Polyhedral Outer Approximations in Convex Mixed-Integer Nonlinear Programming

Jan Kronqvist

PhD Thesis in Process Design and Systems Engineering
Faculty of Science and Engineering
Åbo Akademi University

Åbo, Finland 2018

Polyhedral Outer Approximations in Convex Mixed-Integer Nonlinear Programming

Jan Kronqvist

PhD Thesis in Process Design and Systems Engineering
Faculty of Science and Engineering
Åbo Akademi University

Åbo, Finland 2018

Dissertations published by Process Design and Systems Engineering
ISSN 2489-7272
978-952-12-3734-8
978-952-12-3735-5 (pdf)

Painosalama Oy
Åbo 2018

Preface

My time as a PhD student began back in 2014 when I was given a position in the Optimization and Systems Engineering (OSE) group at Åbo Akademi University. It has been a great time, and many people have contributed to making these years a wonderful experience. During these years, I was given the opportunity to teach several courses in both environmental engineering and process systems engineering. The opportunity to teach these courses has been a great experience for me.

I want to express my greatest gratitude to my supervisors Prof. Tapio Westerlund and Docent Andreas Lundell. Tapio has been a true source of inspiration and a good friend during these years. Thanks to Tapio, I have been able to be a part of an international research environment. Andreas has played an essential role in this thesis; he has not only been a great friend, but also an excellent mentor and research colleague. Without Andreas, the SHOT solver would not have seen the light of day, at least not in such a sophisticated form. Furthermore, I want to thank Andreas for always taking the time to listen to my ideas.

I had the pleasure to spend a half year as a visitor of Prof. Ignacio Grossmann at Carnegie Mellon University (CMU) in Pittsburgh USA during 2017. I am genuinely grateful that Ignacio hosted me at the department, and I want to thank him for his mentoring during this time. It was a real privilege to attend both courses and seminars at CMU. The stay at CMU resulted in several fruitful research collaborations, which I hope to continue in the future. I had a wonderful time at CMU; I want to thank all my friends and colleagues at CMU for making the time in Pittsburgh such a great experience, especially David Bernal. I hope we can continue with new interesting projects in the future.

I also want to thank all my friends and colleagues at the Process Design and Systems Engineering Laboratory and Process Control Laboratory for creating a nice work environment. In particular, Anders Brink, Anders Skjäl, and Annika Fougstedt deserve thank-yous for helping with different aspects of writing the thesis. I also want to thank Mikael Manngård for many interesting scientific discussions and for always being keen on taking tough courses at different locations in the world. I also want to thank Juan Javaloyes and Jarl Ahlbeck for interesting collaborations during these years.

I am grateful for having had a reliable source of funding during my entire time as a PhD student, first by the OSE group and later by the Graduate School in Chemical Engineering (GSCE). Furthermore, I want to express my gratitude to: Svenska Tekniska

Vetenskapsakademien i Finland (STViF), Tekniska Föreningen i Finland (TFiF), Tekniikan Edistämissäätiö (TES), Tutkijat Maailmalle (Walter Ahlström foundation), and Waldemar von Frenckells stiftelse. Thanks to the grants given by these organizations, I was able to visit CMU for six months and to annually attend international conferences relevant for my research.

Last but definitely not least, I want to express my gratitude to my family for their never-ending support. Without your support, I would not be in a situation about to complete the highest form of education in Finland. My lovely wife Jessica deserves a special homage for always being supportive and understanding, and I apologize for almost constantly being late during the last years.

Åbo, Midsummer Eve 2018

Jan Kronqvist

Svensk sammanfattning

Matematisk optimering är ett tvärvetenskapligt forskningsområde med starka rötter i matematik, ekonomi och teknik. Målet med matematisk optimering är kort sagt att finna den bästa möjliga lösningen till ett givet problem. Optimeringsproblemen kan t.ex. handla om att finna det bästa möjliga produktionsplanen, hitta de effektivaste produktionsförhållandena för en process eller hitta den kortaste vägen mellan två städer. Genom att optimera produktionsförhållanden och använda den effektivaste produktionsplanen är det ofta möjligt att både minska på råvaruanvändningen och skadliga utsläpp, samtidigt som lönsamheten ökar. Matematisk optimering kan därmed bidra med ett viktigt verktyg för att stärka konkurrenskraften hos våra industrier.

För att effektivt kunna analysera optimeringsproblem skrivs problemen i en matematisk form. I den matematiska formen består ett optimeringsproblem av beslutsvariabler, en objektfunktion och bivillkor. Beslutsvariablerna kan t.ex. beskriva hur mycket av olika produkter som ska produceras, ja/nej beslut eller olika designalternativ. Objektfunktionen beskriver sambandet mellan beslutsvariablerna och det kriterium man antingen vill minimera eller maximera. Bivillkoren används i sin tur för att beskriva samband mellan beslutsvariablerna och kan ange gränser för vissa egenskaper, t.ex. produktkvalitet.

Optimeringsproblem klassificeras ofta enligt vilken typ av beslutsvariabler och funktioner som ingår i problemets matematiska form. Kontinuerliga variabler används för att beskriva beslutsvariabler som inte är begränsade till specifika värden, medan heltalsvariabler eller binära variabler används för att beskriva specifika beslut. Optimeringsproblem som innehåller både heltalsvariabler och kontinuerliga variabler kallas ibland blandade heltalsoptimeringsproblem, men den engelska termen *mixed-integer programming* är mycket vanligare. Ifall alla funktioner dessutom är linjära så hör problemet till kategorin *mixed-integer linear programming* (MILP). För att noggrant beskriva vissa storheter och fenomen krävs också ickelinjära funktioner, och sådana problem hör till kategorin *mixed-integer nonlinear programming* (MINLP). MINLP är ett flexibelt ramverk för att beskriva optimeringsproblem och många praktiska optimeringsproblem kan skrivas som MINLP-problem. Däremot tenderar MINLP-problem att vara mycket svåra att lösa. Även med dagens datorkraft och avancerade algoritmer kan det vara en stor utmaning att lösa ett MINLP-problem.

Denna avhandling fokuserar på en viss typ av MINLP-problem som kallas konvexa, vilket innebär att de ickelinjära funktionerna har vissa egenskaper. Syftet har varit att

utveckla nya effektiva metoder för att lösa optimeringsproblem av denna typ. Dessa optimeringsproblem har vissa egenskaper som gör det möjligt att effektivt dela upp och approximera problemen med enklare problem. Genom att dela upp problemet och lösa en serie av enklare problem, är det möjligt att finna den optimala lösningen av det ursprungliga problemet. Hur man effektivt ska dela upp problemen och skapa de enklare optimeringsproblemen är centrala frågor genom avhandlingen.

De flesta effektiva metoder för att lösa konvexa MINLP-problem bygger på att man skapar linjära approximationer av det ursprungliga problemet. De linjära approximationerna bör i slutändan vara tillräckligt noggranna för att ge den optimala lösningen till det ursprungliga problemet. Dessutom bör de linjära approximationerna resultera i tillräckligt enkla problem att de effektivt kan lösas. I avhandlingen visas tydligt vikten av en effektiv teknik för att skapa de linjära approximationerna, så att det verkliga problemet kan lösas på ett effektivt sätt.

I avhandlingen presenteras olika metoder för att skapa linjära approximationer, men också olika strategier för att dela upp det ursprungliga problemet i lättare problem. Från teorin som presenteras i avhandlingen har även en ny effektiv lösare för konvexa MINLP-problem utvecklats. Med lösare avses här en programvara för hitta den optimala lösningen. Avhandlingen bygger på de sex artiklar som är bifogade i slutet av avhandlingen. Kapitel 2 ger en kort beskrivning av olika typers optimeringsproblem, vilka också har en central roll för att lösa konvexa MINLP-problem. Kapitel 3 kan ses som en litteraturstudie och sammanfattar det arbete som tidigare gjorts inom området. De resultat som presenterats i artiklarna sammanfattas i kapitel 4. Slutligen ges en sammanfattning och tankar kring fortsatt forskning inom området i kapitel 5.

Abstract

This thesis is focused on a specific type of optimization problems commonly referred to as convex MINLP problems. The goal has been to investigate and develop efficient methods for solving such optimization problems. The thesis focuses on decomposition-based algorithms in which a polyhedral outer approximation is used for approximating the integer relaxed feasible set. An introduction and summary of different types of optimization problems are given in Chapter 2, and a summary of work previously done within the field is given in Chapter 3.

The thesis is written as a collection of articles, and the results presented in the papers are summarized in Chapter 4. Different approaches for constructing and utilizing polyhedral outer approximations are presented and analyzed within the thesis. An algorithm, called the extended supporting hyperplane (ESH) algorithm, which uses supporting hyperplanes to the integer relaxed feasible set to construct the polyhedral approximation is presented. The ESH algorithm is utilized in a new solver, called SHOT, which is described in the thesis and presented in detail in the enclosed papers. Different techniques for utilizing the polyhedral approximations are presented in the thesis, and numerical test verifies their efficiency. Reformulation techniques for utilizing separability of the nonlinear functions to construct tighter approximations are also described and analyzed.

Contents

Preface iii

Svensk sammanfattning v

Abstract vii

Contents ix

1 Introduction 1

1.1 Structure of the thesis 3
1.2 List of publications 4
1.3 Contributions by the author 6

2 Background 7

2.1 Convexity 7
2.2 Linear programming 9
2.3 Mixed-integer linear programming 11
2.4 Nonlinear programming 14
2.5 Mixed-integer nonlinear programming 17
2.6 Conclusion 18

3 Convex MINLP 19

3.1 Convex MINLP problems 19
3.2 NLP based branch and bound 20
3.3 Polyhedral outer approximations 22
3.4 The extended cutting plane method 24
3.5 Outer approximation 26
3.6 Generalized Benders decomposition 29
3.7 LP/NLP-based branch and bound 34
3.8 Other techniques for convex MINLP 36
3.9 Software 37
3.10 Conclusion 37

4 Contributions to Convex MINLP 39

4.1 The extended supporting hyperplane algorithm (Papers I–II) 39
4.2 Reformulations for separability in convex MINLP (Paper III) 44

4.3 The center-cut algorithm (Paper IV) 46
4.4 Using regularization and second order derivatives with the outer approximation method (Paper V) 49
4.5 The supporting hyperplane optimization toolkit solver (Paper VI) 53
4.6 Conclusion 57

5 Conclusion and Future Ideas **59**
5.1 Conclusion 59
5.2 Future ideas 61

Bibliography **63**

Chapter 1

Introduction

Mathematical optimization has become a valuable tool for dealing with a variety of design and decision-making tasks within science, engineering, and economics. The goal of an optimization task is simply to find the best possible, or a good enough, solution to a given problem. Real-world optimization problems usually contain many decision variables, and to rigorously analyze and solve such an optimization problem requires the problem to be stated in a mathematical form. Writing an optimization problem in a mathematical form is often a non-trivial task, and there often exist several formulations of the same problem. Choosing a good problem formulation is important since different formulations can result in practically intractable problems or problems that can easily be solved. Optimization problems are often classified based on the attributes included in their mathematical forms; and some problem classes can easily be solved whereas other classes are far more demanding. Mathematical optimization has been an active research area since the 1960s, motivated by important applications within both industries and academia. The ability to find optimal solutions to problems can yield significant benefits, *e.g.*, the use of optimization within the chemical industry has resulted in reduced use of raw materials and reduction of green gas emissions. Typical optimization problems originating from industrial applications are, *e.g.*, production planning [164] , heat integration [203], process design [32], model predictive control [81], vehicle routing [187], and assignment problems [119].

Many real-world optimization problems contain some form of distinct decision making, where one has to choose among specific options. To incorporate such distinct decisions in a mathematical model requires the use of integer-valued decision variables, and such optimization problems are usually referred to as mixed-integer problems. How the decision variables interact and describe certain properties are modeled using linear and nonlinear functions, forming constraints and an objective. If all the functions are linear and the problem contains both continuous and integer variables, then the problem is commonly referred to as a mixed-integer linear programming (MILP) problem. Many natural phenomena cannot be modeled accurately by linear functions, but require the use of nonlinear functions in the model. Optimization problems containing both continuous and integer variables, as well as some nonlinear functions are commonly classified as mixed-integer nonlinear programming (MINLP) problems. MINLP is one of the most versatile modeling paradigms and, thus, there are numerous practical problems that can be modeled as MINLP problems [31, 41, 72, 91, 133, 185]. Unfortunately, MINLP problems tend to be very difficult to solve; as a consequence, real-world problems often have to be simplified and reduced in size to obtain computationally tractable problems. Further work in algorithmic research and solver development is, thus, well motivated in order to fully utilize and benefit from MINLP as a tool for design and decision-making tasks.

This thesis focuses on a specific type of MINLP problems that are referred to as convex MINLP problems. In convex MINLP problems, the nonlinear functions have certain desirable properties that can be exploited when solving the problem and enable an efficient decomposition of the problems. Decomposition-based methods for convex MINLP obtain the optimal solution by solving a sequence of easier optimization problems, often referred to as sub-problems. Progress in both algorithms and software for solving the sub-problems has been a motivation for further studying and developing new decomposition-based methods for solving convex MINLP problems.

The work presented here is focused around deterministic decomposition techniques, and four new methods for solving convex MINLP problems have been presented in the enclosed papers. The thesis is limited to deterministic methods, and methods such as evolutionary algorithms are not considered here. My research career, and time as a PhD student, basically began with the question "Can we improve the performance of the extended cutting plane method?", which resulted in the extended supporting hyperplane (ESH) algorithm. Later the scope grew, and the goal has been to investigate how to efficiently construct and utilize polyhedral approximations to solve convex MINLP problems. The work has resulted in new general algorithms for convex MINLP problems, which are based on polyhedral outer approximations of the feasible set. Based on the theoretical work, and in close collaboration with Docent Andreas Lundell, we have developed a state-of-the-art solver called SHOT, which is now commonly available as an open-source solver.

From a complexity point of view, even convex MINLP problems are $\mathcal{NP}$-hard. This follows from the fact that MILP problems are $\mathcal{NP}$-hard, and convex MINLP is a generalization of MILP. However, $\mathcal{NP}$-hard should not be used as a certificate that such problems cannot be solved, which seems to be the custom in some fields. Instead, it should be seen as an indicator that such a problem might be difficult, but not impossible to solve. MINLP may be a difficult type of optimization problem, but it should not stop us from trying to develop new algorithms and software. As John F. Kennedy said in one of his iconic speeches, " *We choose to go to the moon in this decade and do the other things, not because they are easy, but because they are hard,*" encouraging people to continue working on the hard, but ambitious, goals. Today, we are actually able to solve many $\mathcal{NP}$-hard optimization problems efficiently, and significant progress has been made. For example, extensive tests on MILP solvers in [3, 35] have shown a 100x speedup due to algorithmic developments over a 12 year period. In combination with the evolution of computer hardware, the progress has been truly impressive.

The goal of the research has been to contribute to the development of convex MINLP into a ready-to-use technology. There have been general-purpose solvers available for convex MINLP since the beginning of the 1990s. However, when compared to MILP or NLP solvers they have not yet reached the same level of maturity. Industry-relevant convex MINLP problems are still challenging to solve and may require expert knowledge to obtain good solutions. Further research is still needed for society to fully benefit from convex MINLP as a tool for dealing with difficult decision tasks; however, progress is continuously being made. The new methods presented here provide a new set of tools for dealing with convex MINLP problems, and the solver comparisons have, hopefully, resulted in a friendly competition among researchers within the field. Through the release of the SHOT solver, we have provided a new efficient open-source solver, which openly shares the knowledge we have learned.

1.1 Structure of the thesis

The work presented within this thesis is focused on techniques to efficiently solve convex MINLP problems by utilizing polyhedral approximations, and the main results have been presented in the enclosed articles. This thesis is written as a collection of articles, and the articles are enclosed at the end of the thesis. Chapter 2 is intended as a brief introduction to some basic theory and optimization methods. The theory is later used to derive and analyze the algorithms in the following chapters. Chapter 3 is intended as a literature review of convex MINLP and gives a summary of work previously done within the field. The methods presented in Chapter 3 also serves as a reference point to provide a better understanding of the methods presented in Chapter 4.

Chapter 4 gives a summary of the contributions to the field of convex MINLP presented within this thesis. The extended supporting hyperplane (ESH) algorithm is described in Section 4.1. The ESH algorithm was initially presented in Paper I, and in Paper II it was shown that it is applicable to a more general class of problems. Section 4.2 describes an extended reformulation technique, resulting in tighter polyhedral approximations, which can give significant benefits to several state-of-the-art solvers as shown in Paper III. Section 4.3 describes the center-cut algorithm for convex MINLP problems, which is presented in Paper IV. The center-cut algorithm can either be used as a deterministic method with guaranteed convergence, or as a so-called primal heuristic intended to obtain good solutions quickly. Section 4.4 describes a technique that enables the use of trust-regions and quadratic approximations within the outer approximation (OA) algorithm, and gives a summary of the results from Paper V. Section 4.5 is dedicated to the SHOT solver, giving a summary of the solver's main features and the benchmark results from Paper VI. Finally, the work is concluded in Chapter 5 along with some future research ideas within the field of convex MINLP.

1.2 List of publications

This thesis is based on the results presented in the following papers.

Paper I J. Kronqvist, A. Lundell, and T. Westerlund. The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. *Journal of Global Optimization*, Volume 64, pp. 249–272, 2016.

Paper II V.P. Eronen, J. Kronqvist, T. Westerlund, M. Mäkelä and N. Karmitsa. Method for solving generalized convex nonsmooth mixed-integer nonlinear programming problems. *Journal of Global Optimization*, Volume 69, pp. 443–459, 2017.

Paper III J. Kronqvist, A. Lundell, and T. Westerlund. Reformulations for utilizing separability when solving convex MINLP problems. *Journal of Global Optimization*, Volume 71, pp. 571–592, 2018.

Paper IV J. Kronqvist, D.E. Bernal, A. Lundell, and T. Westerlund. A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems. Accepted for publication in *Computers & Chemical Engineering*.

Paper V J. Kronqvist, D.E. Bernal and I.E. Grossmann. Regularization and second order information in outer approximation for convex MINLP. Manuscript submitted, preprint available at: `http://www.optimization-online.org/DB_HTML/2018/01/6405.html`.

Paper VI A. Lundell, J. Kronqvist, and T. Westerlund. The supporting hyperplane optimization toolkit: A polyhedral outer approximation based convex MINLP solver utilizing a single branching tree approach. Manuscript submitted, preprint available at: `http://www.optimization-online.org/DB_HTML/2018/06/6680.html`.

Some results have also been presented in the following conference papers.

1 J. Kronqvist, A. Lundell and T. Westerlund. A center-cut algorithm for solving convex mixed-integer nonlinear programming problems. A. Espuña, M. Graells and L. Puigjaner editors, *27rd European Symposium on Computer Aided Process Engineering*, Volume 32 of *Computer Aided Chemical Engineering*, pp. 2131–2136. Elsevier, 2017.

2 J. Kronqvist, A. Lundell and T. Westerlund. Lifted polyhedral approximations in convex mixed integer nonlinear programming. *XIII GLOBAL OPTIMIZATION WORKSHOP GOW'16 4-8 September 2016*, Volume 16, pp. 117–120, 2016.

3 A. Lundell, J. Kronqvist and T. Westerlund. Improvements to the supporting hyperplane optimization toolkit solver for convex MINLP. *XIII GLOBAL OPTIMIZATION WORKSHOP GOW'16 4-8 September 2016*, Volume 16, pp. 101–104, 2016.

4 A. Lundell, J. Kronqvist and T. Westerlund. An extended supporting hyperplane algorithm for convex MINLP problems. *XII GLOBAL OPTIMIZATION WORKSHOP MAGO'14*, Volume 16, pp. 21–24, 2014.

However, these conference papers are not included since the most important results are all given in Papers I–VI.

1.3 Contributions by the author

To clarify my (Jan Kronqvist's) role in the papers, descriptions of my contributions to the papers are given below.

Paper I The ESH algorithm in the paper is a result of a collaboration between all the authors. I am mainly responsible for writing the paper and the proof of convergence. The implementation of the SHOT solver was mainly done by A. Lundell.

Paper II The paper continues on the ESH algorithm from Paper I, and I have contributed to the theory and examples in the paper. I have also contributed to the writing process; however, V.P. Eronen is the main author. I also made a first implementation of the algorithm. However, the algorithm was later implemented in the GAECP solver by T. Westerlund, which was used in the paper.

Paper III The idea for the reformulations originated from observations that several polyhedral outer approximation based solvers struggled with some types of problems, and we began investigating different problem formulations. I am responsible for writing the paper, analyzing the reformulation technique, and the numerical experiments. The co-authors assisted in the writing process.

Paper IV I got the idea for the center-cut algorithm from investigating different approaches to obtaining the interior point to be used in the ESH algorithm. I am responsible for writing the paper, proving convergence of the algorithm, and making the center-cut implementation used in the numerical comparison. The co-authors assisted in writing the paper and with the numerical comparison.

Paper V I spent some time studying bundle methods, and the nice convergence properties of the level method inspired me to investigate a similar approach for convex MINLP problems. I am mainly responsible for writing the paper, proving convergence of the algorithms, and making the implementations used in the numerical comparison. The co-authors assisted in writing the paper and with the numerical comparison.

Paper VI The features used in the SHOT solver are a combination of different ideas of all the authors, and I tested some of the techniques before they were fully implemented in SHOT. As a co-author, I have also participated in writing the paper. However, A. Lundell is mainly responsible for implementing the SHOT solver and writing the paper.

Chapter 2

Background

Optimization problems containing integer restrictions cannot usually be solved directly, and most methods rely on some decomposition or relaxation technique. The main idea behind such methods is to construct simpler approximations of the original problem, that can be solved efficiently, and to obtain the optimal solution by solving a sequence of simplified problems. The "simplified" problems are often referred to as sub-problems, and to fully understand the main methods requires knowledge of some basic properties of these sub-problems. The most commonly used decomposition and relaxation techniques for convex MINLP problems are described later in Chapter 3. Here we begin with a summary of convex properties and an introduction to optimization problems frequently occurring as sub-problems in methods for convex MINLP problems.

2.1 Convexity

This section is dedicated to convexity, and we will start by the formal definitions of convex sets and convex functions. Conditions for ensuring convexity of functions are provided, and the concept of pseudo-convexity is also presented.

The concept of convex sets is important within optimization, and we start by defining a convex set.

Definition 1. A set $C \subset \mathbb{R}^n$ is convex if the entire line segment between any two points in the set lies in C, *i.e.*, for any $\mathbf{x}, \mathbf{y} \in C$ and $\theta \in [0,1]$ it must be true that

$$\theta\mathbf{x} + (1-\theta)\mathbf{y} \in C.$$

A set that is not convex is, in general, referred to as a non-convex set. An important example of convex sets is sets defined by linear constraints. A set defined by finitely many linear constraints is commonly referred to as a polyhedral set, and convexity can easily be proven, *e.g.*, see page 22 of [42]. There are several operations that preservers convexity of sets, *e.g.*, the intersection of convex sets results in a convex set [42]. For more details on convex sets, see the classic textbook by R.T. Rockafellar [169].

It is also of interest to study convex properties of functions, and we begin by the definition of a convex function.

Definition 2. A function $f : \mathbb{R}^n \to \mathbb{R}$ is defined as convex on the convex set C if

$$f(\theta \mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}) \quad \forall \mathbf{x}, \mathbf{y} \in C,\ \theta \in [0,1]. \tag{2.1}$$

The inequality defining a convex function is commonly called Jensen's inequality [42, 46]. If the condition in eq. (2.1) holds with strict inequality, then the function is referred to as *strictly convex*. If the function f is convex, then the function $-f$ will be *concave*. Thus, both *convexity* and *concavity* can be defined from Jensen's inequality. An important property that follows directly from Jensen's inequality is described in the following theorem.

Theorem 1. *Any local minimum of a convex function is also a global minimum.*

The theorem can be easily be proven by contradiction, *e.g.*, see [42].

Theorem 1 has important implications in optimization, since it ensures that it is sufficient to obtain a local minimum for globally minimizing a convex function. From Jensen's inequality, it may be difficult to determine directly if a function is convex. For continuously differentiable functions it is possible to identify convexity based on properties of their derivatives, which leads us to the first- and second-order convexity condition.

