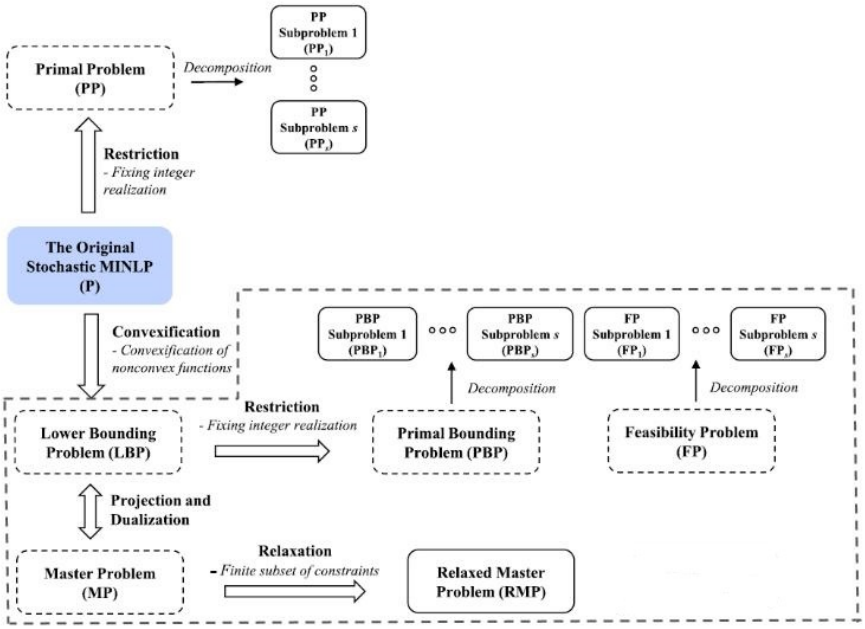



Figure 3.3: Relationship between GBD and NGBD

Figure 3.3 shows the approximate relationship between the GBD and the NGBD method. The GBD method is contained in the dashed box an corresponds to the inner loop of the algorithm. The NGBD extension is the part above the problem (P) box and the outer loop of the algorithm which uses the GBD-like procedure to generate interesting integer realizations for the non-convex subproblems.

## 3.4 Bundle methods in the GBD and NGBD method

Cutting planes-based methods, such as Benders Decomposition, tend to have problems with instability that can lead to slow convergence (Zaourar and Malick, 2014). The instability is usually exhibited in two ways: Firstly, the methods tend to make large steps in the first iterations. Secondly, when they get close to the optimal solution, they can spend a lot of iterations oscillating around it. The second point can be shown in the commonly seen tailing-off effect: The quality of the solutions improves less and less per iteration the closer you get to the optimal solution.

Bundle methods are a family of methods that aims to reduce this instability. As described in Zaourar and Malick (2014), the general idea of these methods is to

encourage the next iterate to stay close to the best iterate, while also considering the objective value of the model. The current best iterate is termed the "stability center", denoting that the next iteration is stabilized around it. One thing to note is that the stability center is only updated when a new solution is a significant improvement upon the current stability center; the definition of "significant" differs between the methods.

Three of the most common bundle methods are the proximal bundle method, the trust region bundle method, and the level bundle method. While they differ in several respects, one important similarity is that they all have parameters that dictate the balance between staying close to the stability center and optimizing the original objective. Management of these parameters is very important for performance, and update strategies are not necessarily straightforward. In theory, these three methods have been shown to be equivalent; there is always a choice of parameters that yields identical iterates. However, in terms of implementation and performance they differ quite widely (de Oliveira and Solodov, 2016).

Given a model $\hat{v}_k(x)$ that approximates the problem (P), or , and a stability center $\hat{x}_k$:

The *proximal* method finds the next iterate by solving:

$$\min_{x \in X} \hat{v}_k(x) + \frac{1}{2t_k}\|x - \hat{x}_k\|^2 \qquad \text{(Proximal bundle method)}$$

The concept of this method is to simply add a term to the objective function that penalizes distance from the stability center. The parameter $t_k$ decides how much this penalty is weighted.

The *trust region* method finds the next iterate by solving:

$$\min_{x \in X} \hat{v}_k(x) \text{ s.t. } \|x - \hat{x}_k\|^2 \leq R_k \qquad \text{(Trust region bundle method)}$$

The trust region method is in a way stricter than the proximal: It restricts the distance from the stability center, so that the next solution is guaranteed to be within the given distance. The parameter $R_k$ decides this maximum distance.

The *level* method finds the next iterate by solving:

$$\min_{x \in X} \|x - \hat{x}_k\|^2 \text{ s.t. } \hat{v}_k(x) \leq L_k \qquad \text{(Level bundle method)}$$

This method works by finding the solution closest to the stability center, while restricting the objective value to be better than some boundary. The boundary is decided by the parameter $L_k$. The solutions allowed by this restriction are members of the *level set*.

These methods work well on their own, but what if they are combined? de Oliveira and Solodov (2016) proposes a combination of the proximal and level bundle methods, into what they term a *doubly stabilized bundle method*. The combined problem is as follows:

$$\min_{x \in X} \hat{v}_k(x) + \frac{1}{2t_k}\|x - \hat{x}_k\|^2 \text{ s.t. } \hat{v}_k(x) \leq L_k \quad \text{(Doubly stabilized bundle method)}$$

This method combines the strengths of the proximal and level methods. The presence of $\hat{v}_k$ in the objective functions means that this method can look for good solutions inside the level set, instead of only on the boundaries of it, as the pure level method does. Additionally, the level constraint $\hat{v}_k(x) \leq L_k$ provides a dual variable that is useful for updating the proximal parameter $t_k$. In practice, the doubly stabilized method "chooses" between the level and proximal method in each iteration. If the level constraint is binding, the iteration is a *level iteration*, and otherwise it is a *proximal iteration*.

One important thing to note here with regards to the GBD and NGBD algorithms is that the objective function can not be used to update the lower bound. Because of the proximal penalty term, the optimal objective value will not be the same as the actual objective value of the model. Therefore, the algorithm employs a *lower bounding problem*, which is used to update the lower bound. Please note that this is not the same problem as the lower bounding problem presented in section 3.2. Given a given a current lower bound $v_k^{\min}$, it is defined as follows:

$$\min_{x \in X} \hat{v}_k(x) \text{ s.t. } \hat{v}_k(x) \geq v_k^{\min} \quad \text{(Lower bounding problem)}$$

The optimal objective value of this problem is the new lower bound. Note the restriction $\hat{v}_k(x) \geq v_k^{\min}$, which ensures that the next lower bound is no smaller than the current lower bound.

# Chapter 4

# The distributed implementation

This chapter describes the distributed NGBD method and its implementation. We present the algorithm first and restate some of the assumptions made earlier. The implementation makes use of two optimization solvers, GUROBI and IPOPT, which are presented briefly. Finally, we present and explain the most important aspects of the implementation and some of the issues that arose during the implementation.

## 4.1 The algorithm

The distributed version of the NGBD method follows naturally from chapter 3. The distributed algorithm has a master node that solves the master problem, creates the necessary cuts at each iteration and that controls the whole information flow. The subproblems are solved at other computational nodes that receive a first-stage solution from the master problem, solves the resulting subproblem before returning the ($\epsilon$-)optimal value and its associated ($\epsilon$-)optimal multiplier vector that are assumed to be provided by the solvers used. This information is used to create the cuts that are added to the successive relaxed master problems ($\text{RMP}^{\text{k}}$).

Figure 4.1 shows the decomposition of the original problem (P) presented in chapter 3. The dashed boxes are the problems that are actually solved by the solvers in the algorithm.

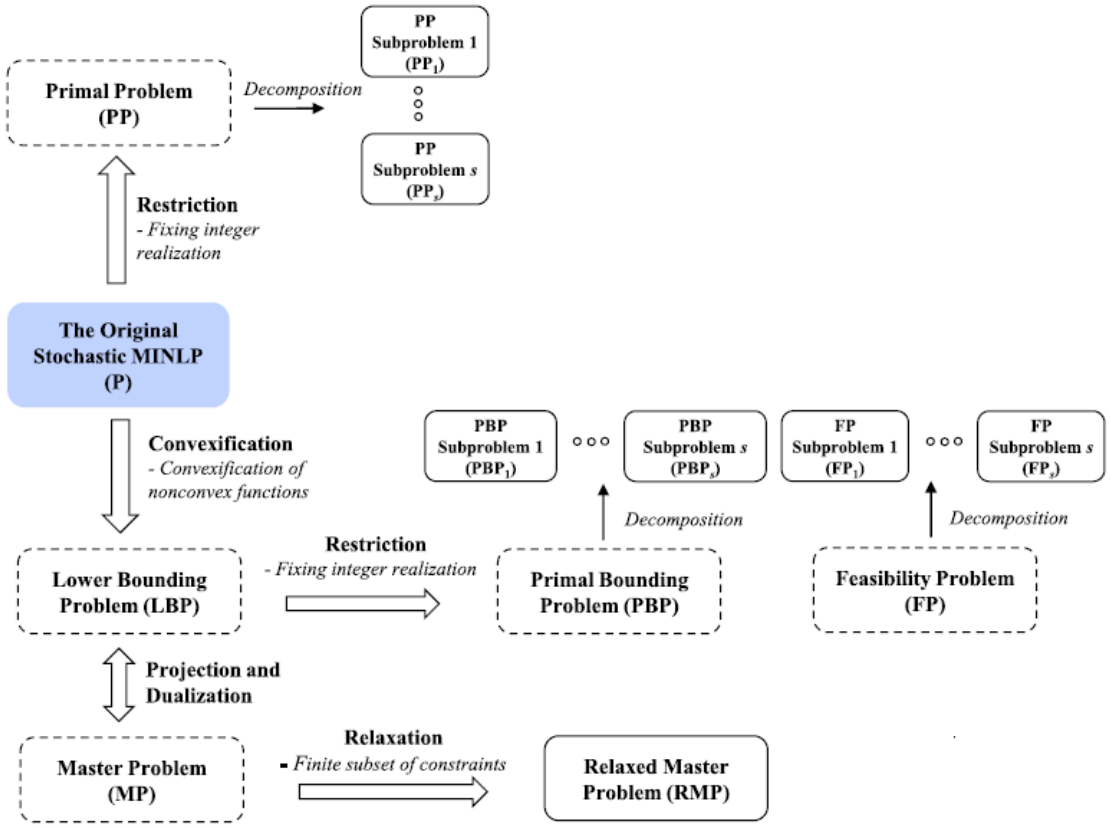
$N$ refers to the number of computational nodes used. On line 19 the algorithm starts solving feasibility problems from scenario $\bar{h}$ which refers to the unsolved scenario with the lowest index $h$. This value of $h$ is chosen so that feasibility problems are solved for all potentially infeasible scenarios.

The master problem receives optimal multiplier vectors from all the subproblems as well as their dual objective functions, which are used to create optimality cuts:

**Algorithm 3** Master node

Initialize iteration counters: $k \leftarrow 0$, $l \leftarrow 1$
Initialize index sets: $T^0 \leftarrow \emptyset$, $I^0 \leftarrow \emptyset$, $U^0 \leftarrow \emptyset$
Initialize bounds: $LBD \leftarrow -\infty$, $UBD \leftarrow \infty$, $UBDPB \leftarrow \infty$
First feasible integer realization $x^{(0)}$ is given.
Send $x^{(0)}$ to all the $|S|$ subproblems of type ($\mathrm{PBP^k_h}$)
Receive optimal objective value for each $PBP_h^0$
Receive optimal multiplier vectors $\lambda_h^k$, create optimality cut and add it to $RMP^{k+1}$
$UBDPB \leftarrow obj_{PBP}(x^{(k)})$, $x^* \leftarrow x^{(0)}$, $k^* \leftarrow k$
$T^0 \leftarrow \emptyset \cup \{k\}$
**while** ($UBDPB < UBD - \epsilon$ or $RMP^k$ is feasible) and $LBD < UBD - \epsilon$ **do**
    **if** ($RMP^k$ is feasible and $LBD < UBDPB$ and $LBD < UBD - \epsilon$) **then**
        **while** $LBD < UBDPB$ and ($RMP^k$) is feasible **do**
            $k \leftarrow k + 1$
            Solve $RMP^k$ and get integer realization $x^{(k)}$
            $LBD \leftarrow obj_{RMP^k}$
            Send $x^{(k)}$ to subproblems $PBP_h^k \forall h$
            **if** $PBP_h^k$ is infeasible for some $\hat{h}$ **then**
                Stop solving subproblems of type $PBP_h^k$
                $\sigma_h^k \leftarrow 0$ for $1, ..., \bar{h}$
                Send $x^{(k)}$ to $FP_h^k$ for $h = \bar{h}, ..., |S|$
                Receive optimal multiplier vectors, $\sigma_h^k$, from feasibility problems and create feasibility cut
                $RMP^{k+1}$ gets feasibility cut
                $I^k \leftarrow I^{k-1} \cup k$
            **else if** $PBP_h^k$ is feasible for all $h$ **then**
                Get objective values and optimal multiplier vectors $\lambda_h^k$ and create optimality cut
                $RMP^{k+1}$ gets optimality cut
                $T^k \leftarrow T^{k-1} \cup \{k\}$
                **if** $obj_{PBP}(x^{(k)}) < UBDPB$ **then**
                    $UBDPB \leftarrow obj_{PBP}(y^k)$, $x^* \leftarrow x^k$, $k* \leftarrow k$
    **if** $UBDPB < UBD - \epsilon$ **then**
        Send $x^*$ to subproblems of type ($PP_h$)
        **while** Receiving solutions from subproblems **do**
            **if** ($obj_{PP_h}(x^*) > UBD - \sum_{i \in S^S} obj_{PP_i}(x^*) - \sum_{j \in S^U} obj_{PBP_j}(x^*)$ **then**
                $obj_{PP}(x^*) \leftarrow UBD + \epsilon$ and stop solving subproblems
        $U^l \leftarrow U^{l-1} \cup \{k*\}$
        **if** $obj_{PP}(x^*) < UBD$ **then**
            $UBD \leftarrow obj_{PP}(y*)$
            $x_p^* = x^*$, $y*_{p,h} = y*_h$ for all $h$
        **if** $T^k \setminus U^l = \emptyset$ **then**
            $UBDPB = \infty$
        **else**
            Choose $i \in T^k \setminus U^l$ such that $obj_{PBP}(x^i) = \min_{j \in T^k \setminus U^l} obj_{PBP}(x^j)$
            $UBDPB \leftarrow obj_{PBP}(x^{(i)}$, $x_p^* \leftarrow x^{(i)}$, $k* \leftarrow i$
            $l \leftarrow l + 1$.
Return $\epsilon$-optimal solution given by $(y*_p, x^*_{p,1}, ..., x^*_{p,s})$.

Figure 4.1: Decomposition of problem (P)




$$\theta \geq w_h(f(\bar{x}^i, \bar{y}_h^i) + (\lambda_h^i)^T(g_h(\bar{y}_h^i) + B_h x) \quad \forall h \in S$$

In many scenario generation methods and the one we use, every scenario is assumed to be equally likely as high probability scenarios are generated more often. This leads to a simpler form for the optimality cut:

$$\theta \geq \frac{1}{|S|}(f(\bar{x}^i, \bar{y}_h^i) + (\lambda_h^i)^T(g_h(\bar{y}_h^i) + B_h x) \quad \forall h \in S$$

The nodes that solve the different subproblems are much simpler as the flow of the algorithm is controlled in the master node.

**Algorithm 4** Subproblem node

1: Receive first stage decision $x^{(i)}$ from master problem
2: Fix all $x$ in the subproblem to $x^{(i)}$
3: Solve
4: **if** Subproblem is of type $PBP_h$ **then**
5: Send the objective value and optimal multiplier vector $\lambda_h^k$ to master node
6: **else if** Subproblem is of type $FP_h$ **then**
7: Send optimal multiplier vector $\sigma_h^k$ to master node
8: **else**
9: Send optimal objective value $obj_{PP_h}(x^*)$

When the subproblems receive a first stage decision $\bar{x}$ all the integer variables in the problem are fixed so that the remaining problem is easier to solve. Under the assumption that Slater's condition is satisfied so that strong duality holds for the convex subproblems $PBP_h$ and $FP_h$, an optimal multiplier vector exists for them, by theorem 2.1.2. This is not the case for non-convex subproblems $PP_h$, but it is not necessary since only the objective value is sent.

