

**Decomposition in Multistage Stochastic Programming  
and  
a Constraint Integer Programming Approach to  
Mixed-Integer Nonlinear Programming**

DISSERTATION

zur Erlangung des akademischen Grades

Dr. rer. nat.  
im Fach Mathematik

eingereicht an der  
Mathematisch-Naturwissenschaftlichen Fakultät II  
Humboldt-Universität zu Berlin

von  
**Dipl.-Inf. Dipl.-Math. Stefan Vigerske**

Präsident der Humboldt-Universität zu Berlin:  
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:  
Prof. Dr. Elmar Kulke

Gutachter:

1. Prof. Dr. Werner Römisch
2. Prof. Dr. Rüdiger Schultz
3. Pierre Bonami, Ph.D.

**eingereicht am:** 11.01.2012

**Tag der mündlichen Prüfung:** 21.08.2012

## Abstract

This thesis contributes to two topics in mathematical programming: stochastic optimization and mixed-integer nonlinear programming.

Stochastic programming allows to consider uncertainty in the parameters of an optimization problem. Assuming the uncertainty can be modeled by a stochastic process  $\xi$ , the task of a stochastic program is to find a decision process  $x(\xi)$  that obeys certain constraints and minimizes a function on  $x(\cdot)$ . For the latter, usually a combination of expected costs and a measure on the risk of deviating from the expected costs is used. To solve a stochastic program, the probability distribution is often approximated by a discrete one, which raises questions on the behavior of the optimal value and the set of optimal solutions of a stochastic program under perturbations of the underlying probability distribution. In this thesis, we extend quantitative continuity results for two-stage stochastic mixed-integer linear programs to include situations with simultaneous uncertainty in costs and right-hand side.

A meaningful approximation by a discrete distribution often requires the use of many scenarios. The deterministic equivalent that corresponds to the discretized stochastic program can then easily be too large for standard solution approaches. However, at least from the discretization, the problem is well structured, which makes it attractive for decomposition algorithms. After an extended review on decomposition algorithm for two- and multistage stochastic linear and mixed-integer linear programs, we discuss an extension of the Nested Benders Decomposition method for multistage stochastic linear programs to exploit the advantages of so-called recombining scenario trees. Further, we investigate a combination of Nested Benders Decomposition with Nested Column Generation. Recombining scenario trees have the advantage that they can be used to approximate stochastic processes with a certain short-term memory in a way that much less nodes are required than for an equivalent non-recombining tree. However, time coupling constraints in a stochastic program prohibit a dynamic programming formulation, as it is fundamental for certain fast solution algorithms, on a recombining tree. Our extensions to the Nested Benders and Column Generation methods show how to overcome these difficulties, so that especially “manystage” stochastic linear programs with recombining scenario trees can be solved efficiently. As an application of this method, we consider the optimal scheduling of a regional energy system including wind power and energy storages and further to the planning of investment decisions for the expansion of an energy supply system.

The second part of this thesis concerns the solution of mixed-integer nonlinear optimization problems (MINLPs). Discrete decisions, nonlinearity, and possible nonconvexity of the involved nonlinear functions combines the areas of mixed-integer linear programming, nonlinear programming, and global optimization into a single problem class. We give a comprehensive overview on the state-of-the-art in algorithms and solver technology for MINLPs.

Subsequently, we show that some of these algorithm can be applied for the solution of MINLPs within the constraint integer programming framework SCIP. The paradigm of constraint integer programming combines modeling and solving techniques from the fields of constraint programming (CP), mixed-integer linear programming (MIP), and satisfiability testing. For MINLP, the availability of a CIP framework like SCIP allows us to utilize the power of already existing MIP and CP technologies to handle the linear and discrete parts of the problem. Thus, we focus mainly on the handling of convex and nonconvex nonlinear constraints. The corresponding domain propagation, outer-approximation, and reformulation techniques are discussed in this thesis.

Finally, in an extensive computational study, we investigate the performance of our approach on applications from open pit mine production scheduling and water distribution network design and on various benchmarks sets. The results show that SCIP has become a competitive solver for MINLPs, even though it does not yet include all techniques that are available in other solvers.

# Preface

This thesis reviews much of my work in mathematical optimization of the last six years.

In 2001, I started to work in this field as a student assistant in a research project lead by Ivo Nowak at the Department of Mathematics at Humboldt University Berlin and Turang Ahadi-Oskui and George Tsatsaronis at the Institute for Energy Engineering at Technical University Berlin. The goal of this cooperation was the development of methods to solve mixed-integer (nonconvex) nonlinear programs (MINLP) and their application to the simultaneous optimization of structural and operational parameters of a complex energy conversion plant [Ahadi-Oskui, 2006, Ahadi-Oskui, Vigerske, Nowak, and Tsatsaronis, 2010]. The optimization model combined discrete decisions for the structure of the plant with nonconvex nonlinear equations to describe the thermodynamic behavior of the components. My contribution to this project comprised participation in the development of the MINLP solver LAGO (**L**agrangian **G**lobal **O**ptimizer) [Nowak, Alperin, and Vigerske, 2003, Nowak and Vigerske, 2008]. While LAGO never reached the maturity of state-of-the-art solvers for MINLP, it proved a useful tool for experimenting with novel relaxation and decomposition methods for MINLPs [Nowak, 2005] such as quadratic underestimation of general nonconvex functions, Lagrangian heuristics, and branch-cut-and-price. At that time, LAGO was the only MINLP solver that was available to us as source code. The latter was crucial for the optimization of the energy conversion system, since it allowed for both debugging the optimization model and tuning the solver so that at least good feasible solutions were found. After the project finished, the source code of LAGO became publicly available at COIN-OR<sup>1</sup> [Lougee-Heimer, 2003].

In 2005, when working on my diploma thesis in mathematics [Vigerske, 2005, Nowak and Vigerske, 2007], Werner Römisich and Ivo Nowak introduced me to another – similarly exciting – field of mathematical programming: stochastic programming (optimization under uncertainty).

After finishing my diploma thesis, I continued working in this field in a project that investigated the use of renewable energy sources in a decentralized energy supply system and the development of associated novel mathematical optimization methods [Schultz and Wagner, 2009]. As part of a network of seven German research groups (from both energy science and mathematical optimization), Alexa Epe at Ruhr-University Bochum, Oliver Woll at the University of Duisburg-Essen, and Christian Küchler and I at Humboldt-University investigated the potential of energy storages to decouple supply and demand for a cost optimal electricity supply in the context of increasing generation of wind energy [Epe, Küchler, Römisich, Vigerske, Wagner, Weber, and Woll, 2007, 2009a,b] (cf. Chapter 5). While the optimization of energy storages and investment decisions required

---

<sup>1</sup><https://projects.coin-or.org/LaGO>

the consideration of long-term planning horizons, the highly fluctuating wind energy input required a detailed temporal resolution. Due to its dimension, the resulting multistage stochastic optimization problems could not be tackled by standard solution approaches. Motivated by previous work of Christoph Weber on the use of recombining scenario trees for similar problems, Christian and I developed an extension of the well-known Nested Benders Decomposition method to exploit the advantages of recombining scenario tree also in the context of multistage stochastic programs with time-coupling constraints [Küchler and Vigerske, 2007, 2009]. After the project had finished, the software was still used by Alexa Epe to extend our investigations on the use of energy storages from a small dispersed energy supply system to an aggregated system of all existing power plants in Germany [Epe, 2011]. Further, Susanne Wruck investigated the effect of transmission losses on the use of renewable energy sources for energy supply [Wruck, 2009]. To obtain a quantitative and numerically computable estimate on the benefit of using a recombining scenario tree instead of a classical one, Christian and I developed an out-of-sample based method to evaluate approximation techniques for stochastic programs [Küchler and Vigerske, 2010] (cf. Section 4.7).

While the considered models for energy supply were essentially huge linear optimization problems, Werner Römisch motivated me to look also into the area of mixed-integer linear stochastic programming. As a result, we extended quantitative continuity results for two-stage stochastic mixed-integer linear programs under perturbation of the probability measure from Schultz [1996] to include situations with simultaneous uncertainty in costs and right-hand side [Römisch and Vigerske, 2008] (cf. Chapter 2). Moreover, I surveyed the state-of-the-art of decomposition methods for two-stage stochastic mixed-integer linear programs [Römisch and Vigerske, 2010] (cf. Section 3.2).

During these years, I tried to keep the development of LAGO alive, e.g., together with Marc Jüdes at Technical University Berlin, who used LAGO to extend Turang's work on the optimization of complex energy conversion plants by considering also partial load operation when optimizing a plant design [Jüdes, Tsatsaronis, and Vigerske, 2007, 2009, Jüdes, 2009]. However, in 2008, the main impulse to continue research on MINLP was the suggestion of Martin Grötschel, Thorsten Koch, and Marc Pfetsch from Zuse Institute Berlin to contribute my experience with LAGO for the development of MINLP extensions to the constraint integer programming (CIP) framework SCIP<sup>2</sup>. CIP combines modeling and solving techniques from the fields of constraint programming, mixed-integer linear programming (MIP), and satisfiability testing. The combination of the generality of constraint programming with the solving techniques for MIPs and satisfiability problems often allows to solve optimization problems that are intractable with either of the two methods alone. Tobias Achterberg implemented the concept of CIP in the solver SCIP [Achterberg, 2007] (see also Section 7.2). Due to its plugin-based design, it is an easily customizable and extendable framework for many kinds of branch-cut-and-price algorithms. Furthermore, SCIP already provides a lot of machinery necessary to tackle practically relevant MIPs.

Since 2008, I have been working with Timo Berthold, Ambros Gleixner, and Stefan

---

<sup>2</sup><http://scip.zib.de>

Heinz at Zuse Institute Berlin in a MATHEON<sup>3</sup> project on the development of MINLP plugins in SCIP (cf. Chapter 7). In the beginning, we concentrated on support for (convex and nonconvex) quadratic constraints [Berthold, Heinz, and Vigerske, 2009b, Berthold, Gleixner, Heinz, and Vigerske, 2010a, 2012]. A first real-world application was the optimization of open pit mine production scheduling problems with stockpiles [Bley, Gleixner, Koch, and Vigerske, 2012b] (cf. Section 8.1), where an already challenging combinatorial model (precedence constrained selection of the next block to excavate) is extended by nonconvex quadratic constraints arising from mixing materials of different quality on a stockpile. The convincing results for this application indicated the potential for solving MINLPs with a strong MIP core by SCIP. In the following years, the development of MINLP techniques in SCIP continued and in autumn 2011 resulted in the release of SCIP 2.1.0, which supports the general class of factorable MINLPs. Even though SCIP does not yet include all techniques that are available in other MINLP solvers, the computational study in Chapter 8 indicates, that SCIP has become a competitive solver for MINLPs. It shows its strength particularly on problems which combine a difficult MIP model with a few nonlinear constraints. We hope, that the fundamental support for MINLP in SCIP makes it a useful tool for further research in this area. Some indicators that justify this hope are already available, see Ballerstein et al. [2013], Berthold and Gleixner [2009], Berthold et al. [2011], Gleixner et al. [2012], Huang [2011], and Pfetsch et al. [2012].

Further contributions to the field of MINLP are a comprehensive survey of available MINLP solver software in a joint work with Michael Bussieck from GAMS [Bussieck and Vigerske, 2010] (cf. Section 6.2) and work on a convexity test for rational functions over a polyhedral set [Neun, Sturm, and Vigerske, 2010]. For the latter, Winfried Neun at Zuse Institute Berlin suggested to reduce the problem to a real quantifier elimination problem and to apply methods from the computer logic package REDLOG<sup>4</sup>. Thomas Sturm (now at Max-Planck-Institut für Informatik in Saarbrücken) is a main developer of REDLOG and implemented the convexity tests. The method improves over previous convexity detection methods by giving conclusive results and allowing for arbitrary polyhedral sets as domain.

It may serve as an indicator of the computationally challenging nature of both stochastic programming and MINLP, that although I have been active in both areas alternatingly, I have never worked on a combination of the two. Thus, also for this thesis, both fields are presented separately, which has the added advantage that each part can be read independently from the other one.

---

<sup>3</sup><http://www.matheon.de>

<sup>4</sup><http://redlog.dolzmann.de>

## **Acknowledgments**

First of all, I want to thank Werner Römisch for his support during the last ten years and for giving me all the freedom to conduct my own research. Further, I like to thank his working group (Hernan Alperin, Andreas Eichhorn, Konstantin Emich, Holger Heitsch, René Henrion, Christian Küchler, Hernan Leovey, Andris Möller, Ivo Nowak, Gabriele Radtke, Thorsten Sickenberger, Isabel Wegner-Specht, Renate Winkler), the friday-afternoon-cake-break group, and all other people from the right-hand side of the fourth floor at the second house of the John-von-Neumann building for a most pleasant atmosphere at Humboldt University. Equally, I thank the people from the optimization department at Zuse Institute for a warm and inspiring environment, especially all the people I joined in innumerable coffee breaks (Timo Berthold, Gerald Gamrath, Ambros Gleixner, Stefan Heinz, Matthias Miltenberger, Christian Raack, Markus Reuther, Thomas Schlechte, Jonas Schweiger, Yuji Shinano, Dan Steffy, Rüdiger Stephan, Elmar Swarat, Stefan Weltge, Axel Werner, Michael Winkler, Kati Wolter). Similarly, I thank Michael and Susanne Bussieck, Steven Dirkse, Jan-Hendrik Jagla, Alex Meeraus, Franz Nellisen, and Lutz Westermann at GAMS for repeatedly inviting me to Braunschweig and Washington D.C. (including visits to Amy’s, Boeoymonger, and Clyde’s), for supporting me in my “hobby” of writing software interfaces, and for all the fun with compiling, linking, and debugging on exotic platforms.

I am very thankful to Ivo Nowak for introducing me to MINLP, Werner Römisch for introducing me to stochastic programming, and Marc Pfetsch and Thorsten Koch for offering me the possibility to work at Zuse Institute. I am very grateful to Turang Ahadi-Oskui, Martin Ballerstein, Timo Berthold, Michael Bussieck, Alexa Epe, Ambros Gleixner, Stefan Heinz, Jan-Hendrik Jagla, Marc Jüdes, Christian Küchler, Dennis Michaels, Winfried Neun, Ivo Nowak, Werner Römisch, Thomas Sturm, Lutz Westermann, and Oliver Woll, who worked with me on one or the other project. I thank Tobias Achterberg for making SCIP, which has shown me a lot not only about efficient implementation of branch-and-cut algorithms, but also about the amenities of good software design and coding style. I thank again Timo Berthold, Stefan Heinz, and Marc Pfetsch for their help in getting me started with SCIP and the whole LP/IP developers group at Zuse Institute for their patience when I occupied the new computer blades for the computational experiments in this thesis.

I thank Arnold Neumaier for inviting me to the GICOLAG visiting program in Vienna 2006, where I learned about the COCONUT project. I am grateful to Jon Lee and Andreas Wächter for inviting me to IBM Watson in 2007 and to Jon Lee and Sven Leyffer for inviting me to the IMA Hot Topics Workshop on MINLP in Minneapolis in 2008. I thank Rüdiger Schultz for organizing the “asparagus-meeting” on stochastic optimization in Geldern-Walbeck in 2008. For an opportunity to see the world from down under in 2010, I thank Christina Burt, Gary Froyland, Thorsten Koch, and MATHEON.

I acknowledge financial support from the “Bundesministerium für Bildung und Forschung” (BMBF) under the grant 03SF0312E, from the “Wiener Wissenschafts-, Forschungs- und Technologiefonds” in Vienna, and from MATHEON in the projects B19 and B20. I am grateful to GAMS for providing me with an evaluation license of their

software and the included solvers.

Almost finally, I thank Ambros Gleixner, Timo Berthold, Gerald Gamrath, Dennis Michaels, and Jarungjit Parnjai for supporting me on writing this thesis by proofreading parts of it and I thank Holger Heitsch's cheerful disposition that distracted me well while attempting to write this thesis ☺.

Finally, I thank my family and Jeed for their support and love. I thank Jeed for taking so good care of me and my "pung" in the recent years. I am grateful to Rakporn Dokchan to acquaint me with Jeed and, as a consequence, to Roswitha März and René Lamour for letting Rakporn study at Humboldt University.



# Contents

<b>Preface</b>	<b>iii</b>
<b>1. Stochastic Programming</b>	<b>1</b>
<b>1. Introduction</b>	<b>3</b>
1.1. Two-Stage Stochastic Programs . . . . .	4
1.2. Multistage Stochastic Programs . . . . .	5
1.3. Discretization of Stochastic Programs . . . . .	6
1.3.1. Scenarios . . . . .	6
1.3.2. Scenario Trees . . . . .	7
<b>2. Stability of Two-Stage Stochastic Mixed-Integer Linear Programming Problems</b>	<b>9</b>
2.1. Infima of Mixed-Integer Linear Programs . . . . .	9
2.2. Structural Properties of Two-Stage Stochastic Mixed-Integer Linear Programs	12
2.3. Quantitative Stability of Fully-Random Two-Stage Stochastic Mixed-Integer Linear Programs . . . . .	16
<b>3. Decomposition Algorithms for Stochastic Programming Problems</b>	<b>21</b>
3.1. Temporal Decomposition for Stochastic Linear Programs . . . . .	22
3.1.1. Benders Decomposition . . . . .	22
3.1.2. Nested Benders Decomposition . . . . .	25
3.2. Temporal Decomposition for Two-Stage Stochastic Mixed-Integer Linear Programs . . . . .	29
3.2.1. Convexification of the Expected Recourse Function . . . . .	30
3.2.2. Convexification of the Value Function . . . . .	32
3.3. Scenario Decomposition . . . . .	39
3.3.1. Two-Stage Problems . . . . .	39
3.3.2. Multistage Problems . . . . .	40
<b>4. Decomposition with Recombining Scenario Trees</b>	<b>43</b>
4.1. Problem Setting . . . . .	44
4.2. Nested Benders Decomposition for Multistage Stochastic Linear Programs with Recombining Scenarios . . . . .	46
4.2.1. Dynamic Recombination of Scenarios . . . . .	46

Contents

4.3. Nested Column Generation for Multistage Stochastic Linear Programs with Recombining Scenarios . . . . .	53
4.3.1. Dynamic Recombination of Scenarios . . . . .	59
4.4. Combining Nested Benders Decomposition and Nested Column Generation	64
4.5. Construction of Recombining Scenario Trees . . . . .	65
4.6. Numerical Experiment . . . . .	70
4.6.1. Notes on Implementation . . . . .	70
4.6.2. A Power Scheduling Problem . . . . .	72
4.6.3. Numerical Results . . . . .	74
4.7. Out-Of-Sample Evaluation . . . . .	89
4.7.1. Problem Setting . . . . .	90
4.7.2. Adaptation of Approximate Solutions to Out-Of-Sample Scenarios	91
4.7.3. Numerical Example . . . . .	94
<b>5. Optimization of Dispersed Energy Supply</b>	<b>101</b>
5.1. Introduction . . . . .	101
5.2. Cost Optimal Operation Planning . . . . .	102
5.2.1. Model . . . . .	102
5.2.2. Case study . . . . .	105
5.2.3. Numerical Results . . . . .	105
5.3. Expansion Planning . . . . .	108
5.3.1. Model . . . . .	108
5.3.2. Case Study . . . . .	110
5.3.3. Numerical Results . . . . .	111
5.4. Conclusions . . . . .	112
<b>II. Mixed Integer Nonlinear Programming</b>	<b>113</b>
<b>6. Introduction</b>	<b>115</b>
6.1. Algorithmic Concepts . . . . .	117
6.1.1. Convex MINLP . . . . .	118
6.1.2. Convexification for Nonconvex MINLPs . . . . .	125
6.1.3. Spatial Branch-and-Bound . . . . .	132
6.1.4. Branching . . . . .	135
6.1.5. Bound Tightening . . . . .	138
6.1.6. Cutting Planes . . . . .	142
6.1.7. Primal Heuristics . . . . .	144
6.2. Solvers . . . . .	146
6.2.1. History . . . . .	146
6.2.2. Groupings . . . . .	147
6.2.3. List of solvers . . . . .	149
6.2.4. Outlook . . . . .	156

<b>7. A Constraint Integer Programming Approach to MINLP</b>	<b>159</b>
7.1. Constraint Integer Programming . . . . .	159
7.2. The CIP Framework SCIP . . . . .	162
7.3. The Expression Graph . . . . .	167
7.3.1. Simplification . . . . .	169
7.3.2. Bound Tightening . . . . .	170
7.3.3. Convexity Detection . . . . .	177
7.4. Bound Tightening for Quadratic Functions . . . . .	181
7.4.1. Univariate Quadratic Expressions . . . . .	181
7.4.2. Bivariate Quadratic Expressions . . . . .	183
7.5. Outer-Approximation . . . . .	189
7.5.1. General Nonlinear Functions . . . . .	190
7.5.2. Odd and Signed Power Functions . . . . .	191
7.5.3. Quadratic Functions . . . . .	193
7.6. Reformulation . . . . .	196
7.6.1. Reformulating the Expression Graph . . . . .	196
7.6.2. Quadratic Constraints with Binary Variables . . . . .	199
7.6.3. Quadratic Complementarity Constraints . . . . .	200
7.6.4. Signed Square Functions . . . . .	201
7.7. Branching . . . . .	201
7.8. Primal Heuristics . . . . .	202
7.8.1. NLP Local Search . . . . .	202
7.8.2. Undercover Heuristic . . . . .	203
7.8.3. Sub-MINLP Heuristics . . . . .	203
<b>8. Computational Study</b>	<b>205</b>
8.1. Open Pit Mine Production Scheduling with a Single Stockpile . . . . .	206
8.1.1. Open Pit Mine Production Scheduling with Stockpiles . . . . .	206
8.1.2. MIQCP Formulations . . . . .	207
8.1.3. Application-specific Benchmark Algorithm . . . . .	211
8.1.4. Test Instances . . . . .	212
8.1.5. Computational Results . . . . .	213
8.2. Water Distribution Network Design . . . . .	218
8.2.1. Model . . . . .	218
8.2.2. Test Instances . . . . .	220
8.2.3. Computational Results . . . . .	221
8.3. Benchmarks . . . . .	222
8.3.1. MINLPLib . . . . .	225
8.3.2. Convex MINLPs . . . . .	230
8.3.3. MIQCPs . . . . .	234
8.3.4. MISOCPs . . . . .	236
8.4. Impact of Solver Components . . . . .	238

<b>A. Tables</b>	<b>243</b>
A.1. Problem Statistics . . . . .	243
A.1.1. MINLPLib . . . . .	243
A.1.2. Convex MINLPs . . . . .	249
A.1.3. Mittelmann’s MIQPs . . . . .	252
A.2. Detailed Benchmark Results . . . . .	253
A.2.1. MINLPLib . . . . .	253
A.2.2. Convex MINLPs . . . . .	258
A.2.3. MIQPs . . . . .	268
A.2.4. MISOCPs . . . . .	270
<b>Bibliography</b>	<b>275</b>

**Part I.**

# **Stochastic Programming**



# 1. Introduction

In many applications of decision making, data is affected by uncertainty, i.e., the consequences of decisions are not perfectly known at the time they are made, but depend on parameters that are observable in the future only. The search for an optimal decision can then be modeled as an optimization problem that contains uncertainty w.r.t. one or several parameters. Often, some statistical information about the “stochastic” parameters is available, e.g., in form of a probability distribution obtained from historical or simulated data. It is then possible, to replace these parameters in the optimization problem by a random variable or random process. The resulting optimization problem is called a stochastic optimization problem, see Dantzig [1955] and the introductory textbooks Kall and Wallace [1994], Prékopa [1995], Birge and Louveaux [1997], and Ruszczyński and Shapiro [2003], and can be written as

$$\min \left\{ \int_{\Xi} f_0(x, \xi) dP(\xi) : x \in X \right\}, \quad (1.1)$$

where the feasible set  $X \subseteq \mathbb{R}^m$  is closed,  $\Xi$  is a closed subset of  $\mathbb{R}^s$ , the function  $f_0$  from  $\mathbb{R}^m \times \Xi$  to the extended reals  $\overline{\mathbb{R}}$  is a random lower semicontinuous function<sup>1</sup>, and  $P$  belongs to the set of all Borel probability measures  $\mathcal{P}(\Xi)$  on  $\Xi$ .

In the classical case, which is also adopted in (1.1), the objective of the stochastic program is to minimize (or maximize) the expectation of the cost function  $f_0(x, \xi)$ . However, for a particular realization of the stochastic parameters, the cost function may be very different from its expectation, e.g., if the costs vary largely w.r.t. the uncertain data. To find a risk-averse decision, often a weighted sum of expected costs and a measure of its variability is minimized, see, e.g., Föllmer and Schied [2004] and Pflug and Römisch [2007].

Two-stage stochastic optimization problems are characterized by the property that decisions on the first (time) stage have to be made without knowledge of the realization of the stochastic parameters while decisions on the second stage can depend on parameters that are observable at this time. Multistage stochastic programs contain further time stages, which allows to model processes where uncertain parameters are revealed over time and a series of decisions has to be made, each of them depending only on parameters observable by the time of decision. Both two- and multistage stochastic programs are discussed in more detail in the next sections.

Since analytic solutions to stochastic programs are rarely available, one has to resort to numerical methods. However, a main difficulty for numerical methods is imposed by

---

<sup>1</sup>A function  $f : \mathbb{R}^m \times \Xi \rightarrow \overline{\mathbb{R}}$  is a random lower semicontinuous function if its epigraphical mapping  $\xi \mapsto \text{epi } f(\cdot, \xi) := \{(x, \alpha) \in \mathbb{R}^m \times \mathbb{R} : f(x, \xi) \leq \alpha\}$  is closed-valued and measurable.

## 1. Introduction

a continuous distributions when it comes to evaluate the integral in (1.1). Therefore, a given continuous distribution is often approximated by a discrete distribution, i.e., a probability measure that assigns positive weights only to a finite number of realizations of  $\xi$ , c.f. Section 1.3.

### 1.1. Two-Stage Stochastic Programs

In so-called *two-stage stochastic programs*, the function  $f_0(x, \xi)$  in (1.1) is assumed to be the optimal value function of a (deterministic parametric) optimization problem, i.e.,

$$f_0(x, \xi) := \inf\{g_0(x, \xi; y) : y \in \Gamma(x, \xi)\}, \quad (1.2)$$

where  $g_0 : \mathbb{R}^m \times \mathbb{R}^s \times \mathbb{R}^{m'} \rightarrow \overline{\mathbb{R}}$  and  $\Gamma : \mathbb{R}^m \times \mathbb{R}^s \rightrightarrows \mathbb{R}^{m'}$  are random lower semicontinuous functions<sup>2</sup>. Since the decisions  $x$  and  $y(\xi)$  are made before and after the realization of  $\xi$ , they are called *first and second stage decisions*, respectively.

In this thesis, we are especially interested in two-stage stochastic *mixed-integer linear* programs. Here, the first stage feasible set  $X$  is a polyhedron given by

$$X = \{x \in \mathbb{Z}^{m_0} \times \mathbb{R}^{m-m_0} : Ax \leq b\}, \quad (1.3)$$

where  $m_0$  denotes the number of first stage variables with integrality restrictions,  $A$  is a  $(r_0, m)$ -matrix, and  $b \in \mathbb{R}^{r_0}$ . Further, the function  $f_0(x, \xi)$ ,  $x \in \mathbb{R}^m$ ,  $\xi \in \Xi$ , is of the form

$$f_0(x, \xi) = \langle c, x \rangle + \Phi(q(\xi), h(\xi) - T(\xi)x), \quad (1.4)$$

where  $c \in \mathbb{R}^m$  and the affine functions  $q : \mathbb{R}^s \rightarrow \mathbb{R}^{m_1+m_2}$ ,  $T : \mathbb{R}^s \rightarrow \mathbb{R}^{r \times m}$ , and  $h : \mathbb{R}^s \rightarrow \mathbb{R}^r$  are the stochastic cost, technology matrix, and right-hand side, respectively. The *recourse function*  $\Phi : \mathbb{R}^{m_1+m_2} \times \mathbb{R}^r \rightarrow \overline{\mathbb{R}}$ , is defined as the infimum function of a mixed-integer linear program (with cost  $u$  and right-hand side  $t$ ),

$$\Phi(u, t) := \inf\{\langle u_1, y_1 \rangle + \langle u_2, y_2 \rangle : y_1 \in \mathbb{R}^{m_1}, y_2 \in \mathbb{Z}^{m_2}, W_1 y_1 + W_2 y_2 \leq t\} \quad (1.5)$$

where  $W_1 \in \mathbb{R}^{r \times m_1}$  and  $W_2 \in \mathbb{R}^{r \times m_2}$  are matrices. Further, we define the *expected recourse function*  $\Psi : X \rightarrow \overline{\mathbb{R}}$  as

$$\Psi(x) := \int_{\Xi} \Phi(q(\xi), h(\xi) - T(\xi)x) P(d\xi). \quad (1.6)$$

Given a realization of  $\xi$ , a possible violation of  $h(\xi) - T(\xi)x \leq 0$  is compensated by the recourse cost  $\langle q_1(\xi), y_1(\xi) \rangle + \langle q_2(\xi), y_2(\xi) \rangle$ , where the pair  $(y_1(\xi), y_2(\xi))$  with integral  $y_2$  satisfies the constraint  $W_1 y_1 + W_2 y_2 \leq h(\xi) - T(\xi)x$ . Here, the cost coefficients  $q_1(\xi)$  and  $q_2(\xi)$  may depend on  $\xi$ . Hence, the modeling idea in (1.1)–(1.5) consists in adding the expected recourse cost  $\mathbb{E}(\langle q_1(\xi), y_2(\xi) \rangle + \langle q_2(\xi), y_2(\xi) \rangle)$  to the original cost  $\langle c, x \rangle$  and in minimizing the total cost with respect to  $(y_1, y_2)$ .

<sup>2</sup>The notation  $f : A \rightrightarrows B$  denotes set-valued mapping, i.e.,  $f(a) \subset B$  for  $a \in A$ .

In the case that no integrality restrictions are present ( $m_0 = 0$ ,  $m_2 = 0$ ), one has a two-stage stochastic linear program

$$\inf \{ \langle c, x \rangle + \mathbb{E} [\Phi(q(\xi), h(\xi) - T(\xi)x)] : Ax \leq b \}, \quad (1.7a)$$

where

$$\Phi(u, t) = \inf \{ \langle u, y \rangle : Wy \leq t \}. \quad (1.7b)$$

## 1.2. Multistage Stochastic Programs

The formulation (1.1)–(1.2) is readily extended to a multistage setting with *time horizon*  $T \in \mathbb{N}$  by replacing the stochastic variable  $\xi$  by a (discrete-time) *stochastic process*  $\{\xi_t\}_{t \in [T]}$ ,  $\xi_t : \Omega \rightarrow \mathbb{R}^{s_t}$ , on a probability space  $(\Omega, \mathcal{F}, P)$ , with associated filtration  $\mathcal{F}_t := \sigma(\xi_1, \dots, \xi_t)$ ,  $t \in [T]$ , such that  $\mathcal{F}_1 = \{\emptyset, \Omega\}$  (i.e., first stage is deterministic) and  $\mathcal{F}_T = \mathcal{F}$ . Here and in the following, we write  $[n : m]$  for the index set  $\{n, \dots, m\}$ , where  $n, m \in \mathbb{N}$  with  $n \leq m$ . Further, let  $[n] := [1 : n]$  for  $n \in \mathbb{N}$ . Accordingly, define  $\xi_{[t]} := (\xi_1, \dots, \xi_t)$ .

The *multistage stochastic programming problem* consists in finding a stochastic process  $x = \{x_t\}_t : \mathbb{R}^{s_1 + \dots + s_T} \rightarrow \mathbb{R}^{m_1 + \dots + m_T}$ , such that  $x_t$  is  $\mathcal{F}_t$ -measurable (also written as  $x_t \in \mathcal{F}_t$ ), certain constraints  $x_t \in X_t$  and  $g_t(x_1, \dots, x_t, \xi_t) \leq 0$  are satisfied  $P$ -almost everywhere, and a cost function  $f(x_1, \dots, x_T, \xi)$  is minimized in expectation:

$$\min \left\{ \mathbb{E} [f(x_1, \dots, x_T, \xi)] : \begin{array}{l} g_t(x_1, \dots, x_t, \xi_t) \leq 0 \text{ } P\text{-a.e.} \\ x_t \in X_t, x_t \in \mathcal{F}_t \end{array} \quad t \in [T] \right\}, \quad (1.8)$$

where  $X_t \subseteq \mathbb{R}^{m_t}$  is closed and  $f : \mathbb{R}^{m_1 + \dots + m_T} \times \mathbb{R}^{s_1 + \dots + s_T} \rightarrow \overline{\mathbb{R}}$  and  $g_t : \mathbb{R}^{m_1 + \dots + m_t} \times \mathbb{R}^{s_t} \rightarrow \mathbb{R}^{d_t}$ ,  $t \in [T]$ , are random lower semicontinuous functions.

The requirement that  $x_t$  is  $\mathcal{F}_t$ -measurable is also called *nonanticipativity constraint* and ensures that decisions  $(x_1(\xi), \dots, x_t(\xi))$  and  $(x_1(\xi'), \dots, x_t(\xi'))$  along different realizations  $\xi, \xi' \in \Xi$  are equal as long as the corresponding observations up to time  $t$ , i.e.,  $\xi_{[t]}$  and  $\xi'_{[t]}$ , are indistinguishable. It further allows to write (1.8) equivalently in form of a *dynamic program* via the *Bellmann equations*

$$f_T(\xi, x_1, \dots, x_T) := \begin{cases} f(x_1, \dots, x_T, \xi), & x_t \in X_t, g_t(x_1, \dots, x_t, \xi_t) \leq 0, t \in [T], \\ +\infty, & \text{otherwise} \end{cases}$$

$$f_{t-1}(\xi, x_1, \dots, x_{t-1}) := \inf_{y \in \mathbb{R}^{m_t}} \{ \mathbb{E}^r [f_t(\cdot, x_1(\cdot), \dots, x_{t-1}(\cdot), y) | \mathcal{F}_t] (\xi) \}, \text{ } P\text{-a.e.}, t \in [2 : T],$$

if there exists a process  $\tilde{x}$  such that  $\tilde{x}_t : \Xi \rightarrow \mathbb{R}^{m_t}$  is  $\mathcal{F}_t$ -measurable,  $t \in [T]$ , and  $\mathbb{E} [f_T(\xi, \tilde{x}_1(\xi), \dots, \tilde{x}_T(\xi))] < \infty$  [Evstigneev, 1976, Theorem 1 and 2]<sup>3</sup>.

<sup>3</sup> $\mathbb{E}^r[f|\mathcal{G}]$  denotes the regular conditional expectation as defined in Evstigneev [1976] and ensures that the Bellmann equations are well-defined.

## 1. Introduction

The Bellmann equations allow to write (1.8) in the form (1.1) as

$$\min \left\{ \int_{\Xi} f_1(\xi, x_1) P(d\xi) : x_1 \in \mathbb{R}^{m_1} \right\}.$$

In this thesis, we are especially interested in multistage stochastic *linear* programs, which have the form

$$\min \left\{ \mathbb{E} \left[ \sum_{t=1}^T \langle b_t(\xi_t), x_t \rangle \right] : \begin{array}{l} x_t \in X_t, x_t \in \mathcal{F}_t, t \in [T], \\ A_{t,0}(\xi_t)x_t + A_{t,1}(\xi_t)x_{t-1} = h_t(\xi_t), P\text{-a.e.}, t \in [2 : T] \end{array} \right\}, \quad (1.9)$$

for closed polyhedral sets  $X_t \subseteq \mathbb{R}^{m_t}$  and matrix-valued mappings  $A_{t,0} : \mathbb{R}^{s_t} \rightarrow \mathbb{R}^{d_t \times m_t}$ ,  $A_{t,1} : \mathbb{R}^{s_t} \rightarrow \mathbb{R}^{d_t \times m_{t-1}}$ , and  $h_t : \mathbb{R}^{s_t} \rightarrow \mathbb{R}^{d_t}$ . A dynamic programming formulation of (1.9) is given by

$$\min \{ \langle b_1, x_1 \rangle + \mathbb{E} [Q_2(x_1, \xi_1, \xi_2)] : x_1 \in X_1 \}, \quad (1.10a)$$

where  $b_1 := b_1(\xi_1)$  for some  $\xi \in \Xi$  (well defined because  $\mathcal{F}_1 = \{\emptyset, \Omega\}$ ),

$$Q_t(x_{t-1}, \bar{\xi}_{[t]}) := \min \left\{ \begin{array}{l} \langle b_t(\bar{\xi}_t), x_t \rangle + \mathbb{E} [Q_{t+1}(x_t, \xi_{[t+1]}) \mid \xi_{[t]} = \bar{\xi}_{[t]}] : \\ x_t \in X_t, A_{t,0}(\bar{\xi}_t)x_t + A_{t,1}(\bar{\xi}_t)x_{t-1} = h_t(\bar{\xi}_t), P\text{-a.e.} \end{array} \right\}, \quad (1.10b)$$

for  $t = 2, \dots, T-1$ , and

$$Q_T(x_{T-1}, \bar{\xi}) := \min \left\{ \langle b_T(\bar{\xi}_T), x_T \rangle : x_T \in X_T, A_{T,0}(\bar{\xi}_T)x_T + A_{T,1}(\bar{\xi}_T)x_{T-1} = h_T(\bar{\xi}_T) \right\}. \quad (1.10c)$$

The functions  $Q_t(x_{t-1}, \bar{\xi}_{[t]})$ ,  $t \in [2 : T]$ , are also called *cost-to-go* functions, since they evaluate to the expected future cost for taking decision  $x_{t-1}$  at time  $t-1$ , provided  $\xi_{[t]}$  takes the value  $\bar{\xi}_{[t]}$ . Formulation (1.10) is also the starting point for temporal decomposition methods like the *Nested Benders Decomposition*, c.f. Section 3.1.2.

## 1.3. Discretization of Stochastic Programs

### 1.3.1. Scenarios

To render possible a numerical solution of (1.1), a given continuous distribution  $P$  is often approximated by a discrete distribution  $Q$ , i.e., a probability measure on  $\Xi$  that assigns positive weights only to a finite number of *scenarios*  $\{\xi^i\}_{i \in I}$ ,  $|I| < \infty$ . Let  $p_i := Q(\{\xi^i\})$  be the *probability of scenario*  $\xi^i$ ,  $i \in I$ . Note that  $p_i > 0$  for all  $i \in I$  and that  $\sum_{i \in I} p_i = 1$ . For simplicity, assume  $\xi^i \neq \xi^j$  for all  $i, j \in I$  with  $i \neq j$ .

When replacing  $P$  by  $Q$  in a two-stage stochastic program (1.1)–(1.2), the integral over  $\Xi$  can be written as a sum over  $I$ . Further, using the notation  $y^i := y(\xi^i)$ , (1.1)–(1.2) (with  $P$  replaced by  $Q$ ) can be written in form of a *deterministic* optimization problem:

$$\min \left\{ \sum_{i \in I} p_i g_0(x, \xi^i, y^i) : x \in X, y^i \in \Gamma(x, \xi^i), i \in I \right\}.$$

Thus, for stochastic linear or mixed-integer linear programs, replacing a continuous distribution by a finite one yields a deterministic linear or mixed-integer linear program, which may be solvable by nowadays efficient solvers for these problem classes.