Theorem 2. ***First-order convexity condition.*** *A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex on the convex set C if and only if*

$$f(\mathbf{y}) \ \geq \ f(\mathbf{x}) + \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) \quad \forall \mathbf{x}, \mathbf{y} \in C. \tag{2.2}$$

A proof of the theorem is, *e.g.*, given in [42].

An important consequence of the first-order condition is the fact that a first-order Taylor series expansion yields a *global underestimator* of the convex function on the set C. This property will later be used to construct polyhedral outer approximations for convex MINLP problems. However, the first-order condition is, usually, not a practical way of proving convexity and this leads us to the next convexity condition.

Theorem 3. ***Second-order convexity condition.*** *A twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex on the convex set C if and only if, its Hessian matrix* $\mathbf{H}(\mathbf{x})$ *is positive semidefinite for all* $\mathbf{x} \in C$, i.e.,

$$\mathbf{H}(\mathbf{x}) \succeq 0 \quad \forall \mathbf{x} \in C. \tag{2.3}$$

The theorem is, *e.g.*, proven in [169].

The second-order convexity condition provides an efficient approach for analyzing convexity of twice differentiable functions. *Strict convexity* can also be identified from the Hessian, which requires the Hessian to be positive definite. For a matrix to be positive semidefinite all of the eigenvalues have to be non-negative, and to be positive definite requires all eigenvalues to be strictly positive. Thus, it is possible to identify convexity by analyzing the eigenvalues of the Hessian. For quadratic functions it is easy to analyze convexity; however, for more complex functions it may be a non-trivial task to verify that a function is convex. Practically, convexity is often analyzed by decomposing the function and analyzing the parts individually, *e.g.*, Boyd and Vandenberghe [42] describe the circumstances under which compositions of functions are guaranteed to result in a convex function.

Pseudoconvexity and quasiconvexity are generalizations of convexity [46], where a quasiconvex function is the most general type of function. Paper II deals with optimization problems containing pseudoconvex constraint functions, and therefore, we will also introduce the concept of pseudoconvex functions.

Definition 3. A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is pseudoconvex on C if,

$$\forall \mathbf{x}, \mathbf{y} \in C \; : \; f(\mathbf{y}) < f(\mathbf{x}) \implies \nabla f(\mathbf{x})^T (\mathbf{y} - \mathbf{x}) < 0. \tag{2.4}$$

It follows from the definition, that the gradient of a pseudoconvex function can only be zero at a global minimum, *i.e.*, a stationary point will correspond to a global minimum [46]. However, a first-order Taylor series expansion of a pseudoconvex function may not be a valid underestimator. Therefore, it is not trivial to construct outer approximations of a constraint given by a pseudoconvex function [198]. An essential property of both pseudo- and quasiconvex functions is described in the following theorem.

Theorem 4. *A function f, which is either pseudo- or quasiconvex on C, has convex level sets in C,* i.e., *$F_l := \{\mathbf{x} \in C \mid f(\mathbf{x}) \leq l\}$ is a convex set $\forall l \in \mathbb{R}$.*

For more details and a proof of the theorem see, *e.g.*, [46].

Since pseudo- and quasiconvex functions are generalizations of convex functions, Theorem 4 is obviously also true for convex functions. The property of convex level sets can be utilized by the ESH method and enables the method to achieve global convergence for a more general type of problems than "purely" convex. In Paper II it was shown that the ESH algorithm can successfully be applied to problems with pseudoconvex constraints functions, and more details are given in Chapter 4.

This section has covered some of the most important convexity properties. However, many details are left out; for more information see [42, 46, 169]. Next, we continue with a brief overview of some types of optimization problems that will be utilized later on.

2.2 Linear programming

Linear programming (LP) was the first general framework for formulating optimization problems [56], and the introduction of the simplex method around the 1950s made it possible to actually solve some real-world planning problems [57]. An LP problem can, without loss of generality, be formulated as

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n, \end{aligned} \tag{P-LP}$$

where **c** is a vector defining the objective. The linear constraints are all defined by the matrix **A** and vector **b**. For example, finding an optimal solution to the task of transporting commodities from a set of facilities with fixed supply to a number of consumers with specific demands can be formulated as an LP problem [101, 118]. Next, some properties of LP are presented that will be useful in the consecutive chapters.

The linear constants in problem (P-LP) forms a polyhedral set L, given by

$$L = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}. \tag{2.5}$$

The set L is obviously a convex set, since it is defined by linear constraints. The following definition introduces a concept widely used in LP, and the next theorem utilizes this concept to describe an optimal solution of an LP problem.

Definition 4. An extreme point $\bar{\mathbf{x}}$ of the set L is a point that cannot be obtained as a convex combination of any two other points $\mathbf{y}, \mathbf{z}$ within the set, *i.e.*,

$$\nexists\, \mathbf{y}, \mathbf{z} \in L \backslash \bar{\mathbf{x}} \ : \ \bar{\mathbf{x}} = \theta \mathbf{z} + (1-\theta)\mathbf{y},\ \theta \in [0,1].$$

Theorem 5. *Assuming that the set L is nonempty and bounded, then the optimal solution of problem* (P-LP) *will be one of the finitely many extreme points of the set L.*

The theorem describes one of the fundamental properties of LP, and a proof can be found, for example, on page 65 in [27].

From Theorem 5 we can easily derive the main idea of the simplex method, which is one of the most commonly used methods for solving LP problems. Since an optimal solution of problem (P-LP) is located at an extreme point of the set L, it is sufficient to merely examine the extreme points of the set. However, there might exist a vast number of extreme points and examining all the extreme points is, thus, not a desirable approach. The simplex method provides an efficient technique for navigating between the extreme points and finding the optimal solution. The method starts at an extreme point of the set, a so-called *basic feasible solution*, and iteratively moves to adjacent extreme points such that the objective value is improved. Due to convexity of the set L, an extreme point is guaranteed to be optimal if none of the adjacent extreme points have a better objective value, and the optimal solution can always be found by iteratively moving to adjacent extreme points with better objective values. The main strength of the simplex method is the simplicity of the calculations needed for moving between the extreme points and choosing the directions. For more details on the simplex method see, *e.g.*, [56, 139].

From a complexity point of view, the basic simplex method has a non-optimal worst-case performance [114]. Polynomial time algorithms for LP, such as Karmarkar's interior point method [108] and the ellipsoid method [111], were presented in the late '70s and early '80s. However, in practice versions of the simplex method, such as the dual simplex method [130], are still considered as one of the most efficient techniques for solving LP problems [36].

There are several solvers, both commercial and academic, available for LP problems, such as CLP [74], CPLEX [105], Gurobi [95], glpk [144], and XPRESS [66]. Solvers for LP problems have already reached a certain maturity, and LP problems are today considered as "easy" problems. In recent benchmarks, solvers have even been able to solve LP problems with more than a million variables in less than a minute [154]. Of course, there are still LP problems, usually very large, that are not easy to solve with the commonly available solvers, but in general, LP problems can be solved efficiently. The ability to efficiently solve LP problems is of utter importance when dealing with integer problems since most techniques are based on relaxations that usually requires the solution of a vast number of LP sub-problems. Many important details of LP are left out here; for more details, see one of the excellent textbooks [27, 56, 177].

2.3 Mixed-integer linear programming

The need to incorporate integer variables in optimization problems was quickly noticed, and the first algorithms for mixed-integer linear programming (MILP) problems were presented in the late 1950s [58, 88]. Integer variables make it possible to describe discrete quantities and distinct decisions; *e.g.*, a decision where either one or another set of equations has to be satisfied can easily be modeled by using a binary variable. A MILP problem can, without loss of generality, be stated as

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y} \\ \text{subject to} \quad & \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{Z}^m. \end{aligned} \tag{P-MILP}$$

Optimization problems with industrial applications that can be formulated as MILP problems are, *e.g.*, allocation problems [194], packing problems [85], process planning [176], scheduling [77, 163], and the famous traveling salesman problem [136].

The primary approach for solving MILP problems is the so-called branch and bound (BB) algorithm [125]. The BB algorithm tackles a MILP problem by dropping the integer requirements of the problem and dividing the search space. By dropping the integer restrictions, the resulting LP-relaxation given by

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y} \\ \text{subject to} \quad & \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m, \end{aligned} \tag{P-MILPr}$$

can easily be solved. However, the LP relaxation may not result in a feasible integer solution , *i.e.*, one of the integer variables may have a fractional value. The LP-relaxation will still provide a valid lower bound (LB), since it is a relaxation of the original problem. Now, suppose $(\mathbf{x}^*, \mathbf{y}^*)$ is the minimizer of the relaxed problem (P-MILPr), and that one of the integer variables y_i takes on a fractional value, *i.e.*, $\text{round}(y_i^*) \neq y_i^*$. To find an integer solution of the problem, the search space is divided into two parts such that the fractional solution is excluded from the search space but all feasible integer solutions are still included in the search. Such a division of the search space is achieved by generating two new sub-problems, and adding one of the constraints $y_i \leq \lfloor y_i^* \rfloor$ and $y_i \leq \lceil y_i^* \rceil$ to each sub-problem. Here, $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ refer to the floor and ceiling operators. After solving the new sub-problems, the lower bound can be updated as the lower objective value of the sub-problems. In case one of the sub-problems returns a feasible integer solution, it provides a valid upper bound (UB) to the MILP problem and it is the optimal integer solution in that region of the search space. If the difference between the upper and lower bounds is not within the desired tolerance, then the search continues by dividing the sub-problems with a non-integer solution into two new sub-problems. The procedure of dividing the search space into sub-regions is referred to as branching, and the variable on which the division is done is called the branching variable. Often there are several branching options, *e.g.*, if several of the integer variables take on fractional values. A branching variable is then chosen based on some criteria, *e.g.*, most infeasible branching, pseudo branching, or strong branching [4]. The solution procedure is often represented as a tree where the initial LP-relaxation corresponds to the *root node*, and the sub-problems are represented by nodes branching out. In case the optimal solution of one of the sub-problems (nodes) exceeds the current upper bound, then the optimal solution cannot be within that specific part of the search space, and there is no need to further explore the node. When the search is stopped along one node, it is referred to as *pruning* the node. Some of the sub-problems may also become infeasible, and such nodes can also be pruned from the search tree. A node that may still contain the optimal solution is referred to as an open node, and several strategies for determining the order in which to explore open nodes have been proposed [7, 50, 104].

To illustrate the BB procedure, consider the following pure integer problem

$$\begin{aligned} \text{minimize} \quad & -3y_1 - 5y_2 \\ \text{subject to} \quad & 2y_1 + 4y_2 \leq 25, \\ & 0 \leq y_1 \leq 8, \\ & 0 \leq y_2 \leq 5, \\ & y_1, y_2 \in \mathbb{Z}. \end{aligned} \tag{ex1}$$

The procedure for solving the problem with the BB algorithm is illustrated in figure 2.1. For this simple problem, the optimal solution is already found after the first branching step; however, verifying optimality requires two additional branching steps. Two of the nodes represents infeasible sub-problems that can be pruned off. After the third branching step, there are no open nodes with a lower bound better than the upper bound and, thus, the search can be terminated. Note that, even if the problem contains only two integer variables, the BB algorithm already requires the solution of seven LP sub-problems.

The first method proposed for solving MILP problems was not the BB algorithm, but a cutting plane algorithm presented by Gomory in 1960 [86]. A similar cutting plane algorithm for pure integer problems had already been presented by Gomory in 1958 [88]. The main idea is, again, to solve LP-relaxations, but instead of branching,

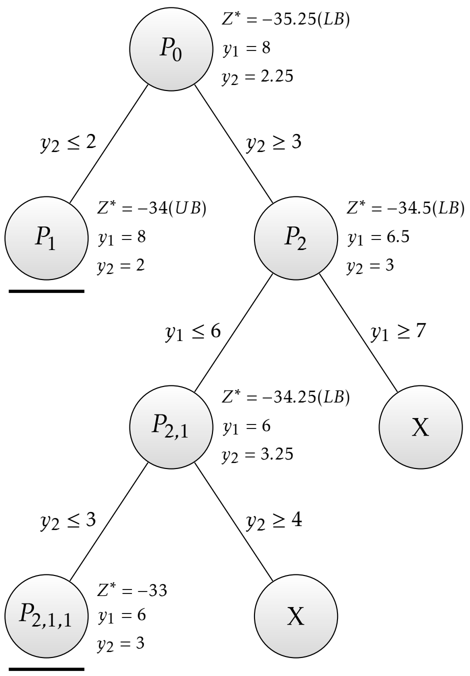

Figure 2.1: Branch and bound tree obtained for problem (ex1). The optimal objective value of the sub-problems are denoted by Z^*, and an X in the node indicates that the sub-problem is infeasible. The inequalities along the branches represent the restrictions added to consecutive sub-problems, and nodes resulting in integer solutions are underlined.

to derive cuts that exclude the non-integer solution of the LP-relaxation. To show how such cuts can be generated, consider a problem with the following constraints in the integer variables $\mathbf{y}$

$$\mathbf{A}_2\mathbf{y} \leq \mathbf{b}, \tag{2.6}$$

where $\mathbf{a_1}, \mathbf{a}_2, \ldots, \mathbf{a_n}$ are the columns of the $\mathbf{A}_2$ matrix. Here it is assumed that all the integer variables $\mathbf{y}$ are restricted to non-negative integers. The individual constraints of eq. (2.6) can be combined into a single constraint according to

$$\boldsymbol{\lambda}^T\mathbf{A}_2\mathbf{y} \leq \boldsymbol{\lambda}^T\mathbf{b}, \tag{2.7}$$

where $\boldsymbol{\lambda}$ is a non-negative scaling vector. Rounding down the coefficients on the left-hand side, given by $\boldsymbol{\lambda}^T\mathbf{a_i}$, results in a valid relaxed constraint. Once the right-hand side contains only integer coefficients, then an integer solution can only result in an integer value on the right-hand side. The coefficient on the left-hand side can then be rounded down to an integer value, thus strengthening the constraint and resulting in the following Chvátal-Gomory cut [49]

$$\left[\lfloor\boldsymbol{\lambda}^T\mathbf{a_1}\rfloor \;\; \lfloor\boldsymbol{\lambda}^T\mathbf{a}_2\rfloor \;\; \ldots \;\; \lfloor\boldsymbol{\lambda}^T\mathbf{a_n}\rfloor\right]\mathbf{y} \leq \lfloor\boldsymbol{\lambda}^T\mathbf{b}\rfloor. \tag{2.8}$$

Chvátal-Gomory cuts will not cut off any feasible integer solution of the problem. However, they can cut off non-integer regions of the polyhedron $L = \{(\mathbf{y} \in \mathbb{R}^m \mid \mathbf{A}_2\mathbf{y} \leq \mathbf{b}\}$, and strengthen the LP-relaxation. There is no unique way of constructing Chvátal-Gomory cuts, and by choosing different scaling vectors $\boldsymbol{\lambda}$, an infinite number of cuts can be obtained. Gomory's cutting plane algorithm for integer problems [88] uses an iterative procedure for generating cuts according to eq. (2.8) to iteratively improve the LP-relaxation, and to obtain an integer solution.

As an example of Chvátal-Gomory cuts, consider again example (ex1). By scaling the first constraint by a factor of 0.5 and rounding down all coefficients we obtain the cut $y_1 + 2y_2 \leq 12$. Adding the cut to the LP-relaxation directly results in a feasible integer solution, and the optimal solution can then be obtained by solving only a single LP problem.

Cutting planes for MILP problems has received a great deal of interest over the years, and many different types of cuts have been proposed, *e.g.*, mixed-integer Gomory cuts [87], disjunctive cuts [12], lift-and-project cuts [13], intersection cuts [11], split cuts [53] and cover cuts [93, 162]. These cuts utilize different properties of MILP problems, with the common goal of obtaining a tight LP-relaxation. However, using cutting planes alone for solving MILP problems is usually not an efficient approach. The number of cutting planes needed may be huge, and the cutting planes tend to become almost parallel, causing problems with degeneracy of the LP sub-problems [204]. Today, most MILP solvers are based on the branch and cut technique, which uses cutting planes to strengthen the LP-relaxations in the branch and bound algorithm [153].

From a complexity point of view, MILP problems are $\mathcal{NP}$-hard [178], meaning we cannot expect MILP problems to be easy. However, thanks to the tremendous work done within the field of MILP, there has been significant progress in general purpose solvers for MILP problems [34]. There are several solvers available for MILP problems, such as CBC [75], CPLEX [105], Gurobi [95], SCIP [2], and XPRESS [66]. Benchmark tests in [115] have shown that these solvers have even been able to solve some MILP problems with more than 100,000 discrete variables. Solving MILP sub-problems is one the key components in several methods for convex MINLP, and the ability to efficiently solve MILP problems is crucial for such methods. Convex MINLP methods utilizing MILP-relaxation will be described in detail later on.

This section is merely intended as a brief introduction to MILP, to give a better understanding of methods for convex MINLP problems. For more details on MILP see, *e.g.*, [28, 51, 157, 177].

2.4 Nonlinear programming

By incorporating nonlinear functions to describe dependencies of the variables, it is possible to accurately describe a wide range of processes and phenomena. There are, thus, a variety of applications of nonlinear programming (NLP) within different fields of science and engineering [68]. Applications of NLP optimization within chemical engineering are, *e.g.*, optimization of process units [30], nonlinear model predictive control [8], and real-time optimization of operating conditions[145].

An NLP problem can be defined as

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h_i(\mathbf{x}) = 0, \quad i = 1,\dots,k, \\ & g_j(\mathbf{x}) \leq 0, \quad j = 1,\dots,l, \\ & \mathbf{x} \in \mathbb{R}^n, \end{aligned} \tag{P-NLP}$$

where f, h_i, and g_j are functions mapping from $\mathbb{R}^n$ to $\mathbb{R}$. The functions f, h_i, and g_j are here not separated into linear and nonlinear, and some of them may be linear and some nonlinear. In general, NLP problems can be considered as difficult problems ($\mathcal{NP}$-hard), *e.g.*, combinatorial problems can easily be modeled as NLP problems by the use of nonlinear equality constraints.

However, specific types of NLP problems, such as convex NLP problems, can usually be solved efficiently. There are different definitions of convex NLP, and here we will use the definition by Boyd [42] which requires the following assumptions to be true:

Assumption 1. The functions f and g_j are all convex.

Assumption 2. h_i are all affine (linear) functions.

Furthermore, here we will assume that all the functions f, h_i, and g_j are continuously differentiable. NLP problems satisfying these assumptions have certain properties that can be exploited to solve the problems efficiently. These properties are also central in some of the methods for convex MINLP methods, and worth describing in some detail here. We begin with the definition of the Lagrangian function.

Definition 5. The *Lagrangian* $\mathcal{L} : \mathbb{R}^{n+k+l} \to \mathbb{R}$ associated with problem (P-NLP) is given by

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^{l} \mu_j g_j(\mathbf{x}).$$

The Lagrangian is frequently used as a technique to account for the constraints by augmenting the objective function by a weighted sum of the constraints functions. The weights λ_i and μ_j are commonly referred to as the *Lagrangian multipliers* of the constraints. Next, we will introduce the concept of the *Lagrangian dual function*.

Definition 6. The *Lagrangian dual function* is defined as the minimum value of *Lagrangian* given specific values of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, *i.e.*,

$$\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x}} \left(f(\mathbf{x}) + \sum_{i=1}^{k} \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^{l} \mu_j g_j(\mathbf{x}) \right).$$

An important property of the Lagrangian dual function is the fact that it provides a lower bound on the objective for the original problem. This property is further described in the following theorem.

Theorem 6. *For any set of valid multipliers, $\boldsymbol{\lambda} \in \mathbb{R}^k$ and $\boldsymbol{\mu} \geq 0$, the following relation holds*

$$\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq z^*,$$

where z^* denotes the optimal value of problem (P-NLP).

The theorem is easily proven, *e.g.*, see [42].

If strong duality holds, then there exist valid multipliers $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ such that the *Lagrangian dual function* is equal to the optimal value of problem (P-NLP), *i.e.*, $\phi(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = z^*$. Convexity alone does not guarantee that strong duality holds for NLP problems; however, there are conditions under which strong duality holds. These conditions are often referred to as constraint qualifications or regularity conditions [15]. Slater's condition [180] is an example of such a condition.

Theorem 7. ***Slater's condition.*** *Strong duality holds for a convex NLP problem, if there exists a feasible point $\bar{\mathbf{x}}$ that strictly satisfies all nonlinear inequality constraints,* i.e., $g_j(\mathbf{x}) < 0$ *for all nonlinear constraints.*

For more details and a proof of the theorem, see [180].

An optimal solution of a convex NLP problem, satisfying Slater's condition, can thus be obtained by maximizing the *Lagrangian dual function*. Furthermore, the optimal solution of such an NLP problem must also correspond to a stationary point of the *Lagrangian*. Due to these properties, it is possible to obtain necessary and sufficient conditions for an optimal solution of a convex NLP problem. These conditions are commonly referred to as the Karush-Kuhn-Tucker (KKT) conditions [25].

Theorem 8. ***KKT conditions.*** *An optimal solution $\mathbf{x}^*$ to a convex NLP problem, satisfying a constraint qualification, must satisfy the following conditions*

$$\begin{aligned}
\nabla f(\mathbf{x}^*) + \sum_{i=1}^{k} \lambda_i^* \nabla h_i(\mathbf{x}^*) + \sum_{j=1}^{l} \mu_j^* \nabla g_j(\mathbf{x}) = \mathbf{0},& \\
h_i(\mathbf{x}^*) = 0, &\quad i = 1, \dots, k, \\
g_j(\mathbf{x}^*) \leq 0, &\quad j = 1, \dots, l, \\
\mu_j^* \geq 0, &\quad j = 1, \dots, l, \\
\mu_j^* g_j(\mathbf{x}^*) = 0, &\quad j = 1, \dots, l,
\end{aligned} \tag{2.9}$$

where $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ are the optimal multipliers (dual variables).

The theorem describes one of the fundamental properties in convex optimization, and proofs of the theorem can be found in [109, 124].

The KKT conditions have an important role in convex NLP by providing necessary and sufficient conditions for an optimal solution. For some specific types of problems it is possible to directly solve the KKT system, and thus obtain the optimal solution. For a non-convex NLP, the KKT conditions still provide a necessary optimality condition, but they do not guarantee global optimality.

There are several methods available for solving convex NLP problems, and most of them can be viewed as an iterative procedure for solving the KKT conditions. For example, the active set technique uses a procedure of selecting active inequality constraints, *i.e.*, binding inequality constraints that hold with equality. By dropping the non-active inequality constraints, the KKT conditions are reduced to a nonlinear equation system that can be solved efficiently, *e.g.*, by Newton's method. The set of active constraints are iteratively updated, since the set of active constraints at the optimum is, generally, not known in advance. The sequential quadratic programming (SQP) method, a commonly used technique for NLP problems, is a special case of the active set method [30]. Other popular methods for convex NLP problems are, *e.g.*, the augmented Lagrangian method [26], the generalized reduced gradient method [126], and interior point methods [76].

There are several efficient solvers available for convex NLP problems, *e.g.*, CONOPT [61], filterSQP [70], IPOPT [193], KNITRO [45], LANCELOT [52], MINOS [155], and SNOPT [84]. NLP has already reached a certain maturity [76], and convex NLP problems are today, generally, considered as "nice" problems. Non-smooth NLP problems are overall more challenging [158], and such problems have not been considered in this section. For details on non-smooth NLP see, *e.g.*, [9].

This section has only covered some basic properties of NLP that are useful in the forthcoming sections on convex MINLP. More details on NLP methods and theory can, *e.g.*, be found in the excellent textbooks [14, 30, 42].

2.5 Mixed-integer nonlinear programming

Mixed-integer nonlinear programming (MINLP) combines the modeling capabilities of MILP and NLP, resulting in a versatile modeling paradigm. The discrete variables enable the modeling of distinct decisions, and the nonlinear functions can describe complex interactions. There are, thus, plenty of real-world optimization tasks that can be modeled as MINLP problems, *e.g.*, cancer treatment planning [47], crude oil scheduling [134], design of heat exchanger networks [202], nuclear reactor core fuel reloading [168], optimization of red blood cell production [152], production planning [156, 175], protein folding [73], pooling problems [149], portfolio optimization [37] and process synthesis [89].