### 4.1.1 Cut strategies

In chapter 3 two extreme cut strategies for optimality cuts were presented, one where one single optimality cut is added for all the scenarios called the single-cut approach and one strategy where one cut is added for each scenario called the multi-cut approach. Theoretically the single-cut approach gives a smaller master problem, but will normally require more iterations to converge. A multi-cut approach makes the master problem larger, but will usually require less iterations to converge. For the distributed version the single-cut or an approach close to the single-cut approach is more suitable because the master problem cannot be parallelized and a bigger master problem will require more time to solve. However, in practice there will be many more factors that complicate this issue. This will be discussed further in chapter 5.

### 4.1.2 Starting point

In our experience, the initial solution given to the GBD and NGBD algorithms has a significant impact on solution time. Although there are no clear rules, we have been more successful with using starting points close to the optimal solution (which is intuitive). In order to consistently provide such starting points, we employ a deterministic version of the problem, i.e. with only one scenario and no decomposition. This is solved before the main part of algorithm starts, and its solution is used as a

starting point. While that solution is rarely the same as the final optimal solution, it is often sufficiently close to be a good starting point.

## 4.2 Solvers

For the implementation of the algorithm the open-source software solver IPOPT (Interior Point OPTimizer) (Wächter, 2009) has been used together with the well-known solver GUROBI (Gurobi Optimization, 2015).The IPOPT solver was developed and is maintained by the Computational Infrastructure for Operations Research (COIN-OR) (Lougee-Heimer, 2003). COIN-OR is an open-source initiative that aims to spur on the development of open-source software in the mathematical programming and operations research community, and make it easier to reuse software and replicate results. GUROBI is used to solve the master problem while IPOPT solves the subproblems, both the non-convex problems ($PP^{l}$), the convexified problems ($PBP^{k}{}_{h}$) and the feasibility problems ($FP^{k}{}_{h}$).

### 4.2.1 IPOPT

IPOPT (Interior Point OPTimizer) is a software package for large-scale nonlinear optimization. IPOPT implements an interior-point algorithm for continuous, nonlinear, nonconvex, constrained optimization problems and works as a general purpose nonlinear programming (NLP) solver. The method uses a primal-dual barrier approach within a line search framework. The primal-dual approach means that both the primal and its dual program is considered at the same time.

#### Interior Point methods

Interior point methods (Boyd and Vandenberghe, 2004) are a class of algorithms for solving both linear and non-linear optimization problems. In the case of a linear program, the optimal solution is known to lie at an extreme point of the feasible area, so the well-known Simplex method for example moves along the edges of the feasible region, which is a convex polyhedron, until the optimal solution is found at an extreme point. Interior-point methods on the other hand tries to take a shortcut in that they traverse the interior in search for an optimal extreme point. In convex optimization the optimal value can often be found in the interior of the feasible region so methods for linear programs, such as the Simplex method, that are based on knowing that the optimal solution lies at an extreme point do not work, but interior-point methods do work for both linear and convex programs. Another interesting difference between the Simplex and interior point methods is that Simplex

have exponential complexity in the worst case, meaning that in the worst case, the time to solve a given linear problem with the Simplex grows exponentially as the problem size grows, while interior point methods have polynomial complexity meaning that the running time can be expressed as a polynomial function of the input size.

Given a convex optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, ..., m \\ & Ax = b \end{aligned} \tag{4.1}$$

where $g_i(x)$ are convex and twice differentiable for all $i$, $\mathbf{rank}\ \mathbf{A}^1 = p$ and $f(x)$ is convex. Assume that an optimal $x^*$ exists so that the problem is feasible. Also assume that there exists a strictly feasible point, that is a point for which all $i$ inequalities hold strictly. This means that Slater's condition is satisfied and that strong duality holds. The dual of the problem can be formulated as:

$$\underset{\lambda \geq 0, u}{\text{maximize}} \left\{ \text{minimize } f(x) + \sum_i \lambda_i g_i(x) + u^T(b - Ax) \right\} \tag{4.2}$$

Due to the assumptions made there exists an optimal $x^*$ for (4.1) and optimal $u^*$ and $\lambda^*$ for the dual problem (4.2). Together these optimal vectors satisfy the Karush-Kuhn-Tucker (KKT) conditions, with which it is assumed the reader is familiar:

$$\begin{aligned} Ax^* &= b \\ g_i(x^*) &\leq 0 \\ \lambda^* &\geq 0 \end{aligned} \tag{4.3}$$

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^{p} u_i^*(b_i - A_i x^*) = 0 \tag{4.4}$$

$$\lambda_i g_i(x^*) = 0 \tag{4.5}$$

where $A_i$ is the $i$th row of $A$. (4.3) denotes primal feasibility, (4.4) ensures dual feasibility and (4.5) is the complementary slackness condiiton. KKT is another sufficient condition for strong duality as every combination of $(x, \lambda, u)$ that satisfies the KKT conditions will have zero duality gap. The KKT conditions give a system of equations that gives an alternative way of finding the optimal solution. Interior-point

[1]The dimension spanned by the rows of A, which is the number of linearly independent rows.

methods solves (4.1) or the equation set given by the KKT conditions (4.3)-(4.5) through solving a sequence of equality constrained problems (remember that every inequality can be made into an equality by adding slack variables), or a sequence of equation sets given by modified versions of the KKT conditions that are solved through e.g. Newton's method.[2]

#### IPOPT algorithm

As mentioned above the IPOPT algorithm utilizes a primal-dual barrier approach, which means that both the primal and its dual program are considered at the same time.

A barrier method solves a sequence of barrier problems:

$$\begin{aligned} \text{minimize} \quad & f(x) + \mu \sum_{i=1}^{m} log(-g_i(x)) \\ \text{subject to} \quad & Ax = b \end{aligned} \tag{4.6}$$

for a decreasing sequence of $\mu > 0$ tconverging towards zero. With the inequalities in the objective function the remaining system is solvable by Newton's method for nonlinear equation systems. The inequalities of (4.1) are added into the barrier term of the objective function in a way that drives the objective to infinity when $g_i(x)$ nears their bounds. The main idea is to solve (4.6) for some initial value of $\mu$ starting from a given point, to some given accuracy. Then one solves a new (4.6) for a smaller $\mu$ more accurately starting from the the solution of the previous barrier problem. The solution of the barrier problem converges to the optimal solution of the original problem (4.1) as $\mu \to 0$.

The IPOPT solver also implements a line-search filter method (Wächter and Biegler, 2006). The line search determines the maximum step size along a search direction, e.g. the direction of maximum descent given minimization objective for the barrier problem, while the filter method by provides a way of avoiding cycles by keeping a list of prohibited trial points, similar to a tabu search heuristic.

### 4.2.2 GUROBI

GUROBI is the brainchild of Zonghao Gu, Edward Rothberg and Robert Bixby from whom the name derives. It is known as one of the best-performing commercial solvers today for mixed-integer problems such as problem our master problem, problem

[2]Newton's method refers here to an iterative procedure for numerically solving a set of nonlinear equations

($RMP^k$). Because of the fact that it is a commercial solver the underlying algorithms are not openly available and will not be described here.

### 4.2.3 Solving the non-convex subproblems

To solve the non-convex subproblems to optimality a global solver is needed, and the most appropriate solver is probably the state-of-the-art global solver BARON (Sahinidis, 2014). However, due to licensing issues and the necessity to run simultaneously on up to several hundred computational nodes we have not been able to use BARON. Another candidate, the open-source solver Couenne (Belotti, 2009) from the COIN-OR project unfortunately did not perform well enough on our case study.

Because of these issues and time constraints we decided to use IPOPT for the non-convex subproblems. IPOPT finds reasonable solutions in a short amount of time, but there is no way to know if those solutions are the optimal ones, since IPOPT is a local optimizer when used on non-convex problems. Furthermore, this means that there is no guarantee that the NGBD algorithm will converge. However, since the optimality gap always can be computed, it is possible to give a worst-case quality guarantee for any solution: For example, if the gap is 10%, then we can say that the solution is within 10% of the best solution. Since the duality gap often will be positive the actual optimal solution to the non-convex problem will often be closer than the bound given. To alleviate the problems of using a local solver, we employ a multi-start strategy. This is carried out by solving the same subproblem several times, each time from a different starting point and then choosing the best solution found.

## 4.3 Parallelization

The fact that all subproblems in each iteration are independent means that they can be solved without passing any information between them. Because of this, the algorithm is well-suited for parallelization of the subproblem phase - all subproblems can in theory be solved at the same time.

The algorithm can be divided into two main phases: The master problem phase and the subproblem phase. Unlike the subproblem phase, the master problem phase can not be parallelized by solving a number of different independent problems. The master problem is a single program that can not be decomposed into independent problems, so any parallelization would be far less straightforward. The amount of speed-up gained from parallelizing the algorithm thus depends on the time required in the master problem and subproblem phases. Intuitively from Amdahl's Law, the speed-up would be better if the subproblem phase dominates the workload than vice versa. Considering that the number of subproblems is proportional to the number of scenarios, it can be deduced that the speed-up increases as the number of scenarios increase.

### 4.3.1 Synchronous/asynchronous subproblem assignment

An important decision to make regarding the parallel implementation is whether to make the subproblem assignment synchronous or asynchronous. In a synchronous implementation, assignment of subproblems is organized in rounds. Each round, all nodes are sent a problem to solve. Once all of them are finished with their problem, a new round starts and all nodes are sent new problems. This goes on until all problems have been solved. In an asynchronous implementation, each node is simply assigned a new problem as soon as it has finished its previous one.

Even though the asynchronous implementation seems easier it can often be more complicated than the synchronous one, due to the more chaotic nature of the process. However, it is generally faster, especially when the problems are different and unpredictable in terms of solution time. This can be seen by comparing Figure 4.2 with Figure 4.3: In the synchronous case all nodes must wait for the slowest one in each round, which is potentially very time-wasting. This is not a problem in the asynchronous case, where no node has to wait for another to finish.

Because the difficulty of each subproblem is unpredictable, a synchronous implementation is potentially significantly slower than an asynchronous one. Seeing as the subproblems are completely independent and can be solved in any arbitrary order, an asynchronous implementation is also not very difficult in this case. Therefore we

are using the asynchronous version.

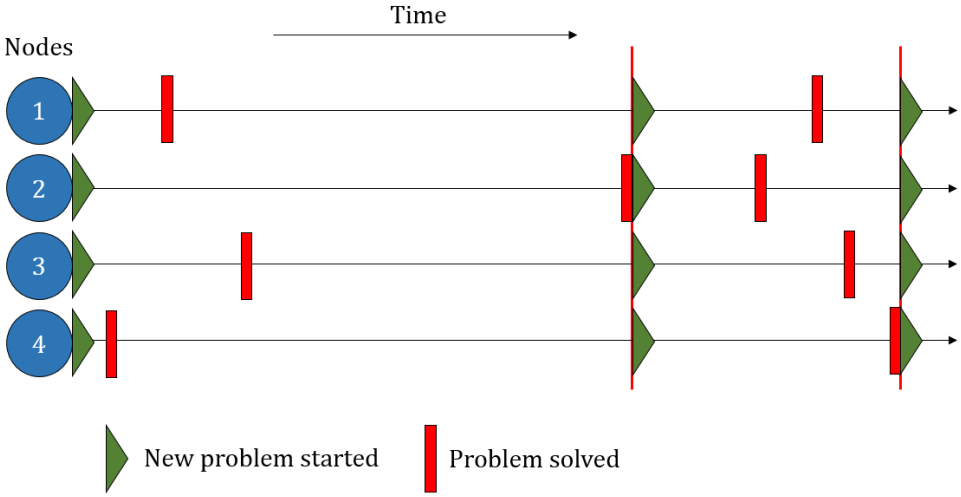



Figure 4.2: Flow of a synchronous parallelization

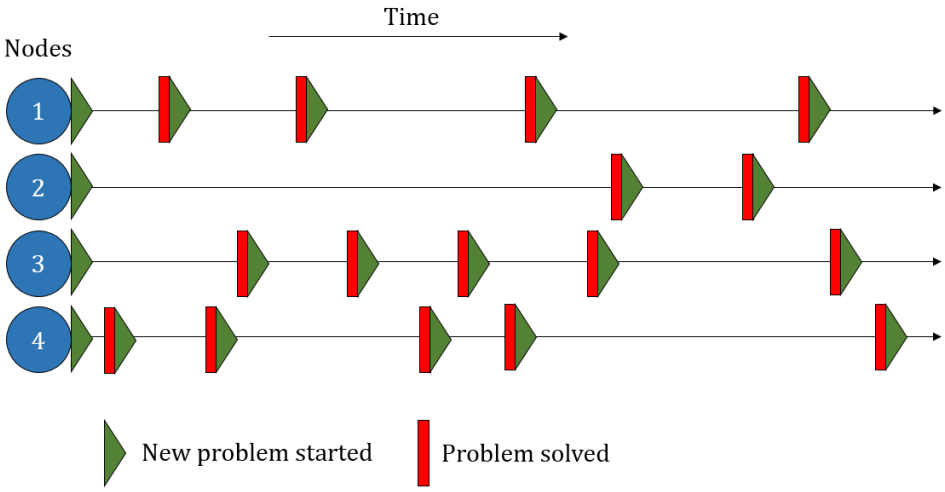



Figure 4.3: Flow of an asynchronous parallelization

## 4.4 The distributed implementation

### 4.4.1 Classes and structure

#### GBD implementation

The C++ implementation takes an object-oriented programming approach. It consists of seven classes, each representing an entity in the algorithm, as shown in table 4.1.

Table 4.1: Implementation classes

| | |
|---|---|
| *SubProb* | Represents the subproblem and contains all necessary parameters to solve it. Is connected to the Ipopt libraries through the TNLP interface. |
| *MasterProb* | Represents the master problem and all necessary parameters to solve it. Contains one instance of *LowerBoundProb*. Includes functionality for adding cuts. Is connected to the Gurobi libraries through a GRBEnv object. |
| *LowerBoundProb* | Represents the lower bounding problem and all necessary parameters to solve it. Is connected to the Gurobi libraries through a GRBEnv object. |
| *DeterministicProb* | Represents the deterministic problem for the expected scenario. Is connected to the Gurobi libraries through a GRBEnv object. |
| *SubWorkerClass* | Represents a worker node responsible for solving subproblems - contains one instance of *SubProb*. Receives first-stage solutions from *GBDclass* and returns the necessary data to make a new cut. |
| *GBDclass* | Represents the node responsible for flow control of the GBD algorithm (the "master node"). Also responsible for solving the master problem - contains one instance each of *MasterProb* and *DeterministicProb*. Allocates scenarios to the *SubWorkerClass* nodes. |
| *Main* | Represents the overarching infrastructure. The top-level class, used for initializing the algorithm and setting up the MPI infrastructure. |

The flow of the implementation is approximately as follows:

1. *Main* starts up and initializes the MPI environment. On the master node an instance of *GBDclass* is initialized. On each of the worker nodes an instance of *SubWorkerClass* is initialized.
2. The master node initiates the algorithm. All parameters, the *DeterministicProb* instance, the *MasterProb* instance and the *LowerBoundProb* instance are initialized.

3. The worker nodes starts listening for MPI messages from the master node.
4. The master node sends its solution of the master problem to the worker nodes and assign each of them a subproblem.
5. The worker nodes initializes an instance of *SubProb* with the data from the master node. The problem is solved, and solution data returned to the master node.
6. The master node collects the subproblem solution data. The data is used to update the upper bound and to apply a new cut in the master problem.
7. The master node solves the master problem and lower bounding problem, to get a new solution and update the lower bound.
8. Number 3-7 are repeated until the lower and upper bound of the objective value converge.
9. When the master node detects convergence, it tells all worker nodes to shut down, writes the results to file and shuts down itself.