Solvability of such a deterministic program depends heavily on the number of scenarios that are used to approximate the distribution  $P$ . A low number of scenarios allows for a smaller deterministic program, which is usually easier to solve. However, the quality of the approximation is usually better if a large number of scenarios is used. The necessary tradeoff led to extensive research on how to find a good approximate of a given distribution  $P$  by a minimal number of scenarios. Established methods are Monte-Carlo methods [Shapiro, 2003a], Quasi-Monte Carlo methods [Lemieux, 2009], optimal quantization methods [Graf and Luschgy, 2000], and quadrature rules using sparse grids [Chen and Mehrotra, 2008], see also the recent survey [Römisch, 2010]. These methods are often applied in combination with scenario reduction methods, which reduce a given set of scenarios to a smaller but still expressive subset [Dupačová et al., 2003, Heitsch and Römisch, 2007, Heitsch, 2007].

The approximation quality for a probability measure can be measured in terms of probability metrics [Rachev, 1991]. Under certain regularity assumptions, it can be shown that the optimal value and solution set of an approximated stochastic program converges to the optimal value and solution set of the original stochastic program, if the approximated probability measures converges to the original one w.r.t. the chosen metric, see Römisch [2003] for a survey on qualitative and quantitative stability results and Chapter 2 for stability properties of two-stage stochastic mixed-integer linear programs. However, the choice of the probability metric cannot be arbitrary, but need to be adapted to the type of stochastic program that is approximated.

### 1.3.2. Scenario Trees

Recall, that a crucial component of a multistage stochastic program are the nonanticipativity constraints  $x_t \in \mathcal{F}_t$ ,  $t \in [T]$ , which are defined by the filtration  $\{\mathcal{F}_t\}_{t \in [T]}$ . This filtration models the “growth of information” over time. If a continuous probability measure is approximated by a finite set of scenarios as in the two-stage setting, e.g., via a sampling approach, then it is unlikely that the filtration defined by the approximated distribution will represent a similar “growth of information” over time as the original one, since the generated scenarios are likely to differ already from the second stage on. In this case, the discretized multistage stochastic program would be equivalent to a two-stage stochastic program. Therefore, when approximating stochastic processes in multistage stochastic programs, special attention must be put on approximating the measure such that it defines a filtration that is in some sense similar to the original one. Heitsch, Römisch, and Strugarek [2006] and Pflug and Pichler [2012] show, that convergence of the optimal value of an approximated multistage stochastic linear programs to the optimal value of the original problem can be expected, if the approximated distribution converges to the original one with respect to both a classic probability metric and a measure for the distance of the induced filtrations.

The scenarios  $\{\{\xi_t^i\}_{t \in [T]}\}_{i \in I}$  in a discretization of a stochastic process together with

## 1. Introduction

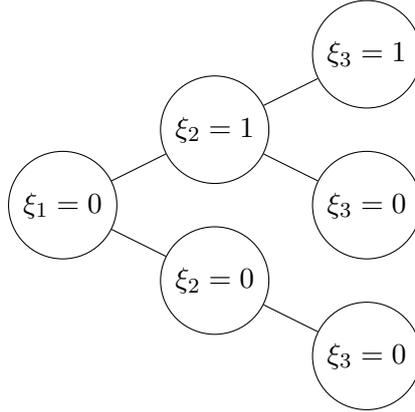


Figure 1.1.: Scenario tree corresponding to a three-stage stochastic process that is specified by the three scenarios  $\xi^1 = (0, 0, 0)$ ,  $\xi^2 = (0, 1, 0)$ ,  $\xi^3 = (0, 1, 1)$ . All scenarios are indistinguishable at the first stage. Further, the second and third scenario are indistinguishable at the second stage, too.

the induced filtration can be represented in a tree form, the so called *scenario tree*. Its nodes correspond to the scenario values  $\xi_{[t]}^i$  up to time  $t$ ,  $t \in [T]$ ,  $i \in I$ . Thus, they can be organized in levels which correspond to the stages  $t = 1, \dots, T$ . Since we assume a deterministic first stage ( $\mathcal{F}_1 = \{\emptyset, \Omega\}$ ), there is only one node at level  $t = 1$ , which is denoted as *root node*. The scenario defines the edges in the tree. That is, scenario  $\xi^i = (\xi_1^i, \dots, \xi_T^i)$ ,  $i \in I$ , implies that  $\xi_{[t+1]}^i$  is an ancestor of node  $\xi_{[t]}^i$ ,  $t \in [T - 1]$ . The leaf nodes of the scenario tree correspond to the scenarios itself (since we assumed  $\mathcal{F}_T = \mathcal{F}$ ). See also Figure 1.1 for a simple example.

There exists several approaches for the construction of scenario trees. Often, the tree structure is (at least partially) predetermined and scenarios are generated according to the prescribed structure by (conditional) random sampling [Dempster, 2006, Kouwenberg, 2001, Shapiro, 2003b], Quasi-Monte Carlo sampling [Pennanen, 2009], bound-based constructions [Frauendorfer, 1996, Kuhn, 2005], moment-matching based optimization [Høyland and Wallace, 2001], or such that a best approximation in terms of probability metrics is achieved [Pflug, 2001, Hochreiter and Pflug, 2007], see also the survey Dupačová et al. [2000] and the references therein. Alternatively, the algorithms presented in [Heitsch and Römisch, 2009b] require only a set of scenarios as input and decide on the tree structure based on probability metrics and estimates on filtration distances. Also an algorithm to reduce an already given scenario tree has been developed [Heitsch and Römisch, 2009a].

## 2. Stability of Two-Stage Stochastic Mixed-Integer Linear Programming Problems

In this chapter, we review stability results for two-stage stochastic mixed-integer linear programs, i.e., results on the dependence of their solutions and optimal values on the underlying probability distribution  $P$ . Such results also provide information how an underlying probability distribution should be approximated such that solutions and optimal values of the approximated problem are close to those of the original one.

The aim is to extend the quantitative continuity properties in Rachev and Römisch [2002] and Schultz [1996] for the optimal value and solution set functions for two-stage stochastic mixed-integer linear programs under perturbation of the probability measure to cover situations with stochastic costs. To this end, we need quantitative continuity and growth properties of optimal value functions and solution sets of parametric mixed-integer linear programs, which are discussed in Section 2.1. Afterwards, in Section 2.2, structural properties of two-stage stochastic mixed-integer linear programs are discussed. The desired quantitative stability result for fully random two-stage stochastic mixed-integer linear programs with fixed recourse is derived in Theorem 2.7 of Section 2.3. The relevant probability metric (2.9) on subsets of  $\mathcal{P}(\Xi)$  and its relations to Fortet-Mourier metrics and polyhedral discrepancies are also discussed (Remark 2.9). The latter metrics may be used for designing moderately sized discrete approximations to  $P$  by optimal scenario reduction of discrete probability measures [Heitsch and Römisch, 2007, Henrion et al., 2008].

The content of this chapter is taken from the publication Römisch and Vigerske [2008].

### 2.1. Infima of Mixed-Integer Linear Programs

To analyze stability of stochastic mixed-integer linear programs, we first need quantitative continuity and growth properties of optimal value functions and solution sets of parametric mixed-integer linear programs. Such properties are known for parametric right-hand sides [Blair and Jeroslow, 1977, 1979, Schultz, 1996] and parametric costs *separately* [Bank et al., 1982, Bank and Mandel, 1988, Cook et al., 1986]. Since to our knowledge *simultaneous* perturbation results with respect to right-hand sides and costs are less familiar, we discuss such properties of optimal value functions in the following.

Consider the parametric mixed-integer linear program

$$\min\{\langle c_x, x \rangle + \langle c_y, y \rangle : A_x x + A_y y \leq b, x \in \mathbb{Z}^n, y \in \mathbb{R}^{m-n}\} \quad (2.1)$$

## 2. Stability of Two-Stage Stochastic Mixed-Integer Linear Programming Problems

with  $c = (c_x, c_y) \in \mathbb{R}^m$  and  $b \in \mathbb{R}^r$  playing the role of the parameters and  $A = (A_x, A_y) \in \mathbb{Q}^{r \times m}$ . Let  $M(b)$ ,  $v(b, c)$ , and  $S(b, c)$  denote the feasible set, optimal value, and solution set of (2.1), respectively, i.e.,

$$\begin{aligned} M(b) &:= \{(x, y) \in \mathbb{Z}^n \times \mathbb{R}^{m-n} : A(x, y) \leq b\} \\ v(b, c) &:= \inf\{\langle c, (x, y) \rangle : (x, y) \in M(b)\} \\ S(b, c) &:= \{(x, y) \in M(b) : \langle c, (x, y) \rangle = v(b, c)\}. \end{aligned}$$

Let  $\mathcal{K}$  denote the polyhedral cone  $\{(x, y) \in \mathbb{R}^m : A_x x + A_y y \leq 0\}$  and  $\mathcal{K}^*$  its polar cone. Observe, that  $v(b, c)$  is finite for  $b \in \mathcal{B} := \text{dom } M$  and  $c \in -\mathcal{K}^*$ . Further, denote by  $\text{Pr}_x M(b)$  the projection of  $M(b)$  onto the  $x$ -space, and let

$$\mathcal{B}^*(b^0) := \{b \in \mathcal{B} : \text{Pr}_x M(b) = \text{Pr}_x M(b^0)\} \quad (b^0 \in \mathcal{B})$$

be the set of right-hand sides on which the projection of  $M(b)$  onto the  $x$ -space is constant. It is well known (see Chapter 5.6 in Bank et al. [1982]) that the sets  $\mathcal{B}^*(b^0)$  are continuity regions of the function  $b \mapsto v(b, c)$ . These regions are further characterized by the following result.

**Lemma 2.1** (Lemma 5.6.1 and 5.6.2 in Bank et al. [1982]).  *$\mathcal{B}$  is a connected set equal to the union of a countable family of convex polyhedral cones each of which is obtained by a translation of the  $r$ -dimensional cone  $T := \{t \in \mathbb{R}^r : \exists y \in \mathbb{R}^{m-n} \text{ such that } t \geq A_y y\}$ .*

*For each  $b^0 \in \mathcal{B}$ , there exists  $t^0 \in \mathcal{B}$  and a finite set  $N \subseteq \mathbb{Z}^n \setminus \text{Pr}_x M(b^0)$  such that*

$$\mathcal{B}^*(b^0) = (t^0 + T) \setminus \bigcap_{z \in N} (A_x z + T).$$

*If  $\text{Pr}_x M(b^0) = \mathbb{Z}^n$ , then  $N = \emptyset$  and  $\mathcal{B}^*(b^0) = t^0 + T$  for some  $t^0 \in \mathcal{B}$ .*

In the following we extend Lemma 2.3 in Schultz [1996] and show local Lipschitz-continuity of the optimal value of (2.1) with respect to simultaneous perturbations of the right-hand side and the objective function coefficients where the right-hand side perturbation does not leave the continuity region  $\mathcal{B}^*(b)$ . Otherwise, for arbitrary right-hand sides, a quasi-Lipschitz property of the value function of (2.1) can be shown.

**Proposition 2.2.** (i) *Let  $b \in \mathcal{B}$ ,  $b' \in \mathcal{B}^*(b)$ , and  $c, c' \in -\mathcal{K}^*$ . Then the estimate*

$$|v(b, c) - v(b', c')| \leq L_1 \max\{\|c\|, \|c'\|\} \|b - b'\| + L_2 \max\{\|b\|, \|b'\|, 1\} \|c - c'\|$$

*holds, where the constants  $L_1$  and  $L_2$  depend on  $A$  only.*

(ii) *Let  $b, b' \in \mathcal{B}$  and  $c, c' \in -\mathcal{K}^*$ . Then we have*

$$|v(b, c) - v(b', c')| \leq \max\{\|c\|, \|c'\|\} (\tilde{L} \|b - b'\| + 2\ell) + \tilde{L} \max\{\|b\|, \|b'\|\} \|c - c'\|,$$

*where the constants  $\tilde{L}$  and  $\ell$  depend on  $A$  only.*

For the proof of Proposition 2.2, we require the following Lemma.

**Lemma 2.3** (Theorem 1.2 in Blair and Jeroslow [1979], Theorem 1 in Cook et al. [1986]).  
 Let  $b \in \mathcal{B}$ ,  $c \in -\mathcal{K}^*$ . Let  $(\tilde{x}, \tilde{y})$  be a solution of

$$\min\{\langle c_x, x \rangle + \langle c_y, y \rangle : A_x x + A_y y \leq b, (x, y) \in \mathbb{R}^m\}. \quad (2.2)$$

Then there exists a solution  $(x, y) \in S(b, c)$  such that

$$\|(x, y) - (\tilde{x}, \tilde{y})\| \leq \ell$$

for some constant  $\ell$  depending on  $A$  only.

*Proof of Proposition 2.2.* Let  $b \in \mathcal{B}$ ,  $b' \in \mathcal{B}^*(b)$ , and  $c, c' \in -\mathcal{K}^*$  be given. To show local Lipschitz continuity of  $v(b, c)$ , we estimate

$$|v(b, c) - v(b', c')| \leq |v(b, c) - v(b', c)| + |v(b', c) - v(b', c')|.$$

For the first difference we can proceed as for the proof of Lemma 2.3 in Schultz [1996]. It is repeated here to keep the presentation self-contained. We write (2.1) as

$$\min\{\langle c_x, x \rangle + \Psi(c_y, b - A_x x) : x \in \text{Pr}_x M(b)\}$$

where  $\Psi(c_y, \tilde{b}) := \min\{\langle c_y, y \rangle : A_y y \leq \tilde{b}\}$ . Since  $\Psi(c_y, \tilde{b})$  is the optimal value function of a linear program and finite for  $b \in \mathcal{B}$ ,  $c' \in -\mathcal{K}^*$ , there exist finitely many matrices  $C_j$ , which depend on  $A_y$  only, such that  $\Psi(c_y, \tilde{b}) = \max_j \langle \tilde{b}, C_j c_y \rangle$  (cf. Walkup and Wets [1969]). Let  $L_1 := \max_j \|C_j\|$ . Then, for  $c_y$  fixed,

$$|\Psi(c_y, \tilde{b}) - \Psi(c_y, \tilde{b}')| \leq L_1 \|c_y\| \|\tilde{b} - \tilde{b}'\|.$$

Let  $(x, y) \in S(b, c)$ ,  $(x', y') \in S(b', c)$ . Since  $\text{Pr}_x M(b) = \text{Pr}_x M(b')$ , we have

$$\begin{aligned} v(b, c) - v(b', c) &\leq \langle c_x, x' \rangle + \Psi(c_y, b - A_x x') - \langle c_x, x' \rangle - \Psi(c_y, b' - A_x x') \\ &\leq L_1 \|c\| \|b - b'\|. \end{aligned}$$

Due to symmetry the same estimate holds for  $v(b', c) - v(b, c)$ .

Next, we derive an estimate for  $|v(b', c) - v(b', c')|$ . Since (2.2) is a linear program, there exist finitely many matrices  $D_j$ , which depend on  $A$  only, and such that each basis solution of (2.2) is given by  $D_j b$  for some  $j$ . We set  $\hat{L} := \max_j \|D_j\|$ . Now let  $(\tilde{x}', \tilde{y}')$  be an optimal basis solutions of problem (2.2) with right-hand side  $b'$  and cost vector  $c'$ . By Lemma 2.3 there exists  $(x', y') \in S(b', c')$  with  $\|(x', y') - (\tilde{x}', \tilde{y}')\| \leq \ell$ . Since  $v(b', c) \leq \langle c_x, x' \rangle + \langle c_y, y' \rangle$  and  $\|(\tilde{x}', \tilde{y}')\| \leq \hat{L} \|b'\|$ , we obtain

$$v(b', c) - v(b', c') \leq \|(x', y')\| \|c - c'\| \leq (\ell + \hat{L} \|b'\|) \|c - c'\|.$$

Due to symmetry, a similar estimate holds for  $v(b', c') - v(b', c)$ . The second part of Proposition 2.2 follows from Lemma 2.3 and stability results for linear programs.  $\square$

Together with Proposition 2.2, the following result is needed to prove Lemma 2.5.

## 2. Stability of Two-Stage Stochastic Mixed-Integer Linear Programming Problems

**Lemma 2.4** (Theorem 2.1 in Blair and Jeroslow [1977], Bank and Mandel [1988]). *Let  $c \in -\mathcal{K}^*$ . The mapping  $b \mapsto S(b, c)$  is quasi-Lipschitz continuous on  $\mathcal{B}$  with constants  $\bar{L}_1$  and  $\bar{L}_2$  not depending on  $b$  and  $c$ , i.e.,*

$$d_H(S(b, c), S(b', c)) \leq \bar{L}_1 \|b - b'\| + \bar{L}_2,$$

where  $d_H$  denotes the Hausdorff distance on subsets of  $\mathbb{R}^m$ .

## 2.2. Structural Properties of Two-Stage Stochastic Mixed-Integer Linear Programs

Recall the definition of a two-stage stochastic mixed-integer linear program from Section 1.1:

$$\min \left\{ \int_{\Xi} f_0(x, \xi) dP(\xi) : x \in X \right\}, \quad (1.1)$$

where  $X \subseteq \mathbb{R}^m$  is the feasible set of first-stage solutions,  $\Xi$  is a closed subset of  $\mathbb{R}^s$ ,  $P \in \mathcal{P}(\Xi)$  is a Borel probability measure on  $\Xi$ , and  $f_0 : \mathbb{R}^m \times \Xi \rightarrow \bar{\mathbb{R}}$  is

$$f_0(x, \xi) = \langle c, x \rangle + \Phi(q(\xi), h(\xi) - T(\xi)x) \quad ((x, \xi) \in \mathbb{R}^m \times \Xi), \quad (1.4)$$

where  $c \in \mathbb{R}^m$ , the affine functions  $q(\xi)$ ,  $T(\xi)$ , and  $h(\xi)$  are the stochastic cost, technology matrix, and right-hand side, respectively, and

$$\Phi(u, t) := \inf \{ \langle u_1, y_1 \rangle + \langle u_2, y_2 \rangle : y_1 \in \mathbb{R}^{m_1}, y_2 \in \mathbb{Z}^{m_2}, W_1 y_1 + W_2 y_2 \leq t \}, \quad (1.5)$$

$u \in \mathbb{R}^{m_1+m_2}$ ,  $t \in \mathbb{R}^r$ , denotes the optimal value of the (second-stage) mixed-integer linear program. Further, assume that  $\Xi$  is a polyhedron in  $\mathbb{R}^s$ .

The following conditions are imposed to have the model (1.1)–(1.5) well-defined:

**(C1)** The matrices  $W_1$  and  $W_2$  have only rational elements.

**(C2)** For each pair  $(x, \xi) \in X \times \Xi$  it holds that  $h(\xi) - T(\xi)x \in \mathcal{T}$ , where

$$\mathcal{T} := \{ t \in \mathbb{R}^r : \exists y = (y_1, y_2) \in \mathbb{R}^{m_1} \times \mathbb{Z}^{m_2} \text{ such that } W_1 y_1 + W_2 y_2 \leq t \}.$$

**(C3)** For each  $\xi \in \Xi$  the recourse cost  $q(\xi)$  belongs to the dual feasible set

$$\mathcal{U} := \left\{ u = (u_1, u_2) \in \mathbb{R}^{m_1+m_2} : \exists z \in \mathbb{R}_-^r \text{ such that } W_1^\top z = u_1, W_2^\top z = u_2 \right\}.$$

**(C4)**  $P \in \mathcal{P}_2(\Xi)$ , i.e.,  $P \in \mathcal{P}(\Xi)$  and  $\int_{\Xi} \|\xi\|^2 P(d\xi) < +\infty$ .

Condition (C2) means that a feasible second stage decision always exists (*relatively complete recourse*). Both (C2) and (C3) imply  $\Phi(u, t)$  to be finite for all  $(u, t) \in \mathcal{U} \times \mathcal{T}$ . Clearly, it holds  $(0, 0) \in \mathcal{U} \times \mathcal{T}$  and  $\Phi(0, t) = 0$  for every  $t \in \mathcal{T}$ . With the convex

## 2.2. Structural Properties of Two-Stage Stochastic Mixed-Integer Linear Programs

polyhedral cone

$$\mathcal{K} := \{t \in \mathbb{R}^r : \exists y_1 \in \mathbb{R}^{m_1} \text{ such that } t \geq W_1 y_1\} = W_1(\mathbb{R}^{m_1}) + \mathbb{R}_+^r$$

one obtains the representation

$$\mathcal{T} = \bigcup_{z \in \mathbb{Z}^{m_2}} (W_2 z + \mathcal{K}).$$

The two extremal cases are (i)  $W_1$  has rank  $r$  implying  $\mathcal{K} = \mathbb{R}^r = \mathcal{T}$  (*complete recourse*) and (ii)  $W_1 = 0$  (*pure integer recourse*) leading to  $\mathcal{K} = \mathbb{R}_+^r$ .

In general, the set  $\mathcal{T}$  is connected (i.e., there exists a polygon connecting two arbitrary points of  $\mathcal{T}$ ) and condition (C1) implies that  $\mathcal{T}$  is closed. The results discussed in the previous section show that the set  $\mathcal{T}$  can be decomposed into a partition of subsets  $\mathcal{B}_i$ ,  $i \in \mathbb{N}$ , which are nonempty and connected (even star-shaped, cf., Theorem 5.6.3 in Bank et al. [1982]), but nonconvex in general (cf. Figure 2.1), such that the function  $\Phi(u, t)$  is Lipschitz-continuous on each  $\mathcal{U} \times \mathcal{B}_i$ :

**Lemma 2.5.** *Assume (C1)–(C3). Then there exists a countable partition of  $\mathcal{T}$  into Borel subsets  $\mathcal{B}_i$ , i.e.,  $\mathcal{T} = \bigcup_{i \in \mathbb{N}} \mathcal{B}_i$  such that*

(i)  $\mathcal{B}_i = (b_i + \mathcal{K}) \setminus \bigcup_{j=1}^{N_0} (b_{i,j} + \mathcal{K})$ , where  $b_i, b_{i,j} \in \mathbb{R}^r$ ,  $i \in \mathbb{N}$ ,  $j \in [N_0]$ , and  $N_0 \in \mathbb{N}$  does not depend on  $i$ . Moreover there exists an  $N_1 \in \mathbb{N}$  such that for all  $t \in \mathcal{T}$  the ball  $\mathbb{B}(t, 1)$  in  $\mathbb{R}^r$  is intersected by at most  $N_1$  different subsets  $\mathcal{B}_i$ .

(ii) the restriction  $\Phi|_{\mathcal{U} \times \mathcal{B}_i}$ , where  $\mathcal{B}'_i := \mathcal{B}_i \cap \{h(\xi) - T(\xi)x : (x, \xi) \in X \times \Xi\}$ , has the property that there exists a constant  $L > 0$  independent of  $i$ , s.t.

$$|\Phi(u, t) - \Phi(\tilde{u}, \tilde{t})| \leq L(\max\{1, \|t\|, \|\tilde{t}\|\} \|u - \tilde{u}\| + \max\{1, \|u\|, \|\tilde{u}\|\} \|t - \tilde{t}\|).$$

Furthermore, the function  $\Phi$  is lower semicontinuous and piecewise polyhedral on  $\mathcal{U} \times \mathcal{T}$  and there exist constants  $D, d > 0$  such that it holds for all pairs  $(u, t), (\tilde{u}, \tilde{t}) \in \mathcal{U} \times \mathcal{T}$ :

$$|\Phi(u, t) - \Phi(\tilde{u}, \tilde{t})| \leq D(\max\{1, \|t\|, \|\tilde{t}\|\} (\|u - \tilde{u}\| + d) + \max\{1, \|u\|, \|\tilde{u}\|\} \|t - \tilde{t}\|).$$

*Proof.* The first part of (i) is Lemma 2.1. The second part is an extension of Lemma 2.5 in Schultz [1996] to the function  $\Phi(u, t)$  since the relevant constants in its proof do not depend on the objective function as recalled in Lemma 2.4.

Part (ii) and the quasi-Lipschitz property of  $\Phi$  is Proposition 2.2.  $\square$

The representation of  $\Phi$  is given on countably many (possibly unbounded) Borel sets. This requires to incorporate the tail behavior of  $P$  and leads to the following representation of the function  $f_0$ .

**Proposition 2.6.** *Assume (C1)–(C4) and  $X$  be bounded. For each  $R \geq 1$  and  $x \in X$  there exist  $\nu \in \mathbb{N}$  and disjoint Borel subsets  $\Xi_{j,x}^R$  of  $\Xi$ ,  $j \in [\nu]$ , whose closures are polyhedra*

## 2. Stability of Two-Stage Stochastic Mixed-Integer Linear Programming Problems

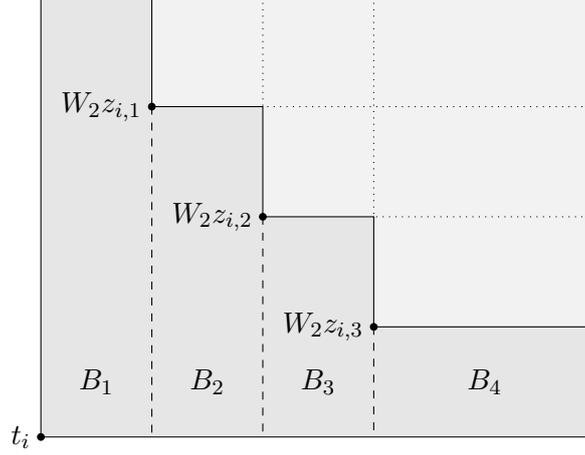


Figure 2.1.: Illustration of  $\mathcal{B}_i$  (see Lemma 2.5) for  $W_1 = 0$  and  $r = 2$ , i.e.,  $\mathcal{K} = \mathbb{R}_+^2$ , with  $N_i = \{1, 2, 3\}$  and its decomposition into the sets  $B_j$ ,  $j = 1, 2, 3, 4$ , whose closures are convex polyhedral (rectangular).

with a uniformly bounded number of faces such that

$$f_0(x, \xi) = \sum_{j=0}^{\nu} (\langle c, x \rangle + \Phi(q(\xi), h(\xi) - T(\xi)x)) \mathbf{1}_{\Xi_{j,x}^R}(\xi) \quad ((x, \xi) \in X \times \Xi)$$

is Lipschitz continuous with respect to  $\xi$  on each  $\Xi_{j,x}^R$ ,  $j \in [1 : \nu]$ , with some uniform Lipschitz constant. Here,  $\Xi_{0,x}^R := \Xi \setminus \bigcup_{j=1}^{\nu} \Xi_{j,x}^R$  is contained in  $\{\xi \in \mathbb{R}^s : \|\xi\|_{\infty} > R\}$ ,  $\nu$  is bounded by a multiple of  $R^r$  and  $\mathbf{1}_A$  denotes the characteristic function of a set  $A$ .

*Proof.* Since  $q(\cdot)$ ,  $h(\cdot)$  and  $T(\cdot)$  are affine linear functions and  $X$  is bounded, there exists a constant  $C > 0$  such that the estimate

$$\max\{\|q(\xi)\|_{\infty}, \|h(\xi) - T(\xi)x\|_{\infty}\} \leq C \max\{1, \|\xi\|_{\infty}\} \quad (2.3)$$

holds for each pair in  $X \times \Xi$ .

Let  $R \geq 1$  and  $\mathcal{T}_R := \mathcal{T} \cap CR\mathbb{B}_{\infty}$ , where  $\mathbb{B}_{\infty}$  is the unit ball w.r.t. the maximum norm  $\|\cdot\|_{\infty}$ . As in Proposition 34 of Römisch [2003] there exist a number  $\nu \in \mathbb{N}$  and disjoint Borel subsets  $\{B_j\}_{j=1}^{\nu}$  of  $CR\mathbb{B}_{\infty}$  such that their closures are polyhedra and their union contains  $\mathcal{T}_R$ . Furthermore, when arguing as in the proof of Proposition 3.1 in Schultz [1996],  $\nu$  is bounded above by  $\kappa R^r$ , where the constant  $\kappa > 0$  is independent of  $R$ .

Now, let  $x \in X$  and consider the following disjoint Borel subsets of  $\Xi$ :

$$\begin{aligned} \Xi_{j,x}^R &:= \{\xi \in \Xi : h(\xi) - T(\xi)x \in B_j, \|\xi\|_{\infty} \leq R\} \quad (j \in [\nu]), \\ \Xi_{0,x}^R &:= \Xi \setminus \bigcup_{j=1}^{\nu} \Xi_{j,x}^R \subseteq \{\xi \in \Xi : \|\xi\|_{\infty} > R\}. \end{aligned}$$

## 2.2. Quantitative Stability Fully-Random Two-Stage Stoch. Mixed-Integer Linear Progr.

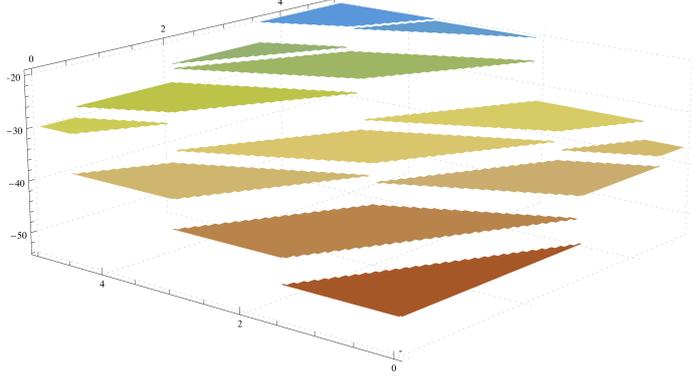


Figure 2.2.: Illustration of an expected recourse function with pure 0–1 recourse, random right-hand side and discrete uniform probability distribution.

Let  $x \in X$  and  $\xi, \xi' \in \Xi_{j,x}^R$  for some  $j \in [\nu]$ . By Lemma 2.5 we obtain

$$\begin{aligned}
 |f_0(x, \xi) - f_0(x, \xi')| &= |\Phi(q(\xi), h(\xi) - T(\xi)x) - \Phi(q(\xi'), h(\xi') - T(\xi')x)| \\
 &\leq L(\max\{1, \|q(\xi)\|_\infty, \|q(\xi')\|_\infty\}(\|h(\xi) - h(\xi')\|_\infty \\
 &\quad + \|(T(\xi) - T(\xi'))x\|_\infty) + \max\{1, \|h(\xi) - T(\xi)x\|_\infty, \\
 &\quad \|h(\xi') - T(\xi')x\|_\infty\}\|q(\xi) - q(\xi')\|_\infty) \\
 &\leq LCR(\|h(\xi) - h(\xi')\|_\infty + \|(T(\xi) - T(\xi'))x\|_\infty + \|q(\xi) - q(\xi')\|_\infty) \\
 &\leq L_1R\|\xi - \xi'\|_\infty,
 \end{aligned}$$

where we used (2.3) for  $\xi, \xi' \in \Xi_{j,x}^R$ , affine linearity of  $q(\cdot)$ ,  $h(\cdot)$ , and  $T(\cdot)$ , and boundedness of  $X$ . We note that the constant  $L_1$  is independent of  $R$ .  $\square$

Since the objective function of (1.1) is lower semicontinuous if the conditions (C1)–(C4) are satisfied, solutions of (1.1) exist if  $X$  is closed and bounded. If the probability distribution  $P$  has a density, the objective function of (1.1) is continuous, but nonconvex in general. If the support of  $P$  is finite, the objective function is piecewise continuous with a finite number of polyhedral continuity regions. The latter is illustrated by Figure 2.2, which shows the expected recourse function

$$x \mapsto \int_{\Xi} \Phi(q, h(\xi) - Tx) dP(\xi) \quad (x \in [0, 5]^2)$$

with  $r = s = 2$ ,  $h(\xi) = \xi$ ,  $m_1 = 0$ ,  $W_1 = 0$ ,  $m_2 = 4$ ,  $q = (-16, -19, -23, -28)^\top$ ,

$$W_2 = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 6 & 1 & 3 & 2 \end{pmatrix}, \quad T = \begin{pmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{pmatrix},$$

and binary restrictions on the second stage variables as in Schultz et al. [1998], but with a uniform probability distribution  $P$  having a smaller finite support than in Schultz et al. [1998], namely,  $\text{supp}(P) = \{5, 10, 15\}^2$ .

### 2.3. Quantitative Stability of Fully-Random Two-Stage Stochastic Mixed-Integer Linear Programs

With  $v(P)$  and  $S(P)$  denoting the optimal value and solution set of (1.1), respectively, the quantitative stability results for stochastic programs developed in Römisch [2003] (see Theorems 5 and 9 in Römisch [2003]) imply, in particular, the estimates

$$|v(P) - v(Q)| \leq L \sup_{x \in X} \left| \int_{\Xi} f_0(x, \xi)(P - Q)(d\xi) \right| \quad (2.4)$$

$$\emptyset \neq S(Q) \subseteq S(P) + \Psi_P \left( L \sup_{x \in X} \left| \int_{\Xi} f_0(x, \xi)(P - Q)(d\xi) \right| \right), \quad (2.5)$$

where  $L > 0$  is some constant,  $X$  is assumed to be compact,  $P$  and  $Q$  belong to a suitable subset of  $\mathcal{P}(\Xi)$ , and  $\Psi_P$  is the *conditioning function*. The function  $\Psi_P$  depends on the growth behavior of the objective function near the solution set and is defined as

$$\Psi_P(\eta) := \eta + \psi_P^{-1}(2\eta) \quad (\eta \in \mathbb{R}_+), \quad (2.6)$$

where the *growth function*  $\psi_P$  is

$$\psi_P(\tau) := \min \left\{ \int_{\Xi} f_0(x, \xi)P(d\xi) - v(P) : d(x, S(P)) \geq \tau, x \in X \right\} \quad (\tau \in \mathbb{R}_+) \quad (2.7)$$

with inverse  $\psi_P^{-1}(t) := \sup\{\tau \in \mathbb{R}_+ : \psi_P(\tau) \leq t\}$ . Finally, we also define the function  $\phi_P$  on  $\mathbb{R}_+$  characterizing the tail behavior of  $P$  by  $\phi_P(0) = 0$  and

$$\phi_P(t) := \inf_{R \geq 1} \left\{ R^{r+1}t + \int_{\{\xi \in \Xi : \|\xi\|_{\infty} > R\}} \|\xi\|_{\infty}^2 P(d\xi) \right\} \quad (t > 0). \quad (2.8)$$

The functions  $\phi_P$  and  $\psi_P$  are nondecreasing,  $\Psi_P$  is increasing and all functions vanish at 0. Furthermore, one has  $\psi_P(\tau) > 0$  if  $\tau > 0$  and  $\Psi_P(\eta) \searrow 0$  if  $\eta \searrow 0$ .

In order to state quantitative stability results for model (1.1) and inspired by the estimates (2.4) and (2.5), we need a distance of probability measures that captures the behavior of  $f_0(x, \cdot)$ ,  $x \in X$ , in its continuity regions and the shape of these regions, respectively. This leads us to the following probability metric on  $\mathcal{P}_2(\Xi)$  for some  $k \in \mathbb{N}$ :

$$\zeta_{2, \text{ph}_k}(P, Q) := \sup \left\{ \left| \int_B f(\xi)(P - Q)(d\xi) \right| : f \in \mathcal{F}_2(\Xi), B \in \mathcal{B}_{\text{ph}_k}(\Xi) \right\}. \quad (2.9)$$

Here,  $\mathcal{B}_{\text{ph}_k}(\Xi)$  denotes the set of all polyhedra being subsets of  $\Xi$  and having at most  $k$  faces. The set  $\mathcal{F}_2(\Xi)$  contains all functions  $f : \Xi \rightarrow \mathbb{R}$  such that

$$|f(\xi)| \leq \max\{1, \|\xi\|^2\} \quad \text{and} \quad |f(\xi) - f(\tilde{\xi})| \leq \max\{1, \|\xi\|, \|\tilde{\xi}\|\} \|\xi - \tilde{\xi}\|$$

holds for all  $\xi, \tilde{\xi} \in \Xi$ .

### 2.3. Quantitative Stability Fully-Random Two-Stage Stoch. Mixed-Integer Linear Progr.

**Theorem 2.7.** *Let the conditions (C1)–(C4) be satisfied and  $X$  be compact. Then there exist constants  $L > 0$  and  $k \in \mathbb{N}$  such that*

$$\begin{aligned} |v(P) - v(Q)| &\leq L\phi_P(\zeta_{2,\text{ph}_k}(P, Q)) \\ \emptyset \neq S(Q) &\subseteq S(P) + \Psi_P(L\phi_P(\zeta_{2,\text{ph}_k}(P, Q)))\mathbb{B}, \end{aligned} \quad (2.10)$$

for each  $Q \in \mathcal{P}_2(\Xi)$ .

If  $\int_{\Xi} \|\xi\|^p P(d\xi) < +\infty$  for some  $p > 2$ , the estimate  $\phi_P(t) \leq Ct^{\frac{p-2}{p+r-1}}$  holds for every  $t \geq 0$  and some constant  $C > 0$ .

*Proof.* Since the function  $\Phi$  is lower semicontinuous on  $\mathcal{U} \times \mathcal{T}$  (Lemma 2.5),  $f_0$  is lower semicontinuous on  $X \times \Xi$  and, hence, a random lower semicontinuous function [Rockafellar and Wets, 1998, Example 14.31]. Using Lemma 2.5 we obtain the estimate

$$\begin{aligned} |f_0(x, \xi)| &\leq \|c\|\|x\| + D[\max\{1, \|h(\xi)\| + \|T(\xi)\|\|x\|\}(\|q(\xi)\| + d) \\ &\quad + \max\{1, \|q(\xi)\|\}(\|h(\xi)\| + \|T(\xi)\|\|x\|)] \\ &\leq C_1 \max\{1, \|\xi\|^2\} \end{aligned}$$

for each pair  $(x, \xi) \in X \times \Xi$  and some constant  $C_1$ . Hence, the objective function  $\langle c, x \rangle + \int_{\Xi} \Phi(q(\xi), h(\xi) - T(\xi)x)Q(d\xi)$  is finite (if  $Q \in \mathcal{P}_2(\Xi)$ ) and lower semicontinuous (due to Fatou's lemma). Since  $X$  is compact, the solution set  $S(Q)$  is nonempty.

From Proposition 2.6 we know that, for each  $R \geq 1$  and  $x \in X$ , there exist Borel subsets  $\Xi_{j,x}^R$ ,  $j \in [\nu]$ , of  $\Xi$  such that the function  $f_{j,x}^R(\cdot) := f_0(x, \cdot)\mathbf{1}_{\Xi_{j,x}^R}$  is Lipschitz continuous on  $\Xi_{j,x}^R$  with constant  $L_1R$ . We extend each function  $f_{j,x}^R(\cdot)$  to the whole of  $\Xi$  by preserving the Lipschitz constant. Then we have  $\frac{1}{L_1R}f_{j,x}^R(\cdot) \in \mathcal{F}_2(\Xi)$ .