A general MINLP problem can be written in the following form

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{subject to} \quad & h_i(\mathbf{x},\mathbf{y}) = 0, \quad i = 1,\dots,k, \\ & g_j(\mathbf{x},\mathbf{y}) \le 0, \quad j = 1,\dots,l, \\ & \mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{Z}^m. \end{aligned}$$

Based on the properties of the functions f, h_i, and g_j the MINLP problem is classified as either convex or non-convex. Convex MINLP is described in detail in the next section; some details regarding non-convex MINLP follow.

Non-convex MINLP problems are in general more difficult than convex MINLP problems. A convex MINLP problem can, under mild assumptions, be solved exactly by a finite number of convex NLP problems, which is not necessarily true for a non-convex MINLP problem. However, progress has also been made within the field of non-convex MINLP. A commonly used technique for non-convex MINLP is to generate convex underestimators of the non-convex functions within a spatial branch and bound framework [185]. With this approach, the convex underestimators become tighter as the search space is divided into smaller subregions. The convex underestimators are used in convex relaxations of the original problem, and the solutions of the convex relaxations provide iteratively improving lower bounds on the original problem. Upper bounds on the original problem are usually obtained by locally optimizing the original problem within the subregions [135]. Spatial branch and bound uses a similar technique to divide the search space as the BB method for MILP problems, although in spatial branch and bound, the continuous variables may also be selected as branching variables. By dividing the search space into smaller subregions, the convex relaxations become tighter within their subregions, and the upper and lower bounds eventually converge within a given tolerance.

To avoid creating unnecessarily large search trees it is important to obtain tight convex underestimators, and the "tightest" convex underestimator of a function is known as the *convex envelope*. For some types of non-convex functions, such as bilinear, concave, and fractional functions, there are known analytical expressions of the convex envelope over a box-constrained region [146, 184]. For twice continuously differentiable functions it is also possible to use the αBB approach for generating convex underestimators [5, 6]. In order to obtain tight convex underestimators, it is of utter importance to obtain tight variable bounds. Bound tightening and range reduction techniques are, therefore, important tools when dealing with non-convex MINLP problems, as shown in [170]. Non-convex MINLP solvers based on the spatial branch and bound are, *e.g.*, ANTIGONE [151], BARON [173], and SCIP [191].

By using piecewise linear functions together with convex underestimators, it is also possible to solve non-convex MINLP problems as a sequence of convex MINLP problems [73, 142]. Certain classes of nonconvex MINLP problems, such as problems with signomial constraints, can also be solved by reformulating them into convex MINLP problems [140, 141, 165]. Progress in methods and solvers for convex MINLP may, therefore, also provide new opportunities within non-convex MINLP. There are also other approaches for solving non-convex MINLP problems, such as the decompositions techniques presented in [160, 161]. Non-convex MINLP is just briefly mentioned here, since this thesis focuses on convex problems. For more details on non-convex optimization see, *e.g.*, [73, 102, 135, 150, 181, 185, 186].

2.6 Conclusion

This chapter gives an introduction to different types of optimization problems and describes some characteristic features. Some methods for solving certain types of optimization problems have briefly been presented as background information. Now, we are ready to move on to the main theme of this thesis, namely convex MINLP.

Chapter 3

Convex MINLP

This chapter covers the fundamental properties of convex MINLP problems, and the standard methods for solving convex MINLP problems are described in detail. The methods presented in this chapter are NLP-based branch and bound, the extended cutting plane method, outer approximation, generalized Benders decomposition, and LP/NLP-based branch and bound. Some details regarding software for convex MINLP problems are also given here. The methods presented in Papers I–VI are not covered here but presented later in the forthcoming chapter.

3.1 Convex MINLP problems

Defining an optimization problem with integer variables as convex might seem odd, since the integer restrictions will obviously result in a non-convex feasible set. However, classifying MINLP problems based on properties of the nonlinear functions still makes sense, because these properties basically determine how the problem can be solved. There are slightly different definitions of convex MINLP problems, but here we use the same definition as Lee and Leyffer in [128].

Definition 7. A MINLP problem is convex if all nonlinear functions defining the objective and inequality constraints are convex, and all equality constraints are given by linear functions.

A convex MINLP problem can, without loss of generality, be defined as

$$\min_{(\mathbf{x},\mathbf{y})\in N\cap L\cap Y} \mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y}, \tag{P-MINLP}$$

where the sets N, L and Y are given by

$$\begin{aligned} N &= \{(\mathbf{x},\mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m \mid g_j(\mathbf{x}) \le 0 \quad \forall j = 1,2,\dots l\}, \\ L &= \{(\mathbf{x},\mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m \mid \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} \le \mathbf{b}\}, \\ Y &= \{\mathbf{y} \in \mathbb{Z}^m\}. \end{aligned} \tag{3.1}$$

The assumption of a linear objective function is not restrictive, since a convex objective function f can easily be transformed into the convex constraints $f(\mathbf{x},\mathbf{y}) - x_{n+1} \leq 0$, by introducing a new continuous variable x_{n+1}. Here, it is also assumed that all the nonlinear functions are continuously differentiable. Paper II shows that the ESH method has guaranteed convergence for some non-differentiable functions, and the method is described in Chapter 4. Throughout this thesis, it is assumed that L is a *compact* set. The compactness assumption is needed for all sub-problems to be well defined, and as a consequence, the intersection $L \cap Y$ will contain only a finite number of different integer combinations. Practically, compactness is not an issue since arbitrary large variable bounds can be assigned to all variables. To avoid any confusion, the assumptions used throughout this chapter are summarized.

Assumption 1. All the nonlinear functions f and g_j are convex.

Assumption 2. All the nonlinear functions f and g_j are, at least once, continuously differentiable.

Assumption 3. The intersection of the sets L and Y defines a compact set.

Binary and integer variables are not treated separately in this thesis, and they are both simply considered as integer variables. The following sections describe commonly used methods for solving convex MINLP problems. We begin with nonlinear branch and bound, which is a natural extension of the BB method for MILP and continues with polyhedral outer approximation based methods. The methods are not presented in chronological order, but in an order in which the algorithmic complexity gradually increases.

3.2 NLP based branch and bound

As mentioned earlier, the branch and bound algorithm for MILP problems was first presented by Land and Doig [125]. A few years later, in 1965, Dakin [54] noted that a similar branch and bound technique could also be used for MINLP problems. However, Dakin's paper mainly focused on linear problems. The branch and bound approach was later revisited by Gupta and Ravindran in 1985 [94], where they focused on convex MINLP problems.

Similar to the branch and bound method for MILP problems, the main idea is to relax the problem by dropping the integer requirements and to divide the search space by branching to obtain integer solutions. When dealing with convex MINLP problems, the integer relaxation results in convex NLP problems. The main difference compared to branch and bound for MILP problems is, thus, that the sub-problems in each node are NLP problems instead of LP problems. The approach of solving NLP problems in each node is often referred to as NLP-based branch and bound (NLP-BB). Branch and cut techniques, utilizing cuts to strengthen the integer relaxation, have also been proposed for convex MINLP problems [182]. Obtaining a tight integer relaxation is essential to reducing the size of the branch and bound tree, and several cut generating procedures

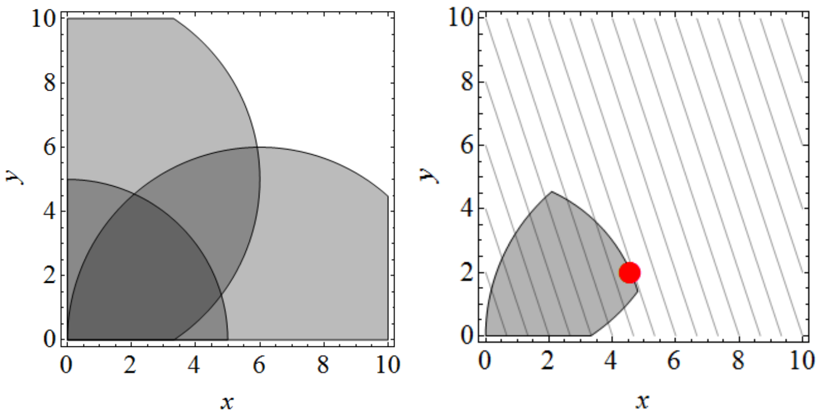

Figure 3.1: The left figure illustrates the feasible regions defined by individual constraints of problem (ex2). The right figure shows contours of the objective function, the feasible region given by the constraints, and the dot represents the optimal solution of the MINLP problem.

have been presented. Some types of cutting planes intended for MILP, such as Gomory cuts [48, 88], can also be used for MINLP problems. Other frequently used cuts are, *e.g.*, lift-and-project cuts [13, 112, 206], and perspective cuts [79]. For more details on cuts for MINLP problems see, *e.g.*, [19]. Early branching is another technique proposed to improve the performance, by efficiently integrating the NLP solver and not solving all NLP sub-problems to optimality [40, 132].

Compared to branch and bound for MILP problems, NLP-BB tends to suffer from the computationally more expensive sub-problems. Even if the sub-problems are convex NLP problems, they are usually more computationally demanding to solve than LP problems. Keep in mind, for an average-sized problem it is not unusual that a BB-based solver has to explore more than 100,000 nodes. Furthermore, due to branching, several of the sub-problems may not satisfy the constraint qualification conditions and some of them may also be infeasible. NLP-BB may, therefore, not be a good strategy for solving problems with many integer variables or a weak integer relaxation. This is clearly shown in a recent solver benchmark presented in [123], where the pure NLP-BB solvers struggle for the problems with a large integer relaxation gap, and overall do not perform as well as the polyhedral outer approximation based solvers. However, NLP based branch and bound may have an advantage for strongly nonlinear problems and problems with a tight integer relaxation. Methods combining polyhedral outer approximations within a branch and bound framework have been proposed to avoid solving NLP problems at each node [167, 186]. Numerical results have shown that such methods are among the most efficient methods for solving convex MINLP problems [1, 38, 121, 123].

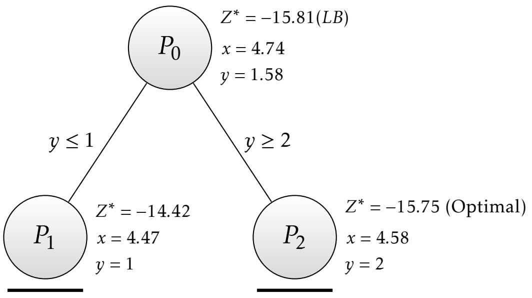

Figure 3.2: Branch and bound tree obtained for problem (ex2). The optimal objective values of the sub-problems are denoted by Z^*, and nodes resulting in integer solutions are underlined. The inequalities along the branches represent the branching restrictions added to consecutive sub-problems.

To illustrate the solution procedure with NLP-BB, consider the following example

$$\begin{aligned} \text{minimize} \quad & -3x - y \\ \text{subject to} \quad & x^2 + y^2 - 25 \leq 0, \\ & x^2 + (5-y)^2 - 36 \leq 0, \\ & (6-x)^2 + y^2 - 36 \leq 0, \\ & 0 \leq x, y \leq 10, \\ & x \in \mathbb{R}, y \in \mathbb{Z}. \end{aligned} \tag{ex2}$$

The basic features of problem (ex2) are illustrated in Figure 3.1, showing the constraints and the optimal solution. The solution procedure with the NLP-BB approach is shown in the branch and bound tree in Figure 3.1. Since the problem contains only one integer variable, only one branching step is needed to obtain the optimal solution. Solving the problem, thus, requires the solution of three convex NLP sub-problems.

3.3 Polyhedral outer approximations

In 1960 Kelley presented a method for solving convex NLP problems, based on iteratively improving polyhedral approximations of the nonlinear constraints [110]. By approximating the nonlinear constraints by linear constraints, Kelley was able to solve convex NLP problems as a sequence of LP sub-problems. Approximating nonlinear constraints with linear constraints is also an appealing approach for convex MINLP problems, since there are efficient solvers for MILP problems available.

Since polyhedral outer approximations are utilized in all the MINLP methods presented in Papers I–V, a detailed description of polyhedral outer approximations is well motivated. The main idea is to utilize linear approximations given by first-order Taylor series expansions. Such a linear approximation $\hat{g}$ of a nonlinear function g is given by

$$g(\mathbf{x}) \approx \hat{g}(\mathbf{x}) = g(\mathbf{x}^k) + \nabla g_j(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k), \tag{3.2}$$

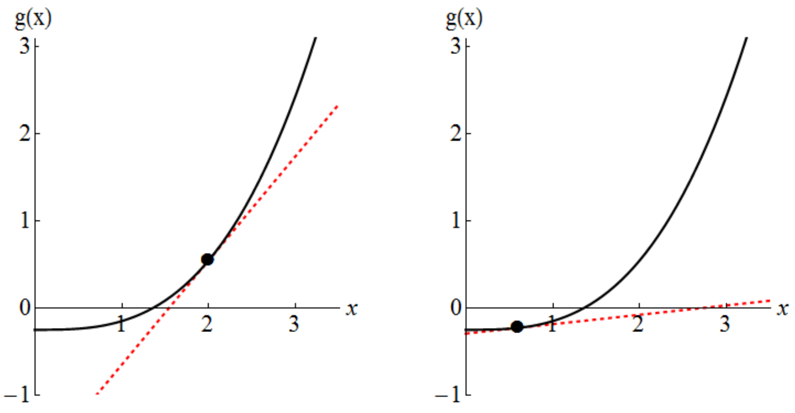

Figure 3.3: The black curves show the nonlinear function $g(x) = 0.1x^3 - 0.25$, and the dashed lines show first-order Taylor series expansion generated at $x = 2$ and $x = 0.5$. The dots in the figures represent the linearization points.

where $\mathbf{x}^k$ is the linearization point. If g is a convex function, then, as stated by Theorem 2, the linearization will be a valid underestimator of the function, *i.e.*, $\hat{g}(\mathbf{x}) \leq g(\mathbf{x})\ \forall \mathbf{x}$. A nonlinear constraint, given by $g(\mathbf{x}) \leq 0$, can then be approximated by a linear constraint by utilizing the linear approximation given by eq. (3.2), resulting in the following linear constraint

$$g(\mathbf{x}^k) + \nabla g_j(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) \leq 0. \tag{3.3}$$

Due to convexity, any point $\mathbf{x}$ that satisfies $g(\mathbf{x}) \leq 0$ will also satisfy the constraint given by eq. (3.3). However, all points satisfying the linearized constraint do not necessarily satisfy the nonlinear constraints. The linear constraint given by eq. (3.3), therefore, overestimates the feasible region of the original nonlinear constraint. To illustrate the overestimation property, consider the following nonlinear constraint

$$g(x) = 0.1x^3 - 0.25 \leq 0, \quad x \in \mathbb{R}_+. \tag{3.4}$$

The function g as well two different linearizations of the function are shown in Figure 3.3. From the figure, it is clear that linearizations underestimate the actual function, and that the linearized constraints generated according to eq. (3.3) results in an overestimation of the feasible region defined by the nonlinear constraint. The figure also illustrates the impact of the linearization point. Generating a linearized constraint at either $x = 2$ or $x = 0.5$, results in the limit $x \leq 1.54$ or $x \leq 2.71$, whereas the actual nonlinear constraint results in the limit $x \leq 1.36$.

To obtain an accurate approximation of the feasible region defined by a constraint, given by a multivariate nonlinear function, it is possible to generate multiple linearized constraints according to eq. (3.3) by using multiple linearizations points. In cases with several nonlinear constraints, linearizations can be generated individually for these constraints. Given a set of points $(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \ldots, (\mathbf{x}^k, \mathbf{y}^k) \in \mathbb{R}^n \times \mathbb{R}^m$, an approximation of the feasible set N defined by the nonlinear constraints in problem (P-MINLP) can be constructed according to

$$\widehat{N}_k = \left\{ (\mathbf{x},\mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m \;\middle|\; g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^T \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq 0 \;\; \forall j = 1,2,\ldots l, \forall i = 1,2,\ldots,k \right\}. \tag{3.5}$$

Since the functions g_j are assumed to be convex, $\widehat{N}_k$ gives an outer approximation of the set N. Furthermore, the approximations $\widehat{N}_k$ have the following property

$$N \subseteq \widehat{N}_k \subseteq \widehat{N}_{k-1} \subseteq \cdots \subseteq \widehat{N}_1, \tag{3.6}$$

which follows directly from the first-order convexity condition. Since $\widehat{N}_k$ is given by a set of linear constraints, we refer to $\widehat{N}_k$ as a polyhedral outer approximation.

An exact polyhedral outer approximation can be obtained using an infinite number of linearization points [100], and an arbitrary accurate polyhedral outer approximation can be obtained with a finite number of linearization points. However, practically we cannot use a huge number of linearization points because it would result in a MILP problem with an even larger number of constraints, and such a MILP problem could be computationally intractable. The goal is, thus, to construct a polyhedral outer approximation $\widehat{N}_k$ with only a small number of linear constraints, but accurate enough such that an optimal solution of problem (P-MINLP) can be obtained and verified by minimizing $\mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y}$ over $\widehat{N}_k \cap L \cap Y$.

The next sections describe different well-known methods for solving convex MINLP problems, by using polyhedral outer approximations. We begin with the extended cutting plane method; while it was not the first method, it is one of the most straightforward approaches for using and generating polyhedral outer approximations.

3.4 The extended cutting plane method

The extended cutting plane (ECP) method was presented by Westerlund and Petterson in 1995 [196] and can be viewed as an extension of Kelley's cutting plane method [110]. The ECP method was first presented as a method for solving convex MINLP problems but has later been further extended to handle both pseudo-convex functions [198] and non-smooth pseudo-convex functions[65]. The main idea of the ECP method is to solve a sequence of MILP sub-problems, where the nonlinear constraints are approximated by polyhedral outer approximations. The polyhedral outer approximation is iteratively improved by accumulating linearizations of the nonlinear constraints, which are here referred to as cutting planes. The cutting planes are generated at the trial solutions obtained by solving the MILP sub-problems, and exclude an infeasible trial solution from the search space.

The MILP sub-problem at iteration k can be defined as

$$\text{find} \quad (\mathbf{x}^k,\mathbf{y}^k) \in \underset{(\mathbf{x},\mathbf{y})\in\widehat{N}_k\cap L\cap Y}{\operatorname{argmin}} \; \mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y}. \tag{MILP-k}$$

The solution of the MILP sub-problem is used as a trial solution, but the solution also provides a lower bound on the optimum of the MINLP problem. Note that the sub-problem is minimizing the original objective function within a set containing the entire feasible region of the MINLP problem ,*i.e.*, $N \cap L \cap Y \subseteq \widehat{N}_k \cap L \cap Y$. A valid lower bound on the optimum of problem (P-MINLP) is, thus, given by $\mathbf{c}_1^T\mathbf{x}^k + \mathbf{c}_2^T\mathbf{y}^k$. If the trial solution $(\mathbf{x}^k, \mathbf{y}^k)$ satisfies all the nonlinear constraints, then it is also an optimal solution to problem (P-MINLP) and the search can be terminated.

If the trial solution $(\mathbf{x}^k, \mathbf{y}^k)$ does not satisfy the nonlinear constraints, then the polyhedral outer approximation can be improved by accumulating cutting planes according to

$$\widehat{N}_{k+1} = \left\{ (\mathbf{x},\mathbf{y}) \in \widehat{N}_k \;\middle|\; g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^T \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \le 0 \quad \forall j \in I_a \right\}, \tag{3.7}$$

where I_a contains the indexes of all violated constraints, or a subset of the violated constraints. In the first iteration the initial polyhedral outer approximation $\widehat{N}_0$ can, *e.g.*, be initialized as $\mathbb{R}^n$. The cuts generated according to eq. (4.6) improve the polyhedral approximation and exclude the infeasible trial solution $(\mathbf{x}^k, \mathbf{y}^k)$ from the search space, *i.e.*, $(\mathbf{x}^k, \mathbf{y}^k) \notin \widehat{N}_{k+1} \cap L \cap Y$. The procedure is repeated by solving sub-problem (MILP-k) with the improved polyhedral approximation $\widehat{N}_{k+1}$.

For convex MINLP problems, the ECP method generates a sequence of trial solutions with the following properties:

$$\begin{aligned} &\mathbf{c}_1^T\mathbf{x}^1 + \mathbf{c}_2^T\mathbf{y}^1 \le \mathbf{c}_1^T\mathbf{x}^2 + \mathbf{c}_2^T\mathbf{y}^2 \le \cdots \le \mathbf{c}_1^T\mathbf{x}^k + \mathbf{c}_2^T\mathbf{y}^k \le \mathbf{c}_1^T\mathbf{x}^* + \mathbf{c}_2^T\mathbf{y}^*, \\ &\lim_{k\to\infty} \left\| \begin{matrix} \mathbf{x}^k - \mathbf{x}^* \\ \mathbf{y}^k - \mathbf{y}^* \end{matrix} \right\| = 0, \end{aligned} \tag{3.8}$$

where $(\mathbf{x}^*, \mathbf{y}^*)$ denotes an optimal solution of the MINLP problem. A solution within an arbitrarily small tolerance can, thus, be obtained within a finite number of iterations.

To illustrate how the ECP method refines the polyhedral outer approximation, the basic ECP method is applied to problem (ex2). The search is initialized by defining $\widehat{N}_0 = \mathbb{R}^2$, and the first MILP sub-problem then minimizes the objective function within the variable bounds. The first solution, $x^1 = 10, y^1 = 10$, violates all the nonlinear constraints, and we choose the approach of generating cutting planes for all violated constraints. Generating cuts for all violated constraints usually results in fewer iterations at the expense of somewhat more difficult sub-problems. The iterative solution procedure is illustrated in Figure 4.4 and shows the six first iterations. Obtaining a solution that satisfies all the constraints, within a tolerance of 10^{-6}, requires nine iterations.

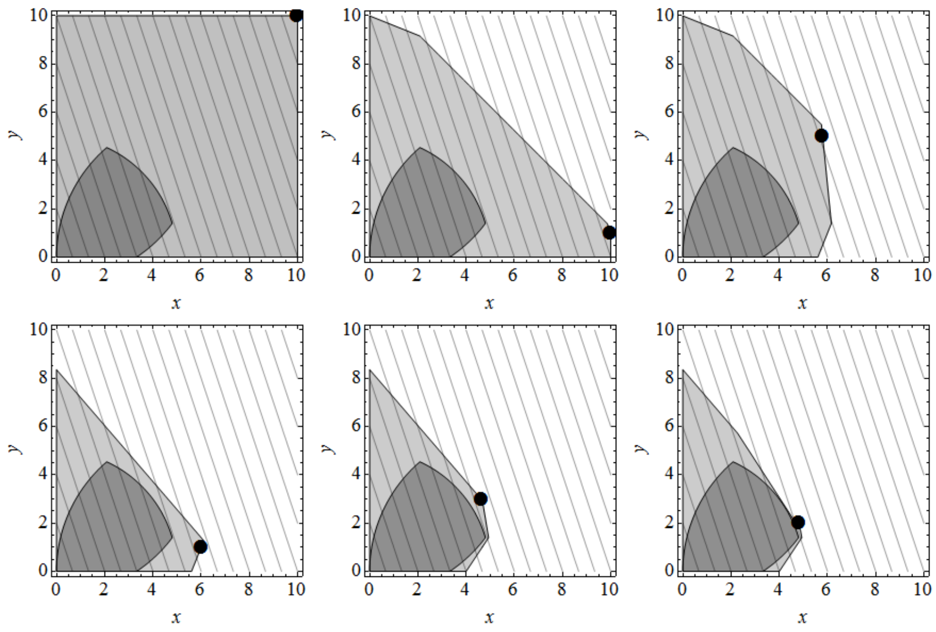

Figure 3.4: The figures show the first six iterations, out of the nine iterations needed, to solve problem (ex2) with the ECP method. The dark gray areas represent the feasible region defined by the nonlinear constraints, the light gray areas represent the polyhedral outer approximations, the lines represent contours of the objective function, and the dots represent an optimal solution of the MILP sub-problems.

3.5 Outer approximation

Duran and Grossmann first presented the outer approximation (OA) algorithm in 1986 [63]. In its original form, OA was restricted to convex MINLP problems that are linear in the integer variables. Later in 1994, Fletcher and Leyffer [69] showed that linearity of the integer variables is not needed for achieving finite convergence, and they proposed a different approach for dealing with infeasible integer combinations. For coping with non-convex problems more efficiently, Kocis and Grossmann proposed an equality relaxation technique in 1987 [116, 117], and two years later Wiswanathan and Grossmann proposed a penalty approach for relaxing the constraints [192].