### NGBD implementation

The NGBD implementation builds on the GBD implementation, and is thus very similar. They are described in table 4.2.

The flow of the implementation is naturally also similar to the GBD one, but with a few more steps and an additional loop in the main flow.

1. *Main* starts up and initializes the MPI environment. On the master node an instance of *GBDclass* is initialized. On each of the worker nodes an instance of *SubWorkerClass* is initialized.
2. The master node initiates the algorithm. All parameters, the *DeterministicProb* instance, the *MasterProb* instance and the *LowerBoundProb* instance are initialized.
3. The worker nodes starts listening for MPI messages from the master node.
4. The master node sends its solution of the master problem to the worker nodes and assign each of them a convex subproblem.
5. The worker nodes initializes an instance of *SubProb* with the data from the master node. The problem is solved, and solution data returned to the master node.
6. The master node collects the subproblem solution data. The data is used to update the upper bound for the bounding problem, and to apply a new cut in the master problem.
7. The master node solves the master problem and lower bounding problem, to get a new solution and update the lower bound.

Table 4.2: Implementation classes

| | |
|---|---|
| *SubProb* | Represents the subproblem and contains all necessary parameters to solve it. Is connected to the Ipopt libraries through the TNLP interface. |
| *NCSubProb* | Represents the nonconvex subproblem and contains all necessary parameters to solve it. Is connected to the Ipopt libraries through the TNLP interface. |
| *MasterProb* | Represents the master problem and all necessary parameters to solve it. Contains one instance of *LowerBoundProb*. Includes functionality for adding cuts. Is connected to the Gurobi libraries through a GRBEnv object. |
| *LowerBoundProb* | Represents the lower bounding problem and all necessary parameters to solve it. Is connected to the Gurobi libraries through a GRBEnv object. |
| *DeterministicProb* | Represents the deterministic problem for the expected scenario. Is connected to the Gurobi libraries through a GRBEnv object. |
| *SubWorkerClass* | Represents a worker node responsible for solving subproblems - contains one instance each of *SubProb* and *NCSubProb*. It receives first-stage solutions from *GBDclass* and returns the solution data. |
| *NGBDclass* | Represents the node responsible for flow control of the NGBD algorithm (the "master node"). Also responsible for solving the master problem - contains one instance each of *MasterProb* and *DeterministicProb*. Allocates scenarios to the *SubWorkerClass* nodes. |
| *Main* | Represents the overarching infrastructure. The top-level class, used for initializing the algorithm and setting up the MPI infrastructure. |

8. Number 3-7 are repeated until the lower and upper bound of the bounding problem converge.
9. The master node sends the best solution to the worker nodes and assigns each of them a nonconvex subproblem.
10. The worker nodes initializes an instance of *NCSubProb* with the data from the master node. The problem is solved, and solution data returned to the master node.
11. Number 3-10 are repeated until the lower and upper bound of the bounding problem converge.
12. When the master node detects convergence, it tells all worker nodes to shut down, writes the results to file and shuts down itself.

# Chapter 5

# The unit-commitment problem and case study

This chapter presents the unit commitment problem and a stochastic MINLP formulation of the problem. The model is first presented as a general model based on a fictitious power system based on the Brazilian power system. A specific version of the model with data is then introduced and its properties discussed, including implications for the implementation.

## 5.1 The unit commitment problem

The unit commitment problem (Castillo et al.) is an optimization problem that arises in short-term planning of electrical power production. The goal is to determine the optimal schedule in terms of some objective while meeting demand for electrical power. A schedule decides when a unit in the system should be generating or not. An example of a unit can be a hydropower turbine, several similar turbines modeled as one superunit or an entire thermal power plant. The problem naturally lends itself to a stochastic model as there are several sources of variability that affects a schedule such as variations in demand and unpredictable energy sources such as wind power and solar power. It can also be viewed as having two main stages: the strategic stage where the schedule is made and the operational stage where given a schedule the goal is to minimize the operational costs. The operational stage is the recourse stage. The problem can be formulated as a stochastic programming problem with multiple stages where the strategic decisions that must be made are the operating schedule for the next operating horizon, usually between 24 and 168 hours. The problem in the operational horizon is then to determine how to produce the necessary power as cost-effective as possible with the given committed units

and the realization of random variables. In the problem there are several types of constraints, both on the strategic and operational level, which will be presented below.

## 5.2 The general model

This section presents the general model of a unit commitment problem in a power system with reservoir-based hydropower plants, run-of-river hydropower plants and thermal power plants, which will be referred to as a hydro-thermal system. First some notational choices will be explained.

Some of the constraints are formulated in terms of absolute values for brevity, but they can be formulated as two linear constraints, as shown below.

$$|x_t - x_{t-1}| \leq M$$

can be reformulated to

$$x_t - x_{t-1} \leq M$$
$$x_{t-1} - x_t \leq M$$

where $x_t$ is some variable for a time period $t$ and $M$ is a fixed parameter restricting state changes.

### 5.2.1 Switching constraints

The following constraints are necessary to connect the state variables $u_{trj}$, which model whether a unit is on or off, to the switching variables $\widetilde{u}_{trj}$, that are necessary to incur potential start-up costs on the units. The $u_{trj}$ variables are defined as:

$$u_{trj} = \begin{cases} 1 & \text{if unit } j \text{ in plant } r \text{ is on at time } t \\ 0 & \text{otherwise} \end{cases}$$

while the $\widetilde{u}_{trj}$ are defined as:

$$\widetilde{u}_{trj} = \begin{cases} 1 & \text{if unit } j \text{ in plant } r \text{ is switched on at time } t \\ 0 & \text{otherwise} \end{cases}$$

Constraints (5.1)-(5.3), (5.9)-(5.11) and (5.16)-(5.18) enforces the switching variables $\widetilde{u}$ to 1 whenever a unit is switched on and restricts the number of times each unit can be switched on in each operational horizon.

### 5.2.2 Hydropower plants with reservoir

The constraints for the hydropower plants with reservoir concerns the volume of the reservoir, the maximum number of start-ups of the units in the plants and their minimum and maximum production capacity. Constraints imposed by rules and regulations are also common such as maximum discharge to avoid flooding and minimal discharge to preserve river flow.

#### Strategic constraints

The strategic constraints (5.1)-(5.3) are the maximum number of start-ups for each operational period.

$$\widetilde{u}^H_{trj} - u^H_{trj} = 0 \qquad t = 0, r \in R^H, j \in J(r) \tag{5.1}$$

$$\widetilde{u}^H_{trj} - u^H_{trj} + u^H_{t-1,rj} \geq 0 \qquad t \in T \setminus \{0\}, r \in R^H, j \in J(r) \tag{5.2}$$

$$\sum_{t \in T} \widetilde{u}^H_{trj} \leq \overline{S}^H \qquad r \in R^H, j \in J(r) \tag{5.3}$$

The set $R^H$ is the set of hydropower plants with reservoirs, indexed with $r$. The set $J(r)$ is the set of units in plant $r$, indexed with $j$. $T$ is the set of time periods, indexed with $t$.

#### Operational constraints

$$g^H_{trj} \leq \mathcal{H}^H_{rj}(q^H_{trj}, s^H_{tr}, v^H_{tr}) \qquad t \in T, r \in R^H, j \in J(r) \tag{5.4}$$

$$v_{tr} - V_{0,r} + \sum_{j \in J(r)} q^H_{trj} + s^H_{tr} = Y^H_{sr} \qquad t = 0, r \in R^H \tag{5.5}$$

$$v_{tr} - v_{t-1,r} + \sum_{j \in J(r)} q^H_{trj} + s^H_{tr} = Y^H_{sr} \qquad t \in T \setminus \{0\}, r \in R^H \tag{5.6}$$

$$u^H_{trj} \underline{G}^H_{rj} \leq g^H_{trj} \leq u^H_{trj} \overline{G}^H_{rj} \qquad t \in T, r \in R^H, j \in J(r) \tag{5.7}$$

$$\underline{O}^H_r \leq \sum_{j \in J(r)} q^H_{trj} + s^H_{tr} \leq \overline{O}^H_r \qquad t \in T, r \in R^H \tag{5.8}$$

$\mathcal{H}^H_{rj}$ refers to the production function for unit $j$ in plant $r$. The function can be linear, concave or non-concave. (5.4) sets the power generated $g^H_{trj}$ by a unit $j$ as a function of the discharge through the unit $q^H_{trj}$, the water spill from the plant $s^H_{tr}$ and the reservoir level $v^H_{rt}$. This non-linear function is usually unique for each plant. (5.5) and (5.6) are the reservoir balance constraints. They make sure that the inflow to the plant is equal to the total water discharge, water spill and reservoir fill-up.

(5.7) forces the power generated to stay within the actual capacities of the units while (5.8) keeps the river flow at the desired levels.

### 5.2.3 Run-of-river hydropower plants

Run-of-river hydropower plants do not have reservoirs but instead depend on the river flow in the same way a wind farm would depend on the wind. However, a run-of-river hydropower plant might be downstream from a reservoir. This will affect the inflows to the run-of-river plant and must be taken into account. This is done through constraint (5.13).

#### Strategic constraints

The strategic constraints for the run-of-river hydropower plants are the maximum number of start-ups for the units.

$$\widetilde{u}^F_{trj} - u^F_{trj} = 0 \qquad t = 0, r \in R^F, j \in J(r) \tag{5.9}$$

$$\widetilde{u}^F_{trj} - u^F_{trj} + u^F_{t-1,rj} \geq 0 \qquad t \in T \setminus \{0\}, r \in R^F, j \in J(r) \tag{5.10}$$

$$\sum_{t \in T} \widetilde{u}^F_{trj} \leq \overline{S}^F \qquad r \in R^F, j \in J(r) \tag{5.11}$$

The set $R^F$ is the set of run-of-river hydropower plants. The set $U(r, \tau)$ is the set of upstream hydropower plants for plant $r$ with a travel time of $\tau$. The outflow from upstream plants must be considered at every downstream power plant so that the river flow is consistent.

#### Operational constraints

$$g^F_{trj} \leq \mathcal{H}^F_{rj}(q^F_{trj}, s^F_{tr}) \qquad t \in T, r \in R^F, j \in J(r) \tag{5.12}$$

$$\sum_{j \in J(r)} q^F_{trj} + s^F_{tr} - \sum_{\tau \in \mathcal{T}} \sum_{r' \in U(r,\tau)} (s_{(t-\tau)r'} + \sum_{j' \in J(r')} q_{(t-\tau)r'j'}) = Y^F_{tr} \qquad t \in T, r \in R^F \tag{5.13}$$

$$u^F_{trj} \underline{G}^F_{rj} \leq g^F_{trj} \leq u^F_{trj} \overline{G}^F_{rj} \qquad t \in T, r \in R^F, j \in J(r) \tag{5.14}$$

$$\underline{O}^F_r \leq \sum_{j \in J(r)} q^F_{trj} + s^F_{tr} \leq \overline{O}^F_r \qquad t \in T, r \in R^F \tag{5.15}$$

$\mathcal{H}^F_{rj}$ refers to the production function for unit $j$ in plant $r$ which can be concave or non-convex. (5.12) describes the power generated $g^F_{trj}$ as a function of discharge $q^F_{trj}$ through a unit and spill $s^F_{tr}$ from its plant. (5.13) forces the discharge and spill for a run-of-river hydropower plant to equal inflows and discharge from the upstream plants. (5.14) and (5.15) gives the generation limits, both upper and lower, and the limits on river flow.

### 5.2.4 Thermal power plants

The thermal power plants behave differently from the hydropower plants and are not dependent on any random variables.

#### Strategic constraints

The strategic constraints for the thermal power plants are the minimum up- and downtime for each plant and possibly maximum start-ups.

$$\widetilde{u}^T_{tr} - u^T_{tr} = 0 \qquad t = 0, r \in R^T \tag{5.16}$$
$$\widetilde{u}^T_{tr} - u^T_{tr} + u^T_{t-1,r} \geq 0 \qquad t \in T \setminus \{0\}, r \in R^T \tag{5.17}$$
$$\sum_{t \in T} \widetilde{u}^T_{tr} \leq \overline{S}^T \qquad r \in R^T \tag{5.18}$$
$$u^T_{tr} \geq u^T_{cr} - u^T_{c-1,r} \qquad c \in [t - T^{up}_i + 1, t - 1], r \in R^T \tag{5.19}$$
$$u^T_{tr} \leq 1 + (u^T_{cr} - u^T_{c-1,r}) \qquad c \in [t - T^{down}_i + 1, t - 1], r \in R^T \tag{5.20}$$

The set $R^T$ is the set of thermal plants. The $c$ index describes a subset of the set $T$ in constraints (5.19) and (5.20). The subset is defined as a moving window where the plant must be on if it is turned on at time $t$ or a window where it must be turned off if the plant is turned off at time $t$.

#### Operational constraints

$$u^T_{tr} \leq \underline{G}^T_r \qquad t = 0, r \in R^T \tag{5.21}$$
$$u^T_{tr} - u^T_{t-1,r} \leq u_{t-1,r}\overline{\Delta}_r + (1 - u_{t-1,r})\underline{G}^T_r \qquad t \in T \setminus \{0\}, r \in R^T \tag{5.22}$$
$$-u^T_{tr} \leq u_{tr}\underline{\Delta}_r + (1 - u_{t,r})\underline{G}^T_r \qquad t = 0, r \in R^T \tag{5.23}$$
$$u^T_{t-1,r} - u^T_{tr} \leq u_{tr}\underline{\Delta}_r + (1 - u_{t,r})\underline{G}^T_r \qquad t \in T \setminus \{0\}, r \in R^T \tag{5.24}$$
$$u^T_{tr}\underline{G}^T_r \leq g^T_{tr} \leq u^T_{tr}\overline{G}^T_r \qquad t \in T, r \in R^T \tag{5.25}$$

(5.25) describes the lower and upper bounds when a thermal plant is producing electricity. Constraints (5.22)-(5.23) describes the ramp constraints on a thermal

plant to more realistically model the response time of a thermal power plant that can only be adjusted gradually.

### 5.2.5 Artificial power plants

The artificial power plants are included to model shortfall in production and are not constrained, meaning that they have an infinite capacity and instant production. They are always available and can generate as much power as needed, however the production cost is very high so that they only are used whenever no other alternatives are possible to satisfy demand. The cost of the artificial plants can for example represent fines or other penalties incurred when demand is not satisfied. Another reason to include artificial plants rather than making the problem infeasible is that infeasibility does not have a natural interpretation and because not meeting demand is a very real possibility.

### 5.2.6 Network and demand constraints

The network is the electrical network of transmission lines from the different power plants to the user. The network constraints defines the relationship between the different buses in terms of where electricity is generated and where it can be sent to as well as how much capacity each transmission line has. Each bus in the network has its own demand that must be satisfied, either with electricity produced at the same bus or by getting power transported to it. Each bus has an electrical angle and the angle difference between two buses gives the power flow between them and the direction of the flow, through constraints (5.26). The positive direction of the flow is given by lower to higher bus number, as shown in figure (5.2). This convention corresponds with the sign of the difference between the angles so that the subtrahend in the constraint receives positive flow while the minuend receives negative flow. The network and demand constraints are all in the operational level of the problem.

$$X_{ii'} f_{tii'} = \omega_{ti} - \omega_{ti'} \qquad i \in \mathcal{I}, i' \in \mathcal{O}(i) \tag{5.26}$$

$$\begin{aligned} &\sum_{j \in \mathcal{J}^H(i)} g^H_{trj} + \sum_{j \in \mathcal{J}^F(i)} g^F_{trj} + \sum_{r \in \mathcal{R}^T(i)} g^T_{tr} \\ &+ \sum_{r \in \mathcal{R}^{Inf}(i)} g^{inf}_{tr} - \sum_{i' \in \mathcal{O}(i)} (f_{tii'} - f_{ti'i}) \geq D_{ti} \qquad t \in \mathcal{T}, i \in \mathcal{I} \end{aligned} \tag{5.27}$$

The sets $\mathcal{J}^H(i)$ and $\mathcal{J}^F(i)$ contains the hydropower units connected to a bus $i \in \mathcal{I}$ while the sets $\mathcal{R}^T(i)$ and $\mathcal{R}^{Inf}(i)$ contain the power plants connected to bus $i$.