Furthermore, Proposition 2.6 implies that the closures of  $\Xi_{j,x}^R$  are contained in  $\mathcal{B}_{\text{ph}_k}(\Xi)$  for some  $k \in \mathbb{N}$ , that the number  $\nu$  is bounded above by  $\kappa R^r$ , where the constant  $\kappa > 0$  is independent on  $R$ , and that  $\Xi_{0,x}^R := \Xi \setminus \bigcup_{j=1}^{\nu} \Xi_{j,x}^R$  is a subset of  $\{\xi \in \Xi : \|\xi\|_{\infty} > R\}$ .

For each  $Q \in \mathcal{P}_2(\Xi)$  and  $x \in X$  we obtain

$$\begin{aligned} \left| \int_{\Xi} f_0(x, \xi)(P - Q)(d\xi) \right| &= \left| \sum_{j=0}^{\nu} \int_{\Xi_{j,x}^R} f_0(x, \xi)(P - Q)(d\xi) \right| \\ &\leq \sum_{j=1}^{\nu} \left| \int_{\Xi_{j,x}^R} f_{j,x}^R(\xi)(P - Q)(d\xi) \right| + I_x^R(P, Q) \\ &\leq \nu L_1 R \sup_{f \in \mathcal{F}_2(\Xi), j \in [\nu]} \left| \int_{\Xi} f(\xi)\chi_{\Xi_{j,x}^R}(\xi)(P - Q)(d\xi) \right| + I_x^R(P, Q), \end{aligned}$$

where the last summand on the right-hand side is given by

$$I_x^R(P, Q) := \left| \int_{\Xi_{0,x}^R} f_0(x, \xi)(P - Q)(d\xi) \right|.$$

## 2. Stability of Two-Stage Stochastic Mixed-Integer Linear Programming Problems

Using  $\nu \leq \kappa R^r$  and arguing as in Theorem 35 of Römisch [2003] we continue

$$\left| \int_{\Xi} f_0(x, \xi)(P - Q)(d\xi) \right| \leq \kappa L_1 R^{r+1} \zeta_{2, \text{ph}_k}(P, Q) + I_x^R(P, Q).$$

For the term  $I_x^R(P, Q)$  we use the estimate  $|f_0(x, \xi)| \leq C_1 \|\xi\|^2$  for any pair  $(x, \xi) \in X \times \{\xi \in \Xi : \|\xi\|_{\infty} > R\}$  and the norming constant  $C_2$  such that  $\|\xi\| \leq C_2 \|\xi\|_{\infty}$  holds for all  $\xi \in \mathbb{R}^s$ . We get

$$I_x^R(P, Q) \leq C_1 C_2^2 \int_{\{\xi \in \Xi : \|\xi\|_{\infty} > R\}} \|\xi\|_{\infty}^2 (P + Q)(d\xi).$$

Since the set  $\{\xi \in \Xi : \|\xi\|_{\infty} > R\}$  can be covered by  $2^s$  intersections of  $\Xi$  with open halfspaces (whose closures belong to  $\mathcal{B}_{\text{ph}_k}(\Xi)$ ), we can estimate

$$\int_{\{\xi \in \Xi : \|\xi\|_{\infty} > R\}} \|\xi\|_{\infty}^2 Q(d\xi) \leq 2^s \zeta_{2, \text{ph}_k}(P, Q) + \int_{\{\xi \in \Xi : \|\xi\|_{\infty} > R\}} \|\xi\|_{\infty}^2 P(d\xi).$$

Hence, combining the last three estimates we get

$$\begin{aligned} \sup_{x \in X} \left| \int_{\Xi} f_0(x, \xi)(P - Q)(d\xi) \right| \\ \leq (\kappa L_1 R^{r+1} + C_1 C_2^2 2^s) \zeta_{2, \text{ph}_k}(P, Q) + 2C_1 C_2^2 \int_{\{\xi \in \Xi : \|\xi\| > R\}} \|\xi\|_{\infty}^2 P(d\xi) \end{aligned}$$

for any  $R \geq 1$ . Taking the infimum with respect to  $R \geq 1$  we obtain

$$\sup_{x \in X} \left| \int_{\Xi} f_0(x, \xi)(P - Q)(d\xi) \right| \leq \hat{C} \phi_P(\zeta_{2, \text{ph}_k}(P, Q))$$

with some constant  $\hat{C} > 0$ .

Now, the result is a consequence of the estimates (2.4) and (2.5). If  $\int_{\Xi} \|\xi\|^p dP(\xi) < +\infty$  for some  $p > 2$ , it holds that

$$\int_{\{\xi \in \Xi : \|\xi\|_{\infty} > R\}} \|\xi\|_{\infty}^2 dP(\xi) \leq R^{2-p} \int_{\Xi} \|\xi\|_{\infty}^p P(d\xi)$$

by Markov's inequality. The desired estimate follows by inserting  $R = t^{-\frac{1}{p+r-1}}$  for small  $t > 0$  into the function whose infimum with respect to  $R \geq 1$  is  $\phi_P(t)$ :

$$\phi_P(t) \leq t^{-\frac{r+1}{p+r-1}+1} + t^{\frac{p-2}{p+r-1}} \int_{\Xi} \|\xi\|_{\infty}^p P(d\xi) \leq C t^{\frac{p-2}{p+r-1}}. \quad \square$$

The boundedness condition on  $X$  may be relaxed if localized optimal values and solution sets are considered (see Römisch [2003]). In case that the underlying distribution  $P$  and its perturbations  $Q$  have supports in some bounded subset  $\Xi$  of  $\mathbb{R}^s$ , the stability result improves slightly.

### 2.3. Quantitative Stability Fully-Random Two-Stage Stoch. Mixed-Integer Linear Progr.

**Corollary 2.8.** *Let the conditions (C1)–(C4) be satisfied,  $P \in \mathcal{P}(\Xi)$ ,  $X$  and  $\Xi$  be bounded. Then there exist constants  $L > 0$  and  $k \in \mathbb{N}$  such that*

$$\begin{aligned} |v(P) - v(Q)| &\leq L\zeta_{2,\text{ph}_k}(P, Q) \\ \emptyset \neq S(Q) &\subseteq S(P) + \Psi_P(L\zeta_{2,\text{ph}_k}(P, Q))\mathbb{B}, \end{aligned}$$

holds for each  $Q \in \mathcal{P}(\Xi)$ .

*Proof.* Since  $\Xi$  is bounded, we have  $\mathcal{P}_2(\Xi) = \mathcal{P}(\Xi)$ . Moreover, the function  $\phi_P(t)$  (see (2.8)) can be estimated by  $R^{r+1}t$  for some sufficiently large  $R > 0$ . Hence, Theorem 2.7 implies the assertion.  $\square$

**Remark 2.9.** Since  $\Xi \in \mathcal{B}_{\text{ph}_k}(\Xi)$  for some  $k \in \mathbb{N}$ , we obtain from (2.9) by choosing  $B := \Xi$  and  $f \equiv 1$ , respectively,

$$\max\{\zeta_2(P, Q), \alpha_{\text{ph}_k}(P, Q)\} \leq \zeta_{2,\text{ph}_k}(P, Q) \quad (2.11)$$

for all  $P, Q \in \mathcal{P}_2(\Xi)$ . Here,  $\zeta_2$  and  $\alpha_{\text{ph}_k}$  denote the second order Fortet-Mourier metric [Rachev, 1991, Section 5.1] and the polyhedral discrepancy

$$\begin{aligned} \zeta_2(P, Q) &:= \sup_{f \in \mathcal{F}_2(\Xi)} \left| \int_{\Xi} f(\xi)P(d\xi) - \int_{\Xi} f(\xi)Q(d\xi) \right| \\ \alpha_{\text{ph}_k}(P, Q) &:= \sup_{B \in \mathcal{B}_{\text{ph}_k}(\Xi)} |P(B) - Q(B)|, \end{aligned}$$

respectively. Hence, convergence with respect to  $\zeta_{2,\text{ph}_k}$  implies weak convergence (see Billingsley [1968]), convergence of second order absolute moments, and convergence with respect to the polyhedral discrepancy  $\alpha_{\text{ph}_k}$ . For bounded  $\Xi \subset \mathbb{R}^s$  the technique in the proof of Proposition 3.1 in Schultz [1996] can be employed to obtain

$$\zeta_{2,\text{ph}_k}(P, Q) \leq C_s \alpha_{\text{ph}_k}(P, Q)^{\frac{1}{s+1}} \quad (P, Q \in \mathcal{P}(\Xi)) \quad (2.12)$$

for some constant  $C_s > 0$ . In view of (2.11) and (2.12) the metric  $\zeta_{2,\text{ph}_k}$  is stronger than  $\alpha_{\text{ph}_k}$  in general, but if  $\Xi$  is bounded, both distances metricize the same topology on  $\mathcal{P}(\Xi)$ .

For more specific models (1.1), improvements of the above results may be obtained by exploiting specific recourse structures, i.e., by using additional information on the shape of the sets  $\mathcal{B}_i$  in Lemma 2.5 and on the behavior of the (value) function  $\Phi$  on these sets. This may lead to stability results with respect to probability metrics that are (much) weaker than  $\zeta_{2,\text{ph}_k}$ . For example, if  $W_1 = 0$ ,  $\Xi$  is rectangular,  $T$  is fixed and some components of  $h(\cdot)$  coincide with some of the components of  $\xi$ , the closures of  $\Xi_{x,j}$ ,  $x \in X$ ,  $j \in \mathbb{N}$ , are rectangular subsets of  $\Xi$ , i.e., belong to

$$\mathcal{B}_{\text{rect}} := \{I_1 \times I_2 \times \cdots \times I_s : \emptyset \neq I_j \text{ is a closed interval in } \mathbb{R}, j \in [s]\},$$

then the stability estimate (2.10) is valid with respect to  $\zeta_{2,\text{rect}}$ . As shown in Henrion et al. [2009] convergence of a sequence of probability distributions with respect to  $\zeta_{2,\text{rect}}$

## 2. Stability of Two-Stage Stochastic Mixed-Integer Linear Programming Problems

is equivalent to convergence with respect to both  $\zeta_2$  and  $\alpha_{\text{rect}}$ . If, in addition to the previous assumptions,  $q$  is fixed and  $\Xi$  is bounded, the estimate (2.10) is valid with respect to the rectangular discrepancy  $\alpha_{\text{rect}}$  (see also Section 3 in Schultz [1996]).

### 3. Decomposition Algorithms for Stochastic Programming Problems

When the size of an optimization problem becomes intractable for standard solution approaches, a decomposition into small tractable subproblems by relaxing certain coupling constraints is often a possible resort. The task of the decomposition algorithm is then to coordinate the search in the subproblems in a way that their solutions can be combined into one that is feasible for the overall problem and has a “good” objective function value. Often, the algorithm also provides a certified lower bound on the optimal value which allows to evaluate the quality of a found solution. Since, on the one hand, stochastic programs easily reach a size that is intractable for standard solution approaches, but, on the other hand, are also very structured, many decomposition algorithms have been developed, see, e.g., the survey Ruszczyński [2003] for stochastic linear problems and the surveys Louveaux and Schultz [2003], Schultz [2003], and Sen [2005] and the very comprehensive bibliography van der Vlerk [2007] for stochastic mixed-integer linear problems. In the following, we discuss some decomposition methods in more detail.

Many decomposition algorithms for stochastic programs are based on a temporal decomposition that decouples the decisions from the various stages by exploring the structure of the value or expected recourse function (in the two-stage case) or the cost-to-go function (in the multistage case), respectively. These algorithms are discussed in more detail in Sections 3.1 and 3.2. Alternatively, *Lagrangian Decomposition* (also called *dual decomposition*) is a class of algorithms where decomposition is achieved by relaxation of problem constraints. By moving certain coupling restrictions from the set of constraints into the objective function as penalty term, the problem decomposes into a set of subproblems, each of them often much easier to handle than the original problem. This relaxed problem then yields a lower bound onto the original optimal value, which is further improved by optimization of the penalty parameters. Since, in general, a solution of the relaxed problem violates the coupling constraints, heuristics and branch-and-bound approaches are applied to obtain good feasible solutions of the original problem. While there are several alternatives to choose a set of coupling constraints, each one providing a lower bound of different quality when relaxed [Dentcheva and Römis, 2004], in general, scenario and geographical decomposition are the preferred strategies [Carøe and Schultz, 1999, Nowak and Römis, 2000]. In scenario decomposition, nonanticipativity constraints  $x_t \in \mathcal{F}_t$  are relaxed, so that the problem decomposes into one deterministic subproblem for each scenario. We discuss this approach in more detail in Section 3.3. In geographical decomposition, model-specific constraints are relaxed, which leads to one subproblem for each component of the model. Even though each subproblem then corresponds to a stochastic program itself, its structure often allows to develop specialized

### 3. Decomposition Algorithms for Stochastic Programming Problems

algorithms to solve them very efficiently. Similarly, the modelers knowledge can be explored to make solutions from the relaxed problem feasible for the original problem. Geographical decomposition is demonstrated for unit commitment problems in Nowak and Römisch [2000], Gröwe-Kuska et al. [2002], and Römisch and Vigerske [2010].

The presentation in Sections 3.2 and 3.3 is taken from Römisch and Vigerske [2010].

## 3.1. Temporal Decomposition for Stochastic Linear Programs

Temporal decomposition algorithms for the two-stage stochastic linear program (1.7) are based on decoupling the recourse decisions  $y(\omega)$  on the second stage from the first-stage decision  $x$ . The decomposition is achieved by replacing the complicated recourse function  $\Phi(u, t)$  or the expected recourse function  $\Psi(x)$  by an easier to handle approximation, i.e., a more explicit formula. In a *Benders Decomposition* [Benders, 1962, Van Slyke and Wets, 1969], the functions  $\Phi(u, t)$  or  $\Psi(x)$  are approximated by supporting hyperplanes. We will discuss this method in more detail in Section 3.1.1.

For multistage stochastic linear programs (1.9), a temporal decomposition can be achieved by applying the Benders Decomposition principle recursively to the cost-to-go functions  $\mathcal{Q}_t(x_{t-1}, \xi_{[t]})$  as defined in (1.10). This *Nested Benders Decomposition* method [Birge, 1985] is discussed in more detail in Section 3.1.2.

### 3.1.1. Benders Decomposition

We consider the two-stage stochastic linear program (1.7) with a finite distribution  $\{\xi^i\}_{i \in I}$ , c.f. Section 1.3.1. That is, (1.7a) can be written as

$$\inf \left\{ \langle c, x \rangle + \sum_{i \in I} p_i \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x) : Ax \leq b \right\}. \quad (3.1)$$

Since the second-stage problem  $\Phi(u, t) = \inf\{\langle u, y \rangle : Wy \leq t\}$ , c.f. (1.7b), is a linear program, its dual can be written in the form

$$\max\{\langle \pi, t \rangle : W^\top \pi = u, \pi \geq 0\} \quad (3.2)$$

According to linear programming duality, the optimal value of (3.2) equals  $\Phi(u, t)$ , unless both problems are infeasible, and if the optimal values are finite, then both (1.7b) and (3.2) have an optimal solution. Note, that the set

$$\Pi(u) := \{\pi \in \mathbb{R}^r : W^\top \pi = u, \pi \geq 0\}$$

is convex, closed, and polyhedral, and, hence, has a finite number of extreme points. Thus, for fixed  $u$  with  $\Pi(u) \neq \emptyset$ , the function

$$t \mapsto \Phi(u, t) = \sup\{\langle \pi, t \rangle : W^\top \pi = u, \pi \geq 0\}$$

### 3.1. Temporal Decomposition for Stochastic Linear Programs

is a *convex polyhedral function*, i.e., its domain is a closed convex polyhedron and the function is convex and piecewise linear on its domain [Ruszczynski and Shapiro, 2003, Chapter 2, Def. 10]. Thus, also the function  $x \mapsto \Phi(q(\xi), h(\xi) - T(\xi)x)$  is convex and polyhedral, if  $\Pi(q(\xi)) \neq \emptyset$  and there exists at least one  $x \in \mathbb{R}^m$  with  $\Phi(q(\xi), h(\xi) - T(\xi)x) < \infty$  [Ruszczynski and Shapiro, 2003, Chapter 2, Prop. 11]. Condition  $\Pi(q(\xi)) \neq \emptyset$  is also called *dual feasibility* and asserts that the primal second-stage problem (1.7b) is not unbounded with respect to the costs  $u = q(\xi)$ .

The *Benders Decomposition* method [Van Slyke and Wets, 1969] for (1.7) is based on the observation that if  $t \mapsto \Phi(u, t)$  is finite at  $t = t_0$ , then its subdifferential at the point  $t_0$  is given by the set of optimal solutions to (3.2) [Ruszczynski and Shapiro, 2003, Chapter 2, Prop. 12]. Therefore, the function  $t \mapsto \Phi(u, t)$  can be represented on its domain as the maximum of a finite number of *supporting hyperplanes*  $\langle \pi, t \rangle$ , each corresponding to an extreme point  $\pi$  of  $\Pi(u)$ . The idea of Benders Decomposition is to successively compute a number of extreme points  $\pi \in \Pi(u)$  that allow for a good approximation of  $t \mapsto \Phi(u, t)$  (and thus  $x \mapsto \Phi(q(\xi), h(\xi) - T(\xi)x)$ ) on its domain via supporting hyperplanes. These supporting hyperplanes are also called *optimality cuts*.

In the case that  $\Phi(u, t) = \infty$ , i.e., infeasibility of the second-stage problem, the dual problem (3.2) is unbounded. Thus, there exists a  $\pi \geq 0$  with  $W^\top \pi = 0$  and  $\langle \pi, t \rangle > 0$ . Such a *dual ray* can be used to approximate the (polyhedral) domain of  $\Phi(u, t)$  via the inequality  $\langle \pi, t \rangle \leq 0$ , which is called a *feasibility cut*. If  $\Phi(q(\xi), h(\xi) - T(\xi)x) = -\infty$  for some scenario  $\xi$  of positive probability and some  $x \in X$ , (1.7) is unboundedness.

The *Benders Decomposition* algorithm alternates between computing first-stage solutions  $x \in X$  from an approximation of (3.1) and improving the approximation by computing new optimality or feasibility cuts from dual solutions of the second-stage problems  $\Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x)$ ,  $i \in I$ . The algorithm for the case that  $X$  is bounded is detailed in Algorithm 3.1. Since  $\Pi(u)$ ,  $u \in \mathbb{R}^{m_1}$ , has only finitely many extreme points, the algorithm terminates in a finite number of steps, either by discovering infeasibility or unboundedness of (1.7) or by finding an optimal solution [Ruszczynski, 2003, Theorem 4]. If  $X$  is not bounded, the master problem may be unbounded. In this case, Van Slyke and Wets [1969] propose to derive optimality or feasibility cuts from solving the second-stage problems with  $t = -T(\xi^i)x_r$ , where  $x_r$  is an unbounded ray of the master problem. This allows to either discover unboundedness of (1.7) or to make the master problem bounded.

Note, that in each iteration the value  $\langle c, \bar{x} \rangle + \bar{\theta}$  gives a lower bound on the optimal value of (1.7). Further, if all second-stage problems could be solved to optimality, then an upper bound is given by  $\langle c, \bar{x} \rangle + \sum_{i \in I} p_i \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)\bar{x})$ . These bounds are used in line 19 to decide whether the current solution  $\bar{x}$  is optimal. If interrupted prematurely, a feasible solution is given by the primal solutions of the first- and second-stage problems (provided none of them was infeasible) and an estimate on the distance between current objective value and optimal value is given by the gap between lower and upper bound.

The aggregation of the optimality cuts in line 22 reduces the number of cuts that are added to the master problem. However, a stronger relaxation may be obtained by replacing the single variable  $\theta$  by a sum  $\sum_{i \in I} p_i \theta_i$  in the objective function and adding scenario-wise optimality cuts  $\langle \pi^i, h(\xi^i) \rangle - \langle T(\xi^i)^\top \pi^i, x \rangle \leq \theta_i$  to the master problem. This variant is called *multicut version* [Birge and Louveaux, 1988, Trukhanov et al., 2010].

### 3. Decomposition Algorithms for Stochastic Programming Problems

---

**Algorithm 3.1:** Benders Decomposition algorithm for two-stage stochastic linear programs.

---

1  $C_{\text{feas}} \leftarrow \emptyset, C_{\text{opt}} \leftarrow \emptyset;$

2 **loop**

3 solve the *master problem*

$$\min \left\{ \langle c, x \rangle + \theta : \begin{array}{l} x \in X, \\ \alpha + \langle a, x \rangle \leq \theta, \quad (\alpha, a) \in C_{\text{opt}}, \\ \gamma + \langle g, x \rangle \leq 0, \quad (\gamma, g) \in C_{\text{feas}} \end{array} \right\}, \quad (3.3)$$

where  $\theta$  is omitted as long as  $C_{\text{opt}} = \emptyset;$

4 **if** (3.3) *is infeasible* **then STOP:** problem (1.7) is infeasible;

5 let  $(\bar{x}, \bar{\theta})$  be an optimal solution of (3.3) (with  $\bar{\theta} := -\infty$  if  $C_{\text{opt}} = \emptyset$ );

6 **foreach**  $i \in I$  **do**

7 solve the second-stage problem  $\Phi(q(\xi^i), h(\xi^i) - T(\xi^i)\bar{x});$

8 **if**  $\Phi(q(\xi^i), h(\xi^i) - T(\xi^i)\bar{x}) = \infty$  **then**

9 | let  $\bar{\pi} \in \Pi(0)$  with  $\langle \bar{\pi}, h(\xi^i) - T(\xi^i)\bar{x} \rangle > 0$  be a dual ray;

10 | add the feasibility cut  $(\gamma, g) := (\langle \bar{\pi}, h(\xi^i) \rangle, -T(\xi^i)^\top \bar{\pi})$  to  $C_{\text{feas}};$

11 | **break;**

12 **else if**  $\Phi(q(\xi^i), h(\xi^i) - T(\xi^i)\bar{x}) \neq -\infty$  **then**

13 | let  $\pi^i \in \Pi(q(\xi^i))$  be an optimal dual solution;

14 | **end**

15 **end**

16 **if** *no feasibility cuts were added* **then**

17 | **if** *some second-stage problem was unbounded* **then**

18 | | **STOP:** (1.7) is unbounded;

19 | **else if**  $\bar{\theta} = \sum_{i \in I} p_i \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)\bar{x})$  **then**

20 | | **STOP:** (1.7) has been solved to optimality;

21 | **end**

22 | add to  $C_{\text{opt}}$  the aggregated optimality cut

$$\alpha := \sum_{i \in I} p_i \langle \pi^i, h(\xi^i) \rangle, \quad a := - \sum_{i \in I} p_i T(\xi^i)^\top \pi^i;$$

23 | **end**

24 **end;**

---

#### Extensions and Variations

A difficulty with the Benders Decomposition method may be a large number of cuts that need to be stored in the master problem. To overcome this problem, and also to make use of starting solution, Ruszczyński [1986] proposed the *Regularized Decomposition* method, where a convex quadratic term  $\frac{\rho}{2} \|x - x^c\|_2^2$ ,  $\rho > 0$ ,  $x^c \in \mathbb{R}^m$ , is added to the

### 3.1. Temporal Decomposition for Stochastic Linear Programs

objective function of the master problem. The term penalizes deviation from the center  $x^c$  and thereby prevents very large steps, which may otherwise be made due to poor approximations of the recourse function in early iterations of the algorithm. The center  $x^c$  is replaced by the solution  $\bar{x}$  of the master problem, if the value of the approximated recourse function is close to the actual value, i.e., if  $\bar{\theta} \approx \sum_i p_i \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)\bar{x})$ . If the latter is not the case, only the approximation is updated, but  $x^c$  remains unchanged.

Alternatively to adding a quadratic penalty term into the objective function to avoid very large steps, an explicit restriction  $\|x - x^c\| \leq \Delta$ ,  $\Delta > 0$ , can be added to the master problem [Linderoth and Wright, 2003]. The norm  $\|\cdot\|$  may be chosen to be either the one- or the supremum-norm, so linearity of the master problem is preserved. Restricting the master problem solutions to  $\{x \in \mathbb{R}^m : \|x - x^c\| \leq \Delta\}$  defines a *trust region*, in which the master problem is believed to be a good approximation of the original problem (1.7) [Moré, 1983]. Updating of the center point  $x^c$  and radius  $\Delta$  is done in a similar way as in the Regularized Decomposition method.

If  $\xi$  is a continuous distribution or a discrete distribution with a very large number of scenarios, evaluating the second-stage problem (1.7b) for all realizations of  $\xi$  in each iteration of the Benders Decomposition method may not be feasible. If an a-priori discretization of  $\xi$  via scenario construction and reduction techniques, c.f. Section 1.3.1, is undesired, one may approximate the recourse function  $\Psi(\cdot)$  by supporting hyperplanes derived from evaluations of  $\Phi(q(\cdot), h(\cdot) - T(\cdot)x)$  on sampled scenarios. The *stochastic decomposition* algorithm in Higle and Sen [1991] alternates between solving a master problem similar to (3.3) and evaluating the recourse function for a sampled scenario  $\xi$ . Together with supporting hyperplanes on  $\Phi(q(\cdot), h(\cdot) - T(\cdot)x)$  from previous samples, an estimate on the expected recourse function  $\Psi(\cdot)$  is constructed.

Alternatively, one may estimate the expected recourse function  $\Psi(x)$  for a fixed  $x \in \mathbb{R}^m$  by a sample average approximation where samples are obtained via Monte Carlo sampling with respect to the distribution  $P$ . To overcome difficulties with a large variance of  $\Psi(x)$ , *importance sampling* may be used [Ruszczynski, 2003, Shapiro, 2003a]. Here, the scenarios are not sampled w.r.t. the original distribution  $P$ , but w.r.t. a distribution that is proportional to an approximation of  $\Phi(q(\cdot), h(\cdot) - T(\cdot)x)/\Psi(x)P(\cdot)$ . The way how such an approximation is constructed is problem dependent.

#### 3.1.2. Nested Benders Decomposition

Consider the multistage stochastic linear program (1.9) and assume a finite distribution  $\{\xi^i\}_{i \in I}$ , c.f. Section 1.3.2. This assumption allows to write (1.9) as the linear program

$$\min \left\{ \sum_{i \in I} p_i \sum_{t \in [T]} \langle b_t(\xi_t^i), x_t^i \rangle : \begin{array}{l} x_t^i \in X_t, i \in I, t \in [T], \\ x_t^i = x_t^j \text{ for all } i, j \in I : \xi_{[t]}^i = \xi_{[t]}^j, t \in [T], \\ A_{t,0}(\xi_t^i)x_t^i + A_{t,1}(\xi_t^i)x_{t-1}^i = h_t(\xi_t^i), i \in I, t \in [2 : T] \end{array} \right\} \quad (3.4)$$

where the second line of constraints are the nonanticipativity constraints, which require decisions along two scenarios to coincide as long as the scenarios are not distinguishable. In other words, we can associate with each node in the scenario tree given by  $\{\xi^i\}_{i \in I}$

### 3. Decomposition Algorithms for Stochastic Programming Problems

a unique decision variable  $x_t$ . Similarly, we can associate the cost-to-go functions  $x_{t-1} \mapsto \mathcal{Q}_t(x_{t-1}, \xi_{[t]}^i)$ ,  $t \in [2 : T]$ ,  $i \in I$ , cf. (1.10b)–(1.10c), with the nodes of the scenario tree, and associate the root node with the discrete version of (1.10a):

$$\min \left\{ \langle b_1, x_1 \rangle + \sum_{i \in I} p_i \mathcal{Q}_2(x_1, \xi_1^i, \xi_2^i) : x_1 \in X_1 \right\}. \quad (3.5)$$

Problem (3.5) has a similar form as problem (1.7a) in the two-stage case. While the recourse function in the two-stage case was the optimal value function of a linear program, in the multistage case the cost-to-go functions are optimal value functions of a linear program that have additional cost-to-go functions in the objective function. However, since the last-stage cost-to-go functions  $x_{T-1} \mapsto \mathcal{Q}_T(x_{T-1}, \xi^i)$  are convex polyhedral functions (analog to  $t \mapsto \Phi(u, t)$  in the two-stage case), also the cost-to-go functions at stage  $T - 1$  are convex polyhedral functions. Recursively, it follows that all cost-to-go functions are convex and polyhedral [Ruszczynski and Shapiro, 2003, Chapter 2, Prop. 30].

Therefore, the idea of the *Nested Benders Decomposition* method [Birge, 1985] is to apply the Benders Decomposition method recursively. That is, cost-to-go functions on each stage  $t$  are approximated by supporting hyperplanes, whereby the functions  $\mathcal{Q}_{t+1}(x_t, \xi_{[t+1]})$  in the objective function are replaced by supporting hyperplane approximations of the same kind. Note, that since the hyperplane approximations yield lower bounds for the true cost-to-go functions, a supporting hyperplane approximation of an approximated cost-to-go function is again a lower bound on the true cost-to-go function.

The Nested Benders Decomposition Algorithm (as multicut version) is detailed in Algorithm 3.2. For simplicity, we assume that all  $X_t$  are bounded, so no unbounded subproblems can occur. The value  $L$  in the initialization of the sets of optimality cuts  $C_{\text{opt}}(\xi_{[t]}^i)$  has to be some finite lower bound on the cost-to-go functions. It's only purpose is to avoid unbounded master problems in the first iterations of the algorithm, i.e., when not enough optimality cuts have been created yet. Alternatively, one may use  $L = 0$  while solving the master problem (3.6), but use  $L = -\infty$  when deciding whether to generate optimality cuts at stage  $t + 1$ . Note, that the master problems (3.6) can be written equivalently as linear optimization problem, where auxiliary variables  $\theta^j$  and constraints  $\theta^j \geq \alpha + \langle a, x_t \rangle$ ,  $(\alpha, a) \in C_{\text{opt}}(\xi_{[t+1]}^j)$ ,  $0 \geq \gamma + \langle g, x_t \rangle$ ,  $(\gamma, g) \in C_{\text{feas}}(\xi_{[t+1]}^j)$ , are added for all successors  $\xi_{[t+1]}^j$  of the current node  $\xi_{[t]}^i$  (i.e., all  $j \in I$  with  $\xi_{[t]}^j = \xi_{[t]}^i$ ). For details on how to construct optimality and feasibility cuts, we refer to Section 3.1.1.

The algorithm terminates when either infeasibility of the first stage master problem is recognized, or if there was a complete forward pass through all nodes in the scenario tree, where each subproblem was solved to optimality, i.e., no feasibility cuts were generated, and also no optimality cuts were generated. For the approximations of the cost-to-go functions at the last stage, this means  $\mathcal{Q}_T(x_{T-1}, \xi^i) = \underline{\mathcal{Q}}_T(x_{T-1}, \xi^i)$ , i.e., the approximation is exact at  $x_{T-1}$ . Accordingly, it follows that also approximated cost-to-go functions  $\underline{\mathcal{Q}}_{T-1}(x_{T-2}, \xi_{[T-1]}^i)$  are exact at the current value for  $x_{T-2}$ . Thus, recursively it follows that all approximations are exact at their current value, and thus the current solutions are optimal for the problem (1.9). Analogously to the two-stage

---

**Algorithm 3.2:** Nested Benders Decomposition Algorithm
 

---

$C_{\text{feas}}(\xi_{[t]}^i) \leftarrow \emptyset$ ,  $C_{\text{opt}}(\xi_{[t]}^i) \leftarrow \{(L, 0)\}$ ,  $t \in [2 : T]$ ,  $i \in I$ ;  $t \leftarrow 1$ ;  
**loop**  
   **foreach** node  $\xi_{[t]}^i$  of the scenario tree at level  $t$  **do**  
     **if**  $t > 1$  **then** get approx. sol.  $x_{t-1}$  from master problem in ancestor  $\xi_{[t-1]}^i$ ;  
     **if**  $t < T$  **then**  
       solve master problem associated with node  $\xi_{[t]}^i$ :  
         
$$\min \left\{ \begin{array}{l} \langle b_t(\xi_t^i), x_t \rangle + \mathbb{E} \left[ \underline{Q}_{t+1}(x_t, \xi_{[t+1]}) \mid \xi_{[t]} = \xi_{[t]}^i \right] : \\ x_t \in X_t, A_{t,0}(\xi_t^i)x_t + A_{t,1}(\xi_t^i)x_{t-1} = h_t(\xi_t^i) \end{array} \right\}, \quad (3.6)$$
  
       where  
         
$$\underline{Q}_{t+1}(x_t, \xi_{[t+1]}^i) := \begin{cases} \infty, & \text{if } \exists (\gamma, g) \in C_{\text{feas}}(\xi_{[t+1]}^i) : \\ & \gamma + \langle g, x_t \rangle > 0, \\ \max_{(\alpha, a) \in C_{\text{opt}}(\xi_{[t+1]}^i)} (\alpha + \langle a, x_t \rangle), & \text{otherwise,} \end{cases}$$
  
       is the current approximation of the cost-to-go functions at stage  $t + 1$ ;  
     **else**  
       evaluate cost-to-go function at final stage ( $\underline{Q}_T(x_{T-1}, \xi^i$ ) by solving  
         
$$\min \left\{ \langle b_T(\xi_T^i), x_T \rangle : x_T \in X_T, A_{T,0}(\xi_T^i)x_T + A_{T,1}(\xi_T^i)x_{T-1} = h_T(\xi_T^i) \right\}; \quad (3.7)$$
  
     **end**  
     **if**  $t = 1$  and (3.6) is infeasible **then STOP**: (1.9) is infeasible;  
     **if** (3.6) or (3.7) is infeasible **then**  
       add feasibility cut created from dual ray of infeas. problem to  $C_{\text{feas}}(\xi_{[t]}^i)$ ;  
       **break**;  
     **end**  
     **if**  $t > 1$  and optimal value of (3.6) or (3.7), respectively, is larger than the approximated value  $\underline{Q}_t(x_{t-1}, \xi_{[t]}^i)$  (computed when evaluating the master problem at stage  $t - 1$ ) **then** construct a supporting hyperplane from the optimal dual solution and add it  $C_{\text{opt}}(\xi_{[t]}^i)$ ;  
     **end**  
     **if** some (3.6) or (3.7) was infeasible **then**  $t \leftarrow t - 1$ ;  
     **else if**  $t = T$  and no new cuts have been created for any node **then**  
       **STOP**: (1.9) has been solved to optimality (the solution corresponds to the current optimal solutions of all master problems);  
     **else**  $t \leftarrow (t \bmod T) + 1$ ;  
   **end**;  
**end**;

---

### 3. Decomposition Algorithms for Stochastic Programming Problems

case, the polyhedrality of the cost-to-go functions ensures also for the Nested Benders Decomposition algorithm termination within a finite number of steps, either by discovering infeasibility of (1.9) or finding an optimal solution [Ruszczynski, 2003, Theorem 18].

Variations of Algorithm 3.2 are due to changing the order in which subproblems are evaluated. For example, instead of incrementing  $t$  in the last step, one may go back to step  $t - 1$  if new optimality cuts have been generated, thus allowing for an update of  $x_{t-1}$  and reevaluation of the master problems at stage  $t$  instead of stepping forward to step  $t + 1$  with solutions  $x_t$  that may already be outdated. Another common strategy is to first evaluate master problems with increasing  $t$  as long as possible, and to reevaluate master problems and passing back optimality cuts when traversing the tree in reverse direction (decreasing  $t$ ). For an evaluation of different sequencing protocols, see Gassmann [1990].

Further, note that the master problems that have to be solved for one time stage may be very similar. E.g., if the costs  $b_T(\cdot)$  and matrices  $A_{T,0}(\cdot)$  are deterministic, then the master problems at the last stage (3.7) only differ w.r.t. the right-hand-side  $h_T(\xi_T^i) - A_{T,1}(\xi_T^i)x_{T-1}$ . Similar, for the master problems at intermediate stages, differences may only be in the right-hand-sides and in the optimality and feasibility cuts. Strategies on how to organize the evaluation of a number of similar linear programs are discussed in Haugland and Wallace [1988], Gassmann [1990], and Gassmann and Wallace [1996].

Finally, due to decomposition along the scenario tree structure, the algorithm is well suited for parallelization, see, e.g., Dempster and Thompson [1998] and Birge et al. [1996].

#### The case of interstage independent random parameters

Consider now the case of a (not necessarily discrete) stochastic process  $\xi$  consisting of *independent* random variables  $\{\xi_t\}_{t \in [T]}$ . In this case, consecutive stages in a multistage stochastic program are only coupled by the decision vectors  $\{x_t\}_{t \in [T]}$  (which are not independent in general). Thus, the dependence of the cost-to-go function  $\mathcal{Q}_t(x_{t-1}, \xi_{[t]})$  on  $\xi_{[t]}$  simplifies to a dependency on  $\xi_t$  only. Hence, the dynamic programming formulation (1.10) of a multistage stochastic linear program can be written as

$$\min\{\langle b_1, x_1 \rangle + \mathbb{E}[\mathcal{Q}_2(x_1, \xi_2)]\},$$

where

$$\mathcal{Q}_t(x_{t-1}, \bar{\xi}_t) := \min \left\{ \begin{array}{l} \langle b_t(\bar{\xi}_t), x_t \rangle + \mathbb{E}[\mathcal{Q}_{t+1}(x_t, \xi_{t+1})] : \\ x_t \in X_t, A_{t,0}(\bar{\xi}_t)x_t + A_{t,1}(\bar{\xi}_t)x_{t-1} = h_t(\bar{\xi}_t), P\text{-a.e.} \end{array} \right\},$$

for  $t = 2, \dots, T - 1$ , and

$$\mathcal{Q}_T(x_{T-1}, \bar{\xi}_T) := \min \left\{ \langle b_T(\bar{\xi}_T), x_T \rangle : x_T \in X_T, A_{T,0}(\bar{\xi}_T)x_T + A_{T,1}(\bar{\xi}_T)x_{T-1} = h_T(\bar{\xi}_T) \right\}.$$

Note, that for every  $\xi_t(\omega)$ ,  $\omega \in \Omega$ , the same expected cost-to-go function  $\mathbb{E}[\mathcal{Q}_{t+1}(\cdot, \xi_{t+1})]$  is used in the objective of  $\mathcal{Q}_t(\cdot, \xi_t)$ . From the point of view of the Nested Benders Decomposition algorithm, this means that the expected cost-to-go functions  $x_t \mapsto \mathbb{E}[\mathcal{Q}_{t+1}(x_t, \xi_{t+1})]$ , which are approximated in the master problems for  $\xi_t(\omega)$ ,  $\omega \in \Omega$ ,

### 3.2. Temporal Decomposition for Two-Stage Stochastic Mixed-Integer Linear Programs

are all equal. This property is explored by Infanger and Morton [1996] for the case of a finite set of scenarios by introducing *cut sharing*. That is, supporting hyperplanes for the cost-to-go function at a stage- $(t + 1)$  master problem are used to approximate the cost-to-go functions in the objective of *all* stage- $t$  master problems. However, note, that for a finite set of scenarios with interstage independence, if there are  $d_t$  outcomes of  $\xi_t$  at time  $t$ , the number of possible outcomes for  $\xi_{[t]}$  is  $\prod_{\tau=1}^t d_\tau$ . As a result, the corresponding scenario tree has  $\prod_{t=1}^T d_t$  nodes. Since the decision process  $\{x_t\}_{t \in [T]}$  still can take different values in each node of the underlying scenario tree, the number of master problems that have to be evaluated during Nested Benders Decomposition may still increase exponentially with the number of time stages. A possibility to overcome this difficulty is discussed in Chapter 4.