The OA algorithm can be viewed as a decomposition technique, which decomposes the problem into linear problems to deal with the integer requirements and into convex nonlinear problems to deal with the nonlinear constraints. The iterative trial solutions are, thus, obtained by solving an alternating sequence of MILP and convex NLP sub-problems. Besides the assumptions made in Section 3.1, OA also requires the following assumption of the MINLP problem to be true.

Assumption 4. By fixing the integer variables to a feasible integer combination, *i.e.*, an integer combination corresponding to a feasible solution of the MINLP problem, the resulting NLP problem satisfies a constraint qualification, *e.g., Slater's condition.*

The assumption is needed to ensure that all feasible integer combinations obtained result in a point satisfying the KKT conditions. The feasible solutions obtained must satisfy the KKT conditions to ensure that the added linearizations prevent cycling.

Like the name implies, the OA algorithm also utilizes polyhedral outer approximations $\widehat{N}_k$ to construct a linear approximation of the MINLP problem. The polyhedral outer approximations $\widehat{N}_k$ are constructed as described previously according to eq. (4.6), by accumulating first-order Taylor series expansions (linearizations) of the nonlinear constraints, and the set $\widehat{N}_k$ is used to construct a MILP relaxation of the original problem. The MILP sub-problems are often referred to as the MILP-master problems and these are given by problem (MILP-k). However, the trial solutions $(\mathbf{x}^k, \mathbf{y}^k)$ are here obtained by a two-step procedure where the integer and continuous variables are chosen separately, and only integer variables are chosen by problem (MILP-k).

At iteration k, the MILP-master problem (MILP-k) is solved to obtain the integer variables $\mathbf{y}^k$. The integer variables in the original MINLP problem are then fixed as $\mathbf{y}^k$, resulting in the following convex NLP sub-problem

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} \\ \text{subject to} \quad & \mathbf{y} = \mathbf{y}^k, \\ & (\mathbf{x}, \mathbf{y}) \in L \cap N. \end{aligned} \tag{NLP-k}$$

Suppose that problem (NLP-k) is feasible with the given integer combination $\mathbf{y}^k$. Then, the optimal solution of problem (NLP-k) is also a feasible solution of the original MINLP problem and is saved as $(\mathbf{x}^k, \mathbf{y}^k)$. Since $(\mathbf{x}^k, \mathbf{y}^k)$ is a feasible solution of the MINLP problem, $\mathbf{c}_1^T \mathbf{x}^k + \mathbf{c}_2^T \mathbf{y}^k$ provides a valid upper bound on the optimal objective value. A lower bound on the optimal objective value is, as before, provided by the optimal solution of the MILP-master problem (MILP-k). If the lower and upper bound is not within the desired tolerance, the polyhedral outer approximation can be improved by generating linearizations at $(\mathbf{x}^k, \mathbf{y}^k)$ according to eq. (4.6). The trial solution $(\mathbf{x}^k, \mathbf{y}^k)$ is now located on the boundary of the set N, and therefore, at least one of the linearized constraints will form a supporting hyperplane to the set N. The set $\widehat{N}_k$ can be updated by generating linearizations for all nonlinear constraints or by the constraints active at $(\mathbf{x}^k, \mathbf{y}^k)$. Only generating linearizations for active constraints results in smaller sub-problems, but might result in a weaker outer approximation.

The convex NLP problem (NLP-k) may also be infeasible with the given integer combination. The original approach for dealing with such a situation was to derive an integer cut to exclude that specific integer combination [63]. The integer cut is not necessarily a strong cut since it only excludes a specific integer combination. A more efficient approach for dealing with an infeasible integer combination was presented in [69], which is based on solving a feasibility problem. The feasibility problem minimizes the violation of the nonlinear constraints with the given integer combination, *e.g.*, with respect to the ℓ_∞ norm it can be defined as

$$\begin{aligned}
\text{minimize} \quad & r \\
\text{subject to} \quad & g_j(\mathbf{x},\mathbf{y}) \leq r \quad \forall j = 1,2,\dots m, \\
& \mathbf{y} = \mathbf{y}^k, \\
& (\mathbf{x},\mathbf{y}) \in L, \ r \in \mathbb{R}_+.
\end{aligned} \tag{NLPf-k}$$

The feasibility problem is obviously convex and satisfies Slater's condition. The feasibility problem can, thus, be solved efficiently and the optimal solution is denoted $(\mathbf{x}^k, \mathbf{y}^k)$. The polyhedral outer approximation can now be updated by generating linearizations at $(\mathbf{x}^k, \mathbf{y}^k)$ according to eq. (4.6). Now, the point $(\mathbf{x}^k, \mathbf{y}^k)$ will be located outside of the feasible region. The linearizations will therefore in general not form supporting hyperplanes to the feasible set, but instead form cutting planes. These cutting planes are sufficient for excluding the infeasible integer combination $\mathbf{y}^k$ from the search space, and unlike the integer cut, they may also exclude other infeasible integer combinations [69]. Unlike the ECP method, OA will not obtain the same integer combination twice unless it is proven as optimal and the search is terminated.

The OA algorithm is usually initialized by solving an integer relaxation of the original MINLP problem. The relaxed solution provides an initial lower bound on the objective, and the initial polyhedral outer approximation $\widehat{N}_0$ is constructed by generating linearizations at the integer relaxed solution. The procedure of solving the MILP-master problem and one of the NLP sub-problems is repeated until the gap between the upper and the lower bound on the optimum of the MINLP problem is within the desired tolerance. Compared to the ECP method, the OA algorithm usually obtains tighter polyhedral outer approximation as well as an upper bound, which is not obtained before the very last iteration with the basic ECP method. As a result, the OA algorithm usually requires fewer iterations for convex MINLP problems than the ECP method. However, each iteration in the OA algorithm is computationally more demanding than an ECP iteration, and therefore, they are difficult to compare directly by the number of iterations.

To illustrate the solution procedure with the OA algorithm, the algorithm is applied to problem (ex2) from Section 3.2. Here we update the polyhedral outer approximation by only generating linearizations of the active constraints. The solution procedure is illustrated in Figure 3.5, showing the three iterations needed to solve the problem. The initial polyhedral outer approximation is constructed as described by solving the integer relaxed problem, and Figure 3.5 shows that the polyhedral approximation is quite tight already in the first iteration. For this simple example, we did not encounter any infeasible integer combinations. The optimal solution of the MINLP problem is obtained in the second iteration, although a third iteration is needed to verify optimality.

The main drawback of using linearizations for describing the nonlinear constraints is the fact that they only provide an accurate approximation in the neighborhood of the linearization points. Paper IV presents a technique for using the concept of trust regions and second order derivatives within the OA algorithm to choose the integer

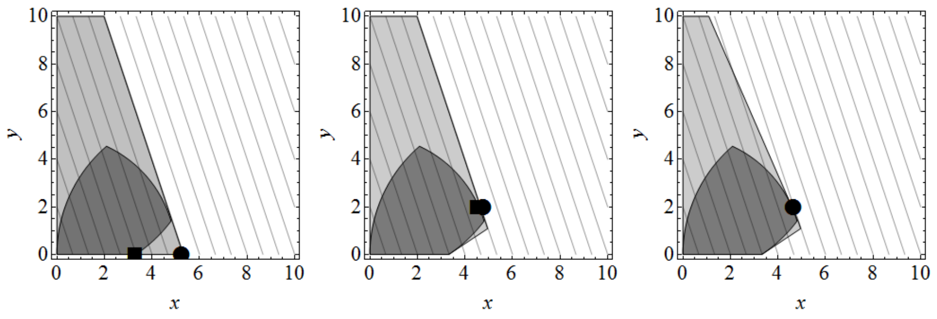

Figure 3.5: The figures show the three iterations needed to solve problem (ex2) with the OA algorithm. The dark gray areas represent the feasible region defined by the nonlinear constraints, the light gray areas represent the polyhedral outer approximations, and the lines represent contours of the objective function. The circular dots represent the solution to the MILP-master problems, and the squared dots represent the solution of the NLP sub-problem.

combinations more carefully and to obtain faster convergence. These techniques are described in more detail in Chapter 4.

3.6 Generalized Benders decomposition

Benders decomposition is a partitioning procedure for solving mixed-integer problems, proposed by Benders [20] in 1962. The partitioning procedure is intended to exploit special structures of optimization problems with complicating variables. Usually, a subset of the variables is considered to be complicating variables if temporarily fixing them results in a significantly more tractable optimization problem. Fixing some of the variables can, *e.g.*, result in a problem that can be separated into smaller individual problems that can be solved in parallel. Benders decomposition was later generalized by Geoffrion [83] to cover a broader class of problems, and this method is known as generalized Benders decomposition (GBD).

When dealing with convex MINLP problems, the integer variables are usually seen as the complicating variables, *i.e.*, fixing these results in a convex NLP problem. Similar to OA, GBD will in each iteration solve a master problem to obtain a trial solution for the integer variables and solve a convex NLP problem to obtain the corresponding continuous variables. Under the assumption that the integer variables occur linearly in the problem, it has been proven that GBD is equivalent to merging all linearizations added in an OA iteration into a single linearization using the Lagrangian multipliers [167]. The main difference between GBD and OA is in the master problem; the master problem in GBD is an optimization problem only in the complicating variables of the original problem [90]. Since the goal here is to solve convex MINLP problems, it is natural to choose the integer variables as complicating variables. Furthermore, we would like the sub-problems to be MILP problems and convex NLP problems since we have efficient solvers for such problems.

GBD has also been considered as one approach for dealing with some types of non-convex problems, and there are different variants of GBD [72]. However, applying GBD to non-convex problem is not trivial, and may not result in optimal solutions, as shown in [174]. Further generalizations of GBD have been proposed by [71, 200], and a somewhat similar approach to GBD was described in [10]. Here we will only consider the basic version of GBD; for a more comprehensive description we refer to, *e.g.*, [72].

Besides the assumptions needed with OA, GBD requires the following assumption of the functions g_j to be true.

Assumption 5. The nonlinear functions g_j are linearly separable in the continuous and integer variables, *i.e.*,

$$g_j(\mathbf{x},\mathbf{y}) = g_{1,j}(\mathbf{x}) + g_{2,j}(\mathbf{y}) \quad \forall j.$$

The need for linear separability will become clear later on. However, linear separability of the continuous and integer variables is not necessarily restrictive.

Remark. By reformulating a convex MINLP problem we can always obtain a problem where the nonlinear functions are linearly separable in the two set of variables. Suppose we have the nonlinear constraint $g_j(\mathbf{x},\mathbf{y}) \leq 0$, which is not linearly separable. Furthermore, suppose it is not separable due to the integer variable y_1. Then we can introduce a new continuous variable x_{n+1}, and replace the integer variable y_1 by the new continuous variable in the function g_j. Replacing y_1 by x_{n+1}, makes the function g_j linearly separable in the the continuous and integer variables. To make the reformulated problem equivalent to the original problem we introduce a new linear equality constraint, namely, $x_{n+1} = u_1$. This reformulation does not reduce the complexity of the problem by any means but shows that a convex MINLP problem can always be reformulated to obtain the desired separability.

One of the key components of GBD is the projection of problem (P-MINLP) onto the $\mathbf{y}$-space. An equivalent problem in the $\mathbf{y}$-space can be written as

$$\min_{\mathbf{y}\in Y\cap V} v(\mathbf{y}) \tag{3.9}$$

where

$$\begin{aligned} v(\mathbf{y}) = \min_{\mathbf{x}} \quad & \mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y} \\ \text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}) \leq 0, \quad \forall j = 1,2,\ldots l, \\ & \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} \leq \mathbf{b}, \\ & \mathbf{y} \text{ is fixed.} \end{aligned} \tag{3.10}$$

The set V in eq.(3.9) is defined as

$$V = \left\{\mathbf{y} \mid \mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} \leq \mathbf{b},\ g_j(\mathbf{x},\mathbf{y}) \leq 0 \ \ \forall j = 1,2,\ldots l \quad \text{for some } \mathbf{x}\right\}. \tag{3.11}$$

The set V is a convex set, and the function v is convex, due to the convex properties of the original problem [42]. The projected problem (3.9) is, thus, a convex pure integer problem. However, unfortunately, neither the function v nor the set V are known explicitly; they are only known through their definitions. Therefore, we cannot directly solve the projected problem (3.9). Instead, we will use a procedure similar to OA to construct an iteratively improving approximation of problem (3.9). To construct the approximation, problem (3.9) is rewritten using a dual representation of V and v into the equivalent problem

$$\begin{aligned}
\min_{\mathbf{y}\in Y,\alpha\in\mathbb{R}} \quad & \alpha \\
\text{s.t.} \quad & \alpha \geq \min_{\mathbf{x}} \mathcal{L}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda},\boldsymbol{\mu}), \quad \forall \boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\mu} \geq \mathbf{0}, \\
& 0 \geq \min_{\mathbf{x}} \bar{\mathcal{L}}(\mathbf{x},\mathbf{y},\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{\mu}}), \quad \forall \bar{\boldsymbol{\lambda}} \geq \mathbf{0}, \bar{\boldsymbol{\mu}} \geq \mathbf{0},
\end{aligned} \tag{3.12}$$

where

$$\begin{aligned}
\mathcal{L}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda},\boldsymbol{\mu}) &= \mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y} + \sum_{j=1}^{l} \lambda_j g_j(\mathbf{x},\mathbf{y}) + \boldsymbol{\mu}^T(\mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} - \mathbf{b}), \\
\bar{\mathcal{L}}(\mathbf{x},\mathbf{y},\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{\mu}}) &= \sum_{j=1}^{l} \bar{\lambda}_j g_j(\mathbf{x},\mathbf{y}) + \bar{\boldsymbol{\mu}}^T(\mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} - \mathbf{b}).
\end{aligned} \tag{3.13}$$

For more details on how to obtain problem (3.12) see, *e.g.*, [72, 83].

Problem (3.12) cannot be solved directly, since it contains an infinite number of constraints. Furthermore, each constraint contains an individual optimization problem which is parametric in the integer variables. The goal is, therefore, to relax problem (3.12) by only considering a finite number of explicit constraints. The constraints of the relaxed problem should preferably be linear in the integer variables, thus, resulting in a MILP problem. To simplify the notation we will introduce the following functions

$$\xi(\mathbf{y},\boldsymbol{\lambda},\boldsymbol{\mu}) = \min_{\mathbf{x}} \mathcal{L}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda},\boldsymbol{\mu}), \tag{3.14a}$$

$$\bar{\xi}(\mathbf{y},\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{\mu}}) = \min_{\mathbf{x}} \mathcal{L}(\mathbf{x},\mathbf{y},\bar{\boldsymbol{\lambda}},\bar{\boldsymbol{\mu}}). \tag{3.14b}$$

An iteratively improving approximation of problem (3.12) can be constructed using a similar approach as in OA. Assume that we have obtained a feasible integer solution $\mathbf{y}^k$. By fixing the integer variables of the MINLP problem as $\mathbf{y}^k$ and solving the resulting problem (NLP-k), we obtain the corresponding optimal continuous variables $\mathbf{x}^k$ as well as the optimal multipliers $\boldsymbol{\lambda}^k$ and $\boldsymbol{\mu}^k$. By utilizing the linear separability of the functions

g_j, we obtain

$$\begin{aligned}
\xi(\mathbf{y},\boldsymbol{\lambda}^k,\boldsymbol{\mu}^k) &= \min_{\mathbf{x}} \mathcal{L}(\mathbf{x},\mathbf{y},\boldsymbol{\lambda}^k,\boldsymbol{\mu}^k) \\
&= \min_{\mathbf{x}} \left(\mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y} + \sum_{j=1}^{l} \lambda_j^k g_j(\mathbf{x},\mathbf{y}) + (\boldsymbol{\mu}^k)^T(\mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{y} - \mathbf{b}) \right) \\
&= \mathbf{c}_2^T\mathbf{y} + \sum_{j=1}^{l} \lambda_j^k g_{2,j}(\mathbf{y}) + (\boldsymbol{\mu}^k)^T\mathbf{A}_2\mathbf{y} + \min_{\mathbf{x}} \left(\mathbf{c}_1^T\mathbf{x} + \sum_{j=1}^{l} \lambda_j^k g_{1,j}(\mathbf{x}) + (\boldsymbol{\mu}^k)^T(\mathbf{A}_1\mathbf{x} - \mathbf{b}) \right) \\
&= \mathbf{c}_2^T\mathbf{y} + \sum_{j=1}^{l} \lambda_j^k g_{2,j}(\mathbf{y}) + (\boldsymbol{\mu}^k)^T\mathbf{A}_2\mathbf{y} + \mathbf{c}_1^T\mathbf{x}^k + \sum_{j=1}^{l} \lambda_j^k g_{1,j}(\mathbf{x}^k) + (\boldsymbol{\mu}^k)^T(\mathbf{A}_1\mathbf{x}^k - \mathbf{b}).
\end{aligned} \tag{3.15}$$

The optimization problem in (3.14a) is, thus, independent of the integer variables for a fixed set of multipliers. Therefore, by solving problem (NLP-k) we obtain one of the constraints of problem (3.12) in explicit form.

If an integer combination $\mathbf{y}^k$ is infeasible, then the continuous feasibility problem (NLPf-k) provides the continuous variables $\mathbf{x}^k$ minimizing the constraint violation as well as the multipliers $\bar{\boldsymbol{\lambda}}^k$ and $\bar{\boldsymbol{\mu}}^k$. Similar to the case with a feasible integer combination, we get

$$\bar{\xi}(\mathbf{y},\bar{\boldsymbol{\lambda}}^k,\bar{\boldsymbol{\mu}}^k) = \sum_{j=1}^{l} \bar{\lambda}_j^k g_{2,j}(\mathbf{y}) + (\bar{\boldsymbol{\mu}}^k)^T\mathbf{A}_2\mathbf{y} + \sum_{j=1}^{l} \bar{\lambda}_j^k g_{1,j}(\mathbf{x}^k) + (\bar{\boldsymbol{\mu}}^k)^T(\mathbf{A}_1\mathbf{x}^k - \mathbf{b}). \tag{3.16}$$

With given multipliers, the functions $\bar{\xi}$ and ξ can now be written as functions of only the integer variables. Once an integer solution is obtained, a constraint of problem (3.12) can be obtained by solving a convex NLP problem. A relaxation of problem (3.12) containing a finite number of explicit constraint is, thus, given by

$$\begin{aligned}
\min_{\mathbf{y}\in Y,\alpha\in\mathbb{R}} \quad & \alpha \\
\text{s.t.} \quad & \xi(\mathbf{y},\boldsymbol{\lambda}^k,\boldsymbol{\mu}^k) \le \alpha, \quad \forall k \in K_f \\
& \bar{\xi}(\mathbf{y},\bar{\boldsymbol{\lambda}}^k,\bar{\boldsymbol{\mu}}^k) \le 0, \quad \forall k \in K \backslash K_f.
\end{aligned} \tag{3.17}$$

Here K_f is an index set with the indices of all iterations in which the integer combination was feasible, and K contains the indices of all iterations. Problem (3.17) is sometimes referred to as the relaxed master problem. A new integer combination $\mathbf{y}^{k+1}$ can be obtained by solving problem (3.17), and a new constraint is generated as described and included in the next iteration. Since problem (3.17) is a relaxation of problem (3.12), its optimal solution also gives a valid lower bound on the optimal solution to the original MINLP problem. Similar to OA, an upper bound is provided by the feasible NLP problems. The procedure is repeated until the upper and lower bounds are within the desired tolerance.

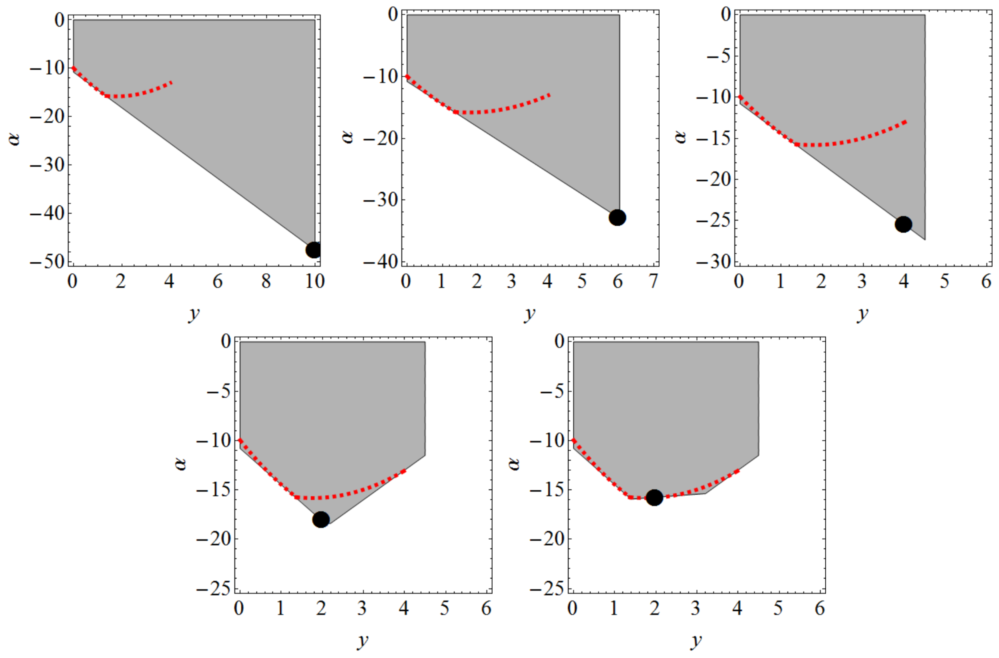

Figure 3.6: The figures show the five iterations needed to solve problem (ex2) with the described GBD algorithm. The dashed curve shows the function v defined by eq. (3.10) within the feasible region. The gray areas represent the feasible region of the relaxed master problem, and the dots represent the minimizer of the relaxed master problem in each iteration. Here, the horizontal axis corresponds to the integer variable y, and the vertical axis shows the α variable.

For MINLP problems were the functions $g_{2,j}$ are all linear, then the relaxed master problem will be a MILP problem. Otherwise, each iteration will require the solution of a convex MINLP problem. For problems with a special structure it might be beneficial to decompose a convex MINLP problem into a sequence of convex MINLP sub-problems but, in general, it is not a desirable approach. The relaxed master problem (3.17) should preferably be linear, such that an efficient MILP solver can be used to solve the problem in each iteration. Since the functions $g_{2,j}$ are all convex, the obvious approach is to approximate them by first-order Taylor series expansions as mentioned in [91, 200].

To illustrate GBD, we have applied the basic algorithm to problem (ex2). The procedure is started at the feasible integer solution $y = 1$, and the nonlinear term in the constraints of the relaxed master problem is approximated by first-order Taylor series expansions in each iteration. The iterative solution procedure is illustrated in Figure 3.6, showing the five iterations needed to solve the problem. The optimal solution is obtained in iteration four; however, verifying optimality requires an additional iteration.

Compared to OA, GBD tends to give weaker lower bounds in each iteration. As mentioned, if the integer variables occur linearly in the problem, the GBD cuts are the equivalent of merging the OA cuts using the Lagrangian multipliers [167]. In such a case it can be proven that OA will never result in weaker bounds than GBD [90]. However, the relaxed master problem in GBD will, in general, contain fewer variables and constraints than the master problem in OA and, thus, the algorithms are difficult to compare directly.

A technique to obtain tighter polyhedral approximations, and tighter lower bounds, within a GBD framework was proposed by Quesada and Grossmann in 1992 [167] and is often referred to as partial surrogate cuts (PSC). With this approach, the continuous variables are classified into linear or nonlinear based on their properties in the original MINLP problem. By only projecting out the nonlinear continuous variables, one can derive Lagrangian cuts similar to the GBD cuts while keeping all the linear constraints of linear continuous variables in the relaxed master problem. It was proven in [167] that the PSC procedure results in a tighter linear relaxation compared to GBD while still adding only one cut per iteration.

3.7 LP/NLP-based branch and bound

When applying the OA algorithm on a convex MINLP problem, the majority of the total solution time is, usually, spent on solving the MILP master problems. A new approach to avoid solving multiple MILP sub-problems was presented by Quesada and Grossmann in 1992 [167]. The method, known as LP/NLP-based branch and bound (LP/NLP-BB), integrates OA within a branch and bound framework. Compared to OA, only one branch and bound tree is generated by dynamically updating the MILP master problem.