(5.27) makes sure that the demand is fulfilled in each period. They are formulated as inequalities since it is a possibility to produce more power than necessary if it is cheaper than turning units on and off due to start-up costs and the like. The generation variable $g^{inf}$ represents high-cost power generation from an artifical plant. This generation "source" makes it possible to satisfy the demand constraints at all times, but at a potentially very high cost.

### 5.2.7 The objective

The objective in this model is to minimize the expected cost of production when demand and water inflows are uncertain. There are several alternatives for an objective, e.g. maximizing profits. The Brazilian power system on which this specific problem is based is highly regulated, so minimization of costs is the most natural objective.

$$\begin{aligned}\min \quad & \sum_{t\in T}\sum_{r\in R^H}\sum_{j\in J(r)}(S^H_{trj}\widetilde{u}^H_{trj}+C^H_{trj}u^H_{trj})+\sum_{t\in T}\sum_{r\in R^F}\sum_{j\in J(r)}(S^F_{trj}\widetilde{u}^F_{trj}+C^F_{trj}u^H_{trj}) \\ & +\sum_{t\in T}\sum_{r\in R^T}(S^T_{tr}\widetilde{u}^T_{tr}++C^T_{tr}u^T_{tr})+\mathbb{E}\left[Q(x,\xi)\right]\end{aligned} \tag{5.28}$$

where

$$\begin{aligned}Q(x,\xi)=\min & \sum_{t\in T}\sum_{r\in R^H}\sum_{j\in J(r)}GC^H(g^H_{trj})+\sum_{t\in T}\sum_{r\in R^F}\sum_{j\in J(r)}GC^F(g^F_{trj}) \\ & +\sum_{t\in T}\sum_{r\in R^T}GC^T(g^T_{tr})+\sum_{t\in T}\sum_{r\in R^{Inf}}c^{inf}g^{inf}_t\end{aligned} \tag{5.29}$$

The $GC$ functions are the generating cost functions for their respective technologies which are all convex.

Now the model will be shown in its entirety.

## Sets

| | |
|---|---|
| $\mathcal{T}$ | Set of time periods |
| $\mathcal{S}$ | Set of scenarios |
| $\mathcal{I}$ | Set of buses |
| $\mathcal{R}^H$ | Set of hydropower plants with reservoir |
| $\mathcal{R}^F$ | Set of run-of-river hydropower plants |
| $\mathcal{R}^T$ | Set of thermal plants |
| $\mathcal{J}(r)^H$ | Set of units in hydropower plant with reservoir $r$ |
| $\mathcal{J}(r)^F$ | Set of units in run-of-river hydropower plant $r$ |
| $\mathcal{O}(i)$ | Set of receiving buses for lines originating in bus $i$ |
| $\mathcal{O}(i)^H$ | Set of units in hydropower plants with reservoirs connected to bus $i$ |
| $\mathcal{O}(i)^F$ | Set of units in run-of-river hydropower plants connected to bus $i$ |
| $\mathcal{O}(r)^T$ | Set of thermal plants connected to bus $i$ |
| $\mathcal{O}(i)^{Inf}$ | Set of artificial infinite plants connected to bus $i$ |
| $\mathcal{U}(r,\tau)$ | Set of upstream hydropower plants for plant $r$ with travel time $\tau$ |

## Parameters

| | |
|---|---|
| $D_{sti}$ | System demand in scenario $s$ and time period $t$ at bus $i$ |
| $S^H_{rj}$ | Start-up cost for unit $j$ in hydropower plant with reservoir $r$ |
| $S^F_{rj}$ | Start-up cost for unit $j$ in run-of-river hydropower plant $r$ |
| $S^T_r$ | Start-up cost for thermal plant $r$ |
| $C^H_{rj}$ | Fixed costs for unit $j$ in hydropower plant with reservoir $r$ |
| $C^F_{rj}$ | Fixed costs for unit $j$ in hydropower run-of-river plant $r$ |
| $C^T_r$ | Fixed costs for thermal plant $r$ |
| $C^{inf}_r$ | Generating cost for artificial plant $r$ |
| $\overline{O}^H_r$ | Maximum total outflow from hydropower plant with reservoir $r$ |
| $\underline{O}^H_r$ | Minimum total outflow from hydropower plant with reservoir $r$ |
| $\overline{G}^H_{rj}$ | Maximum power generation from unit $j$ in hydropower plant with reservoir $r$ |
| $\underline{G}^H_{rj}$ | Minimum power generation from unit $j$ in hydropower plant with reservoir $r$ |
| $\overline{G}^F_{rj}$ | Maximum power generation from unit $j$ in run-of-river hydropower plant $r$ |
| $\underline{G}^F_{rj}$ | Minimum power generation from unit $j$ in run-of-river hydropower plant $r$ |
| $\overline{G}^T_r$ | Maximum power generation from thermal plant $r$ |
| $\underline{G}^T_r$ | Minimum power generation from thermal plant $r$ |
| $\overline{L}_{ii'}$ | Maximum flow capacity over line from bus $i$ to $i'$ |
| $Y^H_{str}$ | Inflow to hydropower plant with reservoir $r$ at time $t$ in scenario $s$ |
| $Y^F_{str}$ | Inflow to run-of-river hydropower plant $r$ at time $t$ in scenario $s$ |
| $V^H_r$ | Reservoir level at starting time in hydropower plant with reservoir $r$ |
| $\Delta_r$ | Ramp limit for thermal plant $r$ |
| $X_{ii'}$ | Line reactance on line between bus $i$ and $i'$ |

**Variables**

| | |
|---|---|
| $u^H_{trj}$ | Binary, on/off for unit $j$ in hydropower plant with reservoir $r$ at time $t$ |
| $u^F_{trj}$ | Binary, on/off for unit $j$ in run-of-river hydropower plant $r$ at time $t$ |
| $u^T_{tr}$ | Binary, on/off for thermal plant $r$ at time $t$ |
| $\widetilde{u}^H_{trj}$ | Binary, switching on unit $j$ in hydropower plant with reservoir $r$ at time $t$ |
| $\widetilde{u}^F_{trj}$ | Binary, switching on unit $j$ in run-of-river hydropower plant $r$ at time $t$ |
| $\widetilde{u}^T_{tr}$ | Binary, switching on thermal plant $r$ at time $t$ |
| $g^H_{trj}$ | Power generated by unit $j$ in hydropower plant with reservoir $r$ at time $t$ |
| $g^F_{trj}$ | Power generated by unit $j$ in run-of-river hydropower plant $r$ at time $t$ |
| $g^T_{tr}$ | Power generated by thermal plant $r$ at time $t$ |
| $g^{inf}_t$ | Power generated by artificial plant at time $t$ |
| $q^H_{trj}$ | Discharge through unit $j$ in hydropower plant with reservoir $r$ at time $t$ |
| $q^F_{trj}$ | Discharge through unit $j$ in run-of-river hydropower plant $r$ at time $t$ |
| $s^H_{tr}$ | Spill in hydropower plant with reservoir $r$ at time $t$ |
| $s^F_{tr}$ | Spill in run-of-river hydropower plant $r$ at time $t$ |
| $v^H_{tr}$ | Reservoir level in hydropower plant with reservoir $r$ at time $t$ |
| $\omega ti$ | Angle at bus $i$ at time $t$ |
| $f_{tii'}$ | Power flow in line from bus $i$ to $i'$ at time $t$ |

$$\text{minimize} \quad \sum_{t\in T}\sum_{r\in R^H}\sum_{j\in J(r)}(S^H_{trj}\widetilde{u}^H_{trj}+C^H_{trj}u^H_{trj})+\sum_{t\in T}\sum_{r\in R^F}\sum_{j\in J(r)}(S^F_{trj}\widetilde{u}^F_{trj}+C^F_{trj}u^H_{trj})$$
$$+\sum_{t\in T}\sum_{r\in R^T}(S^T_{tr}\widetilde{u}^T_{tr}++C^T_{tr}u^T_{tr})+\mathrm{E}[Q(x,\xi)]$$

subject to

$$\widetilde{u}^H_{trj}-u^H_{trj}=0 \qquad t=0, r\in R^H, j\in J(r) \qquad (5.1')$$
$$\widetilde{u}^H_{trj}-u^H_{trj}+u^H_{t-1,rj}\geq 0 \qquad t\in T\setminus\{0\}, r\in R^H, j\in J(r) \qquad (5.2')$$
$$\sum_{t\in T}\widetilde{u}^H_{trj}\leq \overline{S}^H \qquad r\in R^H, j\in J(r) \qquad (5.3')$$
$$\widetilde{u}^F_{trj}-u^F_{trj}=0 \qquad t=0, r\in R^F, j\in J(r) \qquad (5.9')$$
$$\widetilde{u}^F_{trj}-u^F_{trj}+u^F_{t-1,rj}\geq 0 \qquad t\in T\setminus\{0\}, r\in R^F, j\in J(r) \qquad (5.10')$$
$$\sum_{t\in T}\widetilde{u}^F_{trj}\leq \overline{S}^F \qquad r\in R^F, j\in J(r) \qquad (5.11')$$
$$\widetilde{u}^T_{tr}-u^T_{tr}=0 \qquad t=0, r\in R^T \qquad (5.16')$$
$$\widetilde{u}^T_{tr}-u^T_{tr}+u^T_{t-1,r}\geq 0 \qquad t\in T\setminus\{0\}, r\in R^T \qquad (5.17')$$
$$\sum_{t\in T}\widetilde{u}^T_{tr}\leq \overline{S}^T \qquad r\in R^T, \qquad (5.18')$$
$$u^T_{tr}\geq u^T_{cr}-u^T_{c-1,r} \qquad c\in[t-T^{up}_i+1,t-1], r\in R^T \qquad (5.19')$$
$$u^T_{tr}\leq 1+(u^T_{cr}-u^T_{c-1,r}) \qquad c\in[t-T^{down}_i+1,t-1], r\in R^T \qquad (5.20')$$
$$u^H_{trj},\widetilde{u}^H_{trj},u^F_{trj},\widetilde{u}^F_{trj},u^T_{tr},\widetilde{u}_{trj}t^T\in\{0,1\}$$

where

$$Q(x,\xi):=\text{minimize}\sum_{t\in T}\sum_{r\in R^H}\sum_{j\in J(r)}GC^H_{rj}(g^H_{trj})+\sum_{t\in T}\sum_{r\in R^F}\sum_{j\in J(r)}GC^F_{rj}(g^F_{trj})$$
$$+\sum_{t\in T}\sum_{r\in R^T}GC^T_r(g^T_{tr})+\sum_{t\in T}C^{inf}_r g^{inf}_{tr}$$

subject to

$$g^H_{trj} \le \mathcal{H}^H_{rj}(q^H_{trj}, s^H_{tr}, v^H_{tr}) \qquad t \in T, r \in R^H, j \in J(r)$$

(5.4')

$$v_{tr} - V_{0,r} + \sum_{j \in J(r)} q^H_{trj} + s^H_{tr} = Y^H_{sr} \qquad t = 0, r \in R^H$$

(5.5')

$$v_{tr} - v_{t-1,r} + \sum_{j \in J(r)} q^H_{trj} + s^H_{tr} = Y^H_{sr} \qquad t \in T \setminus \{0\}, r \in R^H$$

(5.6')

$$u^H_{trj} \underline{G}^H_{rj} \le g^H_{trj} \le u^H_{trj} \overline{G}^H_{rj} \qquad t \in T, r \in R^H, j \in J(r)$$

(5.7')

$$\underline{O}^H_r \le \sum_{j \in J(r)} q^H_{trj} + s^H_{tr} \le \overline{O}^H_r \qquad t \in T, r \in R^H$$

(5.8')

$$g^F_{trj} \le \mathcal{H}^F_{rj}(q^F_{trj}, s^F_{tr}) \qquad t \in T, r \in R^F, j \in J(r)$$

(5.12')

$$\sum_{j \in J(r)} q^F_{trj} + s^F_{tr} - \sum_{\tau \in \mathcal{T}} \sum_{r' \in U(r,\tau)} (s_{(t-\tau)r'} + \sum_{j' \in J(r')} q_{(t-\tau)r'j'}) = Y^F_{tr} \qquad t \in T, r \in R^F$$

(5.13')

$$u^F_{trj} \underline{G}^F_{rj} \le g^F_{trj} \le u^F_{trj} \overline{G}^F_{rj} \qquad t \in T, r \in R^F, j \in J(r)$$

(5.14')

$$\underline{O}^F_r \le \sum_{j \in J(r)} q^F_{trj} + s^F_{tr} \le \overline{O}^F_r \qquad t \in T, r \in R^F$$

(5.15')

$$u^T_{tr} \le \underline{G}^T_r \qquad t = 0, r \in R^T$$

(5.21')

$$u^T_{tr} - u^T_{t-1,r} \le u_{t-1,r} \overline{\Delta}_r + (1 - u_{t-1,r}) \underline{G}^T_r \qquad t \in T \setminus \{0\}, r \in R^T$$

(5.22')

$$-u^T_{tr} \le u_{tr} \underline{\Delta}_r + (1 - u_{t,r}) \underline{G}^T_r \qquad t = 0, r \in R^T$$

(5.23')

$$u^T_{t-1,r} - u^T_{tr} \le u_{tr} \underline{\Delta}_r + (1 - u_{t,r}) \underline{G}^T_r \qquad t \in T \setminus \{0\}, r \in R^T$$

(5.24')

$$u^T_{tr} \underline{G}^T_r \le g^T_{tr} \le u^T_{tr} \overline{G}^T_r \qquad t \in T, r \in R^T$$

(5.25')

$$X_{ii'} f_{tii'} = \omega_{ti} - \omega_{ti'} \qquad i \in \mathcal{I}, i' \in \mathcal{O}(i)$$

(5.26')

$$\begin{aligned}
&\sum_{j\in\mathcal{J}^H(i)} g^H_{trj} + \sum_{j\in\mathcal{J}^F(i)} g^F_{trj} + \sum_{r\in\mathcal{R}^T(i)} g^T_{tr} \\
&+ \sum_{r\in\mathcal{R}^{Inf}(i)} g^{inf}_{tr} - \sum_{i'\in\mathcal{O}(i)} (f_{tii'} - f_{ti'i}) \geq D_{ti} \qquad t\in\mathcal{T}, i\in\mathcal{I} \\
&g^H_{trj}, g^F_{trj}, g^T_{tr}, g^{inf}_t, q^H_{trj}, q^F_{trj}, s^H_{rt}, s^F_{tr}, v^H_{tr} \geq 0
\end{aligned} \tag{5.27'}$$

## 5.3 The specific system

This section presents the specific system considered, which is fictitious but based on the Brazilian power system.