Another way to make use of interstage independent random variables has been introduced by Pereira and Pinto [1991] under the term *stochastic dual dynamic programming*. Here, the expected cost-to-go functions  $\mathbb{E}[\mathcal{Q}_t(\cdot, \xi_t)]$  are approximated by a sample average approximation of supporting hyperplanes for  $\mathcal{Q}_t(\cdot, \xi_t)$ . That is, at time  $t$ ,  $t \in [2 : T]$ ,  $n$  master problems are evaluated w.r.t. a set of scenarios sampled from  $\xi_t$  and the solutions  $x_{t-1}$  from the  $n$  master problems at time  $t - 1$  (or w.r.t. the first stage decision if  $t = 2$ ). The dual solutions of the  $n$  master problems are then averaged to generate a single cut that is added to the approximation of the stage- $t$  expected cost-to-go function ( $\mathbb{E}[\mathcal{Q}_t(\cdot, \xi_t)]$ ) in the master problems at stage  $t - 1$ . The primal solutions are used to evaluate the master problems at time  $t + 1$  w.r.t.  $n$  samples from  $\xi_{t+1}$ . Convergence of this algorithm is analyzed, e.g., in Shapiro [2011].

Infanger and Morton [1996] also suggested an extension of the cut sharing methodology for interstage independence to restricted forms of interstage dependence, see Morton [1996] for an application. To catch a glimpse, observe that in the two-stage case, any solution to the dual problem (3.2) is feasible for any parameter  $t$ . Thus, if the cost function  $q \equiv q(\xi)$  is deterministic (i.e., independent of  $\xi$ ), then for any dual feasible solution  $\pi$  of  $\Phi(q, h(\xi^i) - T(\xi^i)\bar{x})$  for some  $i \in I$ , the hyperplane  $x \mapsto \langle \pi, h(\xi^j) - T(\xi^j)x \rangle$  underestimates  $x \mapsto \Phi(q, h(\xi^j) - T(\xi^j)x)$  for all  $j \in I$ , but may not be supporting at  $\bar{x}$ .

## 3.2. Temporal Decomposition for Two-Stage Stochastic Mixed-Integer Linear Programs

Consider model (1.1) with (1.3)–(1.5). Further, we denote by

$$\bar{X} := \{x \in \mathbb{R}^m : Ax \leq b\}$$

the *linear relaxation* of  $X$ .

For continuous ( $m_2 = 0$ ) stochastic programs, the *Benders Decomposition* method as discussed in Section 3.1.1 is an established method. Unfortunately, this method relies heavily on the convexity of the value function  $t \mapsto \Phi(u, t)$ . Thus, in the view of Section 2.2, it cannot be directly applied to the case where discrete variables are present.

However, there are several approaches to overcome this difficulty. One of the earliest is

### 3. Decomposition Algorithms for Stochastic Programming Problems

the *Integer L-shaped method* [Laporte and Louveaux, 1993], which assumes that the first stage problem involves only binary variables. This property is exploited to derive linear inequalities that approximate the value function  $\Phi(u, t)$  pointwise. While the algorithm makes only moderate assumptions on the second stage problem, its main drawback is the weak approximation of the value function due to lack of first order information. Thus, the algorithm might enumerate all feasible first stage solutions to find an optimal solution.

A cutting-plane algorithm is proposed in Carøe and Tind [1997]. Here, the deterministic equivalent of (1.1) is solved by improving its linear relaxation with lift-and-project cuts. Decomposition arises here in two ways. First, the linear relaxation (including additional cuts) is solved by Benders Decomposition. Second, lift-and-project cuts are derived scenariowise. Further, in case of a fixed technology matrix  $T(\cdot) \equiv T$ , cut coefficients that have been computed for one scenario can also be reused to derive cuts for other scenarios. This algorithm can be seen as a predecessor of the dual decomposition approach presented in Sen and Hige [2005]. While the cuts in Carøe and Tind [1997] include variables from both stages, Sen and Hige [2005] extend the Benders Decomposition approach to the mixed-integer case by sequentially convexifying the value function  $\Phi(u, t)$ . This method is discussed in detail in Section 3.2.2.

In Klein Haneveld et al. [2006] it is observed, that even though the value function  $\Phi(u, t)$  might be nonconvex and difficult to handle, under some assumptions on the distribution of  $\xi$ , the expected recourse function  $\Psi(x)$  can be convex. Starting with simple integer recourse models and then extending to more general classes of problems, techniques to compute tight convex approximations of the expected recourse function by perturbing the distribution of  $\xi$  are developed in a series of papers [Klein Haneveld et al., 2006, van der Vlerk, 2004, 2010]. We sketch this approach in more detail in Section 3.2.1.

In the case that the second stage problem is purely integer ( $m_1 = 0$ ), the value function  $\Phi(u, t)$  has the nice property to be constant on polyhedral subsets of  $\mathcal{U} \times \mathcal{T}$ , cf. Lemma 2.5. Therefore, in case of a finite distribution, also the expected recourse function  $\Psi(x)$  is constant on polyhedral subsets of  $\bar{X}$ . This property allows to reduce the set  $X$  to a finite set of solution candidates that can be enumerated [Schultz et al., 1998]. Since the expected recourse function  $\Psi(x)$  has to be evaluated for each candidate, many similar integer programs have to be solved. In Schultz et al. [1998] a Gröbner basis for the second stage problem is computed once in advance (which is expensive) and then used for evaluation of  $\Psi(x)$  for every candidate  $x$  (which is then cheap).

Another approach based on enumerating the sets where  $\Psi(x)$  is constant is presented in Ahmed et al. [2004]. Instead of a complete enumeration, here a branch-and-bound algorithm is applied to the first stage problem to enumerate the regions of constant  $\Psi(x)$  implicitly. Branching is thereby performed along lines of discontinuity of  $\Psi(x)$ , thereby reducing its discontinuity in generated subproblems.

#### 3.2.1. Convexification of the Expected Recourse Function

In a simple integer recourse model, the second stage variables are purely integer ( $m_1 = 0$ ) and are partitioned into two sets  $y^+, y^- \in \mathbb{Z}_+^s$  with  $2s = m_2$ . The cost-vector  $q(\xi) \equiv (q^+, q^-)$  and technology matrix  $T(\xi) \equiv T$  are fixed,  $r = 2s$ ,  $h(\xi) \equiv \xi$ , and the value

### 3.2. Temporal Decomposition for Two-Stage Stochastic Mixed-Integer Linear Programs

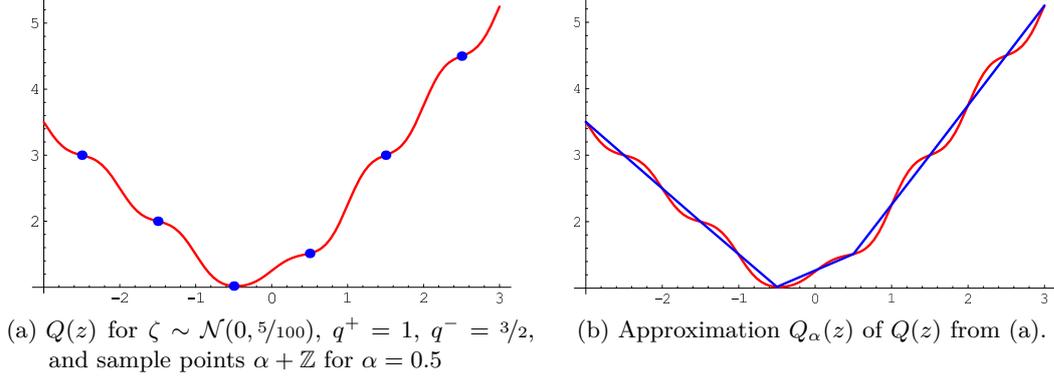


Figure 3.1.: Convex piecewise linear approximation of expected recourse function for a simple integer recourse model according to Klein Haneveld et al. [2006].

function takes the form

$$\Phi(q(\xi), h(\xi) - T(\xi)x) = \inf \left\{ \langle q^+, y^+ \rangle + \langle q^-, y^- \rangle : \begin{array}{l} y^+ \geq \xi - Tx, \\ y^- \geq -(\xi - Tx) \\ y^+, y^- \in \mathbb{Z}_+^s \end{array} \right\}.$$

The simple structure of the value function allows to write the expected recourse function in a separable form,

$$\Psi(x) = \sum_{i=1}^s q_i^+ \mathbb{E}[\lceil \xi_i - (Tx)_i \rceil^+] + q_i^- \mathbb{E}[\lfloor \xi_i - (Tx)_i \rfloor^-],$$

where  $\lceil \alpha \rceil$  denotes the smallest integer that is at least  $\alpha$ ,  $\lfloor \alpha \rfloor$  the largest integer that is at most  $\alpha$ ,  $\alpha^+ = \max(0, \alpha)$ , and  $\alpha^- = \min(0, \alpha)$ . Thus, it is sufficient to consider one-dimensional functions of the form

$$Q(z) := q^+ \mathbb{E}_\zeta[\lceil \zeta - z \rceil^+] + q^- \mathbb{E}_\zeta[\lfloor \zeta - z \rfloor^-],$$

with  $\zeta$  a random variable.

In Klein Haneveld et al. [2006], convex approximations of  $Q(z)$  are derived from a piecewise linear function in the points  $(z, Q(z))$ ,  $z \in \alpha + \mathbb{Z}$ , where  $\alpha \in [0, 1)$  is a parameter, see Figure 3.1. Further, if  $\zeta$  has a continuous distribution, then the approximation of  $Q(z)$  can be realized as expected recourse function of a continuous simple recourse model,

$$Q_\alpha(z) = q^+ \mathbb{E}_{\zeta_\alpha}[(\zeta_\alpha - z)^+] + q^- \mathbb{E}_{\zeta_\alpha}[(\zeta_\alpha - z)^-] + \frac{q^+ q^-}{q^+ + q^-},$$

where  $\zeta_\alpha$  is a discrete random variable with support in  $\alpha + \mathbb{Z}$  [Klein Haneveld et al., 2006].  $\zeta_\alpha$  has the same distribution as  $\lceil \zeta - \alpha \rceil + \alpha - \eta$ , where  $\eta$  is a Bernoulli distributed random variable (independent of  $\zeta$ ) with parameter  $\frac{q^-}{q^+ + q^-}$ .

### 3. Decomposition Algorithms for Stochastic Programming Problems

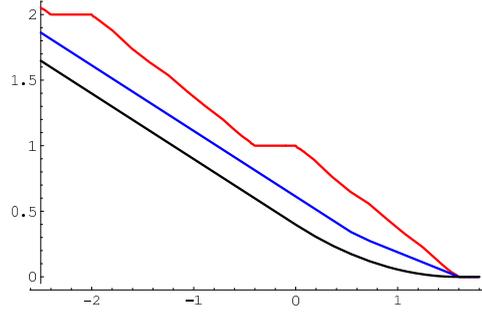


Figure 3.2.: Convex piecewise linear (middle) and linear programming relaxation (down) based underestimators of an expected recourse function (top) for a complete integer recourse model according to van der Vlerk [2004]. The expected recourse function is  $\mathbb{E}_\xi[\min_{y \in \mathbb{Z}_+^{m_2}} \langle q, y \rangle : Tx + Wy \geq \xi]$ , the piecewise linear underestimator is  $\mathbb{E}_{\xi_\alpha}[\min_{y \in \mathbb{R}_+^{m_2}} \langle q, y \rangle : Tx + Wy \geq \xi_\alpha]$  where  $\xi_\alpha := \lceil \xi - \alpha \rceil + \alpha$ , and the linear programming relaxation based underestimator is  $\mathbb{E}_\xi[\min_{y \in \mathbb{R}_+^{m_2}} \langle q, y \rangle : Tx + Wy \geq \xi]$ . Here,  $m = 1$ ,  $T = 1$ ,  $m_2 = 1$ ,  $W = 2$ ,  $\xi \sim \mathcal{U}(0, 1.6)$ , and  $\alpha = 0.6$ .

The results in Klein Haneveld et al. [2006] are extended to derive convex approximations of the expected recourse function for complete integer recourse models, that is (1.1) with (1.3)–(1.5) where  $m_1 = 0$ ,  $h(\xi) \equiv \xi$ ,  $q(\xi) \equiv q$  and  $T(\xi) \equiv T$  are fixed, and  $y_2 \geq 0$  [van der Vlerk, 2004]. Further, the parameter  $\alpha$  can be chosen such that the derived convex approximation underestimates the original expected recourse function. This convex underestimator is at least as good as an underestimator obtained by relaxing the integrality condition on  $y$ , see Figure 3.2. In the case that  $T$  is unimodular, it even yields the convex hull of  $\Psi(x)$ , and thus it can be utilized to derive lower bounds in a branch-and-bound search for a solution of (1.1).

Another extension of the work of Klein Haneveld et al. [2006] considers mixed-integer recourse models with  $r = 1$  and semi-periodic value function, c.f. van der Vlerk [2010].

#### 3.2.2. Convexification of the Value Function

From now on we assume that the random vector  $\xi$  has only finitely many outcomes  $\xi^i$  with probability  $p_i > 0$ ,  $i \in I$ . Thus, we can write the expected recourse function as

$$\Psi(x) = \sum_{i \in I} p_i \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x) \quad (x \in \bar{X}).$$

As discussed in Section 2.2, the nonconvexity of the function  $\Phi(u, t)$  forbids a representation by supporting hyperplanes as used by Benders Decomposition. However, while in the continuous case ( $m_2 = 0$ ) the hyperplanes are derived from dual feasible solutions of the second stage problem, it is conceptually possible to carry over these ideas to the mixed-integer case by introducing (possibly nonlinear) dual price functions

### 3.2. Temporal Decomposition for Two-Stage Stochastic Mixed-Integer Linear Programs

[Tind and Wolsey, 1981]. Indeed, Chvátal and Gomory functions are sufficiently large classes of dual price functions that allow to approximate the value function  $\Phi(u, t)$  [Blair and Jeroslow, 1982]. These dual functions can be obtained from a solution of (1.5) by a branch-and-bound or Gomory cutting plane algorithm [Wolsey, 1981]. In Carøe and Tind [1998], this approach is used to adapt the Benders Decomposition algorithm for two-stage linear stochastic programs to the mixed-integer linear case by replacing the hyperplane approximation of the expected recourse function by an approximation based on dual price functions. While Carøe and Tind [1998] left open how to solve the master problem with its (nonsmooth and nonconvex) dual price functions, Sen and Hingle [2005], Sen and Sherali [2006], and Ntamo and Sen [2005, 2008a,b] show that a careful construction of dual price functions combined with a convexification step based on disjunctive programming allows to implement an efficient Benders decomposition for mixed-integer two-stage stochastic programs. In the following, we provide a short overview on the developed techniques. For simplicity, we assume relatively complete recourse.

We consider the following *master problem* derived from (1.1) with (1.3)–(1.5) by replacing value functions  $x \mapsto \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x)$  by approximations  $\Theta_i : \mathbb{R}^m \rightarrow \mathbb{R}$ :

$$\min \left\{ \langle c, x \rangle + \sum_{i \in I} p_i \Theta_i(x) : x \in X \right\}, \quad (3.8)$$

where each function  $\Theta_i(\cdot)$ ,  $i \in I$ , is given in the form

$$\Theta_i(x) := \max \{ \min \{ \eta_1(x), \dots, \eta_k(x) \} : (\eta_1(\cdot), \dots, \eta_k(\cdot)) \in C_i \} \quad (x \in \bar{X}),$$

and a tuple  $\eta := (\eta_1(\cdot), \dots, \eta_k(\cdot)) \in C_i$  consists of  $k$  (where  $k$  is allowed to vary with  $\eta$ ) affine linear functions  $\eta_j(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}$ ,  $j \in [k]$ . The tuple  $\eta$  takes here the role of an optimality cut in Benders Decomposition for the continuous case, cf. Section 3.1.1. That is, each  $\eta \in C_i$  is constructed in a way such that for all  $x \in X$ ,

$$\Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x) \geq \eta_j(x) \text{ for at least one } j \in [k]. \quad (3.9)$$

Hence, we have  $\Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x) \geq \Theta_i(x)$  and the optimal value of problem (3.8) is a lower bound on the optimal value of (1.1). Before discussing the construction of the tuples  $\eta$ , we shortly discuss an algorithm to solve problem (3.8).

#### Solving the master problem

Note that problem (3.8) can be written as a disjunctive mixed-integer linear problem:

$$\min \left\{ \langle c, x \rangle + \sum_{i \in I} p_i \theta_i : x \in X, \theta_i \geq \eta_1(x) \vee \dots \vee \theta_i \geq \eta_k(x), \eta \in C_i, i \in I \right\}. \quad (3.10)$$

This problem can be solved by a branch-and-bound algorithm [Ntamo and Sen, 2008a]. To this end, assume that for each tuple  $\eta \in C_i$  an affine linear function  $\bar{\eta}(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}$  is known which underestimates each  $\eta_j(\cdot)$ ,  $j \in [k]$ , on  $X$ , i.e.,  $\eta_j(x) \geq \bar{\eta}(x)$  for  $j \in [k]$  and

### 3. Decomposition Algorithms for Stochastic Programming Problems

$x \in X$ .  $\bar{\eta}(\cdot)$  allows to derive a linear relaxation of problem (3.10):

$$\min \left\{ \langle c, x \rangle + \sum_{i \in I} p_i \theta_i : x \in \bar{X}, \theta_i \geq \bar{\eta}(x), \eta \in C_i, i \in I \right\}. \quad (3.11)$$

Let  $(\hat{x}, \hat{\theta})$  be a solution of (3.11). If  $\hat{x}$  is feasible for (3.10), then an optimal solution for (3.10) has been found. Otherwise,  $\hat{x}$  either violates an integrality restriction on a variable  $x_j$ ,  $j \in [m_0]$ , or a disjunction  $\theta_i \geq \min\{\eta_1(x), \dots, \eta_k(x)\}$  for some tuple  $\eta \in C_i$  (with  $k > 1$ ) and some scenario  $i$ . In the former case, that is,  $\hat{x}_j \notin \mathbb{Z}$ , two subproblems of (3.11) are created with additional constraints  $x_j \leq \lfloor \hat{x}_j \rfloor$  and  $x_j \geq \lceil \hat{x}_j \rceil$ , respectively. In the latter case, the tuple  $\eta$  is partitioned into two tuples  $\eta' = (\eta_1(\cdot), \dots, \eta_{k'}(\cdot))$  and  $\eta'' = (\eta_{k'+1}(\cdot), \dots, \eta_k(\cdot))$ ,  $1 \leq k' < k$ , corresponding linear underestimators  $\bar{\eta}'(\cdot)$  and  $\bar{\eta}''(\cdot)$  are computed (where  $\bar{\eta}' = \eta_1$  if  $k' = 1$  and  $\bar{\eta}'' = \eta_k$  if  $k' = k - 1$ ), and two subproblems where the tuple  $\eta \in C_i$  is replaced by  $\eta'$  and  $\eta''$ , respectively, are constructed. Next, the same method is applied to each subproblem recursively. The first feasible solution for problem (3.10) is stored as “incumbent solution”. In the following, new feasible solutions replace the incumbent solution if they have a better objective value. If a subproblem is infeasible or the value of its linear relaxation exceeds the current incumbent solution, then it can be discarded from the list of open subproblems. Since in each subproblem the number of feasible discrete values for a variable  $x_j$  or the length of a tuple  $\eta \in C_i$  is reduced with respect to the ascending problem, the algorithm can generate only a finite number of subproblems and thus terminates with a solution of (3.10).

#### Convexification of disjunctive cuts

A linear function  $\bar{\eta}(\cdot)$  in (3.11) that underestimates  $\min\{\eta_1(\cdot), \dots, \eta_k(\cdot)\}$  can be constructed via disjunctive programming [Balas, 1998, Sen and Hige, 2005]: For a fixed scenario index  $i \in I$  and a tuple  $\eta \in C_i$ , an inequality  $\theta \geq \bar{\eta}(x)$  is valid for the feasible set of (3.10), if it is valid for  $\bigcup_{j=1}^k \{(x, \theta) \in \mathbb{R}^{m+1} : x \in \bar{X}, \theta \geq \eta_j(x)\}$ . That is, we require

$$\bar{\eta}(x) \leq \eta_j(x) \quad \text{for all } x \in \bar{X}, j \in [k]. \quad (3.12)$$

We write  $\bar{\eta}(x) = \bar{\eta}_0 + \langle \bar{\eta}_x, x \rangle$  and  $\eta_j(x) = \eta_{j,0} + \langle \eta_{j,x}, x \rangle$  for some  $\bar{\eta}_0, \eta_{j,0} \in \mathbb{R}$  and  $\bar{\eta}_x, \eta_{j,x} \in \mathbb{R}^m$ ,  $j \in [k]$ . Then (3.12) is equivalent to

$$\begin{aligned} \bar{\eta}_0 - \eta_{j,0} &\leq \min\{\langle \eta_{j,x} - \bar{\eta}_x, x \rangle : x \in \mathbb{R}^m, Ax \leq b\} \\ &= \max\{\langle \lambda_j, b \rangle : \lambda_j \in \mathbb{R}^{r_0}, A^\top \lambda_j = \eta_{j,x} - \bar{\eta}_x\}. \end{aligned}$$

Therefore, choosing  $\lambda_j \in \mathbb{R}^{r_0}$  and  $\bar{\eta}_x \in \mathbb{R}^m$  such that  $A^\top \lambda_j + \bar{\eta}_x = \eta_{j,x}$ , and setting  $\bar{\eta}_0 := \eta_{j,0} + \min_{j \in [k]} \langle \lambda_j, b \rangle$  yields a function  $\bar{\eta}(x)$  that satisfies (3.12).

Sen and Hige [2005] note, that given an extreme point  $\hat{x}$  of  $\bar{X}$ , the linear underestimator  $\bar{\eta}(\cdot)$  can be chosen such that  $\bar{\eta}(\hat{x}) = \min\{\eta_1(\hat{x}), \dots, \eta_k(\hat{x})\}$ . Thus, if only extreme points of  $\bar{X}$  are feasible for (1.1), then it is not necessary to branch on disjunctions  $\eta$  to solve (3.10). This is the case, e.g., if all first stage variables are restricted to be binary.

**Approximation of  $\Phi(u, t)$  by linear optimality cuts**

The simplest way to construct a tuple  $\eta$  with property (3.9) is to derive a supporting hyperplane for the linear relaxation of  $\Phi(u, t)$ , which we denote by

$$\bar{\Phi}(u, t) := \min\{\langle u_1, y_1 \rangle + \langle u_2, y_2 \rangle : y_1 \in \mathbb{R}^{m_1}, y_2 \in \mathbb{R}^{m_2}, W_1 y_1 + W_2 y_2 \leq t\}. \quad (3.13)$$

As discussed in Section 3.1.1,  $\bar{\Phi}(u, t)$  is piecewise linear and convex in  $t$ . Thus, if, for fixed  $(\hat{u}, \hat{t}) \in \mathcal{U} \times \mathcal{T}$ ,  $\hat{\pi}$  is a dual solution of (3.13), we obtain the inequality  $\bar{\Phi}(\hat{u}, t) \geq \bar{\Phi}(\hat{u}, \hat{t}) + \langle \hat{\pi}, t - \hat{t} \rangle = \langle \hat{\pi}, t \rangle$ ,  $t \in \mathcal{T}$ . Letting  $\hat{u} = q(\xi^i)$  and  $\hat{t} = h(\xi^i) - T(\xi^i)\hat{x}$  for a fixed scenario  $\xi^i$  and first stage decision  $\hat{x} \in \bar{X}$ , we obtain

$$\bar{\Phi}(q(\xi^i), h(\xi^i) - T(\xi^i)x) \geq \langle \hat{\pi}, h(\xi^i) - T(\xi^i)x \rangle =: \eta_1(x). \quad (3.14)$$

Since  $\Phi(u, t) \geq \bar{\Phi}(u, t)$ , (3.14) yields the *optimality cut*  $\eta := (\eta_1(\cdot))$  (i.e.,  $k = 1$ ). Due to the polyhedrality of  $\bar{\Phi}(u, t)$ , a finite number of such cuts for each scenario is sufficient to obtain an exact representation of  $\bar{\Phi}(u, t)$  in the master problem (3.10).

**Approximation of  $\Phi(u, t)$  by lift-and-project**

However, in order to capture the nonconvexity of the original value function  $\Phi(u, t)$ , *nonconvex optimality cuts* are necessary, i.e., tuples  $\eta$  of length  $k > 1$ .

For the case that the discrete variables in the second stage are all of binary type, the following method is proposed in Sen and Higle [2005]: Let  $\bar{x} \in X$  be a feasible solution of the master problem (3.8), let  $(\bar{y}_1^i, \bar{y}_2^i)$  be a solution of the relaxed second stage problem (3.13) for  $u = q(\xi^i)$  and  $t = h(\xi^i) - T(\xi^i)\bar{x}$ ,  $i \in I$ . If  $\bar{y}_2^i \in \mathbb{Z}^{m_2}$  for all scenarios  $i \in I$ , then a linear optimality cut (3.14) is derived. Otherwise, let  $j \in [m_2]$  be an index such that  $0 < \bar{y}_{2,j}^i < 1$ . We now seek for inequalities  $\langle \pi_1^i, y_1 \rangle + \langle \pi_2^i, y_2 \rangle \geq \pi_0^i(x)$ ,  $i \in I$ , which are valid for (1.5) for all  $x \in \bar{X}$ , but cut off the solution  $\bar{y}^i$  from (3.13) for at least one scenario  $i$  with fractional  $\bar{y}_{2,j}^i$ . That is, we search for inequalities that are valid for the disjunctive sets

$$\left\{ y \in \mathbb{R}^{m_1+m_2} : \begin{array}{l} W_1 y_1 + W_2 y_2 \leq t, \\ y_{2,j} \leq 0 \end{array} \right\} \cup \left\{ y \in \mathbb{R}^{m_1+m_2} : \begin{array}{l} W_1 y_1 + W_2 y_2 \leq t, \\ -y_{2,j} \leq -1 \end{array} \right\}, \quad (3.15)$$

where  $t = h(\xi^i) - T(\xi^i)\bar{x}$ ,  $i \in I$ . Observe, that points with fractional  $y_{2,j}$  are not contained in (3.15). With an argumentation similar to the derivation of  $\bar{\eta}(\cdot)$  before, it follows that, for fixed  $x$ , valid inequalities for (3.15) are described by the system

$$W_1^\top \lambda_{1,1}^i = \pi_1^i, \quad W_1^\top \lambda_{2,1}^i = \pi_1^i, \quad (3.16a)$$

$$W_2^\top \lambda_{1,1}^i + e_j \lambda_{1,2}^i = \pi_2^i, \quad W_2^\top \lambda_{2,1}^i - e_j \lambda_{2,2}^i = \pi_2^i, \quad (3.16b)$$

$$\langle h(\xi^i) - T(\xi^i)x, \lambda_{1,1}^i \rangle \geq \pi_0^i(x), \quad \langle h(\xi^i) - T(\xi^i)x, \lambda_{2,1}^i - \lambda_{2,2}^i \rangle \geq \pi_0^i(x), \quad (3.16c)$$

$$\lambda_{1,1}^i \in \mathbb{R}^r, \lambda_{1,2}^i \in \mathbb{R}_-, \quad \lambda_{2,1}^i \in \mathbb{R}^r, \lambda_{2,2}^i \in \mathbb{R}_-, \quad (3.16d)$$

where  $e_j \in \mathbb{R}^{m_2}$  is the  $j$ -th unit vector. Observe further, that the coefficients in (3.16a)

### 3. Decomposition Algorithms for Stochastic Programming Problems

and (3.16b) (i.e.,  $W_1, W_2, e_j$ ) are scenario independent. Thus, it is possible to use common cut coefficients  $(\pi_1, \pi_2) \equiv (\pi_1^i, \pi_2^i)$  for all scenarios, thereby reducing the computational effort to the solution of a single linear program [Sen and Hige, 2005]:

$$\max \left\{ \sum_{i \in I} p_i (\pi_0^i(\bar{x}) - \langle \pi_1, \bar{y}_1^i \rangle - \langle \pi_2, \bar{y}_2^i \rangle) : \begin{array}{l} \lambda_{1,1}, \lambda_{2,1} \in \mathbb{R}^r, \lambda_{1,2}, \lambda_{2,2} \in \mathbb{R}_-, \\ \pi_1 \in \mathbb{R}^{m_1}, \pi_2 \in \mathbb{R}^{m_2}, \pi_0^i(\bar{x}) \in \mathbb{R}, \\ W_1^\top \lambda_{1,1} = \pi_1, W_2^\top \lambda_{1,1} + e_j \lambda_{1,2} = \pi_2, \\ W_1^\top \lambda_{2,1} = \pi_1, W_2^\top \lambda_{2,1} - e_j \lambda_{2,2} = \pi_2, \\ \langle h(\xi^i) - T(\xi^i)\bar{x}, \lambda_{1,1} \rangle \geq \pi_0^i(\bar{x}), \\ \langle h(\xi^i) - T(\xi^i)\bar{x}, \lambda_{2,1} \rangle - \lambda_{2,2} \geq \pi_0^i(\bar{x}), \\ \|\pi_1\|_\infty \leq 1, \|\pi_2\|_\infty \leq 1, |\pi_0^i(\bar{x})| \leq 1, \\ i \in I \end{array} \right\} \quad (3.17)$$

The objective function of this simple recourse problem maximizes the expected violation of the computed cuts by  $(\bar{y}_1^i, \bar{y}_2^i)$ . The functions  $\pi_0^i(\cdot)$ ,  $i \in I$ , with  $\langle \pi_1, y_1 \rangle + \langle \pi_2, y_2 \rangle \geq \pi_0^i(x)$  for all  $x \in \bar{X}$  are derived from a solution of (3.17) as

$$\pi_0^i(x) := \min\{\langle h(\xi^i) - T(\xi^i)x, \lambda_{1,1} \rangle, \langle h(\xi^i) - T(\xi^i)x, \lambda_{2,1} \rangle - \lambda_{2,2}\}.$$

Adding these new cuts to (3.13) for  $u = q(\xi^i)$  and  $t = h(\xi^i) - T(\xi^i)\bar{x}$ ,  $i \in I$ , yields the updated second stage linear relaxations

$$\min \left\{ \langle q_1(\xi^i), y_1 \rangle + \langle q_2(\xi^i), y_2 \rangle : \begin{array}{l} W_1 y_1 + W_2 y_2 \leq h(\xi^i) - T(\xi^i)\bar{x} \\ -\langle \pi_1, y_1 \rangle - \langle \pi_2, y_2 \rangle \leq -\pi_0^i(\bar{x}) \\ y_1 \in \mathbb{R}^{m_1}, y_2 \in \mathbb{R}^{m_2} \end{array} \right\}. \quad (3.18)$$

A dual solution  $(\mu, \mu_0)$  of (3.18) can then be used to derive the inequality

$$\Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x) \geq \langle \mu, h(\xi^i) - T(\xi^i)x \rangle - \mu_0 \pi_0^i(x).$$

However,  $\pi_0^i(x)$  yields a nonconvex optimality cut  $\eta := (\eta_1(\cdot), \eta_2(\cdot))$ , where

$$\begin{aligned} \eta_1(x) &:= \langle \mu - \mu_0 \lambda_{1,1}, h(\xi^i) - T(\xi^i)x \rangle, \\ \eta_2(x) &:= \langle \mu - \mu_0 \lambda_{2,1}, h(\xi^i) - T(\xi^i)x \rangle + \mu_0 \lambda_{2,2}. \end{aligned}$$

In a next iteration, when the second stage problems are revisited with a different first stage solution  $\bar{x}$ , the updated relaxation (3.18) takes the place of the original relaxation (3.13). Since the functions  $\pi_0^i(\cdot)$  are known, the right-hand side of the added cut in (3.18) is updated when  $\bar{x}$  changes.

#### Approximation of $\Phi(u, t)$ by branch-and-bound

For the general case where the discrete second stage variables can also be of integer type, the second stage problem (1.5) can be solved by a (partial) branch-and-bound algorithm and a (nonlinear) optimality cut  $\eta$  can be derived from the dual solutions of the linear

### 3.2. Temporal Decomposition for Two-Stage Stochastic Mixed-Integer Linear Programs

programs in each leaf of the branch-and-bound tree [Sen and Sherali, 2006]: Let  $\bar{x} \in X$  be again a feasible point to problem (3.8) and fix a scenario  $i \in I$ . Assume that (1.5) with  $\bar{u} = q(\xi^i)$  and  $\bar{t} = h(\xi^i) - T(\xi^i)\bar{x}$  is (partially) solved by a branch-and-bound algorithm. Denote by  $\mathcal{Q}$  the set of terminal nodes of the generated branch-and-bound tree. For any node  $q \in \mathcal{Q}$  let  $y_{2,l}^q$  and  $y_{2,u}^q$  denote the vectors that define lower and upper bounds on the  $y_2$  variables in this node. Then the LP relaxation of (1.5) for node  $q \in \mathcal{Q}$  is given as

$$\min \left\{ \langle \bar{u}_1, y_1 \rangle + \langle \bar{u}_2, y_2 \rangle : y \in \mathbb{R}^{m_1+m_2}, W_1 y_1 + W_2 y_2 \leq \bar{t}, \begin{array}{l} y_2 \leq y_{2,u}^q, \\ -y_2 \leq -y_{2,l}^q \end{array} \right\}. \quad (3.19)$$

The dual problem to (3.19) is

$$\max \left\{ \langle \mu, \bar{t} \rangle + \langle \pi_u, y_{2,u}^q \rangle - \langle \pi_l, y_{2,l}^q \rangle : \begin{array}{l} \mu \in \mathbb{R}_-^r, \\ \pi_l, \pi_u \in \mathbb{R}_-^{m_2}, \end{array} \begin{array}{l} W_1^\top \mu = \bar{u}_1 \\ W_2^\top \mu + \pi_u - \pi_l = \bar{u}_2 \end{array} \right\}, \quad (3.20)$$

where we assume that a dual variable  $\pi_{l,j}, \pi_{u,j}$  is fixed to 0 if the corresponding bound  $y_{2,l,j}^q, y_{2,u,j}^q$  is  $-\infty$  or  $+\infty$ , respectively,  $j \in [m_2]$ .

We assume, that subproblems are pruned if infeasible or if their lower bound exceeds a known upper bound. Thus, all terminal nodes are associated with a feasible relaxation. Based on a dual solution  $(\mu^q, \pi_l^q, \pi_u^q)$  of (3.20), a supporting hyperplane of each nodes value function can be derived, c.f. (3.14). Since the branch-and-bound tree represents a partition of the feasible set of (1.5), it allows to state a disjunctive description of the function  $t \mapsto \Phi(\bar{u}, t)$  by combining the value function approximations in all nodes  $q \in \mathcal{Q}$ :

$$\Phi(\bar{u}, t) \geq \langle \mu^q, t \rangle + \langle \pi_u^q, y_{2,u}^q \rangle - \langle \pi_l^q, y_{2,l}^q \rangle \quad \text{for at least one } q \in \mathcal{Q}.$$

This result directly translates into a nonlinear optimality cut  $\eta := (\eta_1(\cdot), \dots, \eta_{|\mathcal{Q}|}(\cdot))$  by letting  $\eta_q(x) := \langle \mu^q, h(\xi^i) - T(\xi^i)x \rangle + \langle \pi_u^q, y_{2,u}^q \rangle - \langle \pi_l^q, y_{2,l}^q \rangle$ ,  $q \in \mathcal{Q}$ .

#### Full Algorithm

An adaptation of the Benders Decomposition scheme that uses the previously discussed approximation strategies is stated in Algorithm 3.3. At the beginning, no information about the value function  $\Phi(u, t)$  is available, so the sets of cuts  $C_i, i \in I$ , in (3.10) are empty. Thus, (3.10) should be solved either with the variables  $\theta_i$  removed or bounded from below by a known lower bound on  $\Phi(u, t)$ . In the first iterations, when almost no information about  $\Phi(u, t)$  is available, it is unnecessary to solve master problem (3.10) and second stage problems (1.5) to optimality. Instead, ignoring the integrality conditions at first and constructing a representation of the value function  $\bar{\Phi}(u, t)$  by a usual Benders Decomposition is more efficient. Later, partial solves of (3.10) and the introduction of nonlinear optimality cuts  $\eta$  into (3.10) based on lift-and-project or partial branch-and-bound searches should be performed to capture the nonconvexity of  $\Phi(u, t)$  in the master problem. Finally, to ensure convergence, first and second stage problems need to be solved to optimality, see also Sen and Hgle [2005] and Ntaimo and Sen [2008a].

### 3. Decomposition Algorithms for Stochastic Programming Problems

---

**Algorithm 3.3:** Solving a two-stage stochastic mixed-integer linear program ((1.1) with (1.3)–(1.5)) with relatively complete recourse by an adaption of the Benders Decomposition algorithm via convexifications of the recourse function.

---

```

repeat
  solve the master problem (3.10) by branch-and-bound;
  if (3.10) is infeasible then STOP: (1.1) is infeasible;
  let  $(\bar{x}, \bar{\theta})$  be a solution of (3.10);
  optimal  $\leftarrow$  TRUE;
  foreach scenario  $i \in I$  do
    solve second stage problem (1.5);
    let  $\phi_i := \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)\bar{x})$  be the optimal value of (1.5);
    if  $\phi_i > \bar{\theta}_i$  then
      derive an optimality cut  $\eta$  of the value function
       $x \mapsto \Phi(q(\xi^i), h(\xi^i) - T(\xi^i)x)$  either via linearization of  $\bar{\Phi}(u, t)$  (see (3.14)),
      via lift-and-project (see (3.17)), or from a (partial) branch-and-bound
      search;
      add  $\eta$  to  $C_i$  in (3.10);
      optimal  $\leftarrow$  FALSE;
    end
  end
until optimal;
output :  $\bar{x}$  is an optimal solution to (1.1);

```

---

#### Extension to Multistage Stochastic Mixed-Integer Linear Problems

Algorithm 3.3 extends Benders Decomposition for two-stage stochastic linear programs to the mixed-integer linear case. A further extension to the multistage case seems possible. While in the two-stage case a nonconvex value function is present only in the first stage, in the multistage setting we are faced with such a function in each node of the scenario tree other than the leaves. That is, the master problems in each node before the last stage are of the form (3.10), where the sets  $X$  depend on the decisions from the previous state. Approximating the value function of an optimization problem in the form of (3.10) requires to take nonlinear optimality cuts, which approximate the value functions of successor nodes, into account. However, we have discussed how the master problem (3.10) can be solved by branch-and-bound and how an optimality cut can be derived from a (partial) branch-and-bound search, so that the same techniques as developed for the two-stage case is applicable in an extension of Algorithm 3.3 to the multistage case.