The search is initialized by solving an integer relaxation of the MINLP problem, and the solution is used to construct an initial polyhedral outer approximation. The integer relaxation also provides a valid lower bound on the optimal objective value of the MINLP problem. The polyhedral outer approximation is used to construct the initial MILP master problem, which will be solved by a branch and bound procedure. An LP relaxation is solved in each node of the branch and bound tree, and the search is stopped once an integer solution is obtained in one of the nodes. The integer solution is treated as normal in OA, by solving an NLP problem with the integer variables fixed. If it results in a feasible NLP problem, then it provides a valid upper bound, and new linearizations can be generated. If the NLP problem is infeasible, it is possible to add an integer cut to exclude the solution or solve the feasibility problem and generate new linearizations. The new linear constraints are then added to all open nodes in the branch and bound tree, and the LP relaxation is resolved for the node which resulted in the integer combination. The search continues with the improved polyhedral outer approximation, and the BB procedure continues from the existing search tree. As normally done in BB, nodes can be pruned off in case the optimum of the LP relaxation exceeds the upper bound. However, the search cannot be stopped once an integer solution is obtained at a node; the search must continue until the LP relaxation results in a feasible integer solution or until the node can be pruned off.

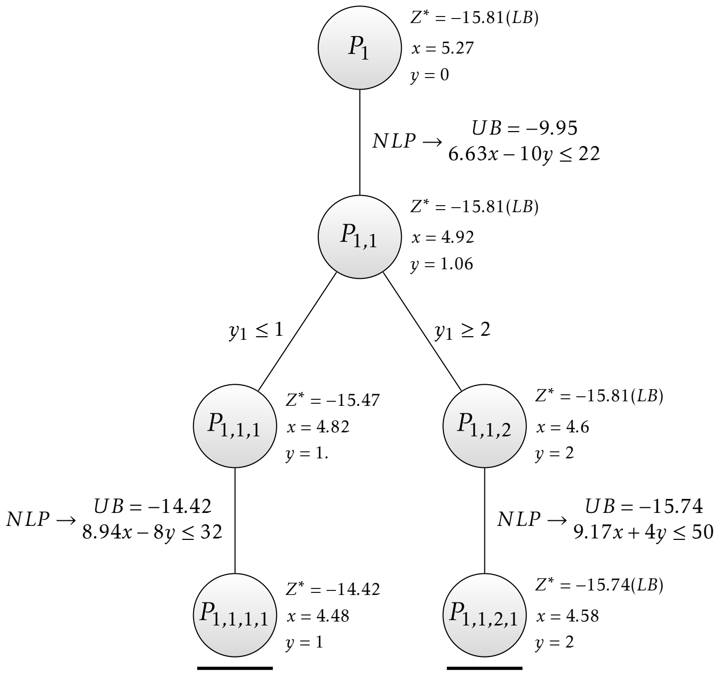

Figure 3.7: The branch and bound tree obtained with LP/NLP-BB for problem (ex2). The optimal objective value of the LP relaxations are denoted by Z^*, and the solutions are given next to the nodes. The upper bounds obtained by the NLP sub-problems as well as the new linear constraints are shown in between the nodes, where an NLP problem was solved. Nodes where the LP relaxation returns an integer solution that also satisfies the nonlinear constraints are underlined.

To illustrate LP/NLP-BB, the algorithm is applied to problem (ex2). The solution procedure is illustrated in Figure 3.7, showing the explored nodes. An integer solution is obtained already at the first node, but after improving the polyhedral approximation it results in a fractional solution. In total, six LP relaxations must be solved, three NLP problems with fixed integer variables, and an initial integer relaxation. One of the nodes and one NLP problem could be avoided in this case by first exploring the right side of the search tree.

By only constructing a single branch and bound tree, LP/NLP-BB may end up exploring fewer nodes than the total number of nodes explored in the multiple branch and bound trees in normal OA. The LP/NLP-BB can also be combined with cut generating procedures for tightening the integer relaxation. Generally, LP/NLP-BB is considered one of the most efficient techniques for convex MINLP [1, 38, 131, 143], which is also supported by a recent benchmark test of convex MINLP solvers in [123]. The SHOT solver, which is described in Chapter 4, can utilize a similar approach as LP/NLP-BB to construct only a single branch and bound tree with the ESH algorithm.

3.8 Other techniques for convex MINLP

Most solvers are not limited to a single method but combine several techniques to improve their practical performance. Solvers utilize a variety of different techniques, and only the most relevant ones for this thesis are mentioned here. For more details on techniques frequently used by MINLP solvers see, *e.g.*, [19].

Primal heuristics

Algorithms intended to quickly obtain good feasible solutions to optimizations problems are commonly referred to as primal heuristics. Primal heuristics were first developed within MILP, and have proven to be an important tool for improving the efficiency of MILP solvers [67]. Knowing a good feasible solution can, *e.g.*, reduce the size of the branch and bound tree.

Different primal heuristics have also been proposed for MINLP problems, such as the feasibility pump [39], rounding heuristics [23], and undercover [24]. A new primal heuristic for convex MINLP, called the center-cut algorithm, is presented in Paper V and is described in Chapter 7. In solvers based on branch and bound, the primal heuristics can also reduce the size of the branch and bound tree for convex MINLP problems. Primal heuristics are also important in solvers based on the ECP or ESH algorithm, since these algorithms do not necessarily obtain a feasible solution before the last iteration. Even if these techniques are referred to as heuristics, some of them are guaranteed to find a feasible solution to a convex MINLP problem in a finite number of iterations. More details on primal heuristics for MINLP problems are, *e.g.*, given in [22] and [55].

Preprocessing

By analyzing an optimization problem, it may be possible to slightly modify the problem to obtain a problem that might be significantly easier to solve. Such procedures are often referred to as preprocessing, or presolving, techniques. Performing preprocessing on a convex MINLP problem can, for example, reduce the problem size or result in tighter relaxations and significantly reduce the total solution time [19].

Bound tightening is a commonly-used preprocessing technique intended to reduce the variable bounds. The most commonly used bound tightening methods are feasibility-based bound tightening [18, 179], and optimization-based bound tightening [135]. Optimization-based bound tightening solves a sequence of relaxed problems, minimizing and maximizing each variable to obtain tighter variable bounds. In feasibility-based bound tightening, the constraints are analyzed, either individually or in groups, to obtain tighter bounds [17, 18, 147]. Several range reduction techniques to reduce the search space were presented in [171].

Reformulation techniques are intended to modify the problem to obtain tighter relaxations while ensuring that the optimal solution to the original problem can be obtained by solving the reformulated problem. For example, disaggregation of some types of linear constraints can result in a tighter integer relaxation [19, 201]. By introducing new continuous variables, it is possible to split some types of nonlinear constraints into several new constraints, which can result in tighter lifted polyhedral outer approximation [183]. Some reformulations for obtaining tighter lifted polyhedral approximations are presented in Paper III, which were shown to have a great impact on several solvers. The reformulations in Paper III are described in more detail in Chapter 5. Lifted reformulations of MINLP problems are also studied in [99, 137].

Only a few preprocessing techniques have been mentioned here, and there exist several other important techniques. A detailed survey of several preprocessing techniques for MINLP is, *e.g.*, given by [19].

3.9 Software

There are several modeling systems for dealing with MINLP problems, such as AIMMS [33], AMPL [78], and GAMS [43]. Lately, there has been a growing interest in modeling and solving optimization problems within the programming languages Python [189] and Julia [29]. Pyomo is a Python based open-source modeling language which supports a wide range of problems types, including MINLP [97]. JuMP is a similar efficient modeling language embedded in Julia [62]. For more details on different modeling systems for mathematical optimization see, *e.g.*, [107].

There are several solvers, both commercial and open-source, available for convex MINLP problems. Reviews of commonly-available solvers are, *e.g.*, given in [19, 44, 188], and a comprehensive comparison of solvers for convex MINLP is given in [123]. Only a few of the best-known solvers are mentioned here, and for more details on convex MINLP solvers see, *e.g.*, [123].

Solvers that are capable of solving convex MINLP problems within GAMS are: AlphaECP [127, 197], ANTIGONE [151], BARON [172, 173], BONMIN [38], Couenne [16], DICOPT [92], Knitro [45], LINDO [106], SBB [80], and SCIP [2, 191]. Several of these solvers are also available within AIMMS, along with the AOA solver [103]. Some of these solvers are also available within both AMPL and Pyomo. Pajarito [137] and Juniper [120] are examples of solvers developed within the Julia language and available in JuMP.

A new solver for convex MINLP, called the Supporting hyperplane toolkit (SHOT), is presented in Paper VI. SHOT was first mentioned in Paper I, and builds upon the ESH algorithm. The solver is described in more detail in Section 4.5.

3.10 Conclusion

This chapter has given a summary of work previously done within the field of convex MINLP. The most commonly-used methods for solving convex MINLP problems have been presented along with an illustrative example to visualize the solution procedures. With the information presented in this chapter, we are now ready to begin with the main contributions of this thesis.

Chapter 4

Contributions to Convex MINLP

The previous chapter was intended as a summary of work done within the field of convex MINLP, and to present the background material needed to fully understand the methods presented here. Some of the techniques described in Chapter 3 will also be used in the methods presented in this chapter, and it is assumed that the reader is acquainted with the MINLP methods from the previous chapter. As mentioned in the introduction, the primary research goal has been to investigate how to efficiently construct and utilize polyhedral approximations to solve convex MINLP problems. This chapter describes the methods presented in the papers and gives a summary of the results.

4.1 The extended supporting hyperplane algorithm (Papers I–II)

The main idea for the extended supporting hyperplane (ESH) algorithm, grew from observations that the cutting planes generated by the ECP method are not, in general, as tight as possible. Figure 4.4, in Section 3.4, clearly shows that it would be possible to generate more efficient linearizations, resulting in tighter polyhedral outer approximations. Obtaining tighter polyhedral outer approximations could, for example, reduce the number of iterations needed to solve the optimization problem. The goal was to come up with an efficient technique for generating tighter polyhedral outer approximations of the integer relaxed feasible region of convex MINLP problems. A quite detailed description of the ESH method is given here since the algorithm has a central role in three of the papers included in this thesis.

Since the linearizations are generated by first-order Taylor series expansion of the nonlinear functions, they are only guaranteed to give an exact approximation in the linearization points, as illustrated in figure 3.3. Due to convexity, the linearizations underestimate the actual function, and the underestimation error tends to grow with the distance from the linearization point. If the linearization is generated outside of the integer relaxed feasible region, the function value of the linearization may be significantly smaller than the function value of the nonlinear functions within the entire integer relaxed feasible region. The resulting linearized constraint may therefore not be as tight as possible if the linearization is generated outside of the integer relaxed feasible region. In order to obtain the tightest possible polyhedral outer approximation of the integer relaxed feasible set, we would like to use supporting hyperplanes of the set to generate the polyhedral approximation. For the sake of completeness and clarity, the definition of a supporting hyperplane has been included.

Definition 8. A supporting hyperplane of a convex set N, in $\mathbb{R}^n$, is a hyperplane with the following properties: the set N is contained entirely within one of the closed half-spaces defined by the hyperplane, and the set N has at least one boundary point on the hyperplane.

An illustration of a supporting hyperplane to a convex set is given in figure 4.1.

The trial solutions will be chosen in this case by the same approach as with ECP method, and they are, thus, chosen as a minimizer of the linear objective function within $\widehat{N}_k \cap L \cap Y$. The main difference between the ESH and ECP algorithms is the technique used for generating linearizations, and updating the polyhedral outer approximation $\widehat{N}_k$. As before, the set N is defined by the nonlinear constraints, L by the linear constraints, and Y by the integer requirements. The trial solution obtained by solving sub-problem (MILP-k) are here denoted as $(\mathbf{x}^k_{\text{MILP}}, \mathbf{y}^k_{\text{MILP}})$. As in the ECP method, the trial solutions provide an iteratively improving lower bound on the optimum of the convex MINLP problem, but they will directly be used for generating the linearizations.

Given a trial solution $(\mathbf{x}^k_{\text{MILP}}, \mathbf{y}^k_{\text{MILP}})$, located outside the set N, it is possible to generate a supporting hyperplane to the integer relaxed feasible set, $N \cap L$, by projecting $(\mathbf{x}^k_{\text{MILP}}, \mathbf{y}^k_{\text{MILP}})$ onto $N \cap L$. The trial solution can be projected onto $N \cap L$ by solving the convex NLP problem

$$(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \underset{(\mathbf{x},\mathbf{y}) \in N \cap L}{\text{argmin}} \quad \left\| \begin{matrix} \mathbf{x}^k_{\text{MILP}} - \mathbf{x} \\ \mathbf{y}^k_{\text{MILP}} - \mathbf{y} \end{matrix} \right\|_2, \tag{4.1}$$

where $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is the projected point [42]. A supporting hyperplane to the set $N \cap L$ is then given by

$$\begin{pmatrix} \bar{\mathbf{x}} - \mathbf{x}^k_{\text{MILP}} \\ \bar{\mathbf{y}} - \mathbf{y}^k_{\text{MILP}} \end{pmatrix}^T \begin{pmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{pmatrix} \geq 0. \tag{4.2}$$

For a proof that the inequality given by eq. (4.2) is a valid supporting hyperplane to $N \cap L$ see, *e.g.*, [9]. The linear constraint given by eq. (4.2) could be used to update the polyhedral outer approximation $\widehat{N}_k$; However, this technique would require the exact solution of problem (4.1) and could be quite sensitive to the numerical accuracy. Since the point $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ is located on the boundary of the set N, a supporting hyperplane of N is simply given by

$$g_j(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \nabla g_j(\bar{\mathbf{x}}, \bar{\mathbf{y}})^T \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix} \leq 0, \tag{4.3}$$

where g_j is an active constraint, *i.e.*, $g_j(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = 0$. Generating supporting hyperplanes according to eq. (4.3) should be less sensitive to the quality of the solution to problem (4.1). However, this approach still requires the solution of a convex NLP problem for

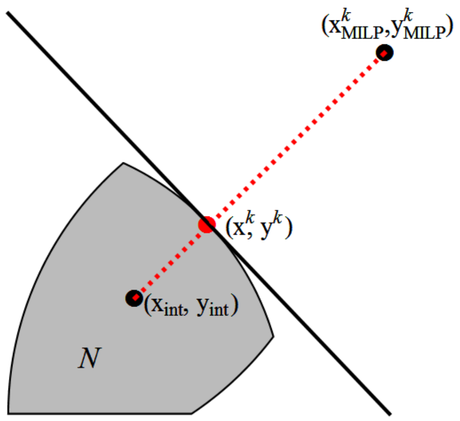

Figure 4.1: The figure illustrates the main principle in the ESH algorithm. A root-search is conducted between the interior point $(\mathbf{x}_{\text{int}}, \mathbf{y}_{\text{int}})$ and the MILP solution $(\mathbf{x}^k_{\text{MILP}}, \mathbf{y}^k_{\text{MILP}})$ to find a point on the boundary of the set N.

generating a supporting hyperplane, and the projection is unlikely to result in integer feasible solutions. A simpler and faster approach for generating supporting hyperplanes is, therefore, used within the ESH algorithm.

For generating supporting hyperplanes, the ESH algorithm uses a one-dimensional root-search to obtain an approximate solution to problem (4.1). Suppose that a point strictly within the interior of N and within $N \cap L$ is known. The interior point will be used within the root-search and is denoted as $(\mathbf{x}_{\text{int}}, \mathbf{y}_{\text{int}})$. To find a point on the boundary of the set N, a new function F is defined as the point-wise maximum of the nonlinear constraints according to

$$F(\mathbf{x}, \mathbf{y}) = \max_j \left\{ g_j(\mathbf{x}, \mathbf{y}) \right\}. \tag{4.4}$$

A new point $\left(\mathbf{x}^k, \mathbf{y}^k\right)$ is now constructed as a convex combination of the trial solution and the interior point, according to

$$\begin{aligned} \mathbf{x}^k &= \lambda^k \mathbf{x}_{\text{int}} + (1 - \lambda^k)\mathbf{x}^k_{\text{MILP}}, \\ \mathbf{y}^k &= \lambda^k \mathbf{y}_{\text{int}} + (1 - \lambda^k)\mathbf{y}^k_{\text{MILP}}. \end{aligned} \tag{4.5}$$

The interpolation parameter λ^k is chosen such that $F(\mathbf{x}^k, \mathbf{y}^k) = 0$, which can be determined by a simple root-search in the interval $[0, 1]$. A supporting hyperplane of N can then be obtained by linearizing a constraint active at $\mathbf{x}^k, \mathbf{y}^k$; the procedure for obtaining supporting hyperplanes is illustrated in Figure 4.1. The polyhedral outer approximation $\widehat{N}_k$ can then be updated by including new supporting hyperplanes, according to

$$\widehat{N}_{k+1} = \left\{ (\mathbf{x}, \mathbf{y}) \in \widehat{N}_k \;\middle|\; \nabla g_j(\mathbf{x}^k, \mathbf{y}^k)^T \begin{bmatrix} \mathbf{x} - \mathbf{x}^k \\ \mathbf{y} - \mathbf{y}^k \end{bmatrix} \le 0 \quad \forall j \in I_a \right\}, \tag{4.6}$$

where I_a contains the indices of the active constraints. A new trial solution $(\mathbf{x}^k_{\text{MILP}}, \mathbf{y}^k_{\text{MILP}})$ is then obtained by solving problem (MILP-k) with the improved polyhedral outer approximation.

The interior point used within the root-search should preferably be located as deep as possible within the set N, to avoid attracting the points $\mathbf{x}^k, \mathbf{y}^k$ to a certain region. For example, if N is given by an n-dimensional ball and $(\mathbf{x}_{\text{int}}, \mathbf{y}_{\text{int}})$ is located at the center of the ball, then the simple root-search will result in the same point as the projection by problem (4.1). The Chebyshev center of N would, thus, have desirable properties, but unfortunately it is nontrivial to obtain for a set defined by convex nonlinear functions [42]. To obtain an interior point with similar properties, the ESH algorithm chooses the interior point by minimizing l_∞-norm of the nonlinear constraints, which is obtained by solving the following convex NLP problem

$$\begin{aligned} \text{minimize} \quad & \mu \\ \text{subject to} \quad & g_j(\mathbf{x}, \mathbf{y}) \leq \mu \quad \forall j = 1, 2, \dots, l, \\ & (\mathbf{x}, \mathbf{y}) \in L, \mu \in \mathbb{R}. \end{aligned} \tag{4.7}$$

Problem (4.7) only has to be solved once, and the interior point does not have to satisfy the integer requirements. The procedure used within the ESH method for generating supporting hyperplanes is, hence, significantly less complex than projecting each trial solution onto $L \cap N$.

The procedure of solving MILP sub-problems and improving the polyhedral outer approximation by generating supporting hyperplanes is repeated until a trial solution satisfies all nonlinear constraints. Once a trial solution obtained by the MILP sub-problem satisfies the nonlinear constraints it is guaranteed to be an optimal solution to the MINLP problem, as in the ECP method. To illustrate the difference between the ESH algorithm and the ECP method, they are both applied to problem (ex2), and the first iterations are shown in Figure 4.2. To obtain a solution that satisfies the constraints, within a tolerance of 10^{-6}, the ESH algorithm requires five iterations and the ECP method requires nine. A technique similar to the ESH method was proposed by Veinott in 1966 [190], and the idea of utilizing supporting hyperplanes for MINLP problems was also mentioned in the PhD thesis of Pörn [166].

A preprocessing procedure for generating an initial polyhedral outer approximation with the ESH algorithm, by solving a sequence of LP relaxations, is presented in Paper I. In the paper it is proven that the ESH algorithm converges to the optimal solution for convex MINLP problems. Assumption 4 is not needed for proving convergence, but it is required that N has a nonempty interior. The paper also describes an early version of the SHOT solver, which implemented the ESH algorithm together with some primal heuristics. The primal heuristics are used for obtaining feasible solutions and a valid upper bound on the optimal objective value of the MINLP problem. The upper bound enables the solver to terminate the search based on the optimality gap, *i.e.*, the difference between the upper and lower bound. Using the ESH algorithm to obtain iteratively improving lower bounds, together with primal heuristics, usually allows SHOT to ter-

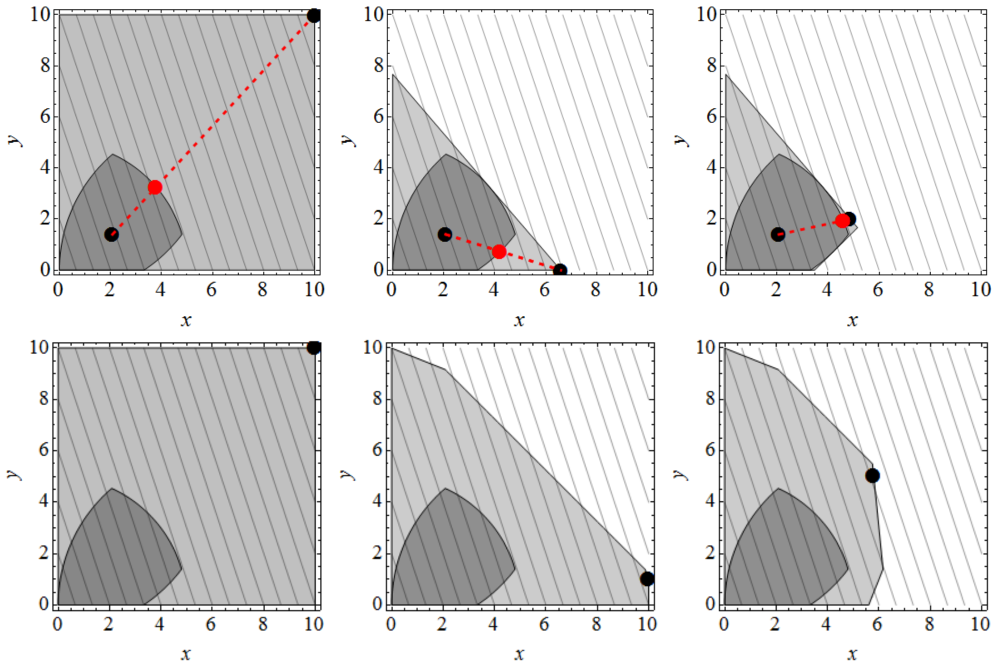

Figure 4.2: The figures show the first three iterations of the ESH algorithm (on top) and ECP method (below) applied to problem (ex2). The dark gray areas represent the feasible region defined by the nonlinear constraints, the light gray areas represent the polyhedral outer approximations, the lines represent contours of the objective function, and the black dots represent an optimal solution of the MILP sub-problems. The dashed lines illustrate the root-search procedure in the ESH algorithm

minate within significantly fewer iterations compared to forcing the trial solutions to satisfy all nonlinear constraints.

Numerical comparisons based on 333 convex MINLP problems were encouraging, and the SHOT solver performed best among the solvers for the test set. More features were later included in the SHOT solver, and the latest version is presented in Paper VI. The numerical results in Paper I showed a clear potential of the ESH strategy for generating tight polyhedral approximations.

In Paper II, it was shown that the ESH algorithm can successfully be applied to a more general class of problems. By using subgradients for generating the supporting hyperplanes, the algorithm can be proven to converge to a global optimum for convex MINLP problems with locally Lipschitz continuous functions. Furthermore, it was proven that the ESH algorithm converges to a globally optimal solution for MINLP problems with constraints given by pseudoconvex locally Lipschitz continuous functions. By generating the linearizations on the boundary of the feasible region, the ESH algorithm is able to efficiently construct valid polyhedral outer approximations of constraints given by pseudoconvex functions. By minor modification, it would be possible to use the same technique for constraints given by quasiconvex functions. However, the task of obtaining an interior point becomes nontrivial for constraints given by quasiconvex functions, and therefore the quasiconvex case has not been studied in more detail. It was later shown in [199] that the ESH technique can be successfully applied to a class of problems referred to as generalized convex, where both the objective and constraints are given by pseudoconvex locally Lipschitz continuous functions.

4.2 Reformulations for separability in convex MINLP (Paper III)

Obtaining an optimal solution of a convex MINLP problem and verifying optimality of the solution with methods utilizing polyhedral outer approximation, such as ECP, ESH, LP/NLP-BB or OA, usually requires a quite accurate approximation of the set N. Obtaining a polyhedral approximation as tight as possible is, therefore, desirable with such methods. Tighter approximations can, *e.g.*, result in a significant reduction of the iterations and computational effort needed to solve a problem.