The system consists of:

- 1 hydropower plant with reservoir, with 3 units
- 2 run-of-river hydropower plants, each with 2 units
- 2 thermal power plants
- 2 wind farms
- 4 artificial power plants (one at each bus with non-zero demand)
- 5 buses

One of the run-of-river hydropower plants is downstream from the reservoir (as shown in figure 5.1 below). The electrical network consists of 5 buses as seen in

Figure 5.1: Topology of the hydropower systems

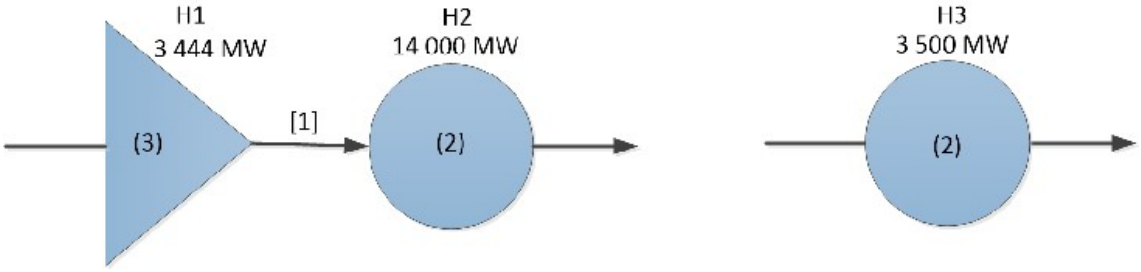
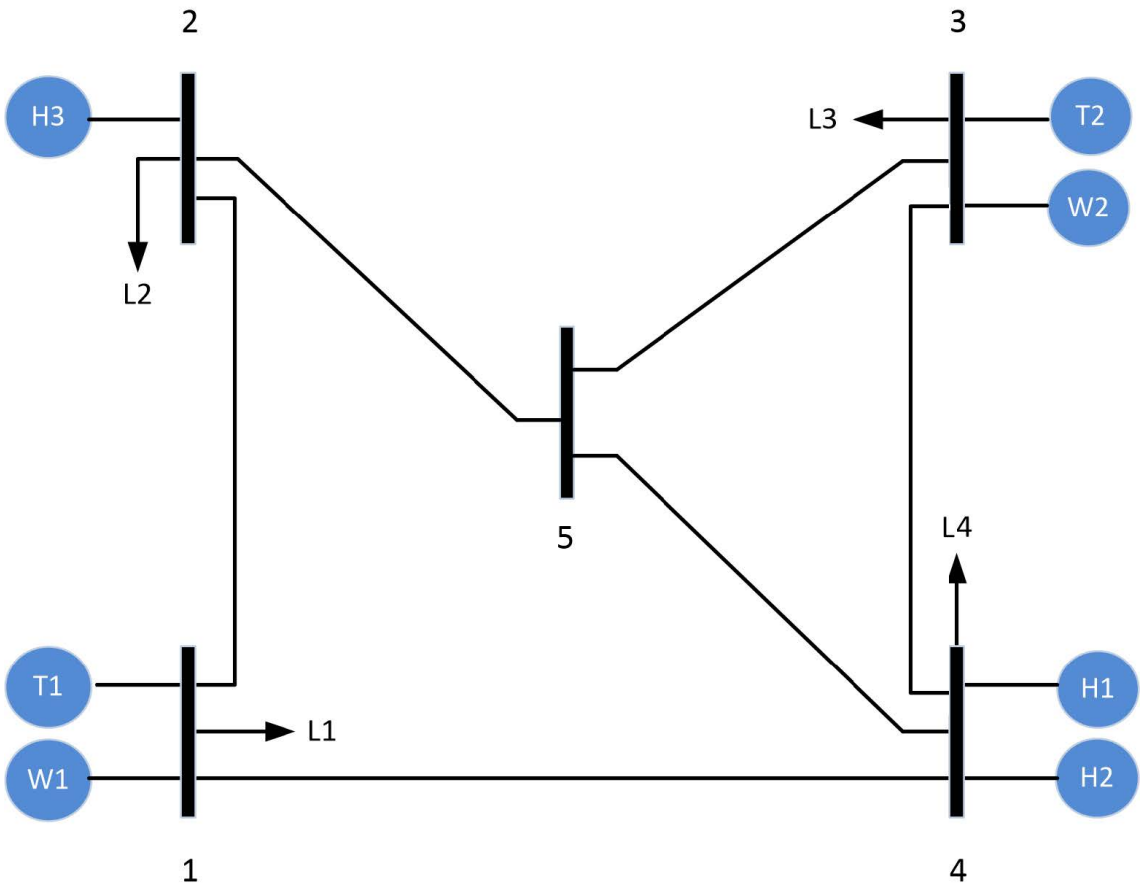



figure 5.2. A bus is a node in the network where power can be produced and used.

In the specific problem there are several versions of the hydropower production functions $\mathcal{H}$. The non-convex functions are given below:

$$\begin{aligned}
\mathcal{H}^H_{t1j} = \; & p_{10}(q^H_{t1j}) + p_{11}(q^H_{t1j})v_{t1} + \\
& p_{20}(q^H_{t1j})^2 + p_{21}(q^H_{t1j})^2 v_{t1} + \\
& p_{30}(q^H_{t1j})^3 + p_{31}(q^H_{t1j})v_{t1} + \\
& p_{40}(q^H_{t1j})^4 \\
\mathcal{H}^F_{t1j} = \; & p_1(q^F_{t1j}) + p_2(q^F_{t1j})^2 + p_3(q^F_{t1j})^3 \\
\mathcal{H}^F_{t2j} = \; & p_{10}(q^F_{t2j}) + p_{11}(q^F_{t2j})Q^F_{t2} + \\
& p_{20}(q^F_{t2j})^2 + p_{21}(q^F_{t2j})^2 Q^F_{t2} + \\
& p_{30}(q^F_{t2j})^3 + P_{31}(q^F_{t2j})^3 Q^F_{t2} + \\
& p_{40}(q^F_{t2j})^4
\end{aligned}$$

Figure 5.2: Electrical system




where $Q^H_{tr}$ and $Q^F_{tr}$ are the total amount of water passing through the system, e.g. $Q^F_{t2} = (\sum_{j \in J^F(2)} q^F_{t1j} + s^F_{t1})$. Their respective concave envelopes

$$\begin{aligned}
\mathcal{H}^H_{1j} &= p'_0 q^H_{t1j} - p'_1 (q^H_{t1j})^2 + p'_2 q_{t1j} v^H_{1j} - p'_3 Q^H_{t1} \\
\mathcal{H}^F_{1j} &= p'_0 q^F_{t1j} - p'_1 (q^F_{t1j})^2 \\
\mathcal{H}^F_{2j} &= p'_0 q^F_{t2j} - p'_1 (q^F_{t2j})^2 - p'_2 q^F_{t2j} Q^F_{t2}
\end{aligned}$$

where $p'_0, p'_1, p'_2$ and $p'_3$ are non-negative so that the envelopes can be seen to be concave. The other $p$ parameters are given in the appendix.

The generating cost functions $GC$ are:

$$\begin{aligned}
GC^T_r &= a_{0r} - a_{1r} g^T_{tr} + a_{2r} (g^T_{tr})^2 \\
GC^H_{rj} &= GC^F_{rj} = 0
\end{aligned}$$

so they are either convex or zero. As the generating cost functions shows the hydropower plants are assumed to have a zero marginal cost of production.

## 5.4 Scenario generation

The stochastic parameters in the model are the inflows to the hydropower plants and the power generated by the wind farms while demand is considered deterministic. However, the wind farms are assumed to always produce as much as possible, since their marginal cost of production is negligible. By subtracting the power generated by the wind farms from the deterministic demand one gets a stochastic parameter for demand for each time period and bus:

$$D_{tb} = d_{tb} - W_{tb}$$

This allows for the wind farm to be removed from the model as it is implicitly modeled through the demand parameter and taken care of in the scenario generation. A scenario then consists of a demand realization for each bus and inflows into the three hydropower plants for each time period.

Scenario generation is not the focus in this thesis for several reasons. The system is fictitious so the specific solutions are not very interesting and the focus is on the solution process rather than the solutions themselves. The inflows are assumed to be normally distributed with parameters estimated from several years of historical data. The power generated by the wind farms have been sampled directly from the data. The random variables are assumed to be independent even though this probably isn't a realistic assumption as wind and rain are correlated to some degree. They have also been assumed to be independent over time for simplicity. These assumptions allows for combining the samples randomly together and over time to generate scenarios.

## 5.5 Decomposition and properties of the problem

This section shows the decomposition of the problem and shows several properties of the decomposed problem.

### 5.5.1 Constraints and infeasibility

The problem has fixed recourse because the first stage decisions do not affect the coefficients of the $y_h$ variables in the $g_h(y_h)$ functions. Whether the problem has complete or relatively complete recourse or not depends on the scenarios used. Infeasibility can only occur in the third run-of-river plant, which is the one that is

not downstream from the reservoir. Infeasibility occurs when there is not enough water in the river so that the units that are active cannot produce more than their lower generation limits. If units are used without sufficient water they break down. As discussed earlier when the artificial power plants were introduced, infeasibility cannot occur with respect to the demand because of the infinite thermal plants that were added to the system as a way of modeling potential shortfalls of production instead of allowing infeasibility.

The model was tested incrementally and there were two sets of constraints that complicated the problem a lot. The minimum up- and downtime constraints for the thermal plants, 5.19-5.20, that connects the time periods in the operational horizon and and the introduction of the network and the related constraints, 5.26-5.27.

### 5.5.2 Optimality and feasibility cuts

As described in section 3.1.1, optimality cuts are formed from the solutions of the subproblems in each iteration. All necessary parameters in the cut, excluding the optimal objective value, come from the constraint group

$$B_h x + g_h(y_h) \leq 0$$

These constraints contain both $x$ and $y$, meaning that they are the ones binding the master and subproblems together. In this case study, these constraints are the generation bounds, which include both the master problem variables $u$ and the subproblem variables $g$:

$$\begin{aligned}
&\bar{u}^H_{trj}\underline{G}^H_{rj} \leq g^H_{trj} \leq \bar{u}^H_{trj}\overline{G}^H_{rj} && t \in T, r \in R^H, j \in J(r)\\
&\bar{u}^F_{trj}\underline{G}^F_{rj} \leq g^F_{trj} \leq \bar{u}^F_{trj}\overline{G}^F_{rj} && t \in T, r \in R^F, j \in J(r)\\
&\bar{u}^T_{tr}\underline{G}^T_{r} \leq g^T_{tr} \leq \bar{u}^T_{tr}\overline{G}^T_{r} && t \in T, r \in R^T
\end{aligned}$$

Transformed to the general form, they become:

$$\begin{aligned}
\underline{G}^H_{rj}\bar{u}^H_{trj} - g^H_{trj} &\leq 0 && [\lambda^{H1}_{trj}]\\
-\overline{G}^H_{rj}\bar{u}^H_{trj} + g^H_{trj} &\leq 0 && [\lambda^{H2}_{trj}]\\
\underline{G}^F_{rj}\bar{u}^F_{trj} - g^F_{trj} &\leq 0 && [\lambda^{F1}_{trj}]\\
-\overline{G}^F_{rj}\bar{u}^F_{trj} + g^F_{trj} &\leq 0 && [\lambda^{F2}_{trj}]\\
\underline{G}^T_{r}\bar{u}^T_{tr} - g^T_{tr} &\leq 0 && [\lambda^{T1}_{tr}]\\
-\overline{G}^T_{r}\bar{u}^T_{tr} + g^T_{tr} &\leq 0 && [\lambda^{T2}_{tr}]
\end{aligned} \tag{5.30}$$

Here, the first term in each corresponds to $B_h x$, the second to $g_h(y)$ and the right hand side, which is zero, to $h_h$. Each constraint has its corresponding dual variable $\lambda$. Consider the general form of the optimality cut as shown in section 3.1.1:

$$\theta \geq \sum_{h \in S} w_h (f_h(x^i, \bar{y}_h^i) + (\lambda_h^i)^T (g_h(y_h^i) + B_h x))$$

Given a solution for each subproblem in iteration $i$: $\bar{y}_s^i$, which includes the optimal generation values $\bar{g}$), and corresponding optimal dual variables $\lambda$ as described in (5.30). The specific optimality cut, in its single-cut form, is:

$$\begin{aligned}
\theta \geq \quad & \frac{1}{|S|} \sum_{h \in S} \Bigg[ \sum_{t \in T} \sum_{r \in R^T} GC^T(\bar{g}_{tr}^T) + \sum_{t \in T} c^{inf} \bar{g}_t^{inf}) \\
& + \sum_{t \in T} \sum_{r \in R^H} \sum_{j \in J(r)} \left( \lambda_{trj}^{H1} (\underline{G}_{rj}^H u_{trj}^H - \bar{g}_{trj}^H) + \lambda_{trj}^{H2} (-\overline{G}_{rj}^H u_{trj}^H + \bar{g}_{trj}^H) \right) \\
& + \sum_{t \in T} \sum_{r \in R^F} \sum_{j \in J(r)} \left( \lambda_{trj}^{F1} (\underline{G}_{rj}^F u_{trj}^F - \bar{g}_{trj}^F) + \lambda_{trj}^{F2} (-\overline{G}_{rj}^F u_{trj}^F + \bar{g}_{trj}^F) \right) \\
& + \sum_{t \in T} \sum_{r \in R^T} \left( \lambda_{tr}^{T1} (\underline{G}_r^T u_{tr}^T - \bar{g}_{tr}^T) + \lambda_{tr}^{T2} (-\overline{G}_r^T u_{tr}^T + \bar{g}_{tr}^T) \right) \Bigg]
\end{aligned} \tag{5.31}$$

The multi-cut would add add one for each scenario and remove the summation over scenarios:

$$\begin{aligned}
\theta_h \geq \quad & \frac{1}{|S|} \sum_{t \in T} \sum_{r \in R^T} GC^T(\bar{g}_{tr}^T) + \sum_{t \in T} c^{inf} \bar{g}_t^{inf}) \\
& + \sum_{t \in T} \sum_{r \in R^H} \sum_{j \in J(r)} \left( \lambda_{trj}^{H1} (\underline{G}_{rj}^H u_{trj}^H - \bar{g}_{trj}^H) + \lambda_{trj}^{H2} (-\overline{G}_{rj}^H u_{trj}^H + \bar{g}_{trj}^H) \right) \\
& + \sum_{t \in T} \sum_{r \in R^F} \sum_{j \in J(r)} \left( \lambda_{trj}^{F1} (\underline{G}_{rj}^F u_{trj}^F - \bar{g}_{trj}^F) + \lambda_{trj}^{F2} (-\overline{G}_{rj}^F u_{trj}^F + \bar{g}_{trj}^F) \right) \\
& + \sum_{t \in T} \sum_{r \in R^T} \left( \lambda_{tr}^{T1} (\underline{G}_r^T u_{tr}^T - \bar{g}_{tr}^T) + \lambda_{tr}^{T2} (-\overline{G}_r^T u_{tr}^T + \bar{g}_{tr}^T) \right) \quad \forall h \in S
\end{aligned} \tag{5.32}$$

Remember that each $g$ variable is part of the $y_h$ vector of second stage variables for scenario $h$. The index $h$ is not used for each variable to avoid excessive indices.

### Feasibility cuts

The general feasibility cuts shown in section 3.1.1 are also made from the constraint group:

$$B_h x + g_h(y_h) \leq 0$$

because infeasibility can only occur when there is not enough water to produce more than the lower generation limit, but the unit is turned on, for units in hydro plant

3. Written out the the constraints in question are:

$$\bar{u}^F_{trj}\underline{G}^F_{rj} = z^-_k + g^F_{trj} \qquad t \in T, r \in R^F, j \in J(r)$$

which becomes, when written in the general form shown above:

$$\underline{G}^F_{rj}\bar{u}^F_{trj} - g^F_{trj} - z^-_k = 0 \qquad [\sigma^{F1}_{trj}] \tag{5.33}$$

where the dual variables $\sigma$ are used to construct the feasibility cuts.

$$0 \geq (\sigma^j_h)^T(g_h(y^j_s) + B_h x)\forall j, \forall h \in S$$

becomes

$$\begin{aligned} 0 \geq \quad & \frac{1}{|S|} \sum_{t\in T}\sum_{r\in R^T} GC^T(\bar{g}^T_{tr}) + \sum_{t\in T} c^{inf}\bar{g}^{inf}_t) \\ & + \sum_{t\in T}\sum_{r\in R^F}\sum_{j\in J(r)} \left(\sigma^{F1}_{trj}(\underline{G}^F_{rj}u^F_{trj} - \bar{g}^F_{trj})\right) \quad \forall h \in S \end{aligned} \tag{5.34}$$

However, due to knowing why infeasibility occurs a better alternative is to use valid inequalities. As mentioned earlier, infeasibility can only occur when there is not enough inflows to hydropower plant 3, but the units are switched on. If there isn't enough inflow in some period then the following inequality will restrict the number of units allowed to be switched on:

$$\sum_{j\in J(2)} u^F_{t2j} \leq |J(2)| - 1 \text{ for } t \text{ where infeasibility occurs} \tag{5.35}$$

and if infeasibility occurs again the inequality can be strengthened:

$$\sum_{j\in J(2)} u^F_{t2j} \leq |J(2)| - 2 \text{ for } t \text{ where infeasibility occurs twice} \tag{5.36}$$

and so on. The third hydropower plant only has two units so the two inequalities above suffice. The valid inequalities are much more efficient for approximating the feasible set than the general feasibility cuts. This also means that since it is possible to extract the time period where infeasibility occurs the feasibility problems are not necessary.