Nevertheless, the efficiency of such an approach might suffer under the large number of disjunctions that are induced from optimality cuts on late stages into the master problems on early stages. That is, while in the two-stage case the disjunctions in (3.10) are caused only by integrality constraints on the second stage, in the multistage setting we have to deal with disjunctions that are induced by disjunctions on succeeding stages. Therefore,

solving a fairly large mixed-integer multistage stochastic program to optimality with this approach seems questionable. However, an interesting application are multistage problem that can be solved efficiently by a temporal decomposition only, e.g., stochastic programs with recombining scenario trees, c.f. Chapter 4.

### 3.3. Scenario Decomposition

In the following, we discuss Lagrangian Decomposition for two-stage and multistage (mixed-integer) linear programs, where the problem is not decomposed along the time axis, but along the scenarios. We assume a finite distribution  $\{\xi^i\}_{i \in I}$ .<sup>1</sup>

#### 3.3.1. Two-Stage Problems

Consider the following reformulation of (1.1) with (1.3)–(1.5) where the first stage variable  $x$  is replaced by one variable  $x^i$  for each scenario  $i \in I$  and an explicit *nonanticipativity constraint* is added (we assume  $1 \in I$  here):

$$\min \left\{ \begin{array}{l} \sum_{i \in I} p_i (\langle c, x^i \rangle + \langle q(\xi^i), y^i \rangle) : \\ x^i \in X, y^i \in \mathbb{R}^{m_1} \times \mathbb{Z}^{m_2}, i \in I, \\ T(\xi^i)x^i + Wy^i \leq h(\xi^i), i \in I, \\ x^i = x^1, i \in I. \end{array} \right\} \quad (3.21)$$

Problem (3.21) decomposes into scenariowise subproblems by relaxing the coupling constraints  $x^i = x^1, i \in I$  [Carøe and Schultz, 1999]. The violation of the relaxed constraints is then added as a penalty term into the objective function. That is, each subproblem has the form

$$D_i(\lambda) := \min \left\{ L_i(x, y; \lambda) : x \in X, y \in \mathbb{R}^{m_1} \times \mathbb{Z}^{m_2}, T(\xi^i)x + Wy \leq h(\xi^i) \right\}, \quad (3.22)$$

where  $\lambda := (\lambda^i)_{i \in I} \in \mathbb{R}^{m|I|}$  is the *Lagrange multiplier* and

$$L_i(x, y; \lambda) := p_i (\langle c, x \rangle + \langle q(\xi^i), y \rangle + \langle \lambda^i, x - x^1 \rangle) \quad (i \in I).$$

For every choice of  $\lambda$ , a lower bound on (3.21) is obtained by computing

$$D(\lambda) := \sum_{i \in I} D_i(\lambda), \quad (3.23)$$

which requires solving the deterministic problem (3.22) for each scenario. To find the best possible lower bound, one now searches for an optimal solution to the *dual problem*

$$\max \{ D(\lambda) : \lambda \in \mathbb{R}^{m|I|} \}. \quad (3.24)$$

---

<sup>1</sup>For multistage stochastic linear programs with a continuous distribution or a discrete distribution with large  $|I|$ , Higle, Rayco, and Sen [2010] proposed a *Stochastic Scenario Decomposition* algorithm that works in a similar form as the Stochastic Decomposition Algorithm mentioned at the end of Section 3.1.1, but applies to a dual formulation of (1.9).

### 3. Decomposition Algorithms for Stochastic Programming Problems

The function  $D(\lambda)$  is a piecewise linear concave function for which subgradients can be computed from a solution of (3.22). A preferred method to solve the nonsmooth convex optimization problem (3.24) is to use a bundle of subgradients of  $D(\lambda)$  [Kiwiel, 1990].

As shown by Proposition 2 in Carøe and Schultz [1999], the problem (3.24) is equivalent to the primal problem

$$\min \left\{ \langle c, x^1 \rangle + \sum_{i \in I} p_i \langle q(\xi^i), y^i \rangle : (x^1, y^i) \in \text{conv} \left\{ (x, y) : \begin{array}{l} x \in X, y \in \mathbb{R}^{m_1} \times \mathbb{Z}^{m_2}, \\ T(\xi^i)x + Wy \leq h(\xi^i) \end{array} \right\}, \right. \\ \left. i \in I \right\} \quad (3.25)$$

For continuous linear problems ( $m_0 = 0, m_2 = 0$ ), the convex hulls in (3.25) do not relax the original sets of constraints ( $x \in X, T(\xi^i)x + Wy \leq h(\xi^i)$ ), so that, due to strong duality [Nemhauser and Wolsey, 1988, Theorem 6.2], the primal solutions  $(x^i, y^i), i \in I$ , of (3.22), associated with a solution of (3.24), satisfy the nonanticipativity constraints and thus yield an optimal solution to the original problem (1.1)–(1.5). In the mixed-integer linear case, however, some nonanticipativity constraints may be violated in the set of subproblem solutions. To still find a feasible solution, usually heuristics are employed that, e.g., select for  $x$  an average or a frequently occurring value among the  $x^i$  and then possibly resolve each second stage problem to ensure feasibility.

To find an optimal solution to (3.21), a branch-and-bound algorithm can be employed. Here, nonanticipativity constraints are insured by branching on the first stage variables. Since the additional bound constraints on  $x^i$  become part of the constraints in (3.22), the lower bound (3.24) improves by a branching operation. An implementation of this algorithm is available in the `ddsip` software package [Märkert and Gollmer, 2008].

An alternative to solving the dual problem (3.24) by a bundle method is proposed in Lulli and Sen [2004]. Here, the equivalent problem (3.25) is solved by a column generation approach, which constructs an inner approximation of the convex hulls in (3.25). Feasible solutions for the original problem are obtained by an application of branch-and-bound.

For problems where all first stage variables are restricted to be binary, Alonso-Ayuso, Escudero, and Ortuño [2003] propose to relax both nonanticipativity and integrality constraints. Thereby, each scenario is associated with a branch-and-bound tree that enumerates the integer feasible solutions to the scenario’s subproblem (i.e., the feasible set of (3.22)). Since each branch-and-bound fixes first stage variables to be either 0 or 1, a coordinated search across all  $n$  branch-and-bound trees allows to select feasible solutions from each subproblem that satisfy the nonanticipativity constraints. If also continuous variables are present in the first stage, Escudero, Garín, Merino, and Pérez [2007] propose to “cross over” to a Benders Decomposition whenever the coordinated branch-and-bound search yields solutions which binary first stage variables satisfy the nonanticipativity constraints and second stage integer variables are fixed.

#### 3.3.2. Multistage Problems

Consider the multistage stochastic linear program (1.9). In the mixed-integer linear case, the requirement that  $X_t, t \in [T]$ , is polyhedral, is replaced by requiring  $X_t$  to be of the

### 3.3. Scenario Decomposition

form  $\bar{X}_t \cap (\mathbb{Z}^{m'_t} \times \mathbb{R}^{m_t - m'_t})$ , where  $\bar{X}_t \subseteq \mathbb{R}^{m_t}$  is polyhedral and  $0 \leq m'_t \leq m_t$ .

The scenario decomposition algorithm for (1.1) with (1.3)–(1.5) is readily extended to the multistage case by introducing scenario-independent variables  $x^i \in \mathbb{R}^{m_1 + \dots + m_T}$  and replacing the nonanticipativity constraint  $x_t \in \mathcal{F}_t$  by an explicit formulation (as in (3.4)). Hence, with a suitable matrix  $H = (H^i)_{i \in I} \in \mathbb{R}^{\ell \times |I| \sum_{t=1}^T m_t}$ , where  $\ell$  is the number of equations required to write the nonanticipativity constraints, (1.9) is reformulated as

$$\min \left\{ \sum_{i \in I} p_i \sum_{t=1}^T \langle b_t(\xi_t^i), x_t \rangle : \begin{array}{l} x_t^i \in X_t, \quad i \in I, t \in [T], \\ A_{t,0}(\xi_t^i)x_t^i + A_{t,1}(\xi_t^i)x_{t-1}^i = h_t(\xi_t^i), \quad i \in I, t \in [2:T], \\ \sum_{i \in I} H^i x^i = 0 \end{array} \right\},$$

This problem decomposes into scenariowise subproblems by relaxing the coupling constraint  $\sum_{i \in I} H^i x^i = 0$  [Carøe and Schultz, 1999]:

$$D_i(\lambda) := \min \left\{ L_i(x^i; \lambda) : \begin{array}{l} x_t^i \in X_t, \quad t = 1, \dots, T, \\ A_{t,0}(\xi_t^i)x_t^i + A_{t,1}(\xi_t^i)x_{t-1}^i = h_t(\xi_t^i), \quad t = 2, \dots, T \end{array} \right\},$$

where  $\lambda \in \mathbb{R}^\ell$  and

$$L_i(x_i; \lambda) := p_i \sum_{t=1}^T \langle b_t(\xi_t^i), x_t \rangle + \langle \lambda, H^i x^i \rangle.$$

The dual problem is then given as in the two-stage case as maximization of  $D(\lambda) := \sum_{i \in I} D_i(\lambda)$ , c.f. (3.23)–(3.24). Its optimal value is equivalent to the multistage analog of (3.25), so that it yields the optimal value of the original problem (1.9) in the continuous case ( $m'_t = 0$ ,  $t \in [T]$ ). In the mixed-integer case, branch-and-bound may be used to find an optimal solution, where branching is now performed not only on the first-stage variables, but on all variables that are involved in the nonanticipativity constraints, i.e., potentially on all variables from the first  $T - 1$  stages.

An implementation of scenario decomposition for multistage stochastic mixed-integer linear programs is discussed in Heinze [2008] and Heinze and Schultz [2008].



## 4. Decomposition with Recombining Scenario Trees

As discussed in Section 1.3.2, in multistage stochastic programming problems the underlying stochastic processes are usually represented by scenario trees. Unfortunately, even under a moderate branching structure, the number of scenarios can grow exponentially with the number of time stages. Consequently, many problems of practical interest are approximated by models that include only few time stages or a small number of scenarios. Under certain circumstances, a further approach to handle uncertainty over long time horizons is to use *recombining scenario trees*. An illustrative example is the binomial model of stock price behavior in Cox et al. [1979], whereby the number of nodes under  $T$  time stages reduces from  $2^T - 1$  to  $T(T + 1)/2$  through recombination of scenarios. However, some information about the history of a node in the tree will be lost. This has no consequences if the represented stochastic process has the Markov property<sup>1</sup> and the optimization problem contains no time-coupling constraints. But whenever one of these properties is not fulfilled, it is difficult to formulate a dynamic programming equation on a recombining scenario tree. Practical problems include often both non-Markovian stochastic input and time-coupling constraints. For such problems, recombining scenario trees seem to be inappropriate at first sight.

However, motivated by the numerical possibilities, we develop an approach that overcomes these difficulties and allows to solve large-scale and long-term multistage stochastic linear programs with recombining scenario trees. The basic idea is related to the cut sharing principle developed by Infanger and Morton [1996], cf. Section 3.1.2. In case of a stochastic process with interstage independence random variables, cut sharing uses the observation that the cost-to-go functions  $Q_t(\cdot, \xi_{[t]})$  (cf. (1.10)) depend only on  $\xi_t$ , but not on the history  $\xi_{[t-1]}$ ,  $t \in [2 : T]$ , which reduces the number of differing expected cost-to-go functions from the number of scenario tree nodes to the number of time stages. For a Nested Benders Decomposition, the reduction in the number of expected cost-to-go functions allows to share cuts from one evaluation of a cost-to-go function among all master problems in the same time stage. However, as discussed in Section 3.1.2, the number of master problem evaluations may not be reduced by this approach.

In this chapter, we show that the cut sharing principle can be applied – in a limited form – to stochastic processes that are non-Markovian, but where certain scenario tree nodes at one stage share the same subtree. Further, to overcome the problem of an exponential number of master problem evaluations, we introduce a dynamic aggregation

---

<sup>1</sup>A process  $\{\xi_t\}_{t \in [T]}$  has the Markov property, if for all  $s, t \in [T]$  with  $s < t$  and for all  $A \in \mathcal{B}(\mathbb{R}^{s_t})$  it holds that  $P[\xi_t \in A | \sigma(\xi_{[s]})] = P[\xi_t \in A | \sigma(\xi_s)]$ , i.e., the conditional probability distribution of future states of the process depends only on the present state, but not the past.

## 4. Decomposition with Recombining Scenario Trees

scheme for the decision variables  $x_t$ .

Section 4.1 defines the problem setting and the concept of recombining scenarios and introduces some notation. The solution algorithm is detailed in Sections 4.2–4.4, as well as convergence results and stopping criteria. In Section 4.5 we sketch a method for constructing recombining scenario trees. A small example and some numerical results that demonstrate the potentials of the method are presented in Section 4.6.

Sections 4.1–4.2 and 4.5–4.6 are based on material that has previously been published in Küchler and Vigerske [2007, 2009], Epe, Küchler, Römisch, Vigerske, Wagner, Weber, and Woll [2007, 2009b], and Küchler [2009]. The development of upper bounds in Section 4.3 and its integration with lower bounds in Section 4.4 is new, only a rudimentary version has been published before. The out-of-sample evaluation in Section 4.7 has previously been published in Küchler and Vigerske [2010] and Küchler [2009]. The numerical results in Sections 4.6.3 and 4.7.3 are newer and more extensive than previously published ones.

### 4.1. Problem Setting

We consider multistage stochastic linear programs of the form (1.9) with corresponding dynamic programming formulation (1.10). Further, we assume a discrete distribution  $\{\xi^i\}_{i \in I}$  with corresponding scenario tree, cf. Section 1.3.2.

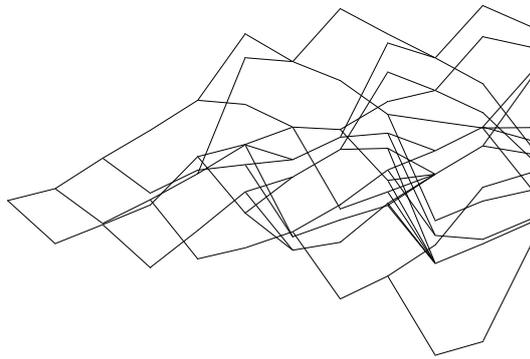
Although in many cases of practical interest the stochastic process  $\xi$  is not Markovian, it has only a comparable short-term memory and can be displayed without great loss of precision by a scenario tree, where scenarios with similar short-term history are *recombined* at certain time stages. We say that a discrete stochastic process  $\{\xi_t^i\}_{t \in [T]}$ ,  $i \in I$ , can be represented by a scenario tree within that the scenarios  $\xi^i$  and  $\xi^j$ ,  $i, j \in I$ , can be recombined at time  $t \in [T]$ , if both share the same associated subtree, i.e., the corresponding conditional distributions of  $\{\xi_\tau\}_{\tau \in [t+1:T]}$  coincide:

$$P \left[ \{\xi_\tau\}_{\tau \in [t+1:T]} \in A \mid \xi_t = \xi_t^i \right] = P \left[ \{\xi_\tau\}_{\tau \in [t+1:T]} \in A \mid \xi_t = \xi_t^j \right] \quad (A \subseteq \mathbb{R}^{s_{t+1} + \dots + s_T}). \quad (4.1)$$

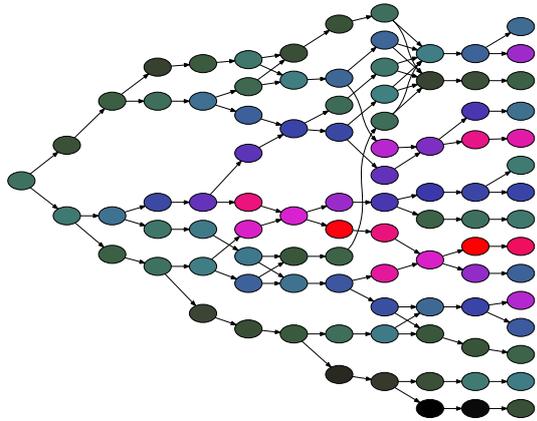
The scenario tree can then be displayed as a recombining tree with much less nodes than a non-recombining tree. Further, repeated recombination may prevent the number of nodes to grow exponentially with the number of time stages, see also Figure 4.1.

However, in the presence of time-coupling constraints, the relevant past of both stochastic process and decision variables has to be available at each node. Thus, even if the stochastic process has a recombining nature, the scenario-dependent decisions  $x$  do not follow the same recombination scheme as  $\xi$ , in general. Therefore, solution methods based on dynamic programming cannot be applied on a recombining tree directly.

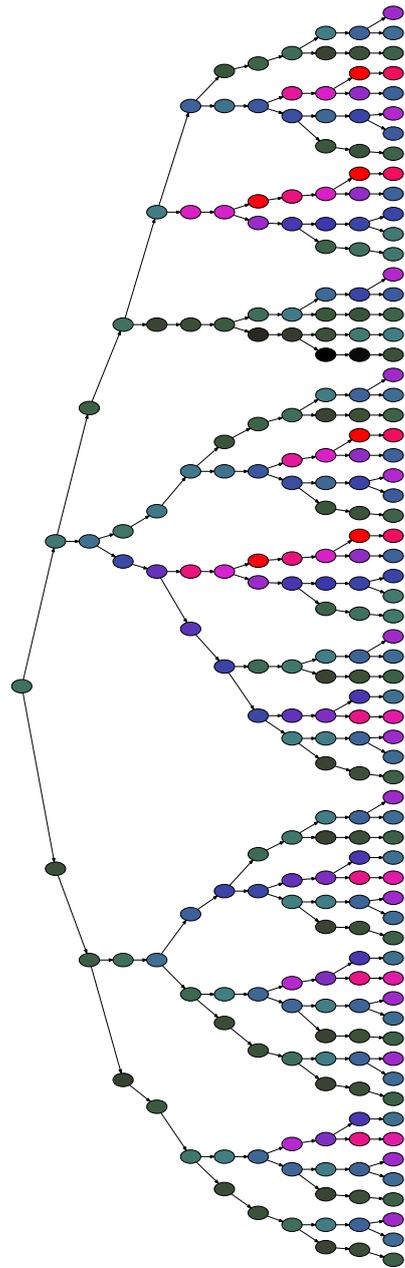
Nevertheless, (4.1) can be useful, since it entails equality of the expected cost-to-go functions  $x_t \mapsto \mathbb{E} \left[ Q_{t+1}(x_t, \xi_{[t+1]}) \mid \xi_t = \xi_t^i \right]$  and  $x_t \mapsto \mathbb{E} \left[ Q_{t+1}(x_t, \xi_{[t+1]}) \mid \xi_t = \xi_t^j \right]$ . The benefit becomes apparent within the Nested Benders Decomposition algorithm, where the cuts for approximating the cost-to-go functions at stage  $t+1$  can be shared among the master problems for nodes  $\xi_t^i$  and  $\xi_t^j$ . In this way, a considerable reduction of the numerical complexity of the problem can be achieved, see also Section 3.1.2.



(a) Stochastic process  $\xi$ .



(b) Structure of scenario tree  $\xi$  where coinciding subtrees are drawn only once, resulting in 93 nodes.



(c) Structure of equivalent non-recombining tree, i.e., coinciding subtrees are not collapsed into one, resulting in 232 nodes.

Figure 4.1.: Values and structure of a scenario tree with  $T = 12$  stages where condition (4.1) is satisfied for certain nodes at stages 6 and 9. Colors in Figures (b) and (c) correspond to value  $\xi_{[t]}^i$  of node.

#### 4. Decomposition with Recombining Scenario Trees

The relation (4.1) divides the set of nodes  $\{\xi_{[t]}^i\}_{i \in I}$  at time  $t \in [T]$  into a family of equivalence classes, which can be represented by a set of *representative nodes*  $\Lambda_t \subseteq \{\xi_{[t]}^i : i \in I\}$ , such that for every  $\xi_{[t]}^i, i \in I$ , there exists exactly one  $\xi_{[t]}^j \in \Lambda_t$  with (4.1) and such that  $\lambda_{[t-1]} \in \Lambda_{t-1}$  for all  $\lambda_{[t]} \in \Lambda_t$ . Further, we associate with  $\bar{\lambda}_t : \{\xi_{[t]}^i : i \in I\} \rightarrow \Lambda_t, t \in [T]$ , the mapping that assigns each node  $\xi_{[t]}^i, i \in I$ , to its representative node. Note, that at time stages where no two coinciding subtrees in the sense of (4.1) exist,  $\Lambda_t = \{\xi_{[t]}^i : i \in I\}$  and  $\bar{\lambda}_t$  is the identity.

Although we do not use trees that are recombining in the strict sense, scenario trees with coinciding subtrees in the sense of (4.1) are denoted as recombining, in the following.

While the number of different expected recourse functions at time  $t$  is reduced to  $r_t$ , the non-recombining nature of the decision process  $\{x_t\}_t$  may still cause an exponential growth (with increasing  $T$ ) of the number of master problem evaluations in a Nested Benders Decomposition. In the following, we present a way to deal with this difficulty.

### 4.2. Nested Benders Decomposition for Multistage Stochastic Linear Programs with Recombining Scenarios

An adaptation of the Nested Benders Decomposition algorithm 3.2 for scenario trees where the identity (4.1) holds for certain nodes is presented in Algorithm 4.1. The sets  $Z(\cdot)$  are used to store approximate solution points that demand for an evaluation of a master problem. In the original Algorithm 3.2, these sets always contained at most one point, and thus were not explicitly mentioned. The value  $L$  constitutes again a lower bound on the cost-to-go functions, c.f. Section 3.1.2. For the presentation, a simple sequencing protocol has been chosen in Algorithm 4.1, i.e., master problems are evaluated with increasing  $t$ , until either  $t = T$  is reached or some subproblem is infeasible. In both cases, the loop restarts with  $t = 1$ . As discussed in Section 3.1.2, different sequencing protocols, which, e.g., reevaluate master problems with decreasing  $t$  if new feasibility or optimality cuts have been passed back, could be applied. For simplicity, we assume that the sets  $X_t, t \in [T]$ , are bounded, so that (4.2)–(4.5) are always bounded from below.

The application of cut sharing allows to reduce the number of master problems to  $|\Lambda_t|$  per time stage, which already yields a reduction in computational complexity. However, every master problem corresponding to some  $\lambda_{[t]} \in \Lambda_t$  for some  $t$  has to be solved for all approximate solution points  $x_{t-1} \in Z(\lambda_{[t]})$  and each evaluation yields a set of new solution points to be added to  $Z(\lambda_{[t+1]}), \lambda_{[t+1]} \in \Lambda_{t+1}$ . This easily leads to an exponential growth of the sets  $Z(\lambda_{[t]})$  with increasing  $t$ . In the following, we will discuss, how to overcome this difficulty.

#### 4.2.1. Dynamic Recombination of Scenarios

Recall from Sections 3.1.1 and 3.1.2, that for two points  $\bar{x}_{t-1}, \bar{x}'_{t-1} \in \text{dom } \mathcal{Q}_t(\cdot, \lambda_{[t]})$ , it holds that

$$|\mathcal{Q}_t(\bar{x}_{t-1}, \lambda_{[t]}) - \mathcal{Q}_t(\bar{x}'_{t-1}, \lambda_{[t]})| \leq \max_{\pi} |\langle \pi, \bar{x}_{t-1} - \bar{x}'_{t-1} \rangle|, \quad (4.6)$$

---

**Algorithm 4.1:** Nested Benders Decomposition for recombining scenario trees
 

---

```

 $C_{\text{feas}}(\lambda_{[t]}) \leftarrow \emptyset, C_{\text{opt}}(\lambda_{[t]}) \leftarrow \{(L, 0)\}, \lambda_{[t]} \in \Lambda_t, t \in [2 : T]; t \leftarrow 1;$ 
loop
  foreach  $\lambda_{[t]} \in \Lambda_t$  do
    if  $t = 1$  then
      infeasible  $\leftarrow$  ProcessBendersMasterFirst();
      alloptimal  $\leftarrow$  TRUE;
       $Z(\lambda_{[\tau]}) \leftarrow \emptyset, \lambda_{[\tau]} \in \Lambda_\tau, \tau \in [2 : T];$ 
    else if  $t < T$  then
      (infeasible, optimal)  $\leftarrow$  ProcessBendersMasterMiddle( $t, \lambda_{[t]}$ );
      alloptimal  $\leftarrow$  alloptimal  $\wedge$  optimal;
    else
      (infeasible, optimal)  $\leftarrow$  ProcessBendersMasterLast( $\lambda_{[t]}$ );
      alloptimal  $\leftarrow$  alloptimal  $\wedge$  optimal;
    end
    if infeasible then
      if  $t = 1$  then STOP: (1.9) is infeasible;
      break;
    end
  end
  if infeasible then  $t \leftarrow 1;$ 
  else if  $t = T$  and alloptimal then STOP: (1.9) has been solved to optimality;
  else  $t \leftarrow (t \bmod T) + 1;$ 
end;

```

---

**Function** ProcessBendersMasterFirst processes first stage master problem within Nested Benders Decomposition for recombining scenario tree

---

**input** : cost-to-go function approximations  $\underline{Q}_2(\cdot, \lambda_{[2]}), \lambda_{[2]} \in \Lambda_2$ , via feasibility cuts  $C_{\text{feas}}(\lambda_{[2]})$  and optimality cuts  $C_{\text{opt}}(\lambda_{[2]})$   
 solve first stage master problem

$$\min \left\{ \langle b_1(\lambda_1), x_1 \rangle + \mathbb{E} \left[ \underline{Q}_2(x_1, \bar{\lambda}_2(\xi_{[2]})) \right] : x_1 \in X_1 \right\} \quad (4.2)$$

where

$$\underline{Q}_2(x_1, \lambda_{[2]}) := \begin{cases} \infty, & \text{if } \exists (\gamma, g) \in C_{\text{feas}}(\lambda_{[2]}) : \gamma + \langle g, x_1 \rangle > 0, \\ \max_{(\alpha, a) \in C_{\text{opt}}(\lambda_{[2]})} (\alpha + \langle a, x_1 \rangle), & \text{otherwise;} \end{cases}$$

**if** (4.2) is infeasible **then return** TRUE;

$Z(\lambda_{[2]}) \leftarrow \{x_1\}$  for all  $\lambda_{[2]} \in \Lambda_2;$

**return** FALSE;

---

#### 4. Decomposition with Recombining Scenario Trees

---

**Function** ProcessBendersMasterMiddle( $t, \lambda_{[t]}$ ) processes  $t$ -stage master problem of representative node  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T - 1]$ , within Nested Benders Decomposition for recombining scenario tree

---

**input**: approximate solutions  $x_{t-1} \in Z(\lambda_{[t]})$  that demand for an evaluation of the cost-to-go function corresponding to  $\lambda_{[t]}$

**input**:  $t + 1$ -stage cost-to-go function approximations  $\underline{Q}_{t+1}(\cdot, \lambda_{[t+1]})$ ,  $\lambda_{[t+1]} \in \Lambda_{t+1}$ , via feasibility cuts  $C_{\text{feas}}(\lambda_{[t+1]})$  and optimality cuts  $C_{\text{opt}}(\lambda_{[t+1]})$

infeasible  $\leftarrow$  FALSE;

optimal  $\leftarrow$  TRUE;

**foreach** *approximate solution*  $x_{t-1} \in Z(\lambda_{[t]})$  **do**

solve master problem

$$\min \left\{ \begin{array}{l} \langle b_t(\lambda_t), x_t \rangle + \mathbb{E} \left[ \underline{Q}_{t+1}(x_t, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] : \\ x_t \in X_t, A_{t,0}(\lambda_t)x_t + A_{t,1}(\lambda_t)x_{t-1} = h_t(\lambda_t) \end{array} \right\} \quad (4.3)$$

where

$$\underline{Q}_{t+1}(x_t, \lambda_{[t+1]}) := \begin{cases} \infty, & \text{if } \exists (\gamma, g) \in C_{\text{feas}}(\lambda_{[t+1]}) : \\ & \gamma + \langle g, x_t \rangle > 0, \text{ or} \\ \max_{(\alpha, a) \in C_{\text{opt}}(\lambda_{[t+1]})} (\alpha + \langle a, x_t \rangle), & \text{otherwise;} \end{cases} \quad (4.4)$$

**if** (4.3) *is infeasible* **then**

add feasibility cut constructed from dual ray to  $C_{\text{feas}}(\lambda_{[t]})$ ;

infeasible  $\leftarrow$  TRUE;

**break**;

**else if** *optimal value of (4.3) is larger than approximated value*  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$  (*computed when evaluating master problem at stage  $t - 1$* ) **then**

add supporting hyperplane constructed from dual solution to  $C_{\text{opt}}(\lambda_{[t]})$ ;

optimal  $\leftarrow$  FALSE;

**end**

**foreach**  $\lambda_{[t+1]} \in \Lambda_{t+1}$  *that is a successor of*  $\lambda_{[t]}$  **do**

add  $x_t$  to  $Z(\lambda_{[t+1]})$  and remember  $\underline{Q}_{t+1}(x_t, \lambda_{[t+1]})$ ;

**end**

**end**

**return** (infeasible, optimal);

---

---

**Function** ProcessBendersMasterLast( $\lambda_{[T]}$ ) processes  $T$ -stage cost-to-go functions of representative node  $\lambda_{[T]} \in \Lambda_T$  within Nested Benders Decomposition for recombining scenario tree

---

**input** : approximate solutions  $x_{T-1} \in Z(\lambda_{[T]})$  that demand for an evaluation of the cost-to-go function corresponding to  $\lambda_{[T]}$

infeasible  $\leftarrow$  FALSE;

optimal  $\leftarrow$  TRUE;

**foreach** *approximate solution*  $x_{T-1} \in Z(\lambda_{[T]})$  **do**

    evaluate cost-to-go function ( $Q_T(x_{T-1}, \lambda_{[T]})$ ) by solving

$$\min \{ \langle b_T(\lambda_T), x_T \rangle : x_T \in X_T, A_{T,0}(\lambda_T)x_T + A_{T,1}(\lambda_T)x_{T-1} = h_T(\lambda_T) \}; \quad (4.5)$$

**if** (4.5) *is infeasible* **then**

        add feasibility cut constructed from dual ray to  $C_{\text{feas}}(\lambda_{[T]})$ ;

        infeasible  $\leftarrow$  TRUE;

**break**;

**else if** *optimal value of (4.5) is larger than approximated value*  $Q_T(x_{T-1}, \lambda_{[T]})$  (*computed when evaluating master problem at stage*  $T - 1$ ) **then**

        add supporting hyperplane constructed from dual solution to  $C_{\text{opt}}(\lambda_{[T]})$ ;

        optimal  $\leftarrow$  FALSE;

**end**

**end**

**return** (infeasible, optimal);

---



---

**Function** AggregatePoints aggregates a set of approximation solution points

---

**input** : a finite set  $Z \subset \mathbb{R}^m$  of approximate solution points

**input** : a scaling vector  $\chi \in \mathbb{R}^m$  and an aggregation parameter  $\rho \geq 0$

construct an undirected graph  $G$  that has  $Z$  as vertices and an edge  $\{x, x'\} \in Z \times Z$  exists if and only if

$$\sum_{k=1}^m \chi_k |x_k - x'_k| \leq \rho;$$

**if**  $G$  *has no edges* **then return** FALSE;

**while**  $G$  *has edges* **do**

    select  $x \in Z$  of maximal degree;

    remove neighborhood of  $x$  from  $Z$  (and  $G$ );

**end**

**return** TRUE;

---

#### 4. Decomposition with Recombining Scenario Trees

where the maximum is taken with respect to all  $\pi$  in the subdifferential of  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$ , see also Ruszczyński [2003]. Thus, aggregation of the sets  $Z(\lambda_{[t]})$  can decrease the number of evaluations of  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$  to a reduced number of solution points, whereby the error caused by not evaluating all points can be controlled via the Lipschitz continuity of  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$ .

The *aggregation* is realized by removing for some  $x_{t-1} \in Z(\lambda_{[t]})$  all points  $x'_{t-1} \in Z(\lambda_{[t]}) \setminus \{x_{t-1}\}$  that are considered as close to  $x_{t-1}$ , that is,

$$\sum_{k=1}^{m_{t-1}} \chi_k(\lambda_{[t]}) |x_{t-1,k} - x'_{t-1,k}| \leq \rho.$$

Here,  $\rho \geq 0$  is an *aggregation parameter* and  $\chi(\lambda_{[t]}) \in \mathbb{R}_+^{m_{t-1}}$  is a scaling vector. The aggregation heuristic is stated in Function `AggregatePoints()` above<sup>2</sup>. Thus, subproblems corresponding to two nodes  $\xi_{[t]}^i$  and  $\xi_{[t]}^j$  of the scenario tree are identified with each other *during* an iteration of the algorithm, if condition (4.1) holds and their current approximate solution points do not differ by more than  $\rho$ . One can interpret this approach as a *dynamic recombination of scenarios*. In difference to a static recombination of scenarios in advance, here the recombination scheme can change with each iteration of the algorithm and thus adjusts itself. In other words, even though we cannot expect the solution process  $\{x_t\}_t$  to follow the same recombination scheme as  $\{\xi_t\}_t$ , it may possess a *more granular* recombination scheme, which is revealed by the algorithm itself.

The scaling vector  $\chi(\lambda_{[t]})$  is chosen as follows. We initialize  $\chi_k(\lambda_{[t]}) := \chi_{\min}$ ,  $k \in [m_{t-1}]$ , for  $\chi_{\min}$  a fixed positive parameter. Whenever an optimality cut  $(\alpha, a) \in C_{\text{opt}}(\lambda_{[t]})$  is created for a reference point  $\bar{x}_{t-1}$ , we update

$$\chi_k(\lambda_{[t]}) := \max \left( \chi_k(\lambda_{[t]}), \frac{|a_k|}{\max(1, |\mathcal{Q}_t(\bar{x}_{t-1}, \lambda_{[t]})|)} \right).$$

The idea is that  $\chi(\lambda_{[t]})$  approximates the (maximal) subgradient of the cost-to-go function  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$  on its domain, scaled by the value of the cost-to-go function, c.f. (4.6).

The “grid-size” parameter  $\rho$  guides the roughness of the aggregation, that is  $\rho = 0$  corresponds to the removal of no points, and  $\rho = 1$  corresponds to an extreme reduction of  $Z(\lambda_{[t]})$ . Starting the decomposition algorithm with a large value for  $\rho$ , a rough approximation of the cost-to-go functions  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$  is obtained. Empirical observations show that this preprocessing may lead to a significant speed-up, cf. Table 4.4 in Section 4.6. This is due to the fact that the rough approximation produces solution points that are already close to an optimal solution of the problem, and, hence, the generation of too many “useless” cuts during the first iterations of the algorithm can be avoided. After having solved the problem roughly, the approximation can be improved by decreasing  $\rho$ .

A modification of Algorithm 4.1 that applies the aggregation operation is given in Algorithm 4.2. If (1.9) is feasible, then Algorithm 4.2 finds the optimal value and an

<sup>2</sup>A best aggregation could be achieved by finding a minimum vertex cover in the graph that has  $Z(\lambda_{[t]})$  as vertices and an edge exists if two points are considered as close. However, minimum vertex cover is known to be an NP problem [Garey and Johnson, 1979], so that we content to use a simple greedy heuristic to find a good aggregation of  $Z(\lambda_{[t]})$ .

---

**Algorithm 4.2:** Nested Benders Decomposition for recombining scenario trees with aggregation of solution points

---

```

1  $C_{\text{feas}}(\lambda_{[t]}) \leftarrow \emptyset, C_{\text{opt}}(\lambda_{[t]}) \leftarrow \{(L, 0)\}, \lambda_{[t]} \in \Lambda_t, t \in [2 : T];$ 
2  $t \leftarrow 1; \rho \leftarrow 1;$ 
3 loop
4   foreach  $\lambda_{[t]} \in \Lambda_t$  do
5     if  $t = 1$  then
6        $\text{infeasible} \leftarrow \text{ProcessBendersMasterFirst}();$ 
7        $\text{alloptimal} \leftarrow \text{TRUE};$ 
8        $\text{aggregation} \leftarrow \text{FALSE};$ 
9        $Z(\lambda_{[\tau]}) \leftarrow \emptyset, \lambda_{[\tau]} \in \Lambda_{\tau}, \tau \in [2 : T];$ 
10    else if  $t < T$  then
11       $\text{aggregation} \leftarrow \text{aggregation} \vee \text{AggregatePoints}(Z(\lambda_{[t]}), \chi(\lambda_{[t]}), \rho);$ 
12       $(\text{infeasible}, \text{optimal}) \leftarrow \text{ProcessBendersMasterMiddle}(t, \lambda_{[t]});$ 
13       $\text{alloptimal} \leftarrow \text{alloptimal} \wedge \text{optimal};$ 
14    else
15       $\text{aggregation} \leftarrow \text{aggregation} \vee \text{AggregatePoints}(Z(\lambda_{[t]}), \chi(\lambda_{[t]}), \rho);$ 
16       $(\text{infeasible}, \text{optimal}) \leftarrow \text{ProcessBendersMasterLast}(\lambda_{[t]});$ 
17       $\text{alloptimal} \leftarrow \text{alloptimal} \wedge \text{optimal};$ 
18    end
19    if  $\text{infeasible}$  then
20      if  $t = 1$  then STOP: (1.9) is infeasible;
21      break;
22    end
23  end
24  if  $\text{infeasible}$  then  $t \leftarrow 1;$ 
25  else if  $t = T$  and  $\text{alloptimal}$  then
26    if  $\text{aggregation}$  then
27       $\rho \leftarrow \rho/2;$ 
28       $t \leftarrow 1;$ 
29    end
30    STOP: (1.9) has been solved to optimality;
31  else  $t \leftarrow (t \bmod T) + 1;$ 
32 end;

```

---

optimal first stage solution, since at some iteration,  $\rho$  will have been decreased sufficiently enough to rule out any further aggregations. However, as shown in Proposition 4.1 below, an early interruption of the algorithm yields a value which difference to the optimal value can be estimated by a multiple of the aggregation tolerance  $\rho$ . Unfortunately, decreasing  $\rho$  until an error tolerance on the aggregation error is satisfied leads to large sets  $Z(\lambda_{[t]})$  again. Thus, a reliable and better adapted stopping criterion is discussed in Section 4.4.