If a nonlinear function defining a nonlinear constraint has certain convex separable properties, then it is possible to utilize a simple reformulation that results in tighter lifted polyhedral outer approximations. To use the reformulation technique described in Paper III the nonlinear function needs to have a separability property referred to as almost additively separable. An almost additively separable function is given by the sum of simpler functions depending only on subsets of the variables. Hence, a function g_j is considered as almost additively separable if it can be written as

$$g_j(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{k} h_i(\mathbf{x},\mathbf{y}), \tag{4.8}$$

where each function h_i is dependent only on subsets of the variables $(\mathbf{x},\mathbf{y})$. Now, if each function h_i is convex, it is possible to use a simple reformulation to obtain tighter polyhedral outer approximations. A nonlinear constraint given by the almost additively separable function g_j can then be reformulated into several nonlinear constraints by introducing new continuous variables, according to

$$g_j(\mathbf{x},\mathbf{y}) \leq 0 \quad \rightarrow \quad \begin{cases} \sum_{i=1}^{k} z_i \leq 0, & \\ h_i(\mathbf{x},\mathbf{y}) \leq z_i & \forall i = 1,\dots k, \\ z_i \in \mathbb{R} & \forall i = 1,\dots k. \end{cases} \tag{4.9}$$

The original constraint and the multiple constraints obtained by the reformulation results in the same set of feasible $(\mathbf{x},\mathbf{y})$ variables, and can be considered as equivalent formulations. However, a polyhedral outer approximation of the multiple constraints in the reformulated form can result in a tighter polyhedral approximation, which is referred to as a lifted polyhedral approximation since it is an approximation in a higher dimensional space.

To illustrate the benefits of the reformulation, consider the following constraint

$$x^2 + y^2 \leq 4. \tag{4.10}$$

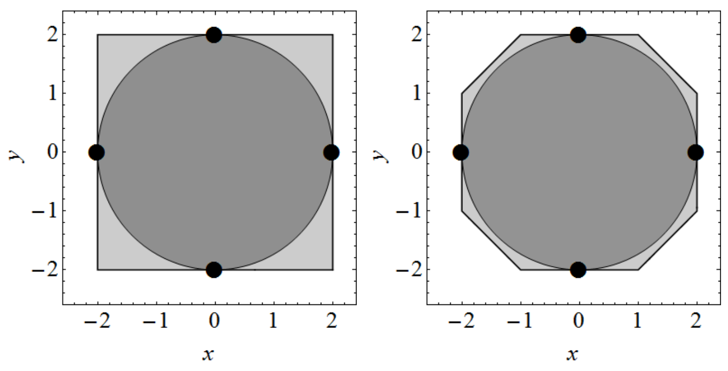

Figure 4.3: The figures show polyhedral approximations of the nonlinear constraint $x^2 + y^2 \leq 4$, based on the original constraint (left figure) and reformulated constraints (right figure). The light gray areas show the polyhedral outer approximations, the dark gray areas represent the feasible region of the nonlinear constraint, and the dots show the linearization points.

Using the described reformulation technique results in the following constraints

$$\begin{aligned} z_1 + z_2 &\leq 4, \\ x^2 &\leq z_1, \\ y^2 &\leq z_2, \end{aligned} \tag{4.11}$$

where z_1 and z_2 are continuous variables. A polyhedral outer approximation of the feasible region defined by the constraints in equations (4.10) and (4.11) is constructed by generating linearizations at the points $(-2,0)$, $(2,0)$, $(0,-2)$, and $(0,2)$. The polyhedral outer approximations obtained with both formulations within the x,y-space are shown in figure 4.3. The figure shows that the reformulation resulted in a significantly tighter polyhedral outer approximation.

The benefits of linearizing the components of a separable convex function individually was first shown by Tawarmalani and Sahinidis [183]. Utilizing lifted polyhedral approximations has also been studied by Hijazi et al. [99], and extended formulations were also studied later by Lubin et al. [138].

In Paper III, properties regarding the reformulation in eq. (4.9) are analyzed, and a numerical comparison shows the practical advantages of using the reformulation. It can easily be shown that the reformulation will not result in weaker polyhedral approximations, and sufficient conditions under which the reformulation results in tighter approximations are given. By combining the reformulation with a power transform and a logarithmic transform, it is shown that the reformulation technique can be applied to some non-separable functions. Numerical results show that the reformulation technique can greatly improve the performance of several solvers based on polyhedral outer approximation techniques. Several of the test problems cannot be solved by either AlphaECP, BONMIN-OA, DICOPT, or SHOT within two hours using the original problem formulations, whereas the reformulated problems can be solved within only a few seconds. Both problem formulations are equivalent and the differences in the solution times are caused by the tighter polyhedral approximations obtained for the reformulated problems. The numerical results highlight the importance of the problem formulation for MINLP problems, and strongly motivates work within automatic reformulations.

4.3 The center-cut algorithm (Paper IV)

The main idea in the center-cut algorithm is to use the polyhedral outer approximation differently, compared to ECP, ESH, and OA, to obtain feasible solutions within only a few iterations. In the ECP and ESH methods, the trial solutions are chosen as the minimizer of the objective function within the current polyhedral outer approximation, generally resulting in trial solutions on the boundary of the outer approximation. As a consequence, the basic versions of the ESH and ECP methods will not obtain a feasible solution before the very last iteration.

Instead of searching along the boundary of the polyhedral approximation, the idea is to search in the center of the polyhedral approximation. Since the true feasible set is contained somewhere within the polyhedral outer approximation, it seems natural to search for a feasible solution in the center of the outer approximation. A similar idea for solving convex NLP problems was presented by Elzinga and Moore in 1975 [64]. By the center of the polyhedral outer approximation we mean to the Chebyshev center, which is defined as the point furthest away from the boundary in all directions. The Chebyshev center of a polyhedral set is given by the center of the largest n-dimensional ball inscribed in the set and can be obtained by simply solving an LP problem [42, 98].

As before, a polyhedral outer approximation of the set N is constructed as

$$\widehat{N}_k = \left\{ (\mathbf{x},\mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m \mid g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^T \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq 0 \ \ \forall i = 1,2,\dots k, j \in I_i \right\}, \quad (4.12)$$

where I_i contains the indexes of the nonlinear constraints for which linearizations were added in iteration i. The Chebyshev center of set $\widehat{N}_k$ can be obtained by solving the problem

$$\begin{aligned} &\underset{\mathbf{x},\mathbf{y},r}{\text{maximize}} \ r \\ &\text{subject to} \\ &g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^T \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} + r\left\|\nabla g_j(\mathbf{x}^i,\mathbf{y}^i)\right\|_2 \leq 0 \quad \forall i = 1,\dots k, j \in I_i, \\ &\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m, r \in \mathbb{R}, \end{aligned} \quad (4.13)$$

where r is the radius of the inscribed ball. However, the optimal solution of problem (4.13) is unlikely to satisfy the integer requirement and linear constraints of problem (P-MINLP). To consider all the constraints of the MINLP problem, trial solutions will be chosen as the center of the largest inscribed balls in the sets $\widehat{N}_k$, such that the center satisfies all linear constraints and integer requirements of the MINLP problem. A new trial solution $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$ is then given by solving the MILP problem

$$\begin{aligned}
&\underset{\mathbf{x},\mathbf{y},r}{\text{maximize}}\ r \\
&\text{subject to} \\
&g_j(\mathbf{x}^i, \mathbf{y}^i) + \nabla g_j(\mathbf{x}^i, \mathbf{y}^i)^T \begin{bmatrix} \mathbf{x} - \mathbf{x}^i \\ \mathbf{y} - \mathbf{y}^i \end{bmatrix} + r \left\| \nabla g_j(\mathbf{x}^i, \mathbf{y}^i) \right\|_2 \le 0 \quad \forall i = 1, \ldots k, \forall j \in I_i, \\
&(\mathbf{x}, \mathbf{y}) \in L \cap Y, r \in \mathbb{R}.
\end{aligned} \tag{4.14}$$

The linear constants defining the set L only affects the center of the inscribed ball, and not directly the radius since the ball is only inscribed in $\widehat{N}_k$. If a trial solution $(\mathbf{x}^k, \mathbf{y}^k)$ does not satisfy the nonlinear constraints of the MINLP problem, then the polyhedral outer approximation can be improved by generating new linearization at $(\mathbf{x}^k, \mathbf{y}^k)$, as in the ECP method. It would be possible to use the ESH strategy for updating the polyhedral approximation, but for simplicity the approximation is improved by the ECP approach. If $(\mathbf{x}^k, \mathbf{y}^k)$ satisfies all constraints, the solution can usually be further improved by solving the MINLP problem with the integer variables fixed as $\mathbf{y}^k$, *i.e.*, feasible integer combinations are used as in OA. Based on a feasible solution $(\mathbf{x}^k, \mathbf{y}^k)$, an objective cut given by

$$\mathbf{c}_1^T \mathbf{x} + \mathbf{c}_2^T \mathbf{y} \le \mathbf{c}_1^T \mathbf{x}^k + \mathbf{c}_2^T \mathbf{y}^k, \tag{4.15}$$

is used to reduce the set $\widehat{N}_k$. By including the objective cut, $\widehat{N}_k$ is no longer an outer approximation of N. However, $\widehat{N}_k$ it still guaranteed to contain all optimal solutions of the MINLP problem. In the next iteration, where the trial solutions are chosen as the center of $\widehat{N}_k$, the objective cut enforces an improvement in the objective. A lower bound on the optimum of the MINLP problem is not directly obtained by the algorithm, and to prove optimality of the best-found solution the search must continue until the radius of the inscribed ball is reduced to zero.

To illustrate the center-cut algorithm, and to highlight the differences to the other methods, it is applied to problem (ex2) and the result is shown in Figure 4.3. The search is initiated by defining the initial polyhedral outer approximation as $\mathbb{R}^n \times \mathbb{R}^m$, and since the problem only contains two variables the solutions are obtained by inscribing circles in the polyhedral approximation. A feasible solution is obtained in the second iteration, and the optimal solution is obtained in iteration four. In the fifth iteration, optimality is proven by obtaining a zero radius of the inscribed circle.

The center-cut algorithm for convex MINLP was first presented in a conference paper by Kronqvist et al. [122], and a more detailed description is given in Paper IV. In Paper IV it was proven that the center-cut algorithm has finite convergence convex MINLP problems. Proving finite convergence requires assumptions similar to those used to prove finite convergence for OA in [69].

Numerical tests in Paper IV show that the center-cut algorithm is able to quickly find feasible solutions to the problems, and for the more difficult problems it was more efficient at finding feasible solutions than the feasibility pump in DICOPT [21, 39]. The results support the hypothesis that the algorithm would be able to obtain feasible

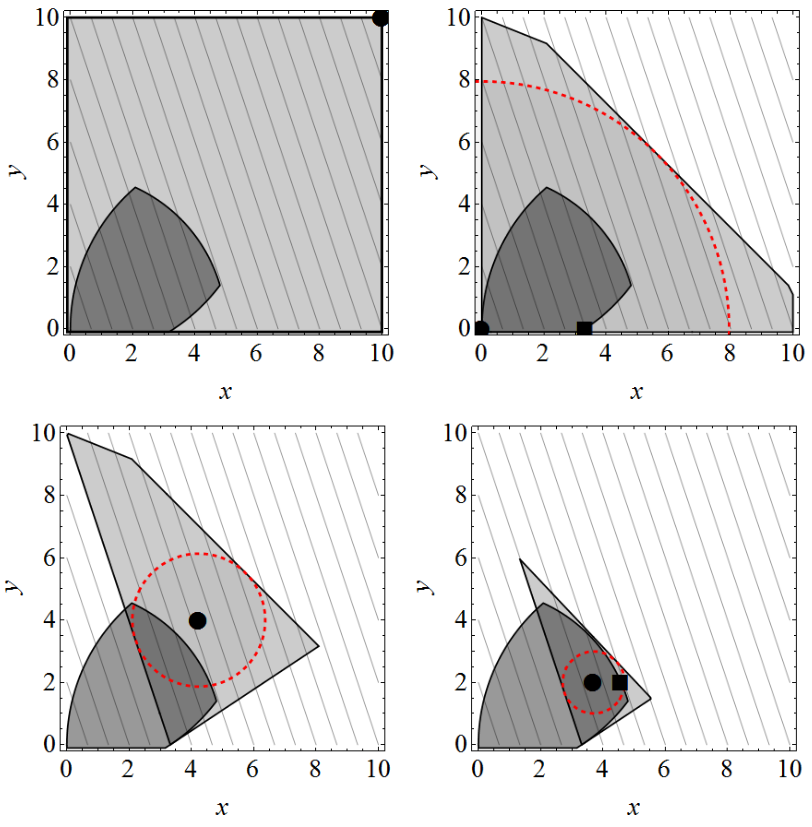

Figure 4.4: The figures show the first four iterations of the center-cut algorithm applied to problem (ex2). The figures show the feasible region defined by the nonlinear constraints in dark gray, the light gray areas show the polyhedral approximations, and the lines show contours of the objective function. The circular dots represent the center of the inscribed circles and the dashed curves represent the circle. The improved solutions, obtained by solving NLP sub-problems for feasible integer combinations, is indicated by the squares.

solutions within only a few iterations. Out of the 295 test problems used, the center cut is able to find a feasible solution for 294 problems, and a solution within 1% of the best-known solution is found for 290 problems. The algorithm seems to be well-suited as a primal heuristic, but it can also be used as a deterministic technique with guaranteed convergence. Details regarding how to efficiently implement the algorithm, and techniques to speed up the algorithm, are also discussed in the paper.

The downside of the center-cut algorithm is the optimality verification. To verify optimality, the radius has to be reduced to zero, which can require quite a number of iterations after the optimal solution was found. Therefore, the center-cut algorithm may be best-suited as a primal heuristic. However, the center-cut algorithm could be implemented as a primal heuristic together with the ECP or ESH algorithm to get a technique for obtaining both feasible solutions (upper bounds) and lower bounds on the optimal objective value.

4.4 Using regularization and second order derivatives with the outer approximation method (Paper V)

In outer approximation, the integer combinations are chosen by minimizing the objective function within a polyhedral outer approximations of the integer relaxed feasible region. The approach for choosing the integer combinations is similar to the technique used for obtaining the iterative trial solutions with Kelley's cutting plane method for convex NLP [110]. Kelly's method is in general not considered efficient at handling nonlinearities, and it has been proven that the method has a poor complexity bound [158]. The method is sometimes even referred to as unstable since the trial solutions tend to make large jumps in search space [60]. Since ECP, ESH, and OA all use the same approach for choosing the integer combinations, they could suffer from the same instability, especially for highly nonlinear MINLP problems.

To reduce the instability and improve the performance of Kelley's method it is often suggested to use regularization or a trust region to reduce the step size [9]. Due to the non-convex properties of all MINLP problems, it is not trivial to directly use a trust region or regularization. For example, the two closest feasible solutions, with different integer combinations, may be located far apart in the search space. Lately, there has been an interest in using regularization techniques for solving convex MINLP problems; *e.g.*, using quadratic stabilization with Benders decomposition was proposed in [205], and a technique using regularization combined with a cutting plane method was presented in [59].

The new methods presented in Paper V are based on OA, and use a regularization technique and quadratic approximations of the Lagrangian for the task of obtaining new integer combinations. The regularization technique is referred to as level-based outer approximation (L-OA), and is inspired by the level method for NLP problems [113, 129]. By modifying the L-OA method it is possible to use a quadratic approximation of the Lagrangian function for the task of choosing integer combinations, and this method is referred to as quadratic outer approximation (Q-OA). The second method, thus, combines ideas from both the level method and sequential quadratic programming (SQP) [96, 159]. The main idea in both methods is to choose the integer combinations more carefully, to obtain the optimal solution in fewer iterations.

The basic steps of L-OA are first described, and from there Q-OA can easily be derived. A feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ of the MINLP problem is needed to initiate the search with both methods, and here it is assumed that such a solution is known. Polyhedral outer approximations $\widehat{N}_k$ are constructed and improved in each iteration, using the same procedure as in OA. In each iteration k, a lower bound LB^k on the optimum of the MINLP problem is obtained by minimizing the objective within $\widehat{N}_k \cap L \cap Y$, *i.e.*, by solving problem (MILP-k). An upper bound UB^k on the optimum is given by the best-known feasible solution. The optimal objective value of the MINLP problem, which is in between LB^k and UB^k, is approximated in each iteration according to

$$\hat{z}_k^* = (1-\alpha)UB^k + \alpha LB^k, \tag{4.16}$$

where $\alpha \in (0,1]$ is a parameter. A new integer combinations $\mathbf{y}^{k+1}$ is then chosen by projecting $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ onto the part of the polyhedral approximation where the MINLP objective function is less than or equal to $\hat{z}_k^*$. The projection is done by solving

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & \left\| \begin{matrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{matrix} \right\|_2^2 \\ \text{subject to} \quad & \mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y} \le \hat{z}_k^*, \\ & (\mathbf{x},\mathbf{y}) \in \widehat{N}_k \cap L \cap Y, \end{aligned} \tag{MIQP-Proj}$$

which is a convex mixed-integer quadratic programming (MIQP) problem. The integer combination $\mathbf{y}^{k+1}$ is thus chosen as the point closest to the feasible solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ solution, such that the objective is reduced to at most $\hat{z}_k^*$. The corresponding continuous variables $\mathbf{x}^{k+1}$ are obtained as in OA, by solving either one of the NLP problems (NLP-k) or (NLPf-k). If a better feasible solution is found, then the upper bound is updated and the solution is stored as $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. The polyhedral outer approximation is updated by generating new linearization at the $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$. In the next iteration the procedure is repeated by obtaining a new improved lower bound LB^{k+1}, improved estimate of the optimum $\hat{z}_{k+1}^*$, and projecting the best found solution. These are the basic steps in the L-OA method, and they are repeated until the upper and lower bounds are within the desired tolerance. In Paper V it is proven that the L-OA method has finite convergence. Furthermore, it is shown that the procedure is equivalent to adding a trust region around the best found feasible solution in the MILP master problem in OA.

It is not necessary to use the Euclidean norm for the projection in problem (MIQP-Proj). For example, a valid norm in the $(\mathbf{x},\mathbf{y})$-space can be defined as

$$\left\| \begin{matrix} \mathbf{x} \\ \mathbf{y} \end{matrix} \right\| = \left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}^T \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}\left(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}, \bar{\mu}\right) \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \right)^{1/2}, \tag{4.17}$$

where $\nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}$ is the Hessian of the Lagrangian with respect to the $(\mathbf{x},\mathbf{y})$-variables. Using the norm given by eq. (4.17) in the projection problem (MIQP-Proj), would favor directions in which the linear approximation is, at least locally, more accurate. However, the approach of using the norm given by eq. (4.17) has so far not been tested properly.

It is also possible to use the L-OA approach with a different objective function in problem (MIQP-Proj), and the convergence proofs in Paper V hold with an arbitrary objective function in the sub-problem. To obtain better integer solutions, it would be desirable to utilize information about the curvature of the problem when choosing the new integer combinations. To include information about both the objective and constraints of the original problem, we can use the Lagrangian function given by eq. (5). Furthermore, to get a tractable sub-problem, we choose to approximate the Lagrangian by a second order Taylor series expansion. By minimizing a quadratic approximation of the Lagrangian function, we can incorporate information about second order derivatives in the task of choosing new integer combinations. The Q-OA method includes the same steps as L-OA for obtaining lower bounds LB^k, and estimates of the optimum $\hat{z}_k^*$. However, the new integer combinations are then obtained by solving the following convex MIQP

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & \nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda},\bar{\mu})^T \begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{y}-\bar{\mathbf{y}}\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{y}-\bar{\mathbf{y}}\end{bmatrix}^T \nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}\left(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda},\bar{\mu}\right)\begin{bmatrix}\mathbf{x}-\bar{\mathbf{x}}\\ \mathbf{y}-\bar{\mathbf{y}}\end{bmatrix} \\ \text{subject to} \quad & \mathbf{c}_1^T\mathbf{x}+\mathbf{c}_2^T\mathbf{y} \le \hat{z}_k^*, \\ & (\mathbf{x},\mathbf{y}) \in \widehat{N}_k \cap L \cap Y, \end{aligned} \tag{QOA-master}$$

where $\bar{\lambda}$ and $\bar{\mu}$ are the optimal Lagrangian multipliers corresponding to the currently best found feasible solution $(\bar{\mathbf{x}},\bar{\mathbf{y}})$. Using similar notation as before, $\nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}$ denotes the gradient of the Lagrangian with respect to the $(\mathbf{x},\mathbf{y})$-variables. The same procedure as in both OA and L-OA, is used for determining the continuous variables and updating the polyhedral approximation.

The constraint $\mathbf{c}_1^T\mathbf{x}+\mathbf{c}_2^T\mathbf{y} \le \hat{z}_k^*$ is needed to ensure that (QOA-master) will obtain new integer combinations. Since the second order Taylor series expansion does not underestimate the Lagrangian, it is possible that the expansion point $(\bar{\mathbf{x}},\bar{\mathbf{y}})$ is the minimizer of the approximation even if it is not the optimal solution to the MINLP problem. Another technique using a quadratic approximation of the Lagrangian within OA was proposed in [69], where they force a small ϵ-improvement on the objective to avoid stagnating at non-optimal solutions. In general, the Q-OA approach enforces a stricter reduction, and should be less sensitive to the quality of the quadratic approximation. Furthermore, the Q-OA approach ensures that the sub-problem (QOA-master) is always feasible. As mentioned in Paper V, Q-OA basically chooses the new integer combinations by interpolating between the minimum of the linear approximation of the MINLP problem and the minimum of the quadratic approximation of the Lagrangian, where α in eq. (4.16) is the interpolation parameter. Setting $\alpha = 1$ forces the integer combination to be chosen as the minimizer of the MILP-master problem, whereas a smaller α allows the integer combination to be closer to the minimum of the quadratic approximation. In general, a smaller α will result in trial solutions closer to the best found solution, and a larger value promotes larger steps in the search space.

To illustrate L-OA and Q-OA, and to show how they differ to the other methods presented in this thesis, the methods are applied to the illustrative example (ex2). Both methods are started at the feasible solution $(3.32, 0)$, and the α parameter is set to 0.5. The first three iterations of both methods are shown in Figure 4.4. L-OA is well-suited for this specific problem and obtains the optimal solution in the second iteration. In the third iteration, the solution is verified as optimal by updating the lower bound. From the figures, it can be seen that the approximation of the Lagrangian function does not attend its minimum at the expansion point, and the solutions are attracted toward the optimum of the approximation. Q-OA obtains the optimal solutions in the third

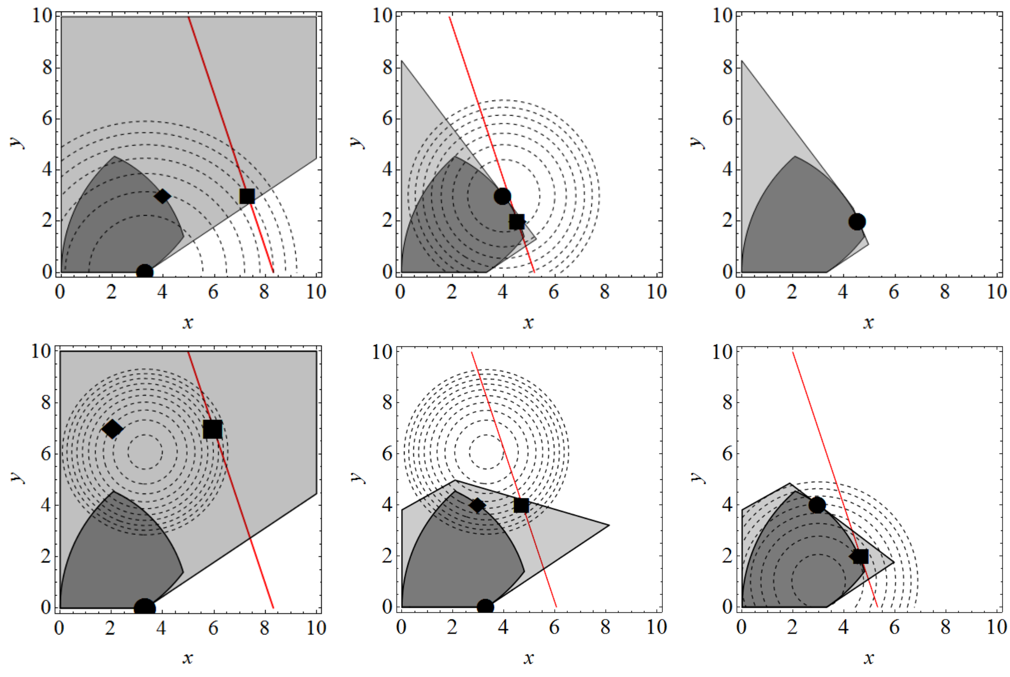

Figure 4.5: The figure shows the first three iterations of L-OA (on top) and Q-OA (below) applied to problem (ex2). The figures show the feasible region defined by the nonlinear constraints in dark gray, and the light gray areas show the polyhedral approximations. The circular dot represents the best found solution, the squared dots the solution of the MIQP sub-problem, and the diamond shaped dots the solution of an NLP sub-problem. The dashed curves are contours of the objective functions in the MIQP sub-problems, and the red lines represent the constraints $\mathbf{c}_1^T\mathbf{x} + \mathbf{c}_2^T\mathbf{y} \leq \hat{z}_k^*$. Contours of the objective function in the MINLP problem have been left out to avoid confusion.

iteration, and optimality can be proven in the following iteration. However, problem (ex2) is not the best example for showing the difference between L-OA and Q-OA, since the Hessian of the Lagrangian will be a scaled identity matrix. The objective function in both types of sub-problems will, therefore, have the same shape. A better illustration of the differences between L-OA and Q-OA is given in Paper V.