## 5.6 Decomposition

By decomposing the problem as described in chapter 3 we get the following master problem:

$$
\begin{aligned}
\min \quad & \sum_{t\in T}\sum_{r\in R^T} S_r^T \widetilde{u}_{tr}^T + \theta \\
\text{subject to} \quad & \theta \geq \sum_{s\in S} \frac{1}{|S|}(u_s^i)^T(h_s - T_s x^i), \qquad i \in T^k \\
& \sum_{j\in J(2)} u_{t2j} \leq |J(2)| - 1, \qquad \text{for every } t \text{ where infeasibility occurs once} \\
& \sum_{j\in J(2)} u_{t2j} \leq |J(2)| - 2, \qquad \text{for every } t \text{ where infeasibility occurs twice} \\
& \sum_{r\in\{r:x_r^{(t)}=1, r=1,\ldots,n_x\}} x_r \\
& - \sum_{r\in\{r:x_r^{(t)}=0, r=1,\ldots,n_x\}} x_r \leq |\{r : x_r^{(t)} = 1\}| - 1 \qquad \forall t \in T^k \\
& (5.1) - (5.3), (5.9) - (5.11), (5.16) - (5.20) \\
& u_{trj}^H, \widetilde{u}_{trj}^H, u_{trj}^F, \widetilde{u}_{trj}^F, u_{tr}^T, \widetilde{u}_{tr}^T \in \{0,1\} \\
& \theta \in \mathbb{R}
\end{aligned}
\tag{RMP$^{\text{k}}$'}
$$

$i$ is the number of iterations. The subproblems, one for each scenario $h \in S$, takes the form:

$$
\begin{aligned}
\min \quad & \sum_{t\in T}\sum_{r\in R^T} GC^T(g_{tr}^T) + \sum_{t\in T} c^{inf} g_t^{inf} \\
\text{subject to} \quad & \bar{u}_{trj}^H \underline{G}_{rj}^H \leq g_{trj}^H \leq \bar{u}_{trj}^H \overline{G}_{rj}^H \qquad t \in T, r \in R^H, j \in J(r) \\
& \bar{u}_{trj}^F \underline{G}_{rj}^F \leq g_{trj}^F \leq \bar{u}_{trj}^F \overline{G}_{rj}^F \qquad t \in T, r \in R^F, j \in J(r) \\
& \bar{u}_{tr}^T \underline{G}_{r}^T \leq g_{tr}^T \leq \bar{u}_{tr}^T \overline{G}_{r}^T \qquad t \in T, r \in R^T \\
& (5.4) - (5.6), (5.8), (5.12) - (5.13), (5.15), (5.21) - (5.24), (5.26) - (5.27) \\
& g_{trj}^H, g_{trj}^F, g_{tr}^T, g_{tb}^{inf}, q_{trj}^H, q_{trj}^F, s_{tr}^H, s_{tr}^F, v_{tr}^H \geq 0
\end{aligned}
\tag{PBP$_{\text{h}}$'}
$$

Where $\bar{u}$ are the first stage decisions concerning which units to turn on at which time. Note that these are treated as parameters in the subproblem, hence the $\bar{u}$ notation. The feasibility problems becomes:

$$\min \quad ||z_k^-|| + ||z_k^+||$$

$$\begin{aligned}
\text{subject to} \quad & {}^{H}_{trj} - z_k^- = \bar{u}^H_{trj}\underline{G}^H_{rj} && t \in T, r \in R^H, j \in J(r) \\
& {}^{H}_{trj} + z_k^+ = \bar{u}^H_{trj}\overline{G}^H_{rj} && t \in T, r \in R^H, j \in J(r) \\
& {}^{F}_{trj} - z_k^- = \bar{u}^F_{trj}\underline{G}^F_{rj} && t \in T, r \in R^F, j \in J(r) \\
& {}^{F}_{trj} + z_k^+ = \bar{u}^F_{trj}\overline{G}^F_{rj} && t \in T, r \in R^F, j \in J(r) \\
& {}^{T}_{tr} - z_k^- = \bar{u}^T_{tr}\underline{G}^T_{r} && t \in T, r \in R^T \\
& {}^{T}_{tr} + z_k^+ = \bar{u}^T_{tr}\overline{G}^T_{r} && t \in T, r \in R^T \\
& (5.4) - (5.6), (5.8), (5.12) - (5.13), (5.15), (5.21) - (5.24), (5.26) - (5.27) \\
& g^H_{trj}, g^F_{trj}, g^T_{tr}, g^{inf}_{tb}, q^H_{trj}, q^F_{trj}, s^H_{tr}, s^F_{tr}, v^H_{tr}, z_k^-, z_k^+ \geq 0
\end{aligned}$$

($\text{FP}_\text{h}$')

where all the numbered constraints are assumed to be rewritten with slack variables as equality constraints.

### 5.6.1 Convexity of the relaxed problem

The objective function of the subproblems, both the convexified problems $\text{PBP}_\text{h}$' and the original problems $\text{PP}^\text{l}$, can be seen to be the sum of different $GC(\cdot)$ functions that are all convex. The sum of convex functions is also convex and so the objective is convex. All constraints in the problem are linear, except for the generation constraints. The generation constraints (5.4) and (5.12) in the convexified problem describe a convex set since the convexified $\mathcal{H}$ functions are seen to be concave and the constraints are less than or equal than a concave surface. Non-negativity are also imposed on the variables in question so the feasible set with regard to the non-linear constraints in the convexified problem is convex. Together, this means that the whole feasible set is convex. Note that equalities in the convexified generation constraints would have made the set non-convex; however, it is not necessary to enforce equality because the objective implicitly enforces this restriction since one always prefers the highest $g$ allowed for a given amount of discharge $q$.

### 5.6.2 Slater's condition

The subproblems for problem ($\text{PBP}_\text{h}$'') are easily seen to contain a Slater point. The only non-linear constraints are the generation constraints (5.4) and (5.12). There will always exist a feasible combination of $q = (q^H_{trj}, q^F_{trj}, q^T_{tr})$ and $g = (g^H_{trj}, g^F_{trj}, g^T_{trj}, g^{inf})$ that makes the generation constraint inequalities hold strictly. These points are

thus Slater points. Slater's condition is therefore satisfied and strong duality holds for problems ($PBP^k{}_h$) and ($FP^k{}_h$). Strong duality also implies that there is a set of optimal multiplier vectors associated with the optimal solution for every subproblem. Additionally each subproblem is finite since the problem is bounded in every direction.

### 5.6.3 Convergence

Given that the integer variables are part of a finite, discrete set $\{0,1\}^n$ and that any given $\bar{x}$ will never appear twice, except for when optimality has been reached, convergence follows from theorem 3.1.5.

Property (P') also holds for the convexified problem (LBP) since the objective function is separable in the variables $\widetilde{u}$ and $g$ while the constraints are also separable in $g$, meaning that no components of $\widetilde{u}$ or $g$ ever occur together in any term.

### 5.6.4 Problem size

The size of the problem, measured in total number of variables and constraints, is most strongly impacted by the number of periods and the number of scenarios. The problem considered in this case study has 24 periods. The number of scenarios ranges from 1 to 10,000, which makes it the main determinant of the overall problem size. Tables 5.1 show the number of variables and constraints for different numbers of scenarios.

Table 5.1: Problem size

| Number of scenarios | 1 | 10 | 100 | 500 | 1000 | 10,000 |
|---|---|---|---|---|---|---|
| Number of binary variables | 432 | 432 | 432 | 432 | 432 | 432 |
| Number of continuous variables | 145 | 1441 | 14,401 | 72,001 | 144,001 | 1,440,001 |
| Number of constraints | 384 | 1464 | 12,264 | 60,264 | 120,264 | 1,200,264 |

## 5.7 Implementation

This section describes several of the implementational choices that have been made in the development and the reasons behind choices. The implementation differs in some respects from the distributed algorithm presented in chapter 4. The hardware and software used to run and test the algorithm is also described.

### 5.7.1 Feasibility cuts and valid inequalities

The general feasibility cuts was replaced with the valid inequalities presented earlier in this chapter. The main reason for this is the fact that the general feasibility cuts are too slow in approximating the feasible set, while the valid inequalities are very efficient. There are a total of 48 possible valid inequalities and if a cut is added the source of infeasibility is eliminated for future iterations. The general feasibility cuts however only cuts away the integer configuration that led to the infeasibility, but not necessarily the source of the infeasibility. This means that the next integer configuration the master problem wants to try can be an integer configuration that is equal except for a few variables. This integer realization will likely lead to another infeasibility stemming from the same source as it is found in the same attractive region of the solution space that the algorithm wants to search. During development we observed long runs of infeasible iterations where the described scenario took place. These iterations provides no progress, but takes just as much time as a normal iteration. The general feasibility cuts weren't fast enough and some other approach like the valid inequalities as specific infeasibility cuts was necessary.

### 5.7.2 Cut strategy in the implementation

The theory on cut strategies presented in chapter 2 and chapter 4 tells us that a single cut or something close to that end of the spectrum would be best for a parallel algorithm because the master problem will be much smaller at every stage compared to the problem in with the multi-cut approach. This is good for parallelization because the non-parallelizable part, which is the master problem, is smaller and the potential for better performance through parallelization is increased. However, through continuous testing in the implementation phase, the effect on the runtime of a smaller master problem turned out to be less important than the effect of needing more iterations to converge. In fact, with a single-cut approach the convexified problem did not converge within 24 hours even for small instances with less than ten scenarios. More cuts consistently performed better in terms of total runtime so the extreme multi-cut approach was adopted in the implementation. There might be several reasons for this, but one reason is that GUROBI, which handles the master problem, is a very good solver for problems like the master problem. The main downside of the multi-cut strategy is that it reduces the potential for parallelization and distribution decreases since the bulk of the time in each iteration is spent in the master problem and the master problem is the non-parallelizable part of the algorithm.

For a reduced version of the case study the single cut strategy performed well and big reductions in runtime were possible. We will come back to this issue in the next chapter.

### 5.7.3 Bundle methods

The bundle method addition turned out to be a necessity. For the problem in its current form the algorithm was not able to solve even small instances with no more than 10 scenarios within 24 hours. The tail-off effect was a major problem: The solution gap would practically stop converging at a certain point in time. Since bundle methods are in part used specifically to counter the tail-off effect, it was only natural to include one. The doubly stabilized method turned out to be the most efficient of the ones we tried; it significantly sped up the solution time and decreased the tail-off effect.

### 5.7.4 Software and hardware

The algorithm has been tested on a computing cluster consisting of 2688 nodes of four different types described in the following table. A computing cluster is a system of several computers that are connected through some network, but that have been designed and set up so to act more or less as a logically single, powerful computer.

Table 5.2: Computational nodes information

| | HP dl140 | HP dl160 G5 | HP dl160 G3 | HP bl165 |
|---|---|---|---|---|
| CPU | 2x1.6GHz Intel E5110 Xeon – 2 core | 2x3.0GHz Intel E5472 Xeon – 4 core | 2x2.4GHz AMD Opteron 2431 – 6 core | 4x2.2GHz AMD Opteron 6274 – 16 core |
| Memory | 8GB RAM | 16GB RAM | 24GB RAM | 128GB RAM |
| Disk | 72Gb 7200rpm SATA | 72Gb SAS 15k rpm | 150Gb SAS 15k rpm | 300Gb SAS 15k rpm |

All nodes are running CentOS 6.4 and are managed with RocksClusters 6.1 SP1. The solvers are IPOPT v3.12.4 and Bonmin v1.8.4, both described above, for the subproblems and the master problem respectively. MPI is implemented with open-MPI v. 1.10. The problem has been implemented in C++, using interfaces provided by COIN-OR and GUROBI, libraries from the solvers and OpenMPI. The program is compiled using the GNU g++ compiler inside an OpenMPI wrapper compiler. Scenarios have been generated with R. The code can be found in the appendix.

# Chapter 6

# Results and discussion

The results are divided into three parts. The algorithm has been tested on a reduced version of the convexified problem (to test the scalability of the implementation), the full convex problem and the full non-convex problem. The results for each problem is presented separately and discussed before we summarize our findings in the last section.

## 6.1 Results

The two first sets of results are concerned with the convexified problem. The only differences between the non-convex and the convex version of the problem are in the hydropower production function. In the convex problem the concave envelopes are used as the hydropower production functions, $\mathcal{H}$. In this case the NGBD method reduces to the GBD method. The third set of results are obtained from the non-convex problem which uses the non-convex hydropower production functions shown in chapter 5.

### 6.1.1 Reduced convexified problem

The smaller problem does not take the network into account and can be thought of as a single bus network where all the power plants and demand is at the same bus. The minimum and maximum uptime constraints for the thermal power plants are not considered either. The cut strategy used is the extreme single-cut strategy meaning that only one aggregated cut is added in each iteration. The number of time periods is reduced to 12 hours which significantly reduces the problem size. The scenarios are all feasible so that the problem has complete recourse.

The details of the smaller problem are given in table 6.1.

Table 6.1: Problem size, 12 periods

| Number of scenarios | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| Binary variables | 216 | 216 | 215 | 216 | 216 | 216 |
| Continuous variables | 289 | 2,881 | 28,801 | 288,001 | 2,880,001 | 28,800,001 |
| Constraints | 593 | 4,895 | 47,915 | 478,115 | 4,780,115 | 47,800,115 |

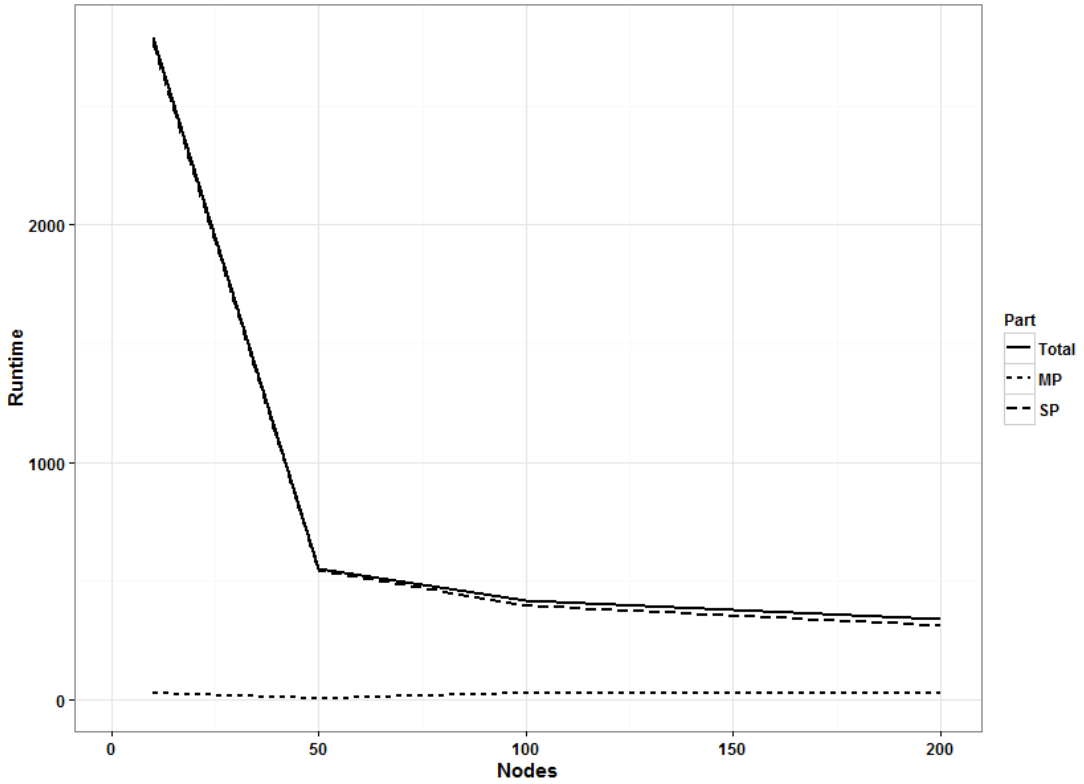



Figure 6.1: Runtimes for 10,000 scenarios in reduced problem

Figures 6.1 and 6.2 show that the runtime can be decreased by a factor of approximately 80 when the numbers of nodes are increased from 1 to 200 in the problem with 10,000 scenarios, and by a factor of approximately 13.5 when the number of nodes are increased from 10 to 200 in the problem with 100,000 scenarios. The parallelization clearly gives significant reductions in runtime for the reduced problem.