#### 4. Decomposition with Recombining Scenario Trees

**Proposition 4.1.** *Assume  $\mathcal{Q}_t(x_{t-1}, \lambda_{[t]}) < \infty$  for all  $x_{t-1} \in X_{t-1}$ ,  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T]$  (relatively complete recourse) and  $X_t$  to be bounded for all  $t \in [T]$  (dual feasibility). Denote by  $v$  the optimal value of (1.10) and by  $\underline{v}$  the optimal value of the first stage master problem (4.2). At line 25 of Algorithm 4.2, a  $\rho$ -optimal solution has been found for (1.10), i.e.,  $|v - \underline{v}| < C_2\rho$  holds true for some constant  $C_2 \geq 0$ .*

*Proof.* To estimate  $|v - \underline{v}|$ , we first show that there exist constants  $C_t$ ,  $t \in [T]$ , such that

$$|\mathcal{Q}_t(x_{t-1}, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(x_{t-1}, \lambda_{[t]})| \leq C_t\rho \quad (x_t \in Z(\lambda_{[t]}), t \in [T]). \quad (4.7)$$

Note, that  $Z(\lambda_{[t]})$ ,  $\lambda_{[t]} \in \Lambda_t$ , contain the solution points for which the master problems have been evaluated since  $t$  has been reset to 1 the last time. Denote the sets of solution points before the aggregation operation has been applied as  $\hat{Z}(\lambda_{[t]})$ ,  $\lambda_{[t]} \in \Lambda_t$ , and let  $\varphi(\lambda_{[t]}) : \hat{Z}(\lambda_{[t]}) \rightarrow Z(\lambda_{[t]})$  be the aggregation mapping that assigns each  $x_{t-1} \in \hat{Z}(\lambda_{[t]})$  to a close representative point in  $Z(\lambda_{[t]})$ , i.e.,

$$\sum_{k=1}^{m_{t-1}} \chi_k(\lambda_{[t]}) |(\varphi(\lambda_{[t]})(x_{t-1}))_k - x_{t-1,k}| \leq \rho.$$

Thus,  $\|\varphi(\lambda_{[t]})(x_{t-1}) - x_{t-1}\|_\infty \leq C^\varphi(\lambda_{[t]})\rho$  with  $C^\varphi(\lambda_{[t]}) := (\min_k \chi_k(\lambda_{[t]}))^{-1} < \chi_{\min}^{-1}$ .

Define  $C_{T+1} = 0$ . Let  $t \in [T]$  and assume that (4.7) holds true for time period  $t+1$  (if  $t < T$ ). Fix  $\lambda_{[t]} \in \Lambda_t$ . Fix  $x_{t-1} \in \hat{Z}(\lambda_{[t]})$  and let  $x'_{t-1} := \varphi(\lambda_{[t]})(x_{t-1}) \in Z(\lambda_{[t]})$ . Due to our assumptions,  $\mathcal{Q}_t(x_{t-1}, \lambda_{[t]})$  and  $\mathcal{Q}_t(x'_{t-1}, \lambda_{[t]})$  are finite. Therefore, we can estimate

$$\begin{aligned} |\mathcal{Q}_t(x_{t-1}, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(x_{t-1}, \lambda_{[t]})| &\leq |\mathcal{Q}_t(x'_{t-1}, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(x'_{t-1}, \lambda_{[t]})| \\ &\quad + |\underline{\mathcal{Q}}_t(x_{t-1}, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(x'_{t-1}, \lambda_{[t]})| + |\mathcal{Q}_t(x_{t-1}, \lambda_{[t]}) - \mathcal{Q}_t(x'_{t-1}, \lambda_{[t]})|. \end{aligned} \quad (4.8)$$

At line 25 of Algorithm 4.2, no optimality cuts have been generated after evaluating the master problem at stage  $t$  for  $x'_{t-1}$  ((4.3) if  $t < T$ , (4.5) otherwise). For  $t = T$ , it follows that  $\underline{\mathcal{Q}}_T(x'_{T-1}, \lambda_{[T]}) = \mathcal{Q}_T(x'_{T-1}, \lambda_{[T]})$ . For  $t < T$ ,  $\underline{\mathcal{Q}}_t(x'_{t-1}, \lambda_{[t]})$  coincides with the value of the master problem (4.3). Let  $x_t^*$  be the solution of (4.3). Then  $x_t^* \in Z(\lambda_{[t+1]})$  for all  $\lambda_{[t+1]} \in \Lambda_{t+1}$  that are a successor of  $\lambda_{[t]}$ . Using (4.7) for time stage  $t+1$ , we can estimate the first term on the right-hand side of (4.8) by

$$\begin{aligned} |\mathcal{Q}_t(x'_{t-1}, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(x'_{t-1}, \lambda_{[t]})| &= \mathcal{Q}_t(x'_{t-1}, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(x'_{t-1}, \lambda_{[t]}) \\ &\leq \mathbb{E} \left[ \mathcal{Q}_{t+1}(x_t^*, \bar{\lambda}_{t+1}(\xi_{[t+1]})) - \underline{\mathcal{Q}}_{t+1}(x_t^*, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] \leq C_{t+1}\rho. \end{aligned}$$

Since  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$  and  $\underline{\mathcal{Q}}_t(\cdot, \lambda_{[t]})$  are Lipschitz-continuous functions, the last two terms of (4.8) can be estimated by  $L(\lambda_{[t]})\|x - x'\|_\infty$  for some Lipschitz constant  $L(\lambda_{[t]}) > 0$ . Thus, we can continue at (4.8) and obtain

$$|\mathcal{Q}_t(x_{t-1}, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(x_{t-1}, \lambda_{[t]})| \leq C_{t+1}\rho + L(\lambda_{[t]})\|x - x'\|_\infty \leq C_t\rho$$

with  $C_t := C_{t+1} + \max\{L(\lambda_{[t]})C^\varphi(\lambda_{[t]}) : \lambda_{[t]} \in \Lambda_t\}$ .

Finally, let  $x_1^*$  be the solution of the first stage master problem (4.2). Then

$$|v - \underline{v}| = v - \underline{v} \leq \mathbb{E}[\mathcal{Q}_2(x_1^*, \lambda_{[2]}) - \underline{\mathcal{Q}}_2(x_1^*, \lambda_{[2]})] \leq C_2\rho. \quad \square$$

**Remark 4.2.** The relatively complete recourse assumption in Proposition 4.1 was necessary, because otherwise the aggregation algorithm could remove a point from some  $Z_t(\lambda_{[t]})$  for which actually a feasibility cut need to be generated. Since the domain of the cost-to-go functions  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$  are convex, this assumption can be dropped if the aggregation is modified such that no extremal points of the convex hull of  $Z_t(\lambda_{[t]})$  is removed from  $Z_t(\lambda_{[t]})$ , see also Corollary 3.2.4 in K uchler [2009].

The parameter  $\rho$  allows to control the approximation quality and has considerable influence on the algorithm’s running time. On the one hand, Proposition 4.1 shows that for sufficiently small  $\rho$  the accuracy of the approximation becomes arbitrarily good. On the other hand, the value of the constant  $C_2$  is usually unknown and the actual approximation error may be much smaller than  $C_2\rho$ . Thus, Proposition 4.1 does not provide a criteria for choosing  $\rho$  such that a predefined approximation quality is guaranteed. In the following, we will show how an estimate on the approximation error can be computed.

### 4.3. Nested Column Generation for Multistage Stochastic Linear Programs with Recombining Scenarios

In the following, we study an approach to estimate the gap  $\mathcal{Q}_t(\cdot, \lambda_{[t]}) - \underline{\mathcal{Q}}_t(\cdot, \lambda_{[t]})$  and therefore the term  $|v - \underline{v}|$  during the solution process. The approach consists of deriving an *upper bounding* approximation that can be combined with the *lower bounding* approximations  $\underline{\mathcal{Q}}_t(\cdot, \lambda_{[t]})$ . The upper bounds are obtained via a Nested Column Generation algorithm [Ford and Fulkerson, 1958, Glassey, 1973, L ubbecke, 2010] based on a Dantzig-Wolfe decomposition [Dantzig and Wolfe, 1960] of (1.9). Instead of approximating  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$  from “outside” by supporting optimality and feasibility cuts, now an “inner” approximation is generated using extremal points and rays.

#### Inner Approximations

First, consider last stage cost-to-go functions, cf. (1.10c), for some  $\lambda_{[T]} \in \Lambda_T$  and the set

$$\Pi(\lambda_{[T]}) := \{(x_{T-1}, x_T) \in \mathbb{R}^{m_{T-1}+m_T} : x_T \in X_T, A_{T,0}(\lambda_T)x_T + A_{T,1}(\lambda_T)x_{T-1} = h_T(\lambda_T)\}.$$

Note, that if  $\mathcal{Q}_T(\cdot, \lambda_{[T]})$  is bounded from below, then  $\text{dom } \mathcal{Q}_T(\cdot, \lambda_{[T]}) = \text{Pr}_{x_{T-1}} \Pi(\lambda_{[T]})$ . Further,  $\Pi(\lambda_{[T]})$  is polyhedral. Thus, due to the representation theorem of Minkowski and Weyl [Schrijver, 1986], it can be written as convex combination of a finite number  $n \in \mathbb{N}$  of points  $v^1 := (v_{T-1}^1, v_T^1), \dots, v^n := (v_{T-1}^n, v_T^n) \in \mathbb{R}^{m_{T-1}+m_T}$  plus a nonnegative combination of a finite number  $s \in \mathbb{N}$  of rays  $w^1 := (w_{T-1}^1, w_T^1), \dots, w^s := (w_{T-1}^s, w_T^s) \in \mathbb{R}^{m_{T-1}+m_T}$ :

$$\Pi(\lambda_{[T]}) = \left\{ \sum_{i=1}^n \alpha_i v^i + \sum_{j=1}^s \beta_j w^j : \alpha \in \Delta, \beta \geq 0 \right\},$$

#### 4. Decomposition with Recombining Scenario Trees

where we denote by

$$\Delta := \left\{ \alpha \in \mathbb{R}_+^k : \sum_{i=1}^k \alpha_i = 1 \right\} \quad (4.9)$$

the standard simplex of suitable dimension. Substituting  $\Pi(\lambda_{[T]})$  in (1.10c) yields

$$\mathcal{Q}_T(x_{T-1}, \lambda_{[T]}) = \min \left\{ \begin{array}{l} \sum_{i=1}^n \alpha_i \langle b_T(\lambda_T), v_T^i \rangle + \sum_{j=1}^s \beta_j \langle b_T(\lambda_T), w_T^j \rangle : \\ \sum_{i=1}^n \alpha_i v_{T-1}^i + \sum_{j=1}^s \beta_j w_{T-1}^j = x_{T-1}, \\ \alpha \in \Delta, \beta \geq 0. \end{array} \right\}$$

This formulation offers a new way for approximating the function  $\mathcal{Q}_T(\cdot, \lambda_{[T]})$ . Instead of underestimation by supporting hyperplanes, an overestimate is obtained by restricting the reformulation of  $\mathcal{Q}_T(\cdot, \lambda_{[T]})$  to a set of explicitly computed vectors  $v^i$  and rays  $w^j$ . That is, given sets of *columns*  $D_{\text{pt}}(\lambda_{[T]}) \subseteq \mathbb{R}^{m_{T-1}+1}$  and  $D_{\text{ray}}(\lambda_{[T]}) \subseteq \mathbb{R}^{m_{T-1}+1}$  such that for each  $(v, \nu) \in D_{\text{pt}}(\lambda_{[T]})$  there exists a vector  $(v_{T-1}, v_T) \in \{v^1, \dots, v^n\}$  with  $v = v_{T-1}$  and  $\nu = \langle b_T(\lambda_T), v_T \rangle$  and for each  $(w, \omega) \in D_{\text{ray}}(\lambda_{[T]})$  there exists a ray  $(w_{T-1}, w_T) \in \{w^1, \dots, w^s\}$  with  $w = w_{T-1}$  and  $\omega = \langle b_T(\lambda_T), w_T \rangle$ , we define

$$\bar{\mathcal{Q}}_T(x_{T-1}, \lambda_{[T]}) := \min \left\{ \begin{array}{l} \sum_{(v, \nu) \in D_{\text{pt}}(\lambda_{[T]})} \alpha_v \nu + \sum_{(w, \omega) \in D_{\text{ray}}(\lambda_{[T]})} \beta_w \omega : \\ \sum_{(v, \nu) \in D_{\text{pt}}(\lambda_{[T]})} \alpha_v v + \sum_{(w, \omega) \in D_{\text{ray}}(\lambda_{[T]})} \beta_w w = x_{T-1} \\ \alpha \in \Delta, \beta \geq 0, \end{array} \right\} \quad (4.10)$$

It is clear, that  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]}) \geq \mathcal{Q}_T(\cdot, \lambda_{[T]})$ . The approximations  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]})$ ,  $\lambda_{[T]} \in \Lambda_T$ , can be utilized to approximate the cost-to-go functions at stage  $T-1$ . That is, analogously to the Nested Benders Decomposition scheme, we define for  $\lambda_{[T-1]} \in \Lambda_{T-1}$  and  $x_{T-2} \in \mathbb{R}^{m_{T-2}}$  the master problem

$$\min \left\{ \begin{array}{l} \langle b_{T-1}(\lambda_{T-1}), x_{T-1} \rangle + \mathbb{E} \left[ \bar{\mathcal{Q}}_T(x_{T-1}, \bar{\lambda}_T(\xi_{[T]})) \mid \xi_{[T-1]} = \lambda_{[T-1]} \right] : \\ x_{T-1} \in X_{T-1}, A_{T-1,0}(\lambda_{T-1})x_{T-1} + A_{T-1,1}(\lambda_{T-1})x_{T-2} = h_{T-1}(\lambda_{T-1}) \end{array} \right\} \quad (4.11)$$

Note, that (4.11) can equivalently be written as a usual linear program:

$$\min \left\{ \begin{array}{l} \langle b_{T-1}(\lambda_{T-1}), x_{T-1} \rangle + \\ \mathbb{E} \left[ \sum_{(v, \nu) \in D_{\text{pt}}(\bar{\lambda}_T(\xi))} \alpha_v \nu + \sum_{(w, \omega) \in D_{\text{ray}}(\bar{\lambda}_T(\xi))} \beta_w \omega \mid \xi_{[T-1]} = \lambda_{[T-1]} \right] : \\ x_{T-1} \in X_{T-1}, \\ A_{T-1,0}(\lambda_{T-1})x_{T-1} + A_{T-1,1}(\lambda_{T-1})x_{T-2} = h_{T-1}(\lambda_{T-1}), \\ \sum_{(v, \nu) \in D_{\text{pt}}(\bar{\lambda}_{[T]})} \alpha_v v + \sum_{(w, \omega) \in D_{\text{ray}}(\bar{\lambda}_{[T]})} \beta_w w = x_{T-1}, \forall \bar{\lambda}_{[T]} \in \Lambda_T : \bar{\lambda}_{[T-1]} = \lambda_{[T]}, \end{array} \right\}$$

### 4.3. Nested Column Gen. for Multistage Stochastic Linear Prog. with Recomb. Scenarios

where we omitted the dependency of  $\alpha$  and  $\beta$  on  $\tilde{\lambda}_{[T]}$  for notational reasons. Thus, also the domain of (4.11) is polyhedral and can be represented by a set of vertices and rays. Again, restricting to some explicitly computed vertices and rays and projection onto  $x_{T-2}$  allows to construct an overestimate of  $\mathcal{Q}_{T-1}(\cdot, \Lambda_{[T-1]})$ . Continuing this process until the first stage defines a set of master problems

$$\min \left\{ \begin{array}{l} \langle b_t(\lambda_t), x_t \rangle + \mathbb{E} \left[ \bar{\mathcal{Q}}_{t+1}(x_t, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] : \\ x_t \in X_t, A_{t,0}(\lambda_t)x_t + A_{t,1}(\lambda_t)x_{t-1} = h_t(\lambda_t) \end{array} \right\} \quad (4.12)$$

and approximate cost-to-go functions

$$\bar{\mathcal{Q}}_t(x_{t-1}, \lambda_{[t]}) := \min \left\{ \begin{array}{l} \sum_{(v,\nu) \in D_{\text{pt}}(\lambda_{[t]})} \alpha_v \nu + \sum_{(w,\omega) \in D_{\text{ray}}(\lambda_{[t]})} \beta_w \omega : \\ \sum_{(v,\nu) \in D_{\text{pt}}(\lambda_{[t]})} \alpha_v v + \sum_{(w,\omega) \in D_{\text{ray}}(\lambda_{[t]})} \beta_w w = x_{t-1} \\ \alpha \in \Delta, \beta \geq 0, \end{array} \right\} \quad (4.13)$$

for  $D_{\text{pt}}(\lambda_{[t]}) \subset \mathbb{R}^{m_{t-1}+1}$ ,  $D_{\text{ray}}(\lambda_{[t]}) \subset \mathbb{R}^{m_{t-1}+1}$ ,  $\lambda_{[t]} \in \Lambda_t$ ,  $x_{t-1} \in \mathbb{R}^{m_{t-1}}$ ,  $t \in [2 : T-1]$ .

The first stage master problem is given by

$$\min \left\{ \langle b_1(\lambda_1), x_1 \rangle + \mathbb{E} \left[ \bar{\mathcal{Q}}_2(x_1, \bar{\lambda}_2(\xi_{[2]})) \right] : x_1 \in X_1 \right\}. \quad (4.14)$$

### Pricing

In the following, we discuss how a given approximation is improved for the final stage  $t = T$ . Assume  $\bar{\mathcal{Q}}_T(x_{T-1}, \lambda_{[T]})$  has been evaluated for some  $x_{T-1} \in X_{T-1}$ . We are now interested to either prove that  $\bar{\mathcal{Q}}_T(x_{T-1}, \lambda_{[T]}) = \mathcal{Q}_T(x_{T-1}, \lambda_{[T]})$  or to find a new column  $(v, \nu)$  or  $(w, \omega)$  (associated with a vertex or ray of  $\Pi(\lambda_{[T]})$ ), such that adding this column to (4.10) improves the approximation in  $x_{T-1}$ . To this end, consider the dual of (4.10):

$$\bar{\mathcal{Q}}_T(x_{T-1}, \lambda_{[T]}) = \max \left\{ \langle \mu, x_{T-1} \rangle + \sigma : \begin{array}{l} \langle \mu, v \rangle + \sigma \leq \nu, \quad (v, \nu) \in D_{\text{pt}}(\lambda_{[T]}) \\ \langle \mu, w \rangle \leq \omega, \quad (w, \omega) \in D_{\text{ray}}(\lambda_{[T]}) \end{array} \right\}. \quad (4.15)$$

Here,  $\mu$  is the dual variable corresponding to  $\sum_{(v,\nu) \in D_{\text{pt}}(\lambda_{[T]})} \alpha_v v + \sum_{(w,\omega) \in D_{\text{ray}}(\lambda_{[T]})} \beta_w w = x_{T-1}$  in (4.10) and  $\sigma$  is the dual variable corresponding to the convexity constraint in (4.9). Formulation (4.15) can further be rewritten as

$$\bar{\mathcal{Q}}_T(x_{T-1}, \lambda_{[T]}) = \max \left\{ \begin{array}{l} \langle \mu, x_{T-1} \rangle + \min_{(v,\nu) \in D_{\text{pt}}(\lambda_{[T]})} (\nu - \langle \mu, v \rangle) : \\ \langle \mu, w \rangle \leq \omega, \quad (w, \omega) \in D_{\text{ray}}(\lambda_{[T]}) \end{array} \right\}.$$

In order to decrease the value of  $\bar{\mathcal{Q}}_T(x_{T-1}, \lambda_{[T]})$ , we seek for a column  $(v, \nu)$  associated with a vertex of  $\Pi(\lambda_{[T]})$  with  $\nu - \langle \mu, v \rangle < \sigma$  or a column  $(w, \omega)$  associated with a ray of  $\Pi(\lambda_{[T]})$  with  $\langle \mu, w \rangle > \omega$ , where  $(\mu, \sigma)$  is a dual solution of (4.10). Hence, we minimize

#### 4. Decomposition with Recombining Scenario Trees

$\nu - \langle \mu, \nu \rangle$  over the set of possible columns, which yields the *pricing problem*

$$\min\{\langle b_T(\lambda_T), x_T \rangle - \langle \mu, x_{T-1} \rangle : (x_{T-1}, x_T) \in \Pi(\lambda_{[T]})\} \quad (4.16)$$

If (4.16) is bounded and has an optimal value equal to  $\sigma$ , then no improving column is available and we know that the approximation  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]})$  is exact at  $x_{T-1}$ . If (4.16) is unbounded and has an optimal value lower than  $\sigma$ , let  $(x_{T-1}, x_T)$  be a solution of (4.16) which corresponds to an extreme point of  $\Pi(\lambda_{[T]})$ . The column  $(v, \nu) := (x_{T-1}, \langle b_T(\lambda_T), x_T \rangle)$  is then added to  $D_{\text{pt}}(\lambda_{[T]})$ . Since  $\langle \mu, v \rangle - \nu > \sigma$ , the value of (4.15) is decreased, and thus the approximation  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]})$  is improved in  $x_{T-1}$ . If (4.16) is unbounded, then there exists a primal ray  $(y_{T-1}, y_T)$  of  $\Pi(\lambda_{[T]})$  such that  $\langle b_T(\lambda_T), y_T \rangle - \langle \mu, y_{T-1} \rangle < 0$ . Thus, adding  $(w, \omega) := (y_{T-1}, \langle b_T(\lambda_T), y_T \rangle)$  to  $D_{\text{ray}}(\lambda_{[T]})$  reduces the value of (4.15) and thus improves the approximation  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]})$  in  $x_{T-1}$ .

For  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T - 1]$ , the pricing problems are derived in a similar manner from the master problem (4.12). Given a dual solution  $(\mu, \sigma)$  of  $\bar{\mathcal{Q}}_t(x_{t-1}, \lambda_{[t]})$ , cf. (4.13), we solve the pricing problem

$$\min \left\{ \begin{array}{l} \langle b_t(\lambda_t), x_t \rangle - \langle \mu, x_{t-1} \rangle + \mathbb{E} \left[ \bar{\mathcal{Q}}_{t+1}(x_t, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] : \\ x_t \in X_t, A_{t,0}(\lambda_t)x_t + A_{t,1}(\lambda_t)x_{t-1} = h_t(\lambda_t) \end{array} \right\} \quad (4.17)$$

If (4.17) is bounded and has an optimal value below  $\sigma$ , then an optimal solution  $(x_{t-1}, x_t)$  gives rise to a new column

$$(v, \nu) := \left( x_{t-1}, \langle b_t(\lambda_t), x_t \rangle + \mathbb{E} \left[ \bar{\mathcal{Q}}_{t+1}(x_t, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] \right)$$

that is added to  $D_{\text{pt}}(\lambda_t)$ . If (4.17) is unbounded, then a primal ray  $(y_{t-1}, y_t)$  that proves unboundedness gives rise to a new column

$$(w, \omega) := \left( y_{t-1}, \langle b_t(\lambda_t), y_t \rangle + \mathbb{E} \left[ \bar{\mathcal{Q}}_{t+1}(y_t, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] \right)$$

that is added to  $D_{\text{ray}}(\lambda_{[t]})$ .

#### Relation to Nested Benders Decomposition

The similarity of column generation as discussed here and cut generation in the Nested Benders Decomposition is apparent. While in a Nested Benders Decomposition, primal information in form of approximate solution points  $x_{t-1}$  is passed forward and dual information in form of optimality and feasibility cuts is passed backward, in a column generation approach, dual information  $(\mu, \sigma)$  is passed forward and primal information in form of extreme points and rays is passed backward. The analogy is even clearer when comparing the dual of an inner approximation of a cost-to-go function, see (4.15), with an outer approximation of the same function, see (4.4). Thus, the extreme points  $\mathcal{D}_{\text{pt}}(\lambda_{[t]})$  in column generation correspond to optimality cuts in Benders Decomposition, while the rays  $\mathcal{D}_{\text{ray}}(\lambda_{[t]})$  correspond to feasibility cuts. In other words, Nested Column Generation for (1.9) corresponds to Nested Benders Decomposition on the dual of (1.9).

### Initialization

To initialize the sets  $\mathcal{D}_{\text{pt}}(\lambda_{[t]})$  and  $\mathcal{D}_{\text{ray}}(\lambda_{[t]})$ ,  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T]$ , first columns may be generated similar to a Phase I initialization in the simplex method for linear programs. For that purpose, the approximate cost-to-go functions (4.13) and (4.10) are replaced by

$$\tilde{\mathcal{Q}}_t(x_{t-1}, \lambda_{[t]}) := \min \left\{ \begin{array}{l} \|y\|_1 + \sum_{(v,\nu) \in \mathcal{D}_{\text{pt}}(\lambda_{[t]})} \alpha_\nu \nu + \sum_{(w,\omega) \in \mathcal{D}_{\text{ray}}(\lambda_{[t]})} \beta_\omega \omega : \\ \sum_{(v,\nu) \in \mathcal{D}_{\text{pt}}(\lambda_{[t]})} \alpha_\nu \nu + \sum_{(w,\omega) \in \mathcal{D}_{\text{ray}}(\lambda_{[t]})} \beta_\omega \omega + y = x_{t-1} \\ y \in \mathbb{R}^{m_{t-1}}, \alpha \in \Delta, \beta \geq 0, \end{array} \right\}$$

where  $\alpha$  is omitted as long as  $\mathcal{D}_{\text{pt}}(\lambda_{[t]})$  is empty,  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T]$ . Thus, we aim at minimizing the slack  $y$  between a point  $x_{t-1} \in X_{t-1}$  and the inner approximation of the domain of  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$ . This inner approximation may be enlarged by generating new columns for  $\mathcal{D}_{\text{pt}}(\lambda_{[t]})$  or  $\mathcal{D}_{\text{ray}}(\lambda_{[t]})$  via solution of the pricing problems (4.17) and (4.16) with  $\langle b_t(\lambda_t), x_t \rangle$  removed from the objective. If no improving columns are found, but the slack is still nonzero, then infeasibility of the original problem is proven. When initialization is finished, columns with nonzero  $\nu$  or  $\omega$  have to be removed, since they may not belong to the domain of the original cost-to-go functions. For all remaining columns, the values  $\nu$  and  $\omega$  need to be recomputed with the restored original objective functions by evaluating the last stage cost-to-go function (1.10c) or master problems (4.12).

### Full Algorithm

The Nested Column Generation algorithm for multistage stochastic linear programs with recombining scenario trees is stated in Algorithm 4.3. We assume for simplicity that the sets  $X_t$ ,  $t \in [T]$ , are nonempty. Otherwise, the modified first-stage master problem (4.18) or a modified pricing problem (4.19) or (4.20) is infeasible during the initialization phase, and thus infeasibility of the original problem is recognized. Note, that if the initialization phase concludes that the problem is feasible, the master and pricing problems are ensured to remain feasible after removing columns with nonzero objective coefficients, since these do not contribute to the feasibility proof. Further, master and pricing problems remain feasible, since only dual information in form of  $(\mu, \sigma)$  is passed forward, and new columns that enlarge the feasible space of the inner approximations are passed backwards. Thus, it is sufficient to consider unboundedness of the pricing problems. Comments on the sequencing protocol for Nested Benders Decomposition, c.f. Section 4.2, are valid one to one also for Nested Column Generation. Further, the phase I initialization can be interrupted once the value of the first stage master problem reaches zero.

If newly generated columns correspond to extreme points or rays of the (polyhedral) feasible set of the corresponding pricing problems, termination of Algorithm 4.3 in a finite number of steps, thereby either proving infeasibility of the original problem or computing its optimal value, can be shown [Glassey, 1973]. The argumentation bases on the finite number of possible columns that could be generated and is thus – without surprise – similar to the termination proof for Nested Benders Decomposition [Ruszczynski, 2003].

#### 4. Decomposition with Recombining Scenario Trees

---

**Algorithm 4.3:** Nested Column Generation for recombining scenario trees

---

```

 $D_{\text{pt}}(\lambda_{[t]}) \leftarrow \emptyset, D_{\text{ray}}(\lambda_{[t]}) \leftarrow \emptyset, \lambda_{[t]} \in \Lambda_t, t \in [2 : T];$ 
for phase =  $I$  to  $II$  do
   $t \leftarrow 1;$ 
  loop
    unbounded  $\leftarrow$  FALSE;
    foreach  $\lambda_{[t]} \in \Lambda_t$  do
      if  $t = 1$  then
         $Y(\lambda_{[\tau]}) \leftarrow \emptyset, \lambda_{[\tau]} \in \Lambda_\tau, \tau \in [2 : T];$ 
        ProcessRestrictedMasterFirst(phase);
        alloptimal  $\leftarrow$  TRUE;
      else if  $t < T$  then
        (unbounded, optimal)  $\leftarrow$  ProcessPricingMiddle(phase,  $t, \lambda_{[t]}$ );
        alloptimal  $\leftarrow$  alloptimal  $\wedge$  optimal;
      else
        (unbounded, optimal)  $\leftarrow$  ProcessPricingLast(phase,  $\lambda_{[t]}$ );
        alloptimal  $\leftarrow$  alloptimal  $\wedge$  optimal;
      end
      if unbounded then break;
    end
    if unbounded then  $t \leftarrow 1;$ 
    else if  $t = T$  and alloptimal then break;
    else  $t \leftarrow (t \bmod T) + 1;$ 
  end;
  if phase =  $I$  then
    if optimal value of modified first-stage master problem (4.18) is nonzero then
      STOP: (1.9) is infeasible;
    end
    for  $t = T$  to 2 do foreach  $\lambda_{[t]} \in \Lambda_t$  do
      remove all columns  $(v, \nu) \in D_{\text{pt}}(\lambda_{[t]})$  with nonzero  $\nu$  and all columns
       $(w, \omega) \in D_{\text{ray}}(\lambda_{[t]})$  with nonzero  $\omega;$ 
      foreach remaining  $(x_{t-1}, \chi) \in D_{\text{pt}}(\lambda_{[t]}) \cup D_{\text{ray}}(\lambda_{[t]})$  do
        update  $\chi$  to value of master problem (4.12) (if  $t < T$ ) or last-stage
        cost-to-go function (1.10c) (if  $t = T$ ) w.r.t.  $x_{t-1};$ 
      end
    end
  else
    (1.9) has been solved to optimality;
  end
end

```

---

---

**Function** ProcessRestrictedMasterFirst processes first stage master problem within Nested Column Generation

---

**input** : cost-to-go function approximations  $\widetilde{\mathcal{Q}}_2(\cdot, \lambda_{[2]})$  or  $\overline{\mathcal{Q}}_2(\cdot, \lambda_{[2]})$ ,  $\lambda_{[2]} \in \Lambda_2$ , via columns  $D_{\text{pt}}(\lambda_{[2]})$  and  $D_{\text{ray}}(\lambda_{[2]})$

**if** phase = I **then**

solve modified first stage master problem

$$\min \left\{ \mathbb{E} \left[ \widetilde{\mathcal{Q}}_2(x_1, \bar{\lambda}_2(\xi_{[2]})) \right] : x_1 \in X_1 \right\}. \quad (4.18)$$

**else**

solve first stage master problem (4.14);

**if** (4.14) is unbounded **then STOP**: (1.9) is unbounded;

**end**

**foreach**  $\lambda_{[2]} \in \Lambda_2$  **do**  $Y(\lambda_{[2]}) \leftarrow \{(\mu, \sigma)\}$ , where  $(\mu, \sigma)$  is dual solution corresponding to  $\widetilde{\mathcal{Q}}_2(x_1, \lambda_{[2]})$  in (4.18) (if phase = I) or  $\overline{\mathcal{Q}}_2(x_1, \lambda_{[2]})$  in (4.14) (otherwise);

---

### 4.3.1. Dynamic Recombination of Scenarios

The Nested Column Generation Algorithm 4.3 utilizes the recombining feature of a scenario tree by *column sharing* – as opposed to cut sharing in Nested Benders Decomposition. However, the algorithm may suffer from an exponential growth (with increasing  $t$ ) of the sets  $Y(\lambda_{[t]})$ ,  $\lambda_{[t]} \in \Lambda_t$ , which demand for an evaluation of pricing problems. Similar as in Section 4.2.1, we can argue that the pricing problems as function of  $\mu$  are Lipschitz continuous. Thus, when extending Algorithm 4.3 by an aggregation step for the sets  $Y(\lambda_{[t]})$ , the number of pricing problem evaluations can be reduced, whereby the aggregation error is controllable due to the Lipschitz continuity of  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$ .

A modification of Algorithm 4.3 that applies an aggregation operation to the sets  $Y(\lambda_{[t]})$  is stated in Algorithm 4.4. The aggregation step is performed as in Algorithm 4.2, i.e., via the Function `AggregatePoints()`. However, the scaling vectors  $\zeta(\lambda_{[t]}) \in \mathbb{R}_+^{m_{t-1}+1}$  are updated differently now. We set the last component, which corresponds to the dual variable  $\sigma$ , to zero<sup>3</sup>, and initialize  $\zeta_k(\lambda_{[t]}) := \zeta_{\min}$ ,  $k \in [m_{t-1}]$ , for  $\zeta_{\min}$  a fixed positive parameter. When a new column  $(v, \nu) \in D_{\text{pt}}(\lambda_{[t]})$  is created, we update

$$\zeta_k(\lambda_{[t]}) := \max \left( \zeta_k(\lambda_{[t]}), \frac{|v_k|}{\max(1, |\nu|)} \right).$$

Again, the approach can be interpreted as a dynamic recombination of scenarios, i.e., in each iteration a recombination scheme for the solution process is computed. It may be finer than the one for the stochastic process  $\{\xi_t\}_t$ , but is not enforced from outside, but revealed during the solution process.

---

<sup>3</sup>For two dual solutions  $(\mu_1, \sigma_1)$  and  $(\mu_2, \sigma_2)$  with  $\mu_1 = \mu_2$ , we can assume  $\sigma_1 = \sigma_2$ , since  $\sigma$  is uniquely determined by the dual formulation of  $\mathcal{Q}_t(x_{t-1}, \lambda_{[t]})$ , see also (4.15). Thus, we can ignore the value of  $\sigma$  when computing the distance for a pair of dual solutions.

#### 4. Decomposition with Recombining Scenario Trees

---

**Function** `ProcessPricingMiddle( $t, \lambda_{[t]}$ )` processes  $t$ -stage pricing problem of node  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T - 1]$ , within Nested Column Generation

---

**input**: dual solutions  $(\mu, \sigma) \in Y(\lambda_{[t]})$  that demand for an evaluation of the pricing problem corresponding to  $\lambda_{[t]}$

**input**:  $t + 1$ -stage cost-to-go function approximations  $\tilde{Q}_{t+1}(\cdot, \lambda_{[t+1]})$  or  $\bar{Q}_{t+1}(\cdot, \lambda_{[t+1]})$ ,  $\lambda_{[t+1]} \in \Lambda_{t+1}$ , via columns  $D_{\text{pt}}(\lambda_{[t+1]})$  and  $D_{\text{ray}}(\lambda_{[t+1]})$

unbounded  $\leftarrow$  FALSE;

optimal  $\leftarrow$  TRUE;

**foreach** *dual solution*  $(\mu, \sigma) \in Y(\lambda_{[t]})$  **do**

**if** phase = I **then**

solve modified pricing problem

$$\min \left\{ \begin{array}{l} -\langle \mu, x_{t-1} \rangle + \mathbb{E} \left[ \tilde{Q}_{t+1}(x_t, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] : \\ x_t \in X_t, A_{t,0}(\lambda_t)x_t + A_{t,1}(\lambda_t)x_{t-1} = h_t(\lambda_t) \end{array} \right\} \quad (4.19)$$

**else**

solve pricing problem (4.17);

**end**

**if** *pricing problem is unbounded* **then**

add new column constructed from primal ray to  $D_{\text{ray}}(\lambda_{[t]})$ ;

unbounded  $\leftarrow$  TRUE;

**break**;

**else if** *optimal value of pricing problem is smaller than  $\sigma$*  **then**

add new column constructed from solution of pricing problem to  $D_{\text{pt}}(\lambda_{[t]})$ ;

optimal  $\leftarrow$  FALSE;

**end**

**foreach**  $\lambda_{[t+1]} \in \Lambda_{t+1}$  *that is a successor of  $\lambda_{[t]}$*  **do**

add dual solution  $(\mu, \sigma)$  corresponding to constraints from  $\tilde{Q}_{t+1}(\cdot, \lambda_{[t+1]})$  in (4.19) (if phase = I) or  $\bar{Q}_{t+1}(\cdot, \lambda_{[t+1]})$  in (4.17) (otherwise) to  $Y(\lambda_{[t+1]})$  ;

**end**

**end**

**return** (unbounded, optimal);

---

For a feasible original problem (1.9), Algorithm 4.4 finds the optimal value and an optimal first stage solution, since eventually the aggregation parameter  $\rho$  is reduced sufficiently enough to prevent removal of points from  $Y(\lambda_{[t]})$ . However, analog to Nested Benders Decomposition, early interruption of the algorithm yields an upper bound on the optimal value whose difference to the optimal value can be estimated by a multiple of  $\rho$ .