As previously mentioned, L-OA and Q-OA are intended to deal with nonlinearities more efficiently than OA by choosing the integer combinations more carefully. For almost linear MINLP problems, L-OA and Q-OA should not have a direct advantage over OA, due to computationally more demanding iterations. Each iteration in both L-OA and Q-OA requires the solution of both a MILP and a convex MIQP sub-problem. Each iteration is, therefore, basically twice as computationally demanding as an iteration in OA. However, as mentioned in Paper V, all of the sub-problems do not need to be solved to optimality, which can significantly reduce the computational effort.

Both L-OA and Q-OA are proven to have finite convergence in Paper V, and a numerical comparison of OA, L-OA, and Q-OA is presented in the paper. The test set in the comparison contained 109 convex MINLP problems, where at least 50% of the variables are included in a nonlinear term. For the test problems, L-OA and Q-OA needed significantly fewer iterations to solve the problems than OA. For 90% of the test problems, OA needed about seven times more iterations than Q-OA to solve the problems. Compared to L-OA, OA needed approximately twice as many iterations. Regarding speed, both L-OA and Q-OA were faster than OA for the test set. The new methods, thus, compensated for the computationally more demanding iterations by significantly reducing the number of iterations needed. L-OA was overall faster than OA, and Q-OA was significantly faster than the other methods. The results showed a clear advantage of utilizing second order derivatives when choosing the integer combinations.

For the test problems, both L-OA and Q-OA encountered significantly fewer infeasible NLP sub-problems than OA. In total, OA encountered 877 infeasible NLP problems, whereas Q-OA and L-OA only encountered 257 and 259 infeasible NLP problems. This is due to the more careful approach of choosing the integer combinations, and the MIQP sub-problems tends to return solutions closer to the feasible region than the solutions returned by the MILP-master problem in OA. From an infeasible integer combination, the methods do not, in general, generate supporting hyperplanes to the integer relaxed feasible set. Therefore, a feasible integer combination tends to result in a tighter polyhedral outer approximation in the next iteration, compared to an infeasible integer combination. The reduction of the infeasible sub-problems, at least partially, explains why L-OA and Q-OA needed fewer iterations than OA to solve the problems.

Techniques similar to those used in L-OA and Q-OA, could also be used within the ECP and ESH algorithm to incorporate regularization and second order derivatives. However, since the optimal Lagrangian multipliers are readily available for feasible integer combinations within OA, it felt natural to integrate the techniques within an OA framework.

4.5 The supporting hyperplane optimization toolkit solver (Paper VI)

The SHOT solver was first presented in Paper I, and a more detailed description and an updated solver are presented in Paper VI. In its current form, SHOT combines a dual strategy based on polyhedral outer approximations with several primal heuristics. The dual strategy is intended to obtain tight lower bounds on the optimal objective value of the MINLP problem and to obtain new trial solutions. The dual strategy uses the ESH algorithm for obtaining supporting hyperplanes of the integer relaxed feasible set, and to generate a tight polyhedral outer approximation. A strategy of solving a sequence of LP-relaxations is used as a preprocessing technique to quickly obtain an initial outer approximation of the feasible set. The polyhedral outer approximation is used either within a so-called, single- or multi-tree strategy. These are referred to as dual strategies since one of their primary tasks is to obtain a lower bound on the optimal objective value of the MINLP problem, which is sometimes referred to as a dual bound.

The multi-tree strategy basically solves a MILP or MIQP sub-problem in each iteration, and minimizes a linear or quadratic objective function within the polyhedral outer approximation to obtain a lower bound and a new trial solution. Currently, MIQP sub-problems are only utilized if the original problem has a quadratic objective. The multi-tree strategy closely resembles the ESH algorithm presented in Paper I; it is referred to as a multi-tree strategy since multiple branch and bound trees are generated. In the early iterations, the polyhedral approximation is usually not accurate enough to get a good representation of the MINLP problem, and to avoid spending time solving poor approximations the multi-tree strategy uses an early termination technique. Early termination allows the subsolver to terminate the search as soon as a good integer solution has been obtained. The requirements of a “ good” solution are dynamically updated, and eventually forces the sub-problems to be solved to optimality.

The main idea behind the single-tree strategy is to construct only a single branch and bound tree and to dynamically update the polyhedral outer approximation within the branch and bound search. The single-tree strategy in SHOT is, thus, based on a method similar to the LP/NLP-BB algorithm described in Section 3.7. The search is initiated as a “ normal” BB procedure of a MILP or MIQP problem, that utilizes the initial polyhedral approximation to approximate the nonlinear constraints. However, the BB procedure is paused as soon as an integer solution is obtained. The obtained integer solution serves as a trial solution and will be used to improve the polyhedral approximation. New supporting hyperplanes to the integer relaxed feasible set are included, and the polyhedral approximation is improved within the open nodes in the branch and bound tree. The integer solutions are also used within the primal heuristics, which may result in a feasible solution and an improved upper bound. A lower bound is obtained, as normally in BB-algorithm, by the lowest objective value of the open nodes. If the upper and lower bounds are not within the desired tolerance, then the BB search continues with an improved polyhedral approximation. The new supporting hyperplanes are added as so-called lazy constraints through callbacks in the MILP/MIQP solver. This approach is implemented with either Gurobi or CPLEX as subsolver, and allows SHOT to update the polyhedral outer approximation without completely rebuilding the BB tree. By using the lazy constraint functionality, SHOT lets the subsolver construct and maintain the BB tree. The tight integration with the subsolver allows SHOT to fully benefit from preprocessing, advanced branching strategies, cut generating procedures, and other features implemented in the subsolver.

Currently, there are three main primal strategies implemented in SHOT, which are used for obtaining feasible solutions and an upper bound to the MINLP problem. The first strategy utilizes the solution pool provided by the MILP/MIQP subsolver, and is only available within the multi-tree strategy. When solving the MILP/MIQP sub-problems, multiple integer solutions are normally found in the BB search, and these alternative solutions are stored in the solution pool. Even if these are non-optimal solutions to the subproblems, they may provide good solutions to the original problem. In the single-tree approach, every integer solution obtained are utilized, and feasible integer solutions are likely to be found in the search procedure. The second strategy uses a technique of temporarily fixing the integer variables in the original MINLP problems to an obtained integer solution, resulting in a convex NLP. By solving the resulting NLP problem, it may be possible to obtain a feasible solution to the MINLP problem. The second primal strategy is, thus, closely related to the OA algorithm. The root-searches used to obtain points on the boundary of the integer relaxed feasible set may also return feasible solutions to the MINLP problem, and the root-searches are considered as the third primal strategy.

SHOT has been released as an open-source COIN-OR project, and the solver can be integrated in the GAMS software. The solver is implemented in C++ and uses some functionality from other COIN-OR projects, such as Optimization Services [82]. The solver implementation is described in more detail in Paper VI, and the solver can be downloaded from `https://github.com/coin-or/shot`. By using CBC and IPOPT as subsolvers, SHOT can be used as completely open-source software. However, the performance is greatly improved by using either Gurobi or CPLEX as subsolver for the MILP/MIQP sub-problems. Through an interface to GAMS, it is also possible to utilize all the NLP solvers available in GAMS as subsolvers with SHOT.

Numerical comparisons of different features of SHOT are presented in Paper VI, along with a benchmark comparison against other state-of-the-art solvers. Figure 4.6, from Paper VI, shows a comparison of SHOT against AlphaECP [195], AOA [103], BARON [173], BONMIN [38], DICOPT [92], Minotaur [143], SBB [80], and SCIP [191]. The comparison is based on all 366 convex MINLP instances currently available in MINLPLib [148]. Figure 4.6 shows how many of the problems that can be solved within a relative objective gap $\leq 0.1\%$ and a primal gap $\leq 0.1\%$ with the different solvers. The graphs do not correspond to the cumulative solution time but show how many individual problems can be solved within a specific time. An objective gap of $\leq 0.1\%$, means that the solver was able to obtain an upper and lower bound within less than 0.1%. A primal gap of less than 0.1% indicates that the solver was able to obtain a feasible solution within 0.1% of the best-known solution, but the solutions were not necessarily verified as optimal. The virtual best and virtual worst lines in the figure show the performance obtained if the best and worst solver is chosen for each individual problem separately.

Figure 4.6 shows that SHOT is one of the most efficient solvers for the test problems, and SHOT is able to solve more problems than any other solver. The efficiency of SHOT is also illustrated in a recent review of solvers for convex MINLP problems [123]. Some of the solvers are able to obtain an optimal solution to significantly more problems, than they are able to verify as optimal. This highlights the importance of obtaining a tight polyhedral outer approximation to obtain a strong lower bound. By comparing the number of problems solved within the desired objective gap and primal gap, it is clear that SHOT is able to obtain tight lower bounds and supports the belief in the ESH algorithm's ability to generate tight polyhedral approximations.

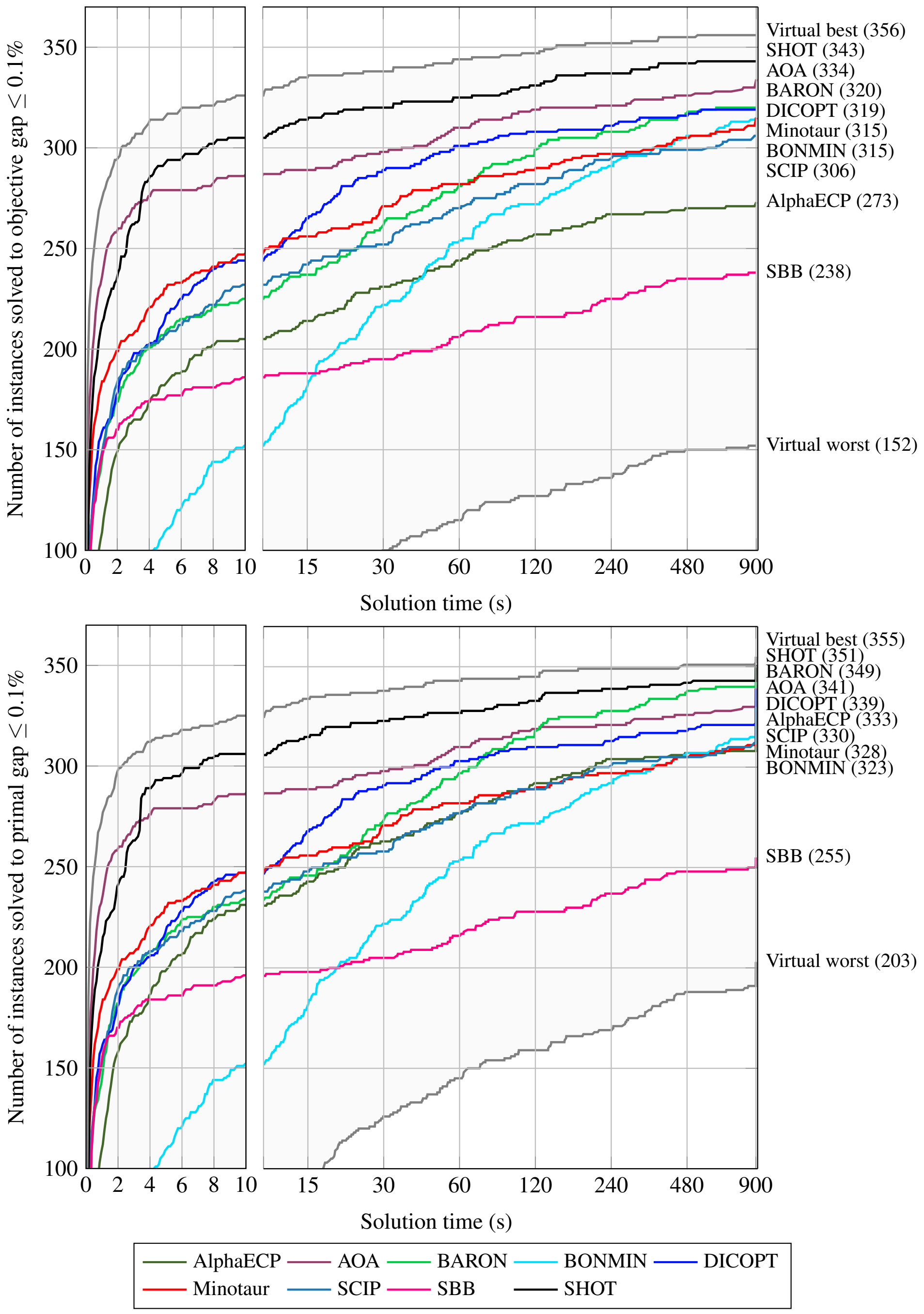

Figure 4.6: Solution profiles showing the number of convex MINLP problems solved, within a specific objective and primal gap, as a function of time. The solution profiles are from Paper VI, where more information about the benchmark is provided.

The two overall fastest solvers were AOA and SHOT, and both solvers utilize a polyhedral outer approximation based on a single-tree approach. The results supports the strong belief in the single tree approach expressed in [1, 19, 131]. However, comparisons of the single- and multi-tree strategies of SHOT, in Paper VI, do not show an obvious advantage of either strategy. The good performance of SHOT is, therefore, not mainly due to the single-tree approach, but rather to an efficient integration of different techniques.

4.6 Conclusion

The new methods presented in Papers I-V have been described in this chapter along with a description of the SHOT solver from Paper VI. This chapter gives a summary of the contributions to convex MINLP presented in Papers I–VI. However, some details and results from the papers are omitted here. Both theoretical aspects, practical implementation details, and numerical results are presented in more detail in the enclosed papers. The next chapter, concludes the thesis and discusses some future research ideas.

Chapter 5

Conclusion and Future Ideas

Finally, this section concludes the thesis, and some future research ideas in the field of convex MINLP are also given.

5.1 Conclusion

Different techniques for constructing and utilizing polyhedral outer approximations have been presented and analyzed in this thesis. It was assumed that the ESH algorithm would be an efficient technique for constructing tight polyhedral outer approximations of the integer relaxed feasible set, and this has been verified by the numerical results presented in Paper I and Paper VI. According to the numerical results, the concept of using the ESH algorithm for generating tight polyhedral outer approximations in combination with primal heuristics seems to be a very competitive approach for solving convex MINLP problems. Furthermore, it was shown in Paper II that the ESH algorithm can be applied to a more general type of problems where the nonlinear function may be non-smooth and pseudoconvex.

The different techniques for utilizing the polyhedral outer approximation, presented in this thesis, resulted in trial solutions with different properties. For example, by using the center-cut technique, from Paper IV, it is usually possible to obtain a feasible solution within only a fraction of the iterations needed to obtain a feasible solution with the basic ECP or ESH algorithm. The results obtained with the L-OA and Q-OA methods, in Paper V, also illustrated the differences between techniques for utilizing the polyhedral outer approximation. OA, L-OA, and Q-OA all use the same technique for generating and updating the polyhedral outer approximations. Still, for many of the test problems Q-OA needed only about 1/7 of the iterations required by OA to solve the problems. As a consequence Q-OA used only about 1/7 of the total number of linearizations added by OA. It is interesting that the quadratic approximation technique was able to construct a polyhedral approximation accurate enough for obtaining the optimal solution and verifying optimality with only a fraction of the linearizations needed by OA. The numerical results in Paper V highlight the importance of efficiently generating linearizations to construct the polyhedral outer approximation, and show that efficient linearizations can significantly reduce the number of iterations needed to solve a problem. By utilizing a quadratic approximation of the Lagrangian function for choosing the integer combination, Q-OA was able to obtain the optimal solution in less time and iterations than the other methods. For convex MINLP problems with a high degree of nonlinearity, the Q-OA approach seems to be a very efficient technique.

It was shown in Paper III that the problem formulation can have a great impact on the quality of the obtained polyhedral outer approximations. By applying the reformulation techniques, it was possible to transform some practically intractable problems into problems that could be solved within a few seconds. The simple reformulations techniques should not be left to the end user but should be taken care of by the solver to improve the ease of use. Further work within the field of automatic reformulations in convex MINLP is, thus, still well motivated. Some solvers, such as BARON, already performs some automatic reformulations. The reformulations, from Paper III, are currently not implemented in SHOT, but the goal is to incorporate automatic reformulations into the solver.

The SHOT solver has been an ongoing project during my entire time as a PhD student. The project has gone through several phases of excitement and frustration. For example, at the end of 2014, we were stressed about how to obtain good feasible solutions (upper bounds), since these are not readily obtained during the iterations of the ESH algorithm. In its current form, SHOT is an efficient integration of several techniques and is able to efficiently solve problems by constructing tight polyhedral outer approximations and obtaining feasible solutions through primal strategies. A detailed description of the solver, as well as numerical comparisons, was presented in Paper VI. According to the numerical results presented in Paper VI, and the solver comparison in [123], SHOT currently seems to be the most efficient solver for convex MINLP problems. The solver is now available as an open-source project, and we hope to spread the knowledge we have obtained. However, SHOT is still quite a simple solver, and there is room for improvement. For example, at the moment SHOT does not use any preprocessing based on the nonlinear functions to perform bound tightening and range reduction.

Comparing algorithms for convex MINLP problems is not entirely straightforward. Due to the combinatorial nature, it is not easy to directly analyze the convergence rate of the algorithms. The convergence rate of decomposition-based algorithms is, of course, limited by the complexity of the sub-problems. However, if we only analyze the main algorithm, would it be possible to obtain a useful limit on the number of main iterations needed to solve a problem with specific properties? The author, at least, is not aware of such results. However, the number of main iterations is not necessarily a good metric for comparing algorithms. Even if two algorithms solve similar sub-problems, the sub-problems may become significantly more expensive in one of the algorithms. The most commonly used metric for comparing MINLP algorithms and solvers is, therefore, the total time needed to solve a problem. The time is obviously computer specific, but it accounts for the complexity of the different sub-problems, and the time difference between solvers should also be comparable on different platforms.

The algorithms and solvers considered here are mainly compared by testing them on problems from the MINLPLib and comparing the time needed to solve the problems. Using this set of test problems is, of course, somewhat limiting and does not necessarily tell how well-suited an algorithm or solver is for a specific problem. However, the problem library contains a large variety of problems originating from different practical applications and theoretical test problems. Currently, MINLPLib contains 366 convex MINLP problems, and the problem library is continuously growing. The test set provided by MINLPLib should, therefore, be a decent test for evaluating the performance of solvers and algorithms.

5.2 Future ideas

In this final section, I want to share some thoughts and ideas on future plans and research topics within convex MINLP.

Some additional primal strategies are planned to be implemented in the SHOT solver, to improve its capability for solving challenging problems. The center-cut algorithm could quite easily be incorporated as a primal heuristic in the solver, and would only require minor changes of the sub-problems. A round-and-project technique, utilizing points on the boundary of the integer relaxed feasible set, could also work well within the solver. Solutions on the boundary of the integer relaxed feasible set are frequently obtained by the root-search procedure. However, these solutions are unlikely to have integer values. An integer solution can be obtained by simply rounding the solution, although the rounded solution is unlikely to satisfy all the linear constraints of the MINLP problem. To hopefully obtain a feasible integer solution, the rounded solution can be projected onto the polyhedral outer approximation by solving an LP problem. The round-and-project technique would not require any expensive computations and could be well suited as a primal heuristic in SHOT.

At the moment, SHOT does not have any functionality for dealing with non-convex MINLP problems. In the future, we are also interested in including some global optimization functionality into the solver. It could also be beneficial to incorporate the techniques used by the Q-OA method in the SHOT solver.

We were surprised by the small difference between the multi- and single-tree strategies in SHOT. The LP/NLP-BB approach is commonly considered more efficient than the standard OA approach, and we expected the same difference with SHOT. The early termination of the MILP/MIQP subsolver and utilization of the solution pool improve the performance of the multi-tree strategy quite significantly, as shown in Paper VI. Similar techniques could also be used within an OA framework, and there is no obvious reason why similar advantages would not be obtained within an OA-based solver. According to the authors best knowledge, there is no OA-based solver directly utilizing these features, and it could definitely be worth testing.

Currently, the author is not aware of any efficient technique for warm-starting a convex MINLP solver. By warm-starting the author refers to techniques to utilize information about a known solution, or information learned from solving a closely related problem, to significantly reduce the computational effort needed to solve the problem. Tests have shown that starting some solvers at the optimal solution may even result in more iterations and slower progress, compared to a normal start. For many practical applications, a feasible solution may be known in advance and it would be desirable to utilize the knowledge somehow. Warm-starting in convex MINLP would have several benefits, but at the moment this remains an open research topic.