Figure 6.3 show how the runtime develops when the number of scenarios increase and confirms the expected linear relationship between runtime and number of scenarios for different numbers of computational nodes used.

Tables for more detailed results are found in tables C1 to C5.

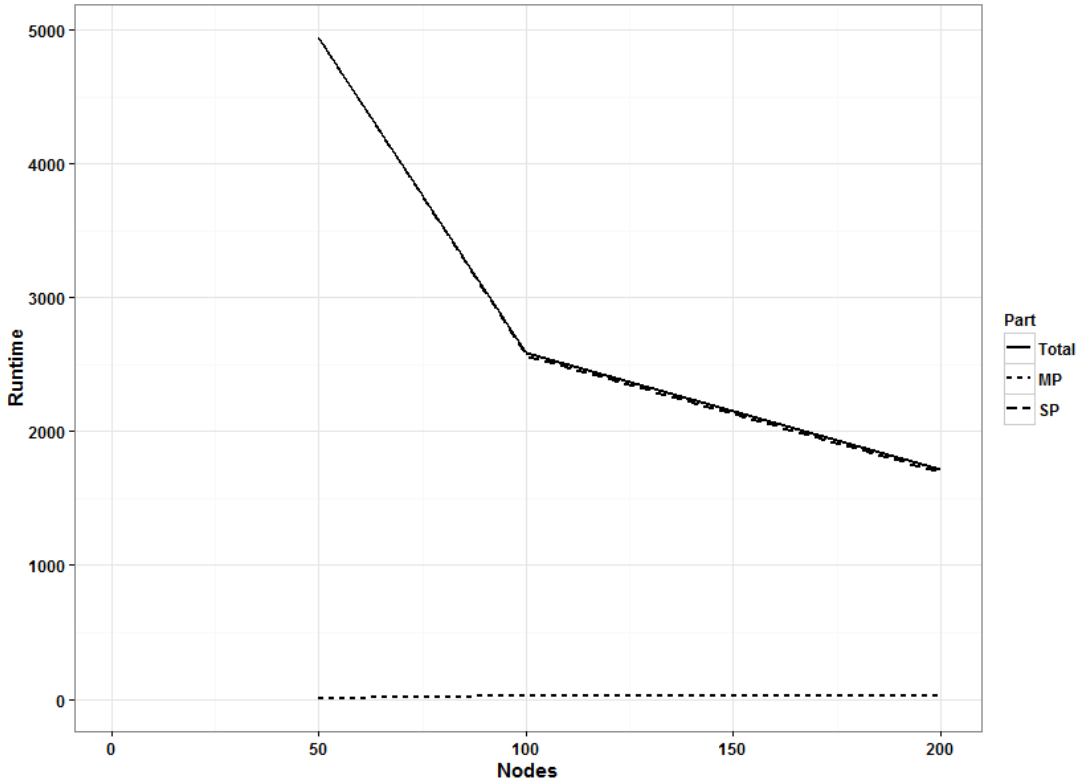



Figure 6.2: Runtimes for 100,000 scenarios in reduced problem

### 6.1.2 Convexified problem

The results from the convex problem show that there is an effect from parallelizing, but that it flattens out when the number of scenarios per node decrease beyond some treshold. Figure 6.4 shows that the time spent in the subproblem phase almost halves when the number of nodes go from 1 to 10. However, when the number of nodes is increased further the time spent in the subproblem phase does not change much, although it does decrease slightly. The time spent in the master problem is more or less constant, which is to be expected as the same cuts are added and the problem is the same in every iteration, no matter how many nodes are used.

The same kinds of results occur when the problem is solved with 1000 scenarios as shown in figure 6.5. It is interesting to note that the problem is solved faster with more scenarios. Closer inspection of the integer realizations visited by the algorithm shows that the additional infeasibilities actually make the problem easier because if there are enough infeasible time periods the third hydropower plant, which is the only source of infeasibility, is practically removed from the problem. This makes the remaining problem easier because the size of the feasible set is decreased and the remaining problem cannot be infeasible.

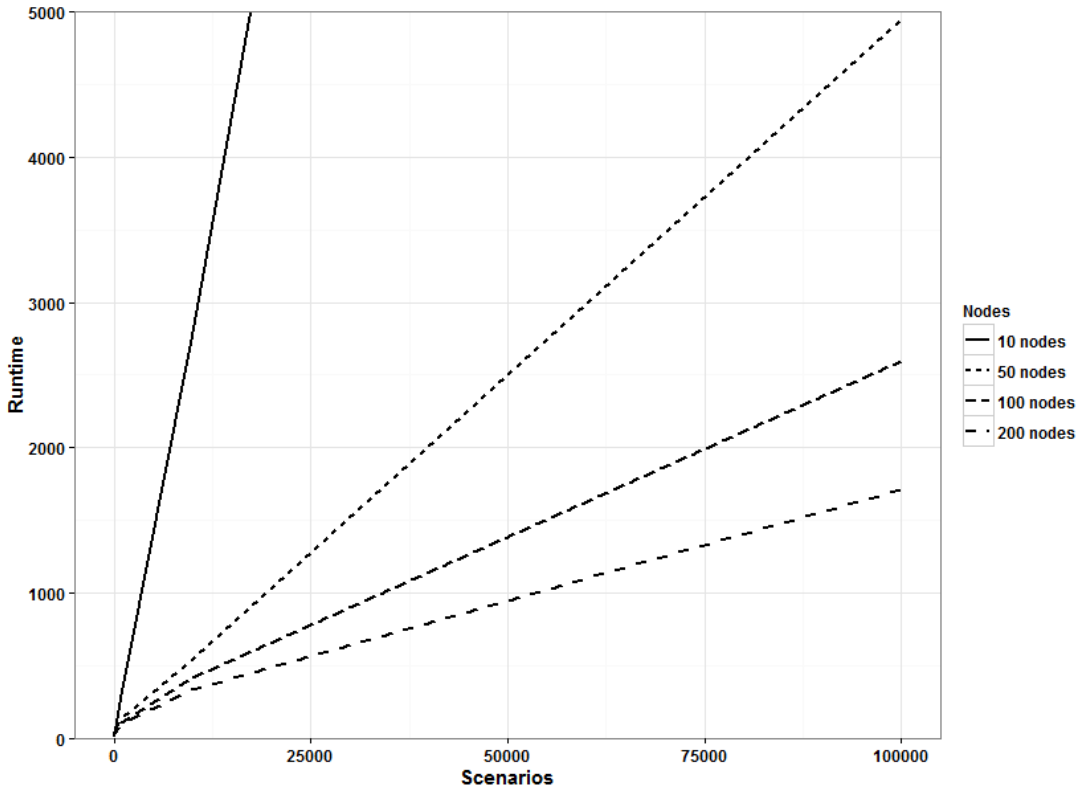


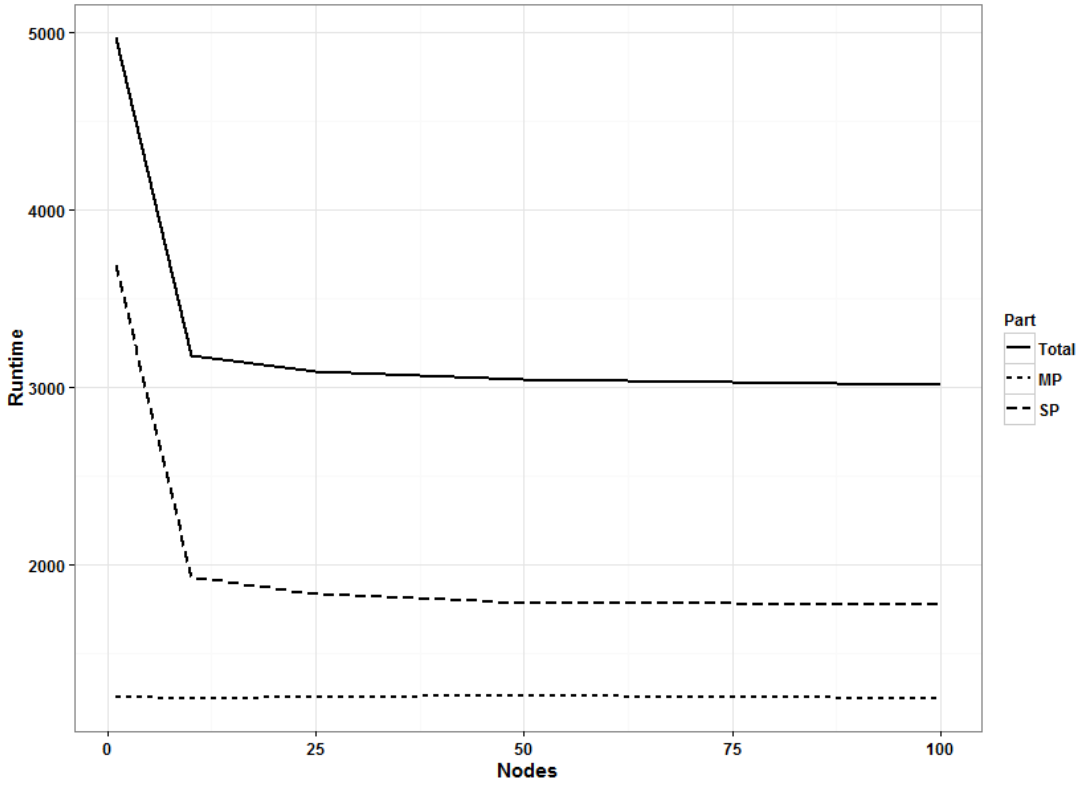

Figure 6.3: Runtimes for increasing number of scenarios with the same number of nodes

The difference can also be explained by looking at the difference in the number of iterations required to converge. While the problem with 1000 scenarios require only 14 iterations to converge, the one with 100 scenarios require 59 iterations. The single scenario problem needed 609 iterations to converge, but in this case the iterations are fast.

This is a weakness of the case study and the scenario generation assumptions that were made about the independence between time periods and the random variables and their respective distributions. A more realistic distribution for the random variable concerned with the inflows into the third hydropower plant would be much more skewed, with fewer occurrences of very high or very low inflow, and most of the time there would be enough to stay within the limits. For a more realistic problem, scenario generation should be reconsidered.

When the number of scenarios are more than 1000 the solution time is longer than 10,000 seconds and the procedure stopped. Only two long runs have been performed. One of them with 10,000 scenarios on one node, which took approximately 70,000 seconds with 25,000 of them spent in the master phase and the remaining 45,000

Figure 6.4: Runtimes for 100 scenarios



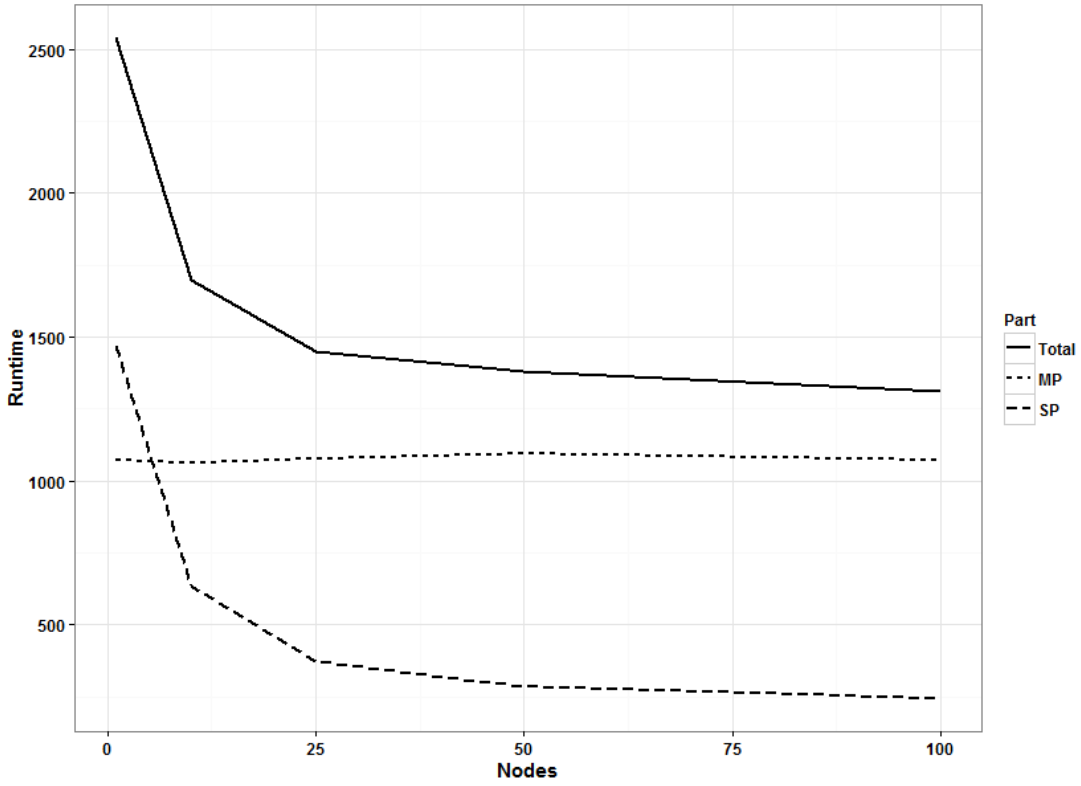

seconds in the subproblem phase. The second long run was done with 100 nodes for the same 10,000 scenarios. It took a total of 45,000 seconds where approximately the same amount of time was spent in the master problem phase and the rest in the subproblem phase. This means that the parallelization decreased the runtime by 25,000 seconds, more than halving the time spent in the subproblem phase. Additional nodes would most likely give further decreases in the runtime as the number of subproblems per node is still high. The flattening out effect is seen at 10 scenarios per node and 40 scenarios per node for the problems with 100 and 1000 scenarios respectively.

Tables for more detailed results are found in tables C6 and C7.

### 6.1.3 Non-convex problem

Due to using a local solver for the non-convex subproblems, the NGBD implementation did not converge within 10,000 seconds for any number of scenarios. The optimality gaps at termination for different numbers of scenarios are shown in table 6.2. As can be seen in the table, most of the gaps hover around 25%, which obvi-

Figure 6.5: Runtimes for 1000 scenarios




ously is not very good. However, it is important to note that this is the worst-case distance from the optimal solution; the actual gap is somewhere between 0% and 25%, there is just no way of knowing without a global optimizer which can provide a real lower bound. A part of the gap is also attributable to the difference between the non-convex functions and their convex relaxations, which mostly lies between 0 and 10%.