**Proposition 4.3.** *Assume that the aggregation function `AggregatePoints()` that is called by Algorithm 4.4 is modified in a way that the convex hull of the set of points to be aggregated is maintained, cf. Remark 4.2. Denote by  $v$  the optimal value of (1.10) and*



#### 4. Decomposition with Recombining Scenario Trees

---

**Algorithm 4.4:** Nested Column Generation for recombining scenario trees with aggregation of dual solutions

---

```

1  $D_{\text{pt}}(\lambda_{[t]}) \leftarrow \emptyset, D_{\text{ray}}(\lambda_{[t]}) \leftarrow \emptyset, \lambda_{[t]} \in \Lambda_t, t \in [2 : T];$ 
2 for phase =  $I$  to  $II$  do
3    $t \leftarrow 1; \rho \leftarrow 1;$ 
4   loop
5     foreach  $\lambda_{[t]} \in \Lambda_t$  do
6       if  $t = 1$  then
7         ProcessRestrictedMasterFirst(phase);
8         alloptimal  $\leftarrow$  TRUE; aggregation  $\leftarrow$  FALSE;
9          $Y(\lambda_{[\tau]}) \leftarrow \emptyset, \lambda_{[\tau]} \in \Lambda_\tau, \tau \in [2 : T];$ 
10      else if  $t < T$  then
11        aggregation  $\leftarrow$  aggregation  $\vee$  AggregatePoints( $Y(\lambda_{[t]}), \zeta(\lambda_{[t]}), \rho$ );
12        (unbounded, optimal)  $\leftarrow$  ProcessPricingMiddle(phase,  $t, \lambda_{[t]}$ );
13        alloptimal  $\leftarrow$  alloptimal  $\wedge$  optimal;
14      else
15        aggregation  $\leftarrow$  aggregation  $\vee$  AggregatePoints( $Y(\lambda_{[t]}), \zeta(\lambda_{[t]}), \rho$ );
16        (unbounded, optimal)  $\leftarrow$  ProcessPricingLast(phase,  $\lambda_{[t]}$ );
17        alloptimal  $\leftarrow$  alloptimal  $\wedge$  optimal;
18      end
19      if unbounded then break;
20    end
21    if unbounded then  $t \leftarrow 1;$ 
22    else if  $t = T$  and alloptimal then
23      if aggregation then  $\rho \leftarrow \rho/2; t \leftarrow 1;$ 
24      else break;
25    else  $t \leftarrow (t \bmod T) + 1;$ 
26  end;
27  if phase =  $I$  then
28    if optimal value of modified first-stage master problem (4.18) is nonzero then
29      STOP: (1.9) is infeasible;
30    end
31    for  $t = T$  to 2 do foreach  $\lambda_{[t]} \in \Lambda_t$  do
32      remove all columns  $(v, \nu) \in D_{\text{pt}}(\lambda_{[t]})$  with nonzero  $\nu$  and all columns
33       $(w, \omega) \in D_{\text{ray}}(\lambda_{[t]})$  with nonzero  $\omega;$ 
34      foreach remaining  $(x_{t-1}, \chi) \in D_{\text{pt}}(\lambda_{[t]}) \cup D_{\text{ray}}(\lambda_{[t]})$  do update  $\chi$  to
35      value of master problem (4.12) (if  $t < T$ ) or last-stage cost-to-go function
36      (1.10c) (if  $t = T$ ) w.r.t.  $x_{t-1};$ 
37    end
38  else (1.9) has been solved to optimality;
39 end

```

---

### 4.3. Nested Column Gen. for Multistage Stochastic Linear Prog. with Recomb. Scenarios

(4.21) in terms of dynamic programming as

$$\max \left\{ \mathbb{E} \left[ \mathcal{R}_2(\kappa_2, \xi_{[2]}) \right] : \kappa_2(\xi) \in \mathbb{R}^{m_1}, \kappa_2 \in \mathcal{F}_2, b_1(\xi_1) - \mathbb{E}[\kappa_2] \in X_1^* \right\}, \quad (4.22a)$$

with dual cost-to-go functions

$$\mathcal{R}_t(\kappa_t, \bar{\xi}_{[t]}) := \max \left\{ \begin{array}{l} \langle h_t(\xi_t), \mu_t \rangle + \mathbb{E} \left[ \mathcal{R}_{t+1}(\kappa_{t+1}, \xi_{[t+1]}) | \xi_{[t]} = \bar{\xi}_{[t]} \right] : \\ \mu_t \in \mathbb{R}^{d_t}, \kappa_{t+1}(\xi) \in \mathbb{R}^{m_t}, \kappa_{t+1} \in \mathcal{F}_{t+1}, \\ b_t(\xi_t) - \mu_t^\top A_{t,0}(\xi_t) + \mathbb{E}[\kappa_{t+1}(\xi) | \xi_{[t]} = \bar{\xi}_{[t]}] \in X_t^*, \\ -\mu_t^\top A_{t,1}(\xi_t) = \kappa_t \end{array} \right\} \quad (4.22b)$$

for  $t \in [2 : T - 1]$ , and

$$\mathcal{R}_T(\kappa_T, \bar{\xi}_{[T]}) := \max \left\{ \langle h_T(\xi_T), \mu_T \rangle : b_T(\xi_T) - \mu_T^\top A_{T,0}(\xi_T) \in X_T^*, -\mu_T^\top A_{T,1}(\xi_T) = \kappa_T \right\}. \quad (4.22c)$$

The additional variables  $\kappa_t$  have been introduced to visualize the correspondence of the dual cost-to-go functions to the dual of the pricing problems (4.17) and (4.16). That is, for given duals  $(\kappa_t, \sigma_t) \in Y(\lambda_{[t]})$ ,  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T]$ , the pricing problems (4.17) and (4.16) can equivalently be formulated as

$$\max \left\{ \begin{array}{l} \langle h_t(\lambda_t), \mu_t \rangle + \mathbb{E} \left[ \bar{\mathcal{R}}_{t+1}(\kappa_{t+1}(\bar{\lambda}_{t+1}(\xi_{[t+1]})), \bar{\lambda}_{t+1}(\xi_{[t+1]})) | \xi_{[t]} = \lambda_{[t]} \right] : \\ \mu_t \in \mathbb{R}^{d_t}, \kappa_{t+1}(\cdot) \in \mathbb{R}^{m_t}, \\ b_t(\lambda_t) - \mu_t^\top A_{t,0}(\lambda_t) + \mathbb{E}[\kappa_{t+1}(\bar{\lambda}_{t+1}(\xi_{[t+1]})) | \xi_{[t]} = \lambda_{[t]}] \in X_t^*, \\ -\mu_t^\top A_{t,1}(\lambda_t) = \kappa_t, \end{array} \right\} \quad (4.23)$$

where for  $\lambda_{[t+1]} \in \Lambda_{t+1}$ ,

$$\bar{\mathcal{R}}_{t+1}(\kappa, \lambda_{[t+1]}) := \begin{cases} -\infty, & \text{if } \exists (w, \omega) \in D_{\text{ray}}(\lambda_{[t+1]}) : \langle \kappa, w \rangle > \omega, \\ \min_{(v, \nu) \in D_{\text{pt}}(\lambda_{[t+1]})} (\nu - \langle \kappa, v \rangle), & \text{otherwise,} \end{cases}$$

and

$$\max \left\{ \langle h_T(\lambda_T), \mu_T \rangle : \mu_T \in \mathbb{R}^{d_T}, b_T(\lambda_T) - \mu_T^\top A_{T,0}(\lambda_T) \in X_T^*, -\mu_T^\top A_{T,1}(\lambda_T) = \kappa_T \right\}. \quad (4.24)$$

Note, that the dual cost-to-go function (4.22c) is equal to the pricing problem (4.24). Thus, approximating (4.22c) by supporting hyperplanes corresponds to computing extreme points and rays of its dual. Since  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]})$  is computed from extreme points and rays of (4.24), we have that the dual of  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]})$ , see also (4.15), corresponds to a hyperplane approximation of  $\mathcal{R}_T(\cdot, \lambda_{[T]})$ . Hence,  $\bar{\mathcal{Q}}_T(\cdot, \lambda_{[T]}) \geq \mathcal{R}_T(\cdot, \lambda_{[T]})$ . Analogously, it follows from (4.22b) and (4.23), that  $\bar{\mathcal{Q}}_t(\cdot, \lambda_{[t]})$  corresponds (in its dual) to a hyperplane approximation of  $\mathcal{R}_t(\cdot, \lambda_{[t]})$ , and thus  $\bar{\mathcal{Q}}_t(\cdot, \lambda_{[t]}) \geq \mathcal{R}_t(\cdot, \lambda_{[t]})$ .

Thus, the claim follows as in the proofs of Proposition 4.1 and Corollary 3.2.4 in Küchler [2009]. The assumption on boundedness of the cost-to-go function  $\mathcal{Q}_t(\cdot, \lambda_{[t]})$  from

#### 4. Decomposition with Recombining Scenario Trees

below is here replaced by requiring that the pricing problems are feasible. This is ensured by phase I of Algorithm 4.4. The assumption on feasibility of the cost-to-go function  $\underline{Q}_t(\cdot, \lambda_{[t]})$  from above is equivalent to assuming boundedness of the pricing problems. Since we assumed that the point aggregation algorithm maintains the convex hull of the sets  $Y_t(\lambda_{[t]})$ , we can argue as in Remark 4.2 and Corollary 3.2.4 in Küchler [2009] that all necessary columns for the sets  $D_{\text{ray}}(\lambda_{[t]})$  are created.  $\square$

### 4.4. Combining Nested Benders Decomposition and Nested Column Generation

As shown in the previous sections, the Nested Benders Decomposition and Nested Column Generation algorithms compute a lower and upper bound, respectively, on the optimal value of a multistage stochastic linear program (with or without recombining scenarios). For both cases, we have discussed that aggregation of primal or dual solution points allows to dynamically recombine scenarios during the solving process and we have shown that the distance between computed bound and true optimal value induced by aggregating similar solution points is proportional to the aggregation parameter  $\rho$ . In both cases, a computational tractable criteria that allows to interrupt the solving process for some  $\rho > 0$  as soon as the distance between computed bound and optimal value is below a specific tolerance was missing.

Clearly, combining both algorithms allows to estimate the distance of any bound to the true optimal value by the distance between both bounds. Thus, in the following we discuss a combination of Algorithms 4.2 and 4.4, where lower and upper bounding approximations of the cost-to-go functions are computed simultaneously. Thereby, master problems (4.2) and (4.3) from Nested Benders Decomposition and master problems (4.14) and (4.12) from Nested Column Generation yield both a “local” (i.e., related to subtree) and a “global” (i.e., related to the whole tree) stopping criteria. That is, let  $x_{t-1} \in Z(\lambda_{[t]})$ ,  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T]$ , be an approximate solution point which corresponding lower bound  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$  on the cost-to-go function in  $\lambda_{[t]}$  has been computed. The purpose of evaluating the master problem (4.3) for  $x_{t-1}$  is to check whether  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$  equals (or is close to) the true value  $Q_t(x_{t-1}, \lambda_{[t]})$ . However, evaluation of master problem (4.3) generates new approximate solution points  $x_t$  that demand for evaluations of master problems at stage  $t+1$ , and so on. Thus, many master problems may have to be evaluated only to check whether  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$  is already accurate enough. With Nested Column Generation, we now have a tool at hand, that allows to estimate the gap between lower bound  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$  and  $Q_t(x_{t-1}, \lambda_{[t]})$  via a single evaluation of the master problem (4.12), since its value yields an upper bound on  $Q_t(x_{t-1}, \lambda_{[t]})$ . Further, applied to the first time stage, we can estimate the approximation error of the master problem (4.2) by additionally solving the master problem (4.14). Thus, the algorithm can be stopped when the gap between the value of these two master problems is small enough.

An analogous discussion is valid for estimating the quality of the upper bounds  $\overline{Q}_t(x_{t-1}, \lambda_{[t]})$  via the master problems from Nested Benders Decomposition. For a given  $(\mu, \sigma) \in Y(\lambda_{[t]})$ , the pricing problem (4.17) is solved, which utilizes the approximations

$\bar{\mathcal{Q}}_{t+1}(\cdot, \lambda_{[t+1]})$ . If the pricing problem was not unbounded, then an indicator whether the approximations  $\bar{\mathcal{Q}}_{t+1}(\cdot, \lambda_{[t+1]})$  can still be improved by passing dual solutions forward to the pricing problems on the next time stage is obtained by replacing the upper bounding approximations  $\bar{\mathcal{Q}}_{t+1}(\cdot, \lambda_{[t+1]})$  by the lower bounding approximations  $\underline{\mathcal{Q}}_{t+1}(\cdot, \lambda_{[t+1]})$  in (4.17). That is, if the value of

$$\min \left\{ \begin{array}{l} \langle b_t(\lambda_t), x_t \rangle - \langle \mu, x_{t-1} \rangle + \mathbb{E} \left[ \underline{\mathcal{Q}}_{t+1}(x_t, \bar{\lambda}_{t+1}(\xi_{[t+1]})) \mid \xi_{[t]} = \lambda_{[t]} \right] : \\ x_t \in X_t, A_{t,0}(\lambda_t)x_t + A_{t,1}(\lambda_t)x_{t-1} = h_t(\lambda_t) \end{array} \right\} \quad (4.25)$$

is close to the one of (4.17), then passing dual solutions to the next stage may be omitted.

An algorithm that combines Nested Benders Decomposition with Nested Column Generation for multistage stochastic linear programs with recombining scenarios is presented in Algorithm 4.5. We write  $a \gg b$  if  $a - b > \varepsilon$  for a given tolerance  $\varepsilon$ . Propositions 4.1 and 4.3 ensure that both  $\underline{v}$  and  $\bar{v}$  converge to the optimal value  $v$  with decreasing  $\rho$  and, thus, the algorithm terminates with either recognizing infeasibility of (1.9) (if phase I initialization of Nested Column Generation terminates with a minimal infeasibility  $> 0$ ), or recognizing unboundedness of (1.9) (if restricted master problem (4.14) gives a value  $\bar{v} = -\infty$ ), or with computing a first stage decision which objective value is within  $\bar{v} - \underline{v}$  of the true optimal value.

Before studying numerical performance of Algorithm 4.4, we discuss how a recombining scenario tree can be constructed from a given stochastic process.

## 4.5. Construction of Recombining Scenario Trees

A variety of approaches for the generation of scenario trees have been developed, relying on different principles, cf. Section 1.3.2. The *forward tree construction* of Heitsch and Römisch [2009b] generates a scenario tree out of a given stochastic process  $\zeta$  on  $\Omega$  by iteratively approximating the conditional distributions

$$P \left[ \zeta_t \in \cdot \mid \zeta_{[t-1]} \in C_{t-1}^i \right], \quad (4.26)$$

given a finite partition  $\{C_{t-1}^i\}_{i \in [n_{t-1}]}$  of  $\text{supp } P[\zeta_{[t-1]} \in \cdot]$ <sup>4</sup>. For a fixed  $i \in [n_{t-1}]$ , this is done by selecting  $n_t \in \mathbb{N}$  points  $\{c_t^j\}_{j \in [n_t]} \subseteq \text{supp } P[\zeta_t \in \cdot \mid \zeta_{[t-1]} \in C_{t-1}^i]$  such that

$$\mathbb{E} \left[ d(\zeta_t, \pi_t(\zeta_t)) \mid \zeta_{[t-1]} \in C_{t-1}^i \right]$$

is small, where  $d(\cdot, \cdot)$  is an appropriate metric on  $\mathbb{R}^{s_t}$  and  $\pi_t : \mathbb{R}^{s_t} \rightarrow \{c_t^j\}_{j \in [n_t]}$  is a ( $d(\cdot, \cdot)$ -nearest neighbor) projection. The mapping  $\pi_t$  defines a partition  $\{\pi_t^{-1}(c_t^j)\}_{j \in [n_t]}$  of  $\text{supp } P[\zeta_t \in \cdot \mid \zeta_{[t-1]} \in C_{t-1}^i]$ . We then define  $C_t^j := C_{t-1}^i \times \pi_t^{-1}(c_t^j)$ ,  $j \in [n_t]$ .

Assuming a scenario tree  $\xi$  has been constructed for the first  $t - 1$  stages, where the nodes at stage  $t - 1$  correspond to the partition  $\{C_{t-1}^i\}_{i \in [n_{t-1}]}$ , the  $n_t$  successors of a

<sup>4</sup> $\text{supp } f(\cdot) := \{x : f(x) \neq 0\}$  denotes the *support* of a function  $f(\cdot)$

#### 4. Decomposition with Recombining Scenario Trees

---

**Algorithm 4.5:** Combined Nested Benders Decomposition and Nested Column Generation for recombining scenario trees

---

```

 $C_{\text{feas}}(\lambda_{[t]}) \leftarrow \emptyset, C_{\text{opt}}(\lambda_{[t]}) \leftarrow \{(L, 0)\}, D_{\text{pt}}(\lambda_{[t]}) \leftarrow \emptyset, D_{\text{ray}}(\lambda_{[t]}) \leftarrow \emptyset, \lambda_{[t]} \in \Lambda_t,$ 
 $t \in [2 : T];$ 
initialize Nested Column Generation as in phase I of Algorithm 4.4;
 $t \leftarrow 1;$ 
 $\rho \leftarrow 1;$ 
loop
  foreach  $\lambda_{[t]} \in \Lambda_t$  do
    if  $t = 1$  then
       $(\underline{v}, \bar{v}) \leftarrow \text{ProcessFirst}();$ 
       $\text{alloptimal} \leftarrow \text{TRUE};$ 
       $Y(\lambda_{[\tau]}) \leftarrow \emptyset, Z(\lambda_{[\tau]}) \leftarrow \emptyset, \lambda_{[\tau]} \in \Lambda_\tau, \tau \in [2 : T];$ 
    else if  $t < T$  then
       $\text{AggregatePoints}(Y(\lambda_{[t]}), \zeta(\lambda_{[t]}), \rho);$ 
       $\text{AggregatePoints}(Z(\lambda_{[t]}), \chi(\lambda_{[t]}), \rho);$ 
       $(\text{unbounded}, \text{infeasible}, \text{optimal}) \leftarrow \text{ProcessMiddle}(t, \lambda_{[t]});$ 
       $\text{alloptimal} \leftarrow \text{alloptimal} \wedge \text{optimal};$ 
    else
       $\text{AggregatePoints}(Y(\lambda_{[t]}), \zeta(\lambda_{[t]}), \rho);$ 
       $\text{AggregatePoints}(Z(\lambda_{[t]}), \chi(\lambda_{[t]}), \rho);$ 
       $(\text{unbounded}, \text{infeasible}, \text{optimal}) \leftarrow \text{ProcessLast}(\lambda_{[t]});$ 
       $\text{alloptimal} \leftarrow \text{alloptimal} \wedge \text{optimal};$ 
    end
    if  $\text{unbounded} \vee \text{infeasible}$  then break;
  end
  if  $\text{unbounded} \vee \text{infeasible}$  then  $t \leftarrow 1;$ 
  else if  $t = T$  and  $\text{alloptimal}$  then
    if  $\underline{v} \ll \bar{v}$  then
       $\rho \leftarrow \rho/2;$ 
       $t \leftarrow 1;$ 
    else
      STOP: (1.9) has been solved accurately enough;
    end
  else  $t \leftarrow (t \bmod T) + 1;$ 
end;

```

---

node  $\xi_{[t-1]}^i$  associated with  $C_{t-1}^i$  are defined by  $\xi_{[t]}^{i,j} := (\xi_1^i, \dots, \xi_{t-1}^i, c_t^j)$  and are associated with  $C_t^j$ . The corresponding distribution is defined by the conditional probabilities

$$P \left[ \xi_t = c_t^j \mid \xi_{[t-1]} = \xi_{[t-1]}^i \right] := P \left[ \pi_t(\zeta_t) = c_t^j \mid \zeta_{[t-1]} \in C_{t-1}^i \right].$$

---

**Function** ProcessFirst processes first stage master problems within Algorithm 4.5
 

---

**input** : cost-to-go function approximations  $\underline{Q}_2(\cdot, \lambda_{[2]})$  and  $\overline{Q}_2(\cdot, \lambda_{[2]})$ ,  $\lambda_{[2]} \in \Lambda_2$   
 solve first stage master problems (4.2) and (4.14);  
 let  $\underline{v}$  be the value of (4.2) and let  $\overline{v}$  be the value of (4.14);  
**if**  $\overline{v} = -\infty$  **then STOP**: (1.9) is unbounded;  
**if**  $\underline{v} \ll \overline{v}$  **then**  
     **foreach**  $\lambda_{[2]} \in \Lambda_2$  **do**  
          $Y(\lambda_{[2]}) \leftarrow \{(\mu, \sigma)\}$  for dual solution  $(\mu, \sigma)$  corresponding to  $\overline{Q}_2(x_1, \lambda_{[2]})$  in  
         (4.14);  
          $Z(\lambda_{[2]}) \leftarrow \{x_1\}$  for primal solution  $x_1$  of (4.2);  
     **end**  
**end**  
**return**  $(\underline{v}, \overline{v})$ ;

---

In order to approximate a Markov process by a recombining tree, Bally et al. [2005] approximate the marginal distributions  $P[\zeta_t \in \cdot]$  instead of the conditional distributions (4.26). That is, the points  $c_t^j$  are selected independently of  $C_{t-1}^i$ , but such that  $\mathbb{E}[d(\xi_t - \pi_t(\xi_t))]$  is small, where  $\pi_t$  is again a projection on  $\{c_t^j\}_{j \in [n_t]}$ . The process  $\xi$  is then constructed as a Markov process with transition probabilities

$$P[\xi_t = c_t^j \mid \xi_{[t-1]} = \xi_{[t-1]}^i] := P[\pi_t(\zeta_t) = c_t^j \mid \pi_{t-1}(\zeta_{t-1}) = \xi_{t-1}^i].$$

For constructing a recombining tree which approximates a non-Markovian stochastic process, we propose an approach that can be seen as a mixture of the methods in Bally et al. [2005] and Heitsch and Römisch [2009b]. First, a set of time points  $R_0 := 0 < R_1 < \dots < R_{n-1} < R_n := T$  where scenarios are allowed to recombine is defined. Then, in a forward tree construction process, non-recombining trees are constructed for time intervals  $[R_{j-1} + 1 : R_j]$ ,  $j \in [n]$ , and scenarios are recombined at times  $R_j$  by means of assigning the same subtree to several nodes at time  $R_j$ . The particular subtree has to approximate  $\zeta$ 's future distribution that can depend, in the non-Markovian case, on  $\zeta$ 's complete history. Thereby, two nodes at time  $R_j$  obtain the same subtree if  $\zeta$ 's values in these nodes are close during the time interval  $[R_j - \tau : R_j]$ , for some value  $\tau \in [0 : R_j - R_{j-1} - 1]$ . This seems to be reasonable whenever  $P[\zeta_{[t+1:T]} \in \cdot \mid \zeta_t = \bar{\zeta}_t]$  depends continuously on  $\bar{\zeta}_{[t-\tau:t]}$  in some sense for all  $t \in [\tau : T]$ , cf. Theorem 4.4 below. The partitioning of the set of nodes at time  $R_j$  such that all nodes in a subset are assigned the same subtree is done by a clustering algorithm. The number of clusters (and thus the number of subtrees originating at  $R_j$ ) determine the structure of the scenario tree and can either be predefined, or, as proposed by Heitsch and Römisch [2009b], determined within the tree construction procedure to not exceed certain local error levels.

The detailed tree generation algorithm is given in Section 3.3 of Küchler [2009] and omitted here. Küchler [2009] has also shown that the tree generation algorithm is consistent in the sense that the optimal value of the multistage stochastic linear program

#### 4. Decomposition with Recombining Scenario Trees

---

**Function** ProcessMiddle( $t, \lambda_{[t]}$ ) processes  $t$ -stage pricing problem of node  $\lambda_{[t]} \in \Lambda_t$ ,  $t \in [2 : T - 1]$ , within Algorithm 4.5

---

**input**: primal solutions  $x_{t-1} \in Z(\lambda_{[t]})$  and dual solutions  $(\mu, \sigma) \in Y(\lambda_{[t]})$  that demand for evaluation of master and pricing problems corresponding to  $\lambda_{[t]}$

**input**:  $t + 1$ -stage cost-to-go function approximations  $\underline{Q}_{t+1}(\cdot, \lambda_{[t+1]})$  and  $\overline{Q}_{t+1}(\cdot, \lambda_{[t+1]})$ ,  $\lambda_{[t+1]} \in \Lambda_{t+1}$

unbounded  $\leftarrow$  FALSE; infeasible  $\leftarrow$  FALSE; optimal  $\leftarrow$  TRUE;

**foreach** approximate solution  $x_{t-1} \in Z(\lambda_{[t]})$  **do**

- solve master problem (4.3);
- if** (4.3) is infeasible **then**
  - add feasibility cut constructed from dual ray of (4.3) to  $C_{\text{feas}}(\lambda_{[t]})$ ;
  - infeasible  $\leftarrow$  TRUE;
  - break**;
- else if** optimal value of (4.3) is larger than  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$  **then**
  - add supporting hyperplane constructed from dual solution to  $C_{\text{opt}}(\lambda_{[t]})$ ;
  - optimal  $\leftarrow$  FALSE;
- end**
- solve column generation master problem (4.12);
- if** value of (4.12)  $\gg$  value of (4.3) **then**
  - foreach**  $\lambda_{[t+1]} \in \Lambda_{t+1}$  that is a successor of  $\lambda_{[t]}$  **do** add  $x_t$  to  $Z(\lambda_{[t+1]})$  and remember  $\underline{Q}_{t+1}(x_t, \lambda_{[t+1]})$ ;
- end**

**end**

**foreach** dual solution  $(\mu, \sigma) \in Y(\lambda_{[t]})$  **do**

- solve pricing problem (4.17);
- if** pricing problem is unbounded **then**
  - add new column constructed from primal ray to  $D_{\text{ray}}(\lambda_{[t]})$ ;
  - unbounded  $\leftarrow$  TRUE;
  - break**;
- else if** optimal value of pricing problem is smaller than  $\sigma$  **then**
  - add new column constructed from solution of pricing problem to  $D_{\text{pt}}(\lambda_{[t]})$ ;
  - optimal  $\leftarrow$  FALSE;
- end**
- solve lower bounding problem (4.25);
- if** value of (4.25)  $\ll$  value of (4.17) **then**
  - foreach**  $\lambda_{[t+1]} \in \Lambda_{t+1}$  that is a successor of  $\lambda_{[t]}$  **do** add dual solution  $(\mu, \sigma)$  corresponding to  $\overline{Q}_{t+1}(\cdot, \lambda_{[t+1]})$  in (4.17) to  $Y(\lambda_{[t+1]})$ ;
- end**

**end**

**return** (unbounded, infeasible, optimal);

---

---

**Function**  $\text{ProcessLast}(\lambda_{[T]})$  processes  $T$ -stage pricing problem of node  $\lambda_{[T]} \in \Lambda_T$  within Algorithm 4.5

---

**input** : primal solutions  $x_{T-1} \in Z(\lambda_{[T]})$  and dual solutions  $(\mu, \sigma) \in Y(\lambda_{[T]})$  that demand for an evaluation of the cost-to-go function and pricing problem corresponding to  $\lambda_{[T]}$

unbounded  $\leftarrow$  FALSE;  
infeasible  $\leftarrow$  FALSE;  
optimal  $\leftarrow$  TRUE;

**foreach** *approximate solution*  $x_{T-1} \in Z(\lambda_{[T]})$  **do**

evaluate cost-to-go function  $(Q_T(x_{T-1}, \lambda_{[T]}))$  by solving (4.5);

**if** (4.5) *is infeasible* **then**

add feasibility cut constructed from dual ray of (4.5) to  $C_{\text{feas}}(\lambda_{[T]})$ ;  
infeasible  $\leftarrow$  TRUE;  
**break**;

**end**

**if** *optimal value of (4.5) is larger than*  $Q_T(x_{T-1}, \lambda_{[T]})$  **then**

add supporting hyperplane constructed from dual solution to  $C_{\text{opt}}(\lambda_{[T]})$ ;  
optimal  $\leftarrow$  FALSE;

**end**

**end**

**foreach** *dual solution*  $(\mu, \sigma) \in Y(\lambda_{[T]})$  **do**

solve pricing problem (4.16);

**if** *pricing problem is unbounded* **then**

add new column constructed from primal ray to  $D_{\text{ray}}(\lambda_{[T]})$ ;  
unbounded  $\leftarrow$  TRUE;  
**break**;

**end**

**if** *optimal value of pricing problem is smaller than*  $\sigma$  **then**

add new column constructed from solution of pricing problem to  $D_{\text{pt}}(\lambda_{[T]})$ ;  
optimal  $\leftarrow$  FALSE;

**end**

**end**

**return** (unbounded, infeasible, optimal);

---

with approximated scenario tree  $\xi$  converges to the optimal value for using the original process  $\zeta$  when the discretization errors controlled by the tree generation algorithm become uniformly small, if the original process  $\zeta$  possesses certain continuity properties:

**Theorem 4.4** (Corollary 3.3.5 in Küchler [2009]). *Assume that  $\zeta$  fulfills the following conditions:*

- i) The values of  $\|\zeta\|$  are bounded by a constant.*

## 4. Decomposition with Recombining Scenario Trees

ii)  $\zeta$  can be written as a recursion

$$\zeta_{t+1} = g(\zeta_{[t-\tau:t]}, \varepsilon_{t+1}), \quad (4.27)$$

with independent random vectors  $\varepsilon_t$ ,  $t \in [T]$ , and a mapping  $g$  that satisfies

$$g(\bar{\zeta}_{[t-\tau:t]}, \bar{\varepsilon}) - g(\tilde{\zeta}_{[t-\tau:t]}, \bar{\varepsilon}) \leq L \sum_{s=t-\tau}^t \|\bar{\zeta}_s - \tilde{\zeta}_s\|,$$

for all  $\bar{\zeta}_{[t-\tau:t]}, \tilde{\zeta}_{[t-\tau:t]} \in \text{supp } P[\zeta_{[t-\tau:t]} \in \cdot]$  and  $\bar{\varepsilon} \in \text{supp } P[\varepsilon_{t+1} \in \cdot]$ . Further,  $g$  is such that the sets  $\{g(\zeta_{[t-\tau:t]}, \varepsilon) : \varepsilon \in \text{supp } P[\varepsilon_{t+1} \in \cdot]\}$  coincide for all values  $\zeta_{[t-\tau:t]} \in \text{supp } P[\zeta_{[t-\tau:t]} \in \cdot]$ ,  $t \in [T]$ .

iii) The measure  $P[\zeta_t \in \cdot]$  is absolutely continuous with respect to the Lebesgue measure on  $\mathbb{R}^{s_t}$ ,  $t \in [2 : T]$ .

Assume that the multistage stochastic linear program (1.9) fulfills the following condition:

- i)  $Q_t(x_{t-1}, \zeta_{[t]}) < \infty$  for all  $x_{t-1} \in \mathbb{R}^{m_{t-1}}$ ,  $t \in [2 : T]$  (complete recourse).
- ii)  $X_t$  is bounded,  $t \in [T]$ .
- iii) The recourse matrices  $A_{t,0}(\cdot)$  are deterministic (i.e., do not depend on  $\zeta_t$ ).

Let  $\xi$  be a recombining scenario tree constructed by Algorithms 3.6 and 3.7 in Küchler [2009] and denote by  $v(\zeta)$  and  $v(\xi)$  the optimal value of (1.9) w.r.t.  $\zeta$  and  $\xi$ , respectively. Then there exists a constant  $L' \geq 0$  such that for every  $\varepsilon > 0$  the following property holds: If the approximation errors for the conditional distributions and the recombination in the tree construction algorithm are smaller than  $\varepsilon$ , then

$$|v(\zeta) - v(\xi)| \leq L'\varepsilon.$$

**Remark 4.5.** The *short-term memory* property (4.27) is a generalization of the Markov property. It is fulfilled by a variety of processes of practical interest and can be verified easily if  $\zeta$ 's distribution is given as a time series model, in general. It is easy to see that, by augmentation of its state space, every discrete-time process  $\zeta$  with short-term memory may be transformed into a Markov process.

## 4.6. Numerical Experiment

### 4.6.1. Notes on Implementation

The decomposition algorithms from Sections 4.2–4.4 have been implemented in the C++ package RECOMBTREE [Vigerske, 2011]. As linear programming solver, currently the open-source code COIN-OR CLP<sup>5</sup> or the commercial code CPLEX<sup>6</sup> can be used, both

<sup>5</sup><https://projects.coin-or.org/Clp>

<sup>6</sup><http://www.cplex.com>

are interfaced via the COIN-OR Open Solver Interface OSI<sup>7</sup>. Further, RECOMBTREE interfaces GAMS<sup>8</sup> to obtain the problem. In a GAMS model, stochastic data ( $\xi$ ) has to be declared as variable by the user, which is then substituted by scenario values in the solver. A scenario tree in XML format is read into RECOMBTREE as separate file.

Even though the algorithm in RECOMBTREE is based on Algorithm 4.5, there are main differences in the order in which subproblems are processed (sequencing protocol) or in the decision when the aggregation parameter  $\rho$  is reduced. As sequencing protocol, we implemented a fast-forward-fast-backward procedure to traverse the tree [Birge, 1985, Gassmann, 1990]. In *forward mode*, master problems are solved and solution sets  $Y(\lambda_{[t]})$  and  $Z(\lambda_{[t]})$ ,  $\lambda_{[t]} \in \Lambda_t$ , are generated. The traversing direction is changed into *backward mode* if either the time horizon has been reached ( $t = T$ ), or a master problem is infeasible or a pricing problem is unbounded. In the latter case, feasibility cuts or columns corresponding to primal rays in the pricing problem are generated and the algorithm resolves the problems at the previous time stage to update the solution sets  $Z(\lambda_{[t]})$  or  $Y(\lambda_{[t]})$ . Otherwise, if the end of the time horizon has been reached, master problems are updated by passing optimality cuts and columns backwards, thereby also resolving master problems if approximations are improved.

For a fixed  $\rho$ , fast-forward-fast-backward processing of the decomposed problem is repeated as long as new cuts or columns are generated, the gap between lower and upper bound is still too large, and the improvement in a lower or upper bound is at least a fixed ratio (10% in default settings) of the best improvement in this bound for the current value of  $\rho$ . If for both bounds, approximations are not improved sufficiently enough,  $\rho$  is reduced geometrically (multiplied by 0.3 in default settings). If  $\rho$  reaches a fixed final value ( $10^{-4}$  in default settings) and no new cuts and columns are created, the algorithm is also stopped. The latter is especially important if one disables the lower or upper bounding approximations and thus no gap estimate is available.

So far, advanced warm starting algorithms as proposed in Haugland and Wallace [1988], Gassmann [1990], and Gassmann and Wallace [1996] or stabilization schemes as in the Regularized Decomposition method [Ruszczynski, 1986] or in stabilized Column Generation [du Merle et al., 1999, Rousseau et al., 2007, Lübbecke, 2010] have not been implemented. Especially the missing stabilization in the Nested Column Generation implementation may be a reason why it does not perform as well as the Nested Benders Decomposition, see Section 4.6.3.

For the Nested Benders Decomposition scheme, a special initialization scheme has been implemented: Consider the expected value problem of (1.9) obtained by replacing the scenario tree  $\xi$  by the single scenario  $\bar{\xi} := \mathbb{E}[\xi]$ . This deterministic linear program is often easy to solve, and, if feasible, then its solution  $\bar{x}$  can serve as a (very rough) estimate on the solution of the stochastic program. RECOMBTREE utilizes  $\bar{x}$  to generate an initial set of cuts by solving the master problems (4.3) and (4.5) in  $\bar{x}_t$ . These evaluations also serve as initialization of the scaling vectors  $\chi(\lambda_{[t]})$ , c.f. Section 4.2.1.

During the processing of the master problems, updates in the approximation of a

---

<sup>7</sup><https://projects.coin-or.org/Osi>

<sup>8</sup><http://www.gams.com>

#### 4. Decomposition with Recombining Scenario Trees

cost-to-go function at stage  $t + 1$  often also lead to updates of approximations at stage  $t$ . As a result, cuts or columns that have originally been computed for stage  $t$  may be dominated by later computed cuts or columns. Therefore, RECOMBTREE tries to recognize such redundant cuts or columns and removes them from the master problems.

Finally, the implementation offers the user some freedom on how the problem should be decomposed. If the subproblems corresponding to single time stages are small but there is a long time horizon, then decomposing the problem at every time stage may computationally be disadvantageous. Therefore, RECOMBTREE allows to aggregate successive time stages without intermediate recombination into a single *time period*. The cost-to-go functions corresponding to such time periods are then approximated as a whole. To simplify notation, we presented the algorithm in the previous chapters without this aggregation. For a discussion of the Nested Benders Decomposition w.r.t. time periods instead of time stages, see the earlier publications Küchler and Vigerske [2007] and Küchler [2009]. By default, for a problem with recombining scenario tree, RECOMBTREE only decomposes at time stages where recombination takes place. Further, the user can choose to decompose every  $n$ 'th time stage, or at time stages where the average degree of the scenario tree nodes is above a certain value, or at time stages where the average number of nodes in a subproblem exceeds a certain value. So far, decomposition has to take place for all nodes at a time stage, or none. Schemes where subproblems starting at the same time stage can have different length are not supported, yet.

The recombining scenario tree construction algorithm, c.f. Section 4.5 has been implemented by C. Küchler as a separate program RECTREECON that utilizes SCENRED<sup>9</sup> [Heitsch and Römis, 2009b] for subtree construction. RECTREECON is fed by a set of scenarios and a parameter file and writes a recombining scenario tree in XML format as it can be understood by RECOMBTREE. RECTREECON allows to choose the length  $R_j - R_{j-1}$  of the time periods for which non-recombining scenario trees are constructed and the number of clusters to use when recombining scenarios at stages  $R_j$ ,  $j \in [n]$ .

##### 4.6.2. A Power Scheduling Problem

We consider a model that describes a power generating system consisting of several coal fired thermal units (index set  $U_{\text{th}}$ ), pumped hydro units (index set  $U_{\text{hy}}$ ), and a wind power plant. The objective is to find cost-optimal operation levels of the thermal units and hydro units under uncertain production of electricity from wind.

Denote by  $p_{u,t}$  the operation level of the thermal unit  $u \in U_{\text{th}}$ , by  $l_{u,t}$  the fill level of the water reservoir  $u \in U_{\text{hy}}$ , by  $w_{u,t}$  the operation level of the pump  $u \in U_{\text{hy}}$ , and by  $v_{u,t}$  the operation level of the turbine  $u \in U_{\text{hy}}$ . Deterministic parameters of the problem are operation ranges for the thermal units  $\underline{p}_u < \bar{p}_u$ ,  $u \in U_{\text{th}}$ , the pumps  $\bar{w}_u > 0$ , and the turbines  $\bar{v}_u > 0$ , the maximal operation gradient for thermal units  $\delta_u \in [0, 1]$ ,  $u \in U_{\text{th}}$ , the capacity of the water reservoirs  $\bar{l}_u > 0$ ,  $u \in U_{\text{hy}}$ , the fill levels of the reservoirs at the beginning and end of the considered time horizon ( $l_{u,\text{in}}$  and  $l_{u,\text{end}}$ , respectively,  $u \in U_{\text{hy}}$ ), the efficiency of the pumps  $\eta_u$ ,  $u \in U_{\text{hy}}$ , fuel costs  $b_u$ ,  $u \in U_{\text{th}}$ , energy demand

<sup>9</sup><http://www.gams.com/solvers/solvers.htm#SCENRED>

$d_t$ , and reserve fraction  $r \in [0, 1]$ . As stochastic parameter we consider the wind power production  $\xi_t$ . The complete model has the form

$$\begin{aligned}
\min \quad & \mathbb{E} \left[ \sum_{t=1}^T \sum_{u \in U_{\text{th}}} b_u p_{u,t} \right] \\
\text{s.t.} \quad & l_{u,1} = l_{u,\text{in}} - (w_{u,1} - \eta_u v_{u,1}), \quad l_{u,T} \geq l_{u,\text{end}} && u \in U_{\text{hy}}, & (4.28a) \\
& l_{u,t} = l_{u,t-1} - (w_{u,t} - \eta_u v_{u,t}), && t \in [2 : T], u \in U_{\text{hy}}, & (4.28b) \\
& |p_{u,t} - p_{u,t-1}| \leq \delta_u (\bar{p}_u - \underline{p}_u), && t \in [2 : T], u \in U_{\text{th}}, & (4.28c) \\
& \sum_{u \in U_{\text{th}}} p_{u,t} + \sum_{u \in U_{\text{hy}}} (w_{u,t} - v_{u,t}) + \xi_t \geq d_t, && t \in [1 : T], & (4.28d) \\
& \sum_{u \in U_{\text{th}}} p_{u,t} \leq \sum_{u \in U_{\text{th}}} \bar{p}_u - r d_t, && t \in [1 : T], & (4.28e) \\
& p_{u,t} \in [\underline{p}_u, \bar{p}_u] && t \in [1 : T], u \in U_{\text{th}} \\
& v_{u,t} \in [0, \bar{v}_u], w_{u,t} \in [0, \bar{w}_u], l_{u,t} \in [0, \bar{l}_u] && t \in [1 : T], u \in U_{\text{hy}}
\end{aligned}$$

Constraint (4.28a) models the initial and final fill level of the reservoirs, (4.28b) couples the fill levels of the reservoirs between successive time stages, (4.28c) bounds the change in the operation of the thermal units between successive time stages, (4.28d) ensures that the electricity demand is covered, and (4.28e) is a reserve requirement. Constraints that couple successive time stages are (4.28b) and (4.28c).

Figure 4.2 summarizes the parameter values and shows the demand  $d_t$  for one week. Note, that even though the capacity of the thermal units is sufficient to cover the maximal load ( $\approx 2000$ ), the reserve requirement (4.28e) limits the usable energy from thermal units to 1800. Thus, at times of high load, alternative energy sources like hydro power or wind energy have to be used. Hence, due to (4.28e) and additionally due to the conditions (4.28a) and (4.28c) on the minimal final fill level of the water storage and the power gradient of the thermal units, this model does not possess relatively complete recourse.