Bibliography

[1] K. Abhishek, S. Leyffer, and J. Linderoth. FilMINT: An outer approximation-based solver for convex mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 22(4):555–567, 2010. (21, 35, 57)

[2] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. (14, 37)

[3] T. Achterberg and R. Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of Combinatorial Optimization*, pp. 449–481. Springer, 2013. (2)

[4] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005. (12)

[5] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998. (18)

[6] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 46(9):1769–1797, 2000. (18)

[7] N. Agin. Optimum seeking with branch and bound. *Management Science*, 13(4):B–176, 1966. (12)

[8] F. Allgöwer and A. Zheng. *Nonlinear model predictive control*, Volume 26. Birkhäuser, 2012. (15)

[9] A. Bagirov, N. Karmitsa, and M. M. Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014. (17, 40, 49)

[10] E. Balas. A duality theorem and an algorithm for (mixed-)integer nonlinear programming. *Linear Algebra and its Applications*, 4(4):341–352, 1971. (30)

[11] E. Balas. Intersection cuts — a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971. (14)

[12] E. Balas. Disjunctive programming. In *Annals of Discrete Mathematics*, Volume 5, pp. 3–51. Elsevier, 1979. (14)

[13] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58(1-3):295–324, 1993. (14, 21)

[14] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 2013. (17)

[15] M. Bazaraa, J. Goode, and C. Shetty. Constraint qualifications revisited. *Management Science*, 18(9):567–573, 1972. (16)

[16] P. Belotti. *Couenne: a user's manual*, 2010. URL `https://www.coin-or.org/Couenne/couenne-user-manual.pdf`. (37)

[17] P. Belotti. Bound reduction using pairs of linear inequalities. *Journal of Global Optimization*, 56(3):787–819, 2013. (36)

[18] P. Belotti, S. Cafieri, J. Lee, and L. Liberti. Feasibility-based bounds tightening via fixed points. In W. Wu and O. Daescu, editors, *International Conference on Combinatorial Optimization and Applications*, pp. 65–76. Springer, 2010. (36)

[19] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013. (21, 36, 37, 57)

[20] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962. (29)

[21] D. E. Bernal, S. Vigerske, F. Trespalacios, and I. E. Grossmann. Improving the performance of DICOPT in convex MINLP problems using a feasibility pump. *Preprint, Optimization Online*, 2017. URL `http://www.optimization-online.org/DB_HTML/2017/08/6171.html`. (47)

[22] T. Berthold. *Heuristic algorithms in global MINLP solvers*. PhD thesis, Technische Universität Berlin, 2014. (36)

[23] T. Berthold. RENS — the optimal rounding. *Mathematical Programming Computation*, 6(1):33–54, 2014. (36)

[24] T. Berthold and A. M. Gleixner. Undercover: a primal MINLP heuristic exploring a largest sub-MIP. *Mathematical Programming*, 144(1-2):315–346, 2014. (36)

[25] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific Belmont, 1999. (16)

[26] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 2014. (17)

[27] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*, Volume 6. Athena Scientific Belmont, 1997. (10, 11)

[28] D. Bertsimas and R. Weismantel. *Optimization over Integers*, Volume 13. Dynamic Ideas Belmont, 2005. (14)

[29] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. arXiv preprint:1209.5145, 2012. (37)

[30] L. T. Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010. (15, 17)

[31] L. T. Biegler and I. E. Grossmann. Retrospective on optimization. *Computers & Chemical Engineering*, 28(8):1169–1192, 2004. (2)

[32] L. T. Biegler, I. E. Grossmann, and A. W. Westerberg. Systematic methods for chemical process design. 1997. (1)

[33] J. Bisschop. *AIMMS Optimization Modeling*. Lulu.com, 2006. (37)

[34] E. R. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice — closing the gap. In *IFIP Conference on System Modeling and Optimization*, pp. 19–49. Springer, 1999. (14)

[35] R. Bixby and E. Rothberg. Progress in computational mixed integer programming- A look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37–41, 2007. (2)

[36] R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1):3–15, 2002. (10)

[37] P. Bonami and M. A. Lejeune. An exact solution approach for portfolio optimization problems under stochastic and integer constraints. *Operations Research*, 57(3):650–670, 2009. (17)

[38] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008. (21, 35, 37, 55)

[39] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, 119(2):331–352, 2009. (36, 47)

[40] B. Borchers and J. E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research*, 21(4):359–367, 1994. (21)

[41] F. Boukouvala, R. Misener, and C. A. Floudas. Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. *European Journal of Operational Research*, 252(3):701–727, 2016. (2)

[42] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. (7, 8, 9, 15, 16, 17, 31, 40, 42, 46)

[43] A. Brook, D. Kendrick, and A. Meeraus. GAMS, a user's guide. *ACM Signum Newsletter*, 23 (3-4):10–11, 1988. (37)

[44] M. R. Bussieck and S. Vigerske. MINLP solver software. In *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library, 2010. (37)

[45] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large-scale Nonlinear Optimization*, pp. 35–59. Springer, 2006. (17, 37)

[46] A. Cambini and L. Martein. *Generalized Convexity and Optimization*. Springer, 2009. (8, 9)

[47] W. Cao and G. J. Lim. Optimization models for cancer treatment planning. In *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library, 2011. (17)

[48] M. T. Çezik and G. Iyengar. Cuts for mixed 0-1 conic programming. *Mathematical Programming*, 104(1):179–202, 2005. (21)

[49] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete mathematics*, 4(4):305–337, 1973. (13)

[50] J. Clausen. Branch and bound algorithms-principles and examples. *Department of Computer Science, University of Copenhagen*, pp. 1–30, 1999. (12)

[51] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer Programming, Volume 271 of Graduate Texts in Mathematics*. Springer Berlin, 2014. (14)

[52] A. R. Conn, G. Gould, and P. L. Toint. *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, Volume 17. Springer Science & Business Media, 2013. (17)

[53] W. Cook, R. Kannan, and A. Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47(1-3):155–174, 1990. (14)

[54] R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255, 1965. (20)

[55] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. A storm of feasibility pumps for nonconvex MINLP. *Mathematical programming*, 136(2):375–402, 2012. (36)

[56] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 2016. (9, 10, 11)

[57] G. B. Dantzig. *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*. Wiley. (9)

[58] G. B. Dantzig. On integer and partial integer linear programming problems. *Rand Corporation*, 1958. (11)

[59] W. de Oliveira. Regularized optimization methods for convex MINLP problems. *TOP*, 24 (3):665–692, 2016. (49)

[60] D. den Hertog, J. Kaliski, C. Roos, and T. Terlaky. A logarithmic barrier cutting plane method for convex programming. *Annals of Operations Research*, 58(2):67–98, 1995. (49)

[61] A. S. Drud. CONOPT — a large-scale GRG code. *ORSA Journal on Computing*, 6(2):207–216, 1994. (17)

[62] I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. (37)

[63] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36(3):307–339, 1986. (26, 27)

[64] J. Elzinga and T. G. Moore. A central cutting plane algorithm for the convex programming problem. *Mathematical Programming*, 8(1):134–145, 1975. (46)

[65] V.-P. Eronen, M. M. Mäkelä, and T. Westerlund. Extended cutting plane method for a class of nonsmooth nonconvex MINLP problems. *Optimization*, 64(3):641–661, 2015. (24)

[66] FICO. *Xpress-optimizer reference manual*, 2017. URL `https://www.artelys.com/uploads/pdfs/Xpress/Xpress_Optimizer_2447PS.pdf`. (11, 14)

[67] M. Fischetti and A. Lodi. Heuristics in mixed integer programming. In *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library, 2011. (36)

[68] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2013. (14)

[69] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(1):327–349, 1994. (26, 27, 28, 47, 51)

[70] R. Fletcher and S. Leyffer. User manual for filterSQP. Numerical Analysis Report NA/181, Department of Mathematics, University of Dundee, 1998. (17)

[71] O. E. Flippo and A. H. R. Kan. Decomposition in general mathematical programming. *Mathematical Programming*, 60(1-3):361–382, 1993. (30)

[72] C. A. Floudas. *Nonlinear and Mixed-integer Optimization: Fundamentals and Applications*. Oxford University Press, 1995. (2, 30, 31)

[73] C. A. Floudas. *Deterministic Global Optimization, vol. 37 of Nonconvex Optimization and its Applications*. Kluwer Academic, 2000. (17, 18)

[74] J. Forrest. Clp user's guide, 2004. URL `https://projects.coin-or.org/Clp`. (11)

[75] J. Forrest. Cbc user's guide, 2005. URL `https://projects.coin-or.org/Cbc`. (14)

[76] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM review*, 44(4):525–597, 2002. (17)

[77] B. A. Foster and D. M. Ryan. An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, 27(2):367–384, 1976. (11)

[78] R. Fourer, D. Gay, and B. Kernighan. *AMPL*. Boyd & Fraser Danvers, MA, 1993. (37)

[79] A. Frangioni and C. Gentile. Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106(2):225–236, 2006. (21)

[80] *SBB GAMS manual*. GAMS, 2018. URL `https://www.gams.com/latest/docs/S_SBB.html`. (37, 55)

[81] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice — a survey. *Automatica*, 25(3):335–348, 1989. (1)

[82] H. Gassmann, J. Ma, K. Martin, and W. Sheng. *Optimization Services 2.10 User's Manual*, 2015. URL `http://projects.coin-or.org/svn/OS/trunk/OS/doc/osUsersManual.pdf`. (55)

[83] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972. (29, 31)

[84] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. (17)

[85] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859, 1961. (11)

[86] R. Gomory. An algorithm for the mixed integer problem. Technical report, RAND Corp. Santa Monica, CA, 1960. (12)

[87] R. E. Gomory. An algorithm for integer solutions to linear programs. *Recent Advances in Mathematical Programming*, 64:260–302, 1963. (14)

[88] R. E. Gomory et al. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958. (11, 12, 14, 21)

[89] I. E. Grossmann. MINLP optimization strategies and algorithms for process synthesis. Technical report, Carnegie Mellon University, 1989. (17)

[90] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, 2002. (29, 34)

[91] I. E. Grossmann and Z. Kravanja. Mixed-integer nonlinear programming: A survey of algorithms and applications. In L. T. Biegler, T. F. Coleman, A. R. Conn, and F. N. Santosa, editors, *Large-scale Optimization with Applications*, pp. 73–100. Springer, 1997. (2, 33)

[92] I. E. Grossmann, J. Viswanathan, A. Vecchietti, R. Raman, E. Kalvelagen, et al. Gams/dicopt: A discrete continuous optimization package. *GAMS Corporation Inc*, 2002. (37, 55)

[93] Z. Gu, G. L. Nemhauser, and M. W. Savelsbergh. Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming*, 85(3):439–467, 1999. (14)

[94] O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management science*, 31(12):1533–1546, 1985. (20)

[95] Gurobi. *Gurobi optimizer reference manual*. Gurobi Optimization, LLC, 2018. URL `http://www.gurobi.com/documentation/8.0/refman.pdf`. (11, 14)

[96] S.-P. Han. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11(1):263–282, 1976. (49)

[97] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. *Pyomo-optimization Modeling in Python*, Volume 67. Springer, 2012. (37)

[98] E. M. Hendrix, C. J. Mecking, and T. H. Hendriks. Finding robust solutions for product design problems. *European Journal of Operational Research*, 92(1):28–36, 1996. (46)

[99] H. Hijazi, P. Bonami, and A. Ouorou. An outer-inner approximation for separable mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 26(1):31–44, 2013. (37, 45)

[100] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms I: Fundamentals*, Volume 305. Springer Science & Business Media, 2013. (24)

[101] F. L. Hitchcock. The distribution of a product from several sources to numerous localities. *Studies in Applied Mathematics*, 20(1-4):224–230, 1941. (10)

[102] R. Horst, P. M. Pardalos, and N. Van Thoai. *Introduction to Global Optimization*. Springer Science & Business Media, 2000. (18)

[103] M. Hunting. The AIMMS outer approximation algorithm for MINLP. Technical report, AIMMS B.V., 2011. (37, 55)

[104] T. Ibaraki. Theoretical comparisons of search strategies in branch-and-bound algorithms. *International Journal of Computer & Information Sciences*, 5(4):315–344, 1976. (12)

[105] IBM ILOG CPLEX Optimization Studio. CPLEX user's manual, version 12.7, 2017. (11, 14)

[106] L. S. Inc. *LINDO User's Manual*, 2017. URL `https://www.lindo.com/downloads/PDF/LindoUsersManual.pdf`. (37)

[107] J. Kallrath. *Modeling Languages in Mathematical Optimization*, Volume 88. Springer Science & Business Media, 2013. (37)

[108] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pp. 302–311. ACM, 1984. (10)

[109] W. Karush. Minima of functions of several variables with inequalities as side conditions. *Master thesis, University of Chicago*, 1939. (16)

[110] J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial & Applied Mathematics*, 8(4):703–712, 1960. (22, 24, 49)

[111] L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. (10)

[112] M. R. Kılınç, J. Linderoth, and J. Luedtke. Lift-and-project cuts for convex mixed integer nonlinear programs. *Mathematical Programming Computation*, 9(4):499–526, 2017. (21)

[113] K. C. Kiwiel. Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities. *Mathematical Programming*, 69(1-3): 89–109, 1995. (49)

[114] V. Klee, G. Minty, and O. Shisha. How good is the simplex method? technical report. *Inequalities III, Academic Press, New York*, 1972. (10)

[115] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, et al. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103, 2011. (14)

[116] G. R. Kocis and I. E. Grossmann. Relaxation strategy for the structural optimization of process flow sheets. *Industrial & Engineering Chemistry Research*, 26(9):1869–1880, 1987. (26)

[117] G. R. Kocis and I. E. Grossmann. Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. *Industrial & Engineering Chemistry Research*, 27(8):1407–1421, 1988. (26)

[118] T. C. Koopmans. Optimum utilization of the transportation system. *Econometrica: Journal of the Econometric Society*, pp. 136–146, 1949. (10)

[119] T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica: Journal of the Econometric Society*, pp. 53–76, 1957. (1)

[120] O. Kröger, C. Coffrin, H. Hijazi, and H. Nagarajan. Juniper: an open-source nonlinear branch-and-bound solver in Julia. arXiv preprint: 1804.07332, 2018. (37)

[121] J. Kronqvist, A. Lundell, and T. Westerlund. The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. *Journal of Global Optimization*, 64 (2):249–272, 2016. (21)

[122] J. Kronqvist, A. Lundell, and T. Westerlund. A center-cut algorithm for solving convex mixed-integer nonlinear programming problems. In *Computer Aided Chemical Engineering*, Volume 40, pp. 2131–2136. Elsevier, 2017. (47)

[123] J. Kronqvist, D. E. Bernal, A. Lundell, and I. E. Grossmann. A Review and Comparison of Solvers for Convex MINLP. *Preprint, Optimization Online*, 2018. URL `http://www.optimization-online.org/DB_HTML/2018/06/6650.html`. (21, 35, 37, 55, 60)

[124] H. Kuhn and T. A.W. Nonlinear programming. In *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pp. 481–492, 1951. (16)

[125] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pp. 497–520, 1960. (11, 20)

[126] L. S. Lasdon, A. D. Waren, A. Jain, and M. Ratner. Design and testing of a generalized reduced gradient code for nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 4(1):34–50, 1978. (17)

[127] T. Lastusilta. GAMS MINLP solver comparisons and some improvements to the AlphaECP algorithm. PhD thesis, Åbo Akademi University, 2011. (37)

[128] J. Lee and S. Leyffer, editors. *Mixed Integer Nonlinear Programming*, Volume 154. Springer Science & Business Media, 2011. (19)

[129] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69(1-3):111–147, 1995. (49)

[130] C. E. Lemke. The dual method of solving the linear programming problem. *Naval Research Logistics (NRL)*, 1(1):36–47, 1954. (10)

[131] S. Leyffer. *Deterministic methods for mixed integer nonlinear programming*. PhD thesis, 1993. (35, 57)

[132] S. Leyffer. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational optimization and applications*, 18(3):295–309, 2001. (21)

[133] S. Leyffer, J. Linderoth, J. Luedtke, A. Miller, and T. Munson. Applications and algorithms for mixed integer nonlinear programming. In *Journal of Physics: Conference Series*, Volume 180. IOP Publishing, 2009. (2)

[134] J. Li, R. Misener, and C. A. Floudas. Scheduling of crude oil operations under demand uncertainty: A robust optimization framework coupled with global optimization. *AIChE Journal*, 58(8):2373–2396, 2012. (17)

[135] L. Liberti and N. Maculan. *Global Optimization: From Theory to Implementation*, Volume 84. Springer Science & Business Media, 2006. (18, 36)

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

[137] M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma. Extended formulations in mixed-integer convex programming. In Q. Louveaux and M. Skutella, editors, *Integer Programming and Combinatorial Optimization: 18th International Conference, IPCO 2016*, pp. 102–113. Springer International Publishing, 2016. (37)

[138] M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma. Polyhedral approximation in mixed-integer convex optimization. *Mathematical Programming*, pp. 1–30, 2017. (45)

[139] D. G. Luenberger, Y. Ye, et al. *Linear and Nonlinear Programming*, Volume 2. Springer, 1984. (10)

[140] A. Lundell and T. Westerlund. Solving global optimization problems using reformulations and signomial transformations. *Computers & Chemical Engineering*, 2017. (available online). (18)

[141] A. Lundell, J. Westerlund, and T. Westerlund. Some transformation techniques with applications in global optimization. *Journal of Global Optimization*, 43(2-3):391–405, 2009. (18)

[142] A. Lundell, A. Skjäl, and T. Westerlund. A reformulation framework for global optimization. *Journal of Global Optimization*, 57(1):115–141, 2013. (18)

[143] A. Mahajan, S. Leyffer, J. Linderoth, J. Luedtke, and T. Munson. Minotaur: A Mixed-Integer Nonlinear Optimization Toolkit. *Preprint, Optimization Online*, 2017. URL `http://www.optimization-online.org/DB_FILE/2017/10/6275.pdf`. (35, 55)

[144] A. Makhorin. *GLPK (GNU linear programming kit)*, 2008. URL `http://www.gnu.org/software/glpk/`. (11)

[145] T. E. Marlin, A. N. Hrymak, et al. Real-time operations optimization of continuous processes. In *AIChE Symposium Series*, Volume 93, pp. 156–164. New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997. (15)

[146] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976. (18)

[147] F. Messine. Deterministic global optimization using interval constraint propagation techniques. *RAIRO-Operations Research*, 38(4):277–293, 2004. (36)

[148] MINLPLib. Mixed-integer nonlinear programming library, 2018. URL `http://www.minlplib.org/`. [Accessed 27-May-2018]. (55)

[149] R. Misener and C. A. Floudas. Global optimization of large-scale generalized pooling problems: quadratically constrained MINLP models. *Industrial & Engineering Chemistry Research*, 49(11):5424–5438, 2010. (17)

[150] R. Misener and C. A. Floudas. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, 57(1):3–50, 2013. (18)

[151] R. Misener and C. A. Floudas. ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014. (18, 37)

[152] R. Misener, M. C. Allenby, M. Fuentes-Garí, K. Gupta, T. Wiggins, N. Panoskaltsis, E. N. Pistikopoulos, and A. Mantalaris. Stem cell biomanufacturing under uncertainty: A case study in optimizing red blood cell production. *AIChE Journal*, 2017. (17)

[153] J. E. Mitchell. Branch and cut. In *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library, 2011. (14)

[154] H. Mittelmann. Benchmarks for optimization software, 2018. URL `http://plato.la.asu.edu/bench.html`. [Accessed 20-April-2018]. (11)

[155] B. A. Murtagh and M. A. Saunders. Minos 5.5 user's guide. Technical report, Stanford University, 1998. (17)

[156] S. M. Neiro and J. M. Pinto. Multiperiod optimization for production planning of petroleum refineries. *Chemical Engineering Communications*, 192(1):62–88, 2005. (17)

[157] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1988. (14)

[158] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, Volume 87. Springer, 2004. (17, 49)

[159] J. Nocedal and S. J. Wright. *Sequential Quadratic Programming*. Springer, 2006. (49)

[160] I. Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*, Volume 152. Springer Science & Business Media, 2006. (18)

[161] I. Nowak, N. Breitfeld, E. M. Hendrix, and G. Njacheun-Njanzoua. Decomposition-based inner-and outer-refinement algorithms for global optimization. *Journal of Global Optimization*, pp. 1–17, 2018. (18)

[162] M. W. Padberg, T. J. Van Roy, and L. A. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33(4):842–861, 1985. (14)

[163] J. M. Pinto and I. E. Grossmann. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. *Industrial & Engineering Chemistry Research*, 34(9):3037–3051, 1995. (11)

[164] Y. Pochet and L. A. Wolsey. *Production Planning by Mixed Integer Programming*. Springer Science & Business Media, 2006. (1)

[165] R. Pörn, I. Harjunkoski, and T. Westerlund. Convexification of different classes of non-convex MINLP problems. *Computers & Chemical Engineering*, 23(3):439–448, 1999. (18)

[166] R. Pörn. Mixed integer non-linear programming: convexification techniques and algorithm development. PhD thesis, Åbo Akademi University, 2000. (42)

[167] I. Quesada and I. E. Grossmann. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers & Chemical Engineering*, 16(10-11):937–947, 1992. (21, 29, 34)

[168] A. Quist, R. Van Geemert, J. Hoogenboom, T. Ílles, C. Roos, and T. Terlaky. Application of nonlinear optimization to reactor core fuel reloading. *Annals of Nuclear Energy*, 26(5): 423–448, 1999. (17)

[169] R. T. Rockafellar. *Convex Analysis. Princeton Landmarks in Mathematics*. Princeton University Press, 1997. (7, 8, 9)

[170] H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995. (18)

[171] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, 1996. (36)

[172] N. V. Sahinidis. *BARON user's manual*, 2018. URL `https://minlp.com/downloads/docs/baron\%20manual.pdf`. (37)

[173] N. V. Sahinidis. BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996. (18, 37, 55)

[174] N. Sahinidis and I. E. Grossmann. Convergence properties of generalized Benders decomposition. *Computers & Chemical Engineering*, 15(7):481–491, 1991. (30)

[175] N. Sahinidis and I. E. Grossmann. MINLP model for cyclic multiproduct scheduling on continuous parallel lines. *Computers & Chemical Engineering*, 15(2):85–103, 1991. (17)

[176] N. Sahinidis, I. Grossmann, R. Fornari, and M. Chathrathi. Optimization model for long range planning in the chemical industry. *Computers & Chemical Engineering*, 13(9):1049–1063, 1989. (11)

[177] A. Schrijver. *Theory of Linear and Integer programming*. John Wiley & Sons, 1998. (11, 14)

[178] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, Volume 24. Springer Science & Business Media, 2003. (14)

[179] J. P. Shectman and N. V. Sahinidis. A finite algorithm for global minimization of separable concave programs. *Journal of Global Optimization*, 12(1):1–36, 1998. (36)

[180] M. Slater et al. Lagrange multipliers revisited. Technical report, Cowles Foundation for Research in Economics, Yale University, 1959. (16)

[181] E. M. Smith and C. C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex minlps. *Computers & Chemical Engineering*, 23(4-5):457–478, 1999. (18)

[182] R. A. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86(3):515–532, 1999. (20)

[183] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005. (37, 45)

[184] M. Tawarmalani and N. V. Sahinidis. Semidefinite relaxations of fractional programs via novel convexification techniques. *Journal of Global Optimization*, 20(2):133–154, 2001. (18)

[185] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Volume 65. Springer Science & Business Media, 2002. (2, 18)

[186] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591, 2004. (18, 21)

[187] P. Toth and D. Vigo. *The vehicle routing problem*. SIAM, 2002. (1)

[188] F. Trespalacios and I. E. Grossmann. Review of mixed-integer nonlinear and generalized disjunctive programming methods. *Chemie Ingenieur Technik*, 86(7):991–1012, 2014. (37)

[189] G. Van Rossum. Python programming language. In *USENIX Annual Technical Conference*, Volume 41, page 36, 2007. (37)

[190] A. F. Veinott Jr. The supporting hyperplane method for unimodal programming. *Operations Research*, 15(1):147–152, 1967. (42)

[191] S. Vigerske and A. Gleixner. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optimization Methods and Software*, 33(3):563–593, 2018. (18, 37, 55)

[192] J. Viswanathan and I. E. Grossmann. A combined penalty function and outer-approximation method for MINLPoptimization. *Computers & Chemical Engineering*, 14(7): 769–782, 1990. (26)

[193] 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. (17)

[194] J. Westerlund, L. G. Papageorgiou, and T. Westerlund. A MILP model for n-dimensional allocation. *Computers & Chemical Engineering*, 31(12):1702–1714, 2007. (11)

[195] T. Westerlund and T. Lastusilta. *AlphaECP GAMS user's manual*, 2008. URL `http://www.gams.com/latest/docs/S_ALPHAECP.html`. (55)

[196] T. Westerlund and F. Petterson. An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19:131–136, 1995. (24)

[197] T. Westerlund and K. Lundqvist. *Alpha-ECP, version 5.01: An interactive MINLP-solver based on the extended cutting plane method*. Technical report, Åbo Akademi, 2001. (37)

[198] T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3(3):253–280, 2002. (9, 24)

[199] T. Westerlund, V.-P. Eronen, and M. M. Mäkelä. On solving generalized convex MINLP problems using supporting hyperplane techniques. *Journal of Global Optimization, available online*, 2018. (44)

[200] L. A. Wolsey. A resource decomposition algorithm for general mathematical programs. In *Mathematical Programming at Oberwolfach*, pp. 244–257. Springer, 1981. (30, 33)

[201] L. A. Wolsey. *Integer Programming. Series in Discrete Mathematics and Optimization*. Wiley-Interscience New Jersey, 1998. (36)

[202] T. F. Yee and I. E. Grossmann. Simultaneous optimization models for heat integration — II. Heat exchanger network synthesis. *Computers & Chemical Engineering*, 14(10):1165–1184, 1990. (17)

[203] T. F. Yee, I. E. Grossmann, and Z. Kravanja. Simultaneous optimization models for heat integration — I. Area and energy targeting and modeling of multi-stream exchangers. *Computers & Chemical Engineering*, 14(10):1151–1164, 1990. (1)

[204] A. Zanette, M. Fischetti, and E. Balas. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Mathematical Programming*, 130(1):153–176, 2011. (14)

[205] S. Zaourar and J. Malick. Quadratic stabilization of Benders decomposition. preprint, 2014. URL `https://hal.archives-ouvertes.fr/hal-01181273`. (49)

[206] Y. Zhu and T. Kuno. A disjunctive cutting-plane-based branch-and-cut algorithm for 0-1 mixed-integer convex nonlinear programs. *Industrial & Engineering Chemistry Research*, 45 (1):187–196, 2006. (21)

Dissertations published by Process Design and Systems Engineering
ISSN 2489-7272
978-952-12-3734-8
978-952-12-3735-5 (pdf)

Painosalama Oy
Åbo 2018