Regarding parallelization, there is definitely potential. A majority of the time spent was in the convex and non-convex subproblem phase, i.e. the phases that are parallelizable. The addition of the non-convex subproblem phase also means that this algorithm theoretically is even better suited than the GBD algorithm, a notion that is supported by our initial runs. However, without any finished runs it is impossible to provide any significant quantification of the actual speed-up.

## 6.2 Further discussion

The difference in the improvement in the runtime for the smaller convex problem and the full convex problem is a result of the difference between the difficulty of the

Table 6.2: Optimality gap in non-convex algorithm after 10,000 seconds

| **Scenarios** | **Optimality gap** |
|---|---|
| 1 | 11.14% |
| 10 | 29.59% |
| 100 | 14.94% |
| 250 | 25.72% |
| 500 | 25.85% |
| 1000 | 26.21% |

master problems. In the smaller problem there are only 216 binary variables and the most difficult constraints are not considered while the full problem has twice as many binary variables due to the time horizon of 24 hours. The full problem also has more constraints such as the minimum and maximum uptime constraints which complicates the problem by connecting the time periods.

The effect of parallelization clearly flattens out when the number of scenarios per node decrease below some treshold. This can be seen from the results for the reduced and full convex problem. This treshold is larger than one subproblem per node. The reason that this happens while the number of scenarios per node is greater than one is that some scenarios are much harder to solve and it is not possible to reduce the time spent in the subproblem phase to less than the amount of time it takes to solve the hardest subproblem. This explains the fact that the parallelization effect flattens while the number of scenarios per node is still greater than one.

Using additional nodes for a problem when the parallelization has started flattening out is not helpful because of the fact that some subproblems are much harder to solve and when the number of subproblems per node is small enough the hardest subproblem becomes the determining factor of the time spent in the subproblem phase. Since parallelization does not reduce the time required for any one subproblem, the effect of additional nodes when the parallelization effect flattens out is limited.

Regarding the non-convex problem, the results show that a global solver, or at least a better heuristic strategy, is required to solve it in a satisfactory way. Even though the solutions might be good, there is no consistent way to verify their quality beyond the optimality gap given by the algorithm. It is important to note that the non-convex algorithm always will use more time than the convex algorithm (given equal parameters and scenarios), since the whole GBD procedure is contained within the NGBD procedure. Thus, the results from the GBD algorithm can be taken as

best-case results for the NGBD algorithm.

# Chapter 7

# Concluding remarks

We have developed a distributed implementation of a parallel version of the NGBD method presented in chapter 3 with bundle methods incorporated into the algorithm. The main goal of the thesis was to explore the potential for parallelization of the method while the second goal was solve a stochastic unit commitment problem which was formulated and presented in the case study in chapter 5. We are able to solve the problem for quite large instances with up to 1000 scenarios in less than 1500 seconds for the convex problem within 2.5% of the true optimal value. The non-convex problem has not been solved to desired optimality, mainly because of the lack of an appropriate global solver.

The results from the case study shows that parallelization improves runtime significantly, up to a factor of approximately 2 for the full convex problem with 1000 scenarios and with a factor of up to approximately 80 for the reduced problem with up to 100,000 scenarios if enough nodes are added. The initial results from the non-convex algorithm also show promise with regards to parallelization, but the actual speed-up remains to be measured. We do recognize that although the results are promising, there are also some limitations to our approach. The rest of the thesis is dedicated to to discussing these limitations and to propose our ideas for further research.

While the results are promising and show that a massively parallel approach has the potential to solve problems with large numbers of scenarios, as can be seen from the results from the reduced version of the convex problem, they also show that it is essential that the master problem is relatively easy to solve. If this is not the case parallelization has limited potential because most of the time is spent solving the master problem which is not parallelizable. This suggests that a direction for further research is to find better cut selection strategies and cut management strategies. The

reduced convex problem is solvable with a single-cut approach and benefits greatly from parallelization, while for the full problem the convergence was too slow with single-cuts. The extreme multi-cut approach adopted in the implementation for the full problem makes the master problem very large as in each iteration one cut for each scenario is added. A cut management strategy that is able to eliminate cuts as they become redundant could potentially reduce the size of the master problem significantly. This would make it possible to solve problems with more scenarios while at the same time increasing the portion of the problem that is parallelizable, so that the problem would benefit more from parallelization. Fischetti et al. (2010) presents an alternative cut selection strategy for the standard Benders decomposition method.

In general it is a good idea to try and utilize the available computing resources as much as possible. In the non-convex and extended convex problem a significant amount of time is spent in the master problem which is contained in one node while the rest of the nodes are idle. There are several possible uses for these nodes. They can be used to solve subproblems that potentially might be useful. An example of this can be subproblems obtained by trying to predict integer realizations for problems (PBP) while the master problem is being solved. Another idea is to have several master nodes that start trying to find new lower bounds and integer realizations before the subproblem phase is finished by using information obtained from the subproblems that are already solved. It can be thought of as doing partial iterations. Li (2013) mentions a similar idea that he calls bounding parallelization.

Finally, the clearest way forward is to generalize the algorithm to multi-stage problems. Kaut et al. (2014) presents what they call a multi-horizon strategy where the strategic decisions and the operational decisions are assumed to be independent. This assumption makes it possible to drastically reduce the number of scenarios and also for parallelization, without sacrificing much if the assumption is reasonable. However, this is often not the case because the operational and strategic stage are very often intimately connected. In the unit commitment problem it is clear that there are several variables that are connected over time, such as reservoir levels and which units are on when one operational horizon ends. Some approach for the transitioning between periods are thus necessary. Another approach to multi-stage problems is a nested NGBD method where a multi-stage problem is solved as a sequence of two-stage problems (Birge et al., 1996).

# Bibliography


G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

M. Bagajewicz and V. Manousiouthakis. On the generalized benders decomposition. *Computers & chemical engineering*, 15(10):691–700, 1991.

P. Belotti. *Couenne: a user's manual*, 2009. URL `https://projects.coin-or.org/Couenne/browser/trunk/Couenne/doc/couenne-user-manual.pdf?format=raw`.

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

J. R. Birge, C. J. Donohue, D. F. Holmes, and O. G. Svintsitski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. *Mathematical Programming*, 75(2):327–352, 1996.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

A. Castillo, C. Laird, C. A. Silva-Monroy, J.-P. Watson, and R. P. O'Neill. The unit commitment problem with ac optimal power flow constraints.

W. de Oliveira and M. Solodov. A doubly stabilized bundle method for nonsmooth convex optimization. *Mathematical Programming*, 156(1):125–159, 2016. ISSN 1436-4646. doi: 10.1007/s10107-015-0873-6. URL `http://dx.doi.org/10.1007/s10107-015-0873-6`.

E. C. Finardi and E. L. Da Silva. Solving the hydro unit commitment problem via dual decomposition and sequential quadratic programming. *Power Systems, IEEE Transactions on*, 21(2):835–844, 2006.

M. Fischetti, D. Salvagnin, and A. Zanette. A note on the selection of benders' cuts. *Mathematical Programming*, 124(1-2):175–182, 2010.

S. Frank, I. Steponavice, and S. Rebennack. Optimal power flow: a bibliographic survey i. *Energy Systems*, 3(3):221–258, 2012.

A. Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1962.

A. Geoffrion. Duality in nonlinear programming: a simplified applications-oriented development. *SIAM review*, 13(1):1–37, 1971.

A. M. Geoffrion. Elements of large-scale mathematical programming part i: Concepts. *Management Science*, 16(11):652–675, 1970.

I. Gurobi Optimization. Gurobi optimizer reference manual, 2015. URL `http://www.gurobi.com`.

J. L. Gustafson. Reevaluating amdahl's law. *Communications of the ACM*, 31(5): 532–533, 1988.

M. Kaut, K. T. Midthun, A. Tomasgard, A. S. Werner, L. Hellemo, and M. Fodstad. Multi-horizon stochastic programming. *Computational management science*, 11 (1):179–193, 2014.

X. Li. Parallel nonconvex generalized benders decomposition for natural gas production network planning under uncertainty. *Computers & Chemical Engineering*, 55:97–108, 2013.

X. Li, A. Tomasgard, and P. I. Barton. Decomposition strategy for the stochastic pooling problem. *Journal of global optimization*, 54(4):765–790, 2012a.

X. Li, A. Tomasgard, and P. I. Barton. Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. *Journal of optimization theory*, 151(3):424–454, 2012b.

R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, 10 (1):147–175, 1976.

A. Pagès-Bernaus, G. Pérez-Valdés, and A. Tomasgard. A parallelized distributed implementation of a branch and fix coordination algorithm. *European Journal of Operations Research*, 244(1):77–85, 2015.

C. Sagastizábal. Divide to conquer: decomposition methods for energy optimization. *Mathematical programming*, 134(1):187–222, 2012.

N. V. Sahinidis. *BARON 14.3.1: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2014.

B. Saravanan, S. Das, S. Sikri, and D. Kothari. A solution to the unit commitment problem—a review. *Frontiers in energy*, 7(2):223–236, 2013.

M. Slater. *Lagrange multipliers revisited*. Springer, 2014.

M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra. Large-scale unit commitment under uncertainty. *4OR*, 13(2):115–171, 2015.

F. Y. Takigawa, E. L. da Silva, E. C. Finardi, and R. N. Rodrigues. Solving the hydrothermal scheduling problem considering network constraints. *Electric Power Systems Research*, 88:89–97, 2012.

R. M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17 (4):638–663, 1969.

A. Wächter. Short tutorial: getting started with ipopt in 90 minutes. *Combinatorial Scientific Computing (U. Naumann, O. Schenk, HD Simon, eds.)*, 34(56):118, 2009.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

Wikimedia. Evolution according to amdahl's law of the theoretical speedup in latency of the execution of a program in function of the number of processors executing it for programs with different potential for parallelization, 2008. URL `https://en.wikipedia.org/wiki/Amdahl%27s_law#/media/File:AmdahlsLaw.svg`.

S. Zaourar and J. Malick. Quadratic stabilization of Benders decomposition. working paper or preprint, 2014. URL `https://hal.archives-ouvertes.fr/hal-01181273`.

# Appendix A

# Source code

The source code can be found in the following dropbox:

https://www.dropbox.com/sh/yvglzo1zayzfx23/AADczVFOPzysLzwuOXEKDP3ka?dl=0

# Appendix B

# Parameters and options

The parameters and options used for the solvers can be found in the parameter files included in the source code. The files are called *ipopt.opt* and *gurobi.prm*. The parameters for the GBD and NGBD algorithms are found in the *parameters* folder.

# Appendix C

# Tables

## C.1 Detailed results for the reduced convex problem

Table C1: Results for the reduced convex problem with one computational node

| Number of scenarios | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| Number of binary variables | 216 | 216 | 216 | 216 | 216 | 216 |
| Number of continuous variables | 289 | 2,881 | 28,801 | 288,001 | 2,880,001 | 28,800,001 |
| Total time | 7.3 | 32.1 | 204.6 | 2671.9 | 27145.44 | - |
| Time for $SP_s$ | 0.3 | 22.0 | 202.3 | 2659.6 | 27121.2 | - |
| Time for MP | 7.0 | 10.1 | 2.3 | 12.3 | 24.24 | - |
| UBD at termination | 7.32e6 | 7.57e5 | 9.26e5 | 1.16e6 | 1.08e06 | - |
| LBD at termination | 7.30e6 | 7.40e5 | 9.05e5 | 1.13e6 | 1.05e06 | - |

Table C2: Results for the reduced convex problem with 10 computational nodes

| Number of scenarios | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| Number of binary variables | 216 | 216 | 216 | 216 | 216 | 216 |
| Number of continuous variables | 289 | 2,881 | 28,801 | 288,001 | 2,880,001 | 28,800,001 |
| Total time | 7.3 | 20.6 | 32.3 | 298.7 | 2791.8 | 23021.6 |
| Time for $SP_s$ | 0.3 | 7.1 | 20.6 | 286.2 | 2767.5 | 22.6 |
| Time for MP | 7.0 | 13.4 | 11.7 | 12.5 | 24.3 | 22999.0 |
| UBD at termination | 7.32e6 | 3.94e6 | 1.78e6 | 1.16e6 | 1.08e6 | 1.02e6 |
| LBD at termination | 7.30e6 | 3.91e6 | 1.74e6 | 1.13e6 | 1.05e6 | 9.95e5 |

Table C3: Results for the reduced convex problem with 50 computational nodes

| Number of scenarios | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| Number of binary variables | 216 | 216 | 216 | 216 | 216 | 216 |
| Number of continuous variables | 289 | 2,881 | 28,801 | 288,001 | 2,880,001 | 28,800,001 |
| Total time | 7.3 | 20.6 | 47.8 | 134.0 | 548.4 | 4963.7 |
| Time for $SP_s$ | 0.3 | 7.1 | 25.5 | 122.4 | 546.1 | 4941.1 |
| Time for MP | 7.0 | 13.4 | 22.3 | 11.6 | 2.3 | 22.6 |
| UBD at termination | 7.32e6 | 3.94e6 | 9.26e5 | 9.79e5 | 9.89e5 | 1.02e6 |
| LBD at termination | 7.30e6 | 3.91e6 | 9.04e5 | 9.59e5 | 9.69e5 | 9.95e5 |

Table C4: Results for the reduced convex problem with 100 computational nodes

| Number of scenarios | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| Number of binary variables | 216 | 216 | 216 | 216 | 216 | 216 |
| Number of continuous variables | 289 | 2,881 | 28,801 | 288,001 | 2,880,001 | 28,800,001 |
| Total time | 7.3 | 20.6 | 29.2 | 106.9 | 418.4 | 2591.7 |
| Time for $SP_s$ | 0.3 | 7.1 | 17.4 | 94.4 | 393.9 | 2569.1 |
| Time for MP | 7.0 | 13.4 | 11.8 | 12.4 | 24.4 | 22.6 |
| UBD at termination | 7.32e6 | 3.94e6 | 1.78e6 | 1.16e6 | 1.08e6 | 1.02e6 |
| LBD at termination | 7.30e6 | 3.91e6 | 1.75e6 | 1.13e6 | 1.05e6 | 9.96e5 |

Table C5: Results for the reduced convex problem with 200 computational nodes

| Number of scenarios | 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|---|
| Number of binary variables | 216 | 216 | 216 | 216 | 216 | 216 |
| Number of continuous variables | 289 | 2,881 | 28,801 | 288,001 | 2,880,001 | 28,800,001 |
| Total time | 7.3 | 20.6 | 29.2 | 99.1 | 336.7 | 1714.7 |
| Time for $SP_s$ | 0.3 | 7.1 | 17.4 | 87.2 | 312.3 | 1691.8 |
| Time for MP | 7.0 | 13.4 | 11.8 | 11.9 | 24.3 | 22.8 |
| UBD at termination | 7.32e6 | 3.94e6 | 1.78e6 | 1.15e6 | 1.08e6 | 1.02e6 |
| LBD at termination | 7.30e6 | 3.91e6 | 1.75e6 | 1.12e6 | 1.05e6 | 9.96e5 |

## C.2 Detailed results for the full convex problem

Table C6: Results from the full convex problem with 100 scenarios

| Number of nodes | 1 | 10 | 25 | 50 | 100 |
|---|---|---|---|---|---|
| Time spent in MP | 1251.23 | 1247.39 | 1252.71 | 1255.50 | 1242.23 |
| Time spent in SPs | 3686.13 | 1929.13 | 1835.40 | 12785.36 | 1774.13 |
| Total time | 4937.36 | 3176.52 | 3088.11 | 3040.86 | 3016.36 |

Table C7: Results from the full convex problem with 1000 scenarios

| Number of nodes | 1 | 10 | 25 | 50 | 100 |
|---|---|---|---|---|---|
| Time spent in MP | 1071.23 | 1064.39 | 1078.57 | 1095.53 | 1070.51 |
| Time spent in SPs | 1470.37 | 635.74 | 372.67 | 287.00 | 332.12 |
| Total time | 2541.97 | 1700.13 | 1451.24 | 1382.53 | 1313.79 |