For our numerical experiments we considered different time horizons between two days and several months in *hourly discretization*. A stochastic wind speed process  $\zeta$  was modeled by adapting a mean-reverting autoregressive stochastic process<sup>10</sup>,

$$\zeta_t := \max \left\{ 0, \frac{6}{7} (\zeta_{t-1} - \mu_{t-1}) + \mathcal{N}(\mu_t, \sigma_t) \right\},$$

that, in particular, exhibits a short-term memory, cf. Remark 4.5. From the distribution of  $\zeta$  we sampled a set of scenarios and transformed them into wind energy scenarios via an *aggregated power curve* [Nørgård et al., 2004]. Finally, the recombining scenario tree construction algorithm as sketched in Section 4.5 was applied, see also Figure 4.3.

<sup>10</sup>The model of the wind speed process is not derived from real-world data here, but has purposely chosen to be sufficiently customizable in a way that scenarios for arbitrary time horizons could easily be generated. For the computations in Chapter 5, realistic wind speed data will be the basis for scenario tree constructions.

#### 4. Decomposition with Recombining Scenario Trees

$U_{\text{th}}$ thermal units	{1,2,3}	$b_1$ fuel cost coal	21
$U_{\text{hy}}$ hydro units	{1}	$b_2$ fuel cost gas & steam	48
$\bar{v}_1$ capacity hydro turbine	2000	$b_3$ fuel cost gas	154
$\bar{w}_1$ capacity hydro pump	2000	$\underline{p}_1$ minimal power coal	500
$\bar{l}_1$ capacity hydro storage	12000	$\underline{p}_2$ minimal power gas & steam	200
$l_{\text{in}}$ initial storage level	6000	$\underline{p}_3$ minimal power gas	50
$l_{\text{end}}$ minimal final storage level	6000	$\bar{p}_1$ capacity coal	1000
$\eta_1$ pump efficiency	0.75	$\bar{p}_2$ capacity gas & steam	500
$r$ reserve fraction	0.1	$\bar{p}_3$ capacity gas	500
		$\delta_{1,2,3}$ power gradient	0.5

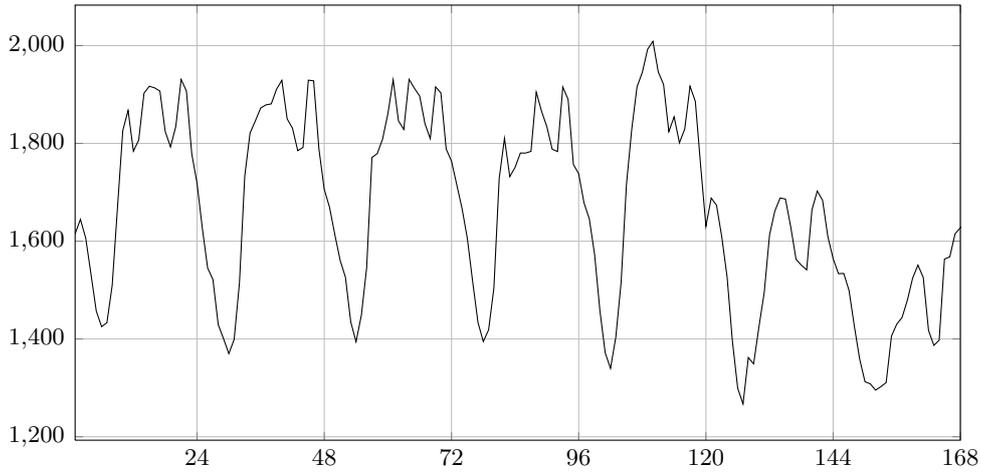


Figure 4.2.: Parameters and demand curve of simple power scheduling model (4.28)

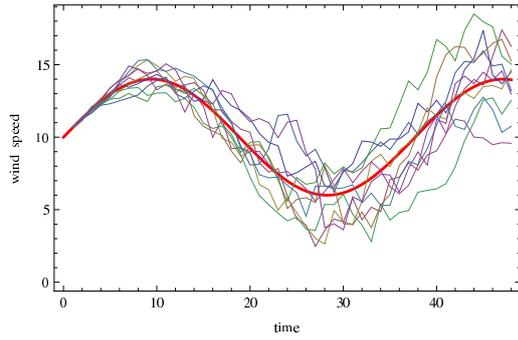
#### 4.6.3. Numerical Results

All computational results presented in this Section have been obtained under openSUSE Linux 11.4 64bit on a Dell PowerEdge M1000e blade with 48GB RAM and two Intel Xeon X5672 CPUs running at 3.20 GHz. RECOMBTREE used CPLEX 12.3.0.0 in single thread mode to solve master problems. RECTREECON used SCENRED 2.1 for scenario tree construction.

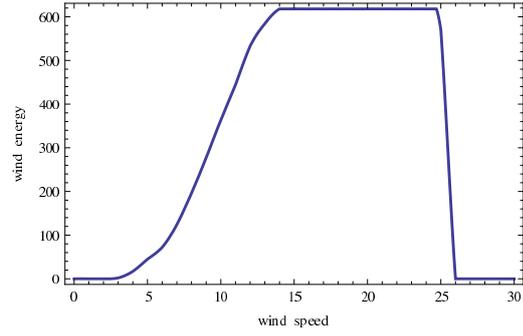
#### Cut Sharing

First, we investigated how the efficiency of the Nested Benders Decomposition improves when it can use cut sharing due to coinciding subtrees, i.e., when relation (4.1) is fulfilled for some nodes at one or several time stages. To this end, we considered time horizons of two, three, and four days, respectively. For each time horizon, several scenario trees were generated by varying SCENREDS parameter `red_percentage` and the number of clusters into which scenarios recombine. In most cases, recombination was applied every 24 hours,

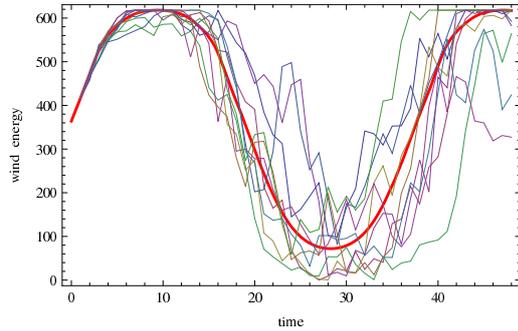
## 4.6. Numerical Experiment



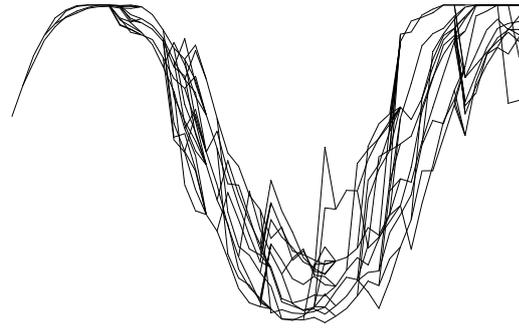
(a) 10 wind speed scenarios sampled from  $\zeta; \mu_t$



(b) Aggregated power curve for transforming wind speed into wind energy.



(c) Wind speed scenarios from (a), transformed into wind energy via (b).



(d) Recombining scenario tree constructed from 1000 wind energy scenarios. Recombination can take place every 6 hours ( $R_1 = 6, R_2 = 12, R_3 = 18, R_4 = 24, R_5 = 30, R_6 = 36$  in Section 4.5) and the number of clusters when recombining was limited to 6. The tree has 436 representative nodes ( $\sum_{t=1}^{48} |\Lambda_t| = 436$ ).

Figure 4.3.: Sample wind speed scenarios, transformation into wind energy, and wind energy scenario tree.

in a few cases also every 12 hours. In the first setting (named ‘with cutsharing’), we run the Nested Benders Decomposition Algorithm 4.2 on the recombining scenario tree, with initial aggregation parameter  $\rho$  set equal to the final  $\rho$  ( $10^{-4}$ ). Thus, the algorithm makes use of coinciding subtrees by sharing the approximation of cost-to-go functions, but aggregation of solution points is reduced to a minimum. In the second setting (named ‘without cutsharing’), we forbid simultaneous approximation of equal cost-to-go functions by replacing the recombining scenario tree by an equivalent non-recombining one and run again the Nested Benders Decomposition algorithm on this much larger scenario tree. Finally, in the setting ‘without decomp.’, we invoked CPLEX directly on the deterministic equivalent of the problem with non-recombining scenario tree. That is, we test the effect of disabling decomposition. Here, CPLEX’s barrier algorithm with subsequent crossover was used in single thread mode.

#### 4. Decomposition with Recombining Scenario Trees

Table 4.1.: Computing times for different time horizons and recombining scenario trees when making use of cutsharing, disabling cutsharing, and disabling decomposition, thus solving the deterministic equivalent directly. Column ‘#nodes’ in the ‘with cutsharing’ section specifies the number of representative nodes in the recombining tree. In the ‘without cutsharing’ section, ‘#nodes’ specifies the number of nodes if recombination of scenarios is not regarded (see also Figures 4.1b and 4.1c). The number of nodes in ‘without decomp.’ equals those in ‘without cutsharing’. Column ‘#subpr’ gives the number of subproblems, if decomposition is applied.

$T$	with cutsharing			without cutsharing			without decomp.
	#nodes	#subpr	time	#nodes	#subpr	time	time
48	151	6	0.1s	12229	43	2.3s	3.4s
48	184	6	0.2s	35081	89	6.3s	14.2s
48	267	10	0.2s	13021	40	2.2s	12.3s
48	292	7	0.8s	350628	601	84.8s	320.3s
48	338	11	0.3s	47924	143	8.5s	58.5s
48	491	19	0.3s	11289	38	2.1s	3.5s
48	528	11	1.2s	431536	473	106.4s	534.7s
48	572	2	0.2s	18527	135	2.2s	20.2s
48	577	20	0.3s	41425	114	7.4s	18.4s
48	662	27	0.2s	10443	38	1.9s	4.9s
48	692	3	0.2s	18370	135	2.3s	10.1s
48	745	3	0.5s	37519	327	4.3s	15.2s
48	811	28	0.4s	34386	94	5.7s	14.1s
48	894	20	1.0s	315486	375	79.6s	278.0s
48	948	4	0.4s	35318	228	4.6s	13.5s
48	1008	5	0.3s	19906	135	2.3s	13.3s
48	1225	8	0.4s	18449	150	2.3s	14.5s
48	1300	31	1.1s	286473	406	67.1s	355.8s
48	1353	6	0.5s	34266	188	4.6s	24.3s
48	1357	3	0.9s	121091	527	17.3s	105.0s
48	1606	8	0.3s	30338	188	4.0s	30.3s
48	1762	4	0.6s	108983	353	18.8s	108.7s
48	2395	6	0.9s	102033	342	16.8s	133.3s
48	2660	10	0.8s	89605	342	16.4s	81.0s
72	682	3	0.8s	149937	1181	13.4s	86.8s
72	844	3	1.0s	302320	2055	30.4s	341.4s
72	919	6	0.7s	138697	1272	14.1s	71.2s
72	1188	6	1.2s	328581	2320	36.4s	285.4s
72	1362	9	0.8s	132622	1226	12.3s	103.8s
72	1718	14	0.7s	111707	1173	9.9s	59.3s
72	1777	9	1.2s	314504	2276	34.0s	277.8s
72	2185	14	1.2s	220240	1847	21.9s	179.6s
96	795	5	1.1s	1241310	19585	120.7s	1283.7s
96	1202	9	1.4s	1569222	22089	152.7s	4056.4s
96	1810	18	2.3s	1386117	22947	183.9s	1682.3s
96	2436	26	2.0s	1148159	18047	172.3s	1449.0s
96	2980	27	3.4s	2667441	35973	411.7s	5638.5s
96	2980	27	3.4s	3900623	43698	1245.9s	10009.7s

Table 4.1 summarizes the results of this experiment. It can be seen, that recombining scenarios avoids an exponential growth of the number of (representative) nodes and significantly reduces the running time of the Nested Benders Decomposition Algorithm. Further, decomposition yields considerably smaller solution times compared to solving the deterministic equivalence as a whole.

### Decomposition Algorithms

Next, we studied the use of lower and/or upper bounds and the aggregation of approximate primal and dual solutions. Table 4.2 shows the results for running all decomposition algorithms for time horizons of two, three, and four days and various scenario trees, see also Figure 4.4 for an illustration. We imposed a timelimit of 3 hours.

The columns entitled with ‘rough’ report the time spent for the *rough phase only*, i.e., the time while the aggregation parameter was at  $\rho = 0.1$ . For the Nested Benders Decomposition, the rough phase turns out to be very fast and its running time appears to depend linearly on the number of nodes of the scenario trees, see the right plots in Figure 4.4. In most cases, the rough phase provides solutions that are close to optimal solutions and that do not significantly change during the remaining optimization procedure, see column ‘diff’ in Table 4.2. Thus, in practice, it may be reasonable to reduce the algorithm’s running time by putting the final  $\rho$  not too small.

The column ‘time’ in Table 4.2 reports the complete solution time, i.e., the time spent to decrease the aggregation parameter  $\rho$  to  $10^{-4}$ . With diminishing  $\rho$ , the non-recombining nature of the decision process, which is due to time-coupling constraints, leads to a considerable growth of the number of approximate solution points for which master problems have to be evaluated, and thus largely increases the running times. Note further, that for similar number of nodes, solutions times increase when the number of subproblems decreases and vice versa. The reason is likely that with a decreasing number of subproblems the average size of the subproblems increases, which increases the time to solve a single subproblem.

Table 4.2.: Performance for different time horizons when using only lower bounds, only upper bounds, or both bounds. Columns  $T$ , ‘#nodes’, and ‘#sub’ show the time horizon, the number of (representative) nodes in the recombining scenario tree, and the number of subproblems in the decomposed problems, respectively. Columns entitled ‘all’ show the overall solution time. Columns entitled ‘rough’ show the time for the *rough phase* only, i.e., the time until  $\rho$  is decreased the first time. Column ‘diff’ shows the relative difference between the bound at the end of the rough phase and the final one (i.e., when either the problem was solved or the time limit was hit). Column ‘gap’ shows the gap between lower and upper bound at the end of the rough phase. Column ‘init’ shows the time to initialize the Nested Column Generation Decomposition. The line marked in bold font corresponds to the tree used in Figure 4.5.

#### 4. Decomposition with Recombining Scenario Trees

instance		only lower bounds			only upper bounds				both bounds			
<i>T</i>	#nodes	#sub	rough	all	diff	init	rough	all	diff	rough	gap	all
48	572	2	0.1s	0.3s	0.01%	0.1s	0.5s	0.8s	0.05%	0.6s	0.04%	1.0s
48	692	3	0.1s	0.2s	0.01%	0.1s	0.4s	0.8s	1.06%	0.5s	0.02%	0.7s
48	745	3	0.2s	0.4s	0.04%	1.0s	4.2s	10.3s	0.09%	1.2s	0.04%	1.6s
48	948	4	0.3s	0.5s	0.00%	0.4s	0.6s	1.5s	0.89%	0.7s	0.01%	1.1s
48	1008	5	0.2s	0.4s	0.04%	0.3s	0.5s	1.0s	1.17%	0.7s	0.04%	1.0s
48	1225	8	0.3s	0.4s	0.01%	0.3s	0.7s	1.2s	0.12%	0.8s	0.01%	1.0s
48	1353	6	0.4s	0.6s	0.02%	0.4s	0.8s	1.6s	0.59%	0.9s	0.03%	1.5s
48	1357	3	0.4s	0.9s	0.01%	0.8s	2.3s	4.3s	0.08%	2.6s	0.02%	3.9s
48	1606	8	0.4s	0.5s	0.01%	0.4s	0.8s	1.4s	0.05%	0.9s	0.03%	1.6s
48	1762	4	0.6s	0.9s	0.01%	0.6s	1.3s	3.0s	0.16%	1.4s	0.06%	2.5s
48	2395	6	0.5s	0.9s	0.01%	0.6s	1.6s	3.4s	0.29%	1.9s	0.02%	2.9s
48	2504	3	0.7s	2.1s	0.11%	0.9s	4.2s	10.2s	0.09%	8.0s	0.14%	13.0s
48	2660	10	0.7s	1.0s	0.02%	0.8s	2.2s	4.2s	0.04%	2.4s	0.03%	3.6s
48	3014	4	0.8s	1.7s	0.01%	0.7s	2.1s	8.0s	0.36%	3.9s	0.06%	6.9s
48	3853	7	0.9s	1.7s	0.03%	0.6s	1.6s	3.5s	0.29%	4.4s	0.08%	7.7s
48	4507	11	1.1s	1.9s	0.01%	0.9s	4.2s	9.0s	0.26%	5.8s	0.02%	8.4s
48	4992	3	1.2s	5.8s	0.02%	1.8s	5.2s	23.9s	1.58%	9.0s	0.32%	18.6s
48	5625	5	1.3s	4.1s	0.03%	1.6s	9.7s	23.6s	0.11%	15.1s	0.04%	19.7s
48	6616	8	1.4s	4.1s	0.05%	1.4s	7.6s	17.6s	0.23%	10.3s	0.09%	16.0s
48	7069	11	1.7s	3.3s	0.01%	1.6s	7.9s	18.8s	0.08%	11.0s	0.05%	15.6s
48	8193	4	1.9s	13.4s	0.05%	3.2s	17.7s	48.0s	0.41%	48.9s	0.06%	80.4s
48	9099	6	2.8s	10.0s	0.02%	3.0s	15.2s	46.5s	0.33%	26.5s	0.07%	48.2s
48	9757	10	2.3s	7.3s	0.02%	2.5s	12.8s	30.7s	0.30%	15.8s	0.10%	34.7s
48	10294	14	2.7s	6.8s	0.04%	2.6s	11.3s	31.7s	1.16%	13.1s	0.14%	29.6s
48	12205	4	3.2s	23.6s	0.10%	5.0s	14.9s	99.4s	1.55%	59.5s	0.12%	112.9s
48	12641	7	3.0s	14.1s	0.06%	4.4s	16.6s	69.5s	1.84%	27.4s	0.33%	60.2s
48	13518	11	3.1s	11.3s	0.02%	4.2s	19.9s	53.7s	0.47%	32.9s	0.07%	52.9s
48	13920	17	3.1s	9.4s	0.04%	4.1s	13.2s	48.9s	1.84%	39.8s	0.06%	55.2s
72	682	3	0.1s	0.7s	0.06%	0.3s	0.5s	1.5s	0.17%	0.4s	0.11%	1.1s
72	844	3	0.1s	0.7s	0.07%	0.2s	0.6s	2.7s	1.42%	0.7s	0.29%	1.8s
72	919	6	0.2s	0.7s	0.08%	0.5s	0.6s	2.6s	1.24%	0.8s	0.12%	1.9s
72	1188	6	0.3s	1.1s	0.07%	0.6s	1.0s	4.3s	0.23%	1.4s	0.24%	3.3s
72	1362	9	0.3s	0.8s	0.02%	0.5s	0.6s	2.6s	1.06%	1.0s	0.09%	2.5s
72	1524	4	0.3s	2.2s	0.05%	0.6s	3.0s	10.5s	0.40%	2.4s	0.36%	8.5s
72	1718	14	0.1s	0.8s	0.05%	0.5s	0.6s	3.3s	0.96%	1.7s	0.09%	2.9s
72	1777	9	0.4s	1.3s	0.09%	0.7s	0.7s	4.3s	1.19%	1.7s	0.19%	3.7s
72	1942	6	0.7s	2.1s	0.02%	0.7s	2.1s	7.1s	0.08%	2.2s	0.23%	5.4s
72	2185	14	0.3s	1.1s	0.04%	0.8s	0.8s	4.8s	1.27%	1.8s	0.17%	4.3s
72	3028	10	0.6s	2.4s	0.06%	0.9s	1.7s	12.7s	1.45%	5.3s	0.15%	9.8s
72	3106	4	0.7s	5.7s	0.05%	1.3s	9.5s	34.4s	0.19%	7.8s	0.30%	19.6s
72	3445	15	0.8s	2.5s	0.06%	1.0s	2.5s	9.3s	0.15%	3.6s	0.17%	9.0s
72	3974	6	1.4s	7.2s	0.05%	1.3s	4.1s	30.4s	1.35%	6.5s	0.21%	21.3s
72	5632	12	1.2s	8.0s	0.02%	2.2s	7.8s	45.8s	0.42%	10.5s	0.13%	29.9s
72	6038	4	1.1s	26.2s	0.04%	2.6s	7.7s	154.5s	1.75%	38.5s	0.27%	81.3s
72	6373	18	1.8s	7.6s	0.02%	1.9s	9.0s	31.5s	0.14%	9.1s	0.22%	25.6s

continue next page...

... continued from previous page

instance			only lower bounds			only upper bounds				both bounds		
$T$	#nodes	#sub	rough	all	diff	init	rough	all	diff	rough	gap	all
72	7646	9	3.1s	21.7s	0.04%	3.1s	20.6s	143.0s	0.65%	46.4s	0.20%	121.5s
72	9118	16	2.9s	20.0s	0.03%	3.6s	19.5s	102.5s	0.60%	30.2s	0.19%	80.0s
72	10003	20	3.0s	18.4s	0.04%	4.5s	16.5s	81.0s	0.43%	28.5s	0.17%	71.5s
<b>72</b>	<b>12211</b>	<b>6</b>	<b>8.9s</b>	<b>175.2s</b>	<b>0.02%</b>	<b>6.8s</b>	<b>52.0s</b>	<b>783.8s</b>	<b>0.88%</b>	<b>171.4s</b>	<b>0.15%</b>	<b>597.2s</b>
72	13804	9	7.1s	68.2s	0.05%	7.1s	44.9s	628.1s	0.86%	90.4s	0.23%	233.8s
72	14607	18	6.1s	51.5s	0.02%	6.6s	73.2s	295.0s	0.20%	69.3s	0.18%	205.5s
72	16014	23	5.5s	40.9s	0.05%	5.9s	59.6s	287.4s	0.20%	42.0s	0.27%	161.6s
72	18432	6	11.2s	457.7s	0.04%	11.3s	124.3s	972.9s	0.49%	256.5s	0.14%	857.6s
72	20139	10	12.1s	191.8s	0.04%	14.6s	116.3s	1108.1s	0.25%	179.3s	0.28%	620.2s
72	21246	20	9.8s	88.9s	0.08%	11.0s	78.8s	549.0s	0.63%	126.3s	0.23%	375.3s
72	22227	27	7.7s	66.9s	0.05%	9.7s	96.7s	409.6s	0.31%	105.0s	0.20%	308.8s
96	795	5	0.2s	0.7s	0.10%	0.4s	0.9s	2.1s	0.32%	0.7s	0.27%	2.2s
96	1075	5	0.3s	1.3s	0.08%	0.6s	0.7s	4.1s	0.29%	1.1s	0.28%	3.8s
96	1202	9	0.2s	0.9s	0.06%	0.3s	1.1s	3.1s	0.25%	1.0s	0.26%	3.3s
96	1548	10	0.3s	1.6s	0.14%	0.6s	1.6s	7.1s	0.65%	2.1s	0.24%	8.3s
96	1810	18	0.2s	1.2s	0.20%	0.8s	1.2s	6.2s	0.77%	1.6s	0.17%	4.9s
96	1883	7	0.4s	5.2s	0.17%	0.8s	3.5s	30.5s	1.21%	5.2s	0.46%	22.9s
96	2342	18	0.3s	2.0s	0.06%	0.7s	2.1s	11.5s	0.60%	2.4s	0.38%	8.0s
96	2436	26	0.4s	1.4s	0.15%	0.9s	1.4s	7.5s	1.11%	2.8s	0.04%	6.1s
96	2599	11	0.5s	5.2s	0.15%	1.0s	4.3s	25.9s	0.60%	4.1s	0.68%	22.6s
96	2980	27	0.6s	2.8s	0.14%	1.0s	2.8s	12.8s	0.23%	3.2s	0.41%	10.6s
96	3728	7	0.8s	51.7s	0.10%	1.4s	10.3s	190.9s	0.37%	17.1s	0.45%	120.3s
96	3884	19	0.5s	6.3s	0.16%	1.1s	6.1s	32.4s	0.29%	4.8s	0.61%	22.1s
96	4901	30	0.9s	6.4s	0.11%	1.3s	5.4s	55.3s	0.87%	6.7s	0.48%	32.2s
96	4987	12	0.8s	28.6s	0.12%	1.6s	9.3s	200.6s	1.49%	11.9s	0.60%	112.0s
96	7031	23	1.2s	58.7s	0.09%	2.0s	14.1s	242.6s	0.71%	39.3s	0.19%	108.4s
96	7838	7	1.4s	286.5s	0.13%	4.2s	43.1s	1004.9s	0.34%	132.0s	0.24%	973.1s
96	8177	34	1.3s	28.3s	0.12%	2.3s	17.6s	213.6s	0.34%	20.5s	0.38%	111.2s
96	9920	15	2.1s	288.0s	0.11%	4.3s	58.0s	754.6s	0.37%	59.9s	0.78%	1062.6s
96	12121	26	2.6s	150.5s	0.13%	8.4s	47.2s	1046.5s	0.49%	70.6s	0.34%	585.4s
96	13783	36	3.7s	181.4s	0.09%	5.8s	41.9s	882.6s	0.68%	44.8s	0.46%	587.7s
96	14528	8	4.6s	7406.1s	0.07%	9.3s	156.9s	7241.6s	0.52%	347.8s	0.41%	7414.2s
96	17472	15	4.9s	1088.5s	0.15%	10.7s	131.6s	4790.9s	1.26%	132.9s	0.80%	6655.2s
96	20086	28	7.2s	771.4s	0.13%	9.1s	107.7s	4080.7s	0.99%	253.9s	0.49%	4288.2s
96	21101	41	7.3s	523.2s	0.10%	17.8s	103.9s	3146.3s	0.65%	168.0s	0.34%	1910.1s
96	23807	10	9.7s	>3h	0.11%	19.7s	498.0s	>3h	1.29%	1038.0s	0.48%	>3h
96	26781	19	20.1s	>3h	0.16%	78.9s	356.7s	>3h	0.47%	451.4s	0.47%	>3h
96	28247	32	9.2s	1433.3s	0.11%	19.7s	252.0s	9229.1s	0.69%	509.2s	0.33%	10336.1s
96	30109	49	9.6s	1050.4s	0.10%	18.2s	220.3s	10297.2s	0.97%	369.0s	0.33%	9908.9s

Nested Column Generation, as detailed in Section 4.3, is an alternative to Nested Benders Decomposition. However, the computing times for the former are always larger than for the latter. Also the difference between the bound at the end of the rough phase and the optimal solution increases. Further, for longer time horizons, the phase I initialization of the Column Generation master problems sometimes failed to reduce the slacks sufficiently enough.

Combining Nested Benders and Nested Column Generation offers the possibility of adaptively choosing  $\rho$  such that a certain error level is not exceeded, cf. Section 4.4.

#### 4. Decomposition with Recombining Scenario Trees

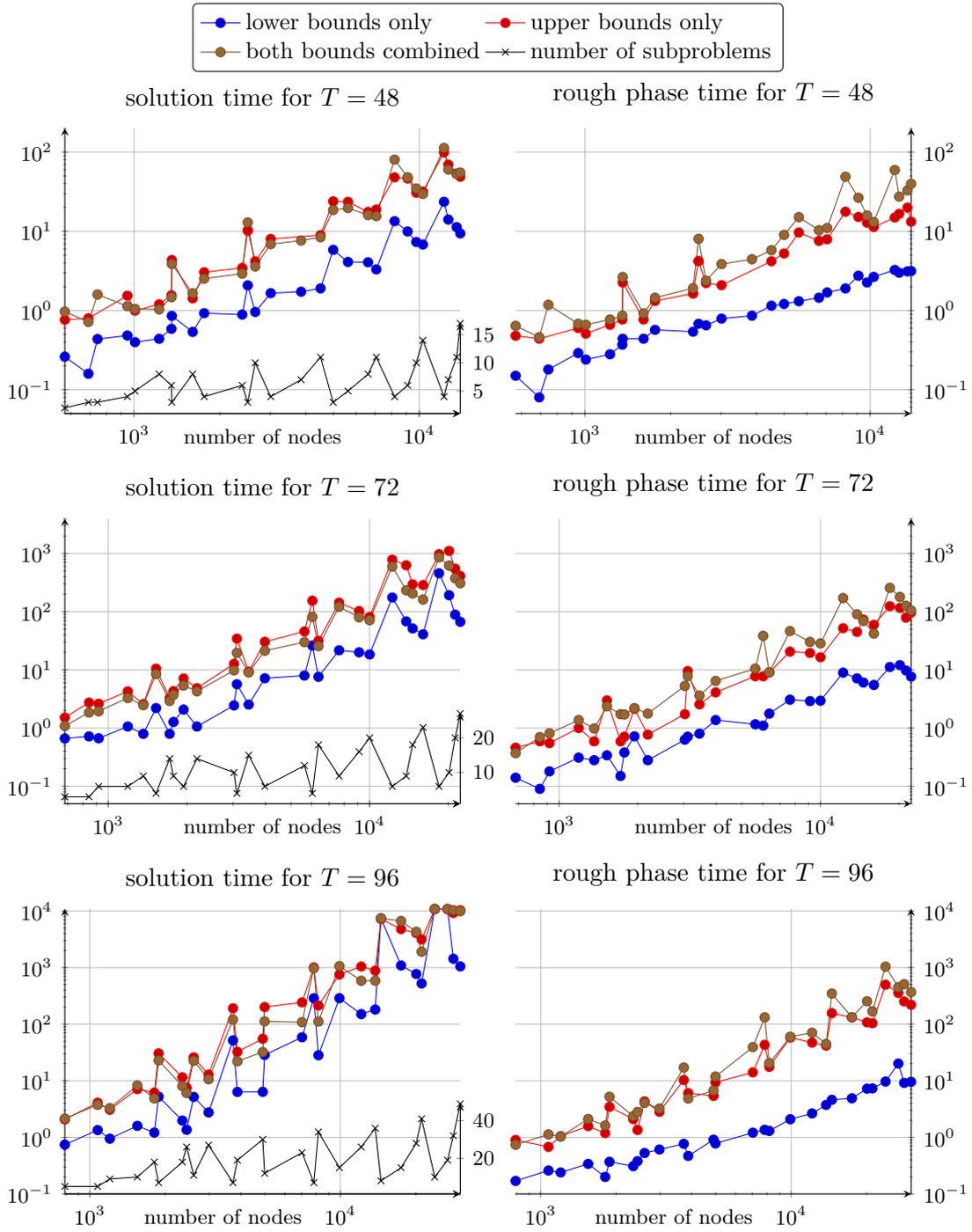


Figure 4.4.: Solution and rough phase time when applying Nested Benders Decomposition only, Nested Column Generation only, and both combined, respectively, to the power scheduling problem (4.28) for different recombining scenario trees and time horizons of two, three, and four days.

The column ‘gap’ reports the relative gap between lower and upper bounds *after having completed the rough phase*. Unfortunately, when comparing overall solution times, the reduction in the number of master problem evaluations is not sufficient to compensate the overhead for additional computation of upper bounds.

For a particular scenario tree with a time horizon of three days (marked in bold font in Table 4.2), Figure 4.5 shows the progress of the decomposition algorithms w.r.t. solution time. For the Nested Benders Decomposition it is seen, that during the rough phase (first 9 seconds, the dashed black line shows the value of the aggregation parameter  $\rho$ ), the lower bound increases to a value that is already very close to the optimal value (the relative difference is 0.02%, see Table 4.2). In the following,  $\rho$  is decreased, which results in an increase in the number of subproblems that are solved within each fast-forward-fast-backward pass. Further, the maximal relative approximation error of the cost-to-go functions is decreasing. This relative approximation error is computed as difference between the approximation value  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$  and the value of the master problem (4.3) or (4.5) when evaluated for the same  $x_{t-1} \in Z(\lambda_{[t]})$ . Recall, that the master problems for  $t < T$  provide only a lower bound on the actual cost-to-go function  $\underline{Q}_t(x_{t-1}, \lambda_{[t]})$ , too.

For the Nested Column Generation, we see that the upper bound converges much slower to the optimal value than for the Nested Benders Decomposition. When the rough phase is finished, the bound is still 0.88% away from the optimal value. However, the number of subproblem evaluations in each fast-forward-fast-backward pass is lower than in Nested Benders Decomposition. The relative approximation error for a subproblem evaluation for some  $(\mu, \sigma) \in Y(\lambda_{[t]})$  is here the relative difference between the dual value  $\sigma$  and the value of the pricing problem (4.17). The plot for Nested Column Generation is shifted to the right due to the time spend for the phase I initialization.

Finally, the combination of Nested Benders Decomposition and Nested Column Generation shows a faster convergence of the upper bound to the optimal value when compared to applying Nested Column Generation alone. After the rough phase, the upper bound is only 0.15% away from the optimal value. Further, it is seen that the algorithm stops already for  $\rho = 0.009$ , which is possible because the gap between lower and upper bound is already sufficiently close. Additionally, the combined algorithm requires only 10 fast-forward-fast-backward passes, while the Nested Benders Decomposition uses 15 passes and the Nested Column Generation uses 20 passes.

### Long Time Horizons

Finally, we investigated the algorithm’s potential for optimization over longer time horizons. Due to the disappointing experience with our implementation of the Nested Column Generation algorithm, we ran only the Nested Benders Decomposition here. Table 4.3 and Figure 4.6 shows the solution times for time horizons up to one year. It is seen, that also such long time horizons with scenario trees having several ten thousand nodes can be handled, even though the solving time is more unpredictable now. Again, the rough phase, which usually takes only a few minutes, already computes a bound that is very close to the value when  $\rho$  is decreased to  $10^{-4}$ , see column ‘diff’ in the table.

#### 4. Decomposition with Recombining Scenario Trees

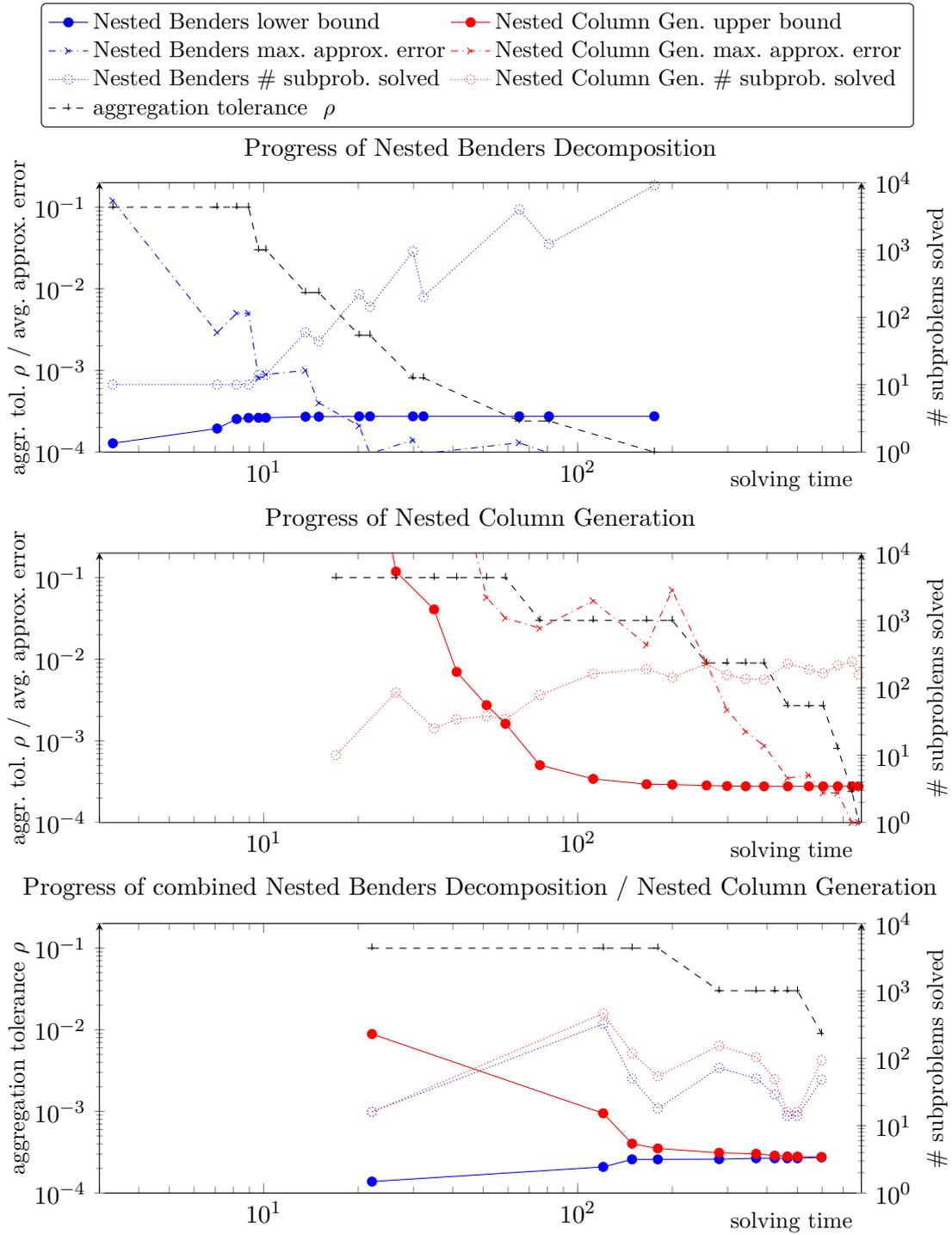


Figure 4.5.: Solving progress of power scheduling problem (4.28) with time horizon  $T = 72$  and a recombining scenario tree having 12211 nodes for Nested Benders Decomposition, Nested Column Generation, and its combination.

Table 4.3.: Performance of Nested Benders Decomposition for different time horizons.  
For an explanation of the columns, see Table 4.2.

$T$	#nodes	#sub	rough	all	diff
168	1260	10	0.4s	2.6s	0.01%
168	1529	11	0.8s	6.8s	0.00%
168	1991	19	0.5s	1.7s	0.01%
168	2517	20	0.6s	9.7s	0.01%
168	2776	12	0.7s	7.6s	0.01%
168	3383	35	0.7s	4.1s	0.05%
168	4120	37	0.8s	8.5s	0.01%
168	4362	21	1.1s	10.7s	0.03%
168	4444	47	0.8s	3.3s	0.00%
168	5450	53	1.1s	7.9s	0.15%
168	5883	13	0.8s	134.7s	0.15%
168	6936	41	1.4s	30.8s	0.00%
168	8834	23	1.6s	160.1s	0.14%
168	8886	56	1.4s	24.3s	0.08%
168	12742	45	2.0s	151.2s	0.09%
168	12991	14	4.0s	3047.9s	0.06%
168	14969	62	2.2s	111.0s	0.03%
168	18044	29	5.1s	1373.3s	0.08%
168	23039	52	4.0s	756.1s	0.12%
168	25912	74	4.5s	497.4s	0.13%
168	26294	15	7.5s	>3h	0.10%
168	31292	30	11.9s	>3h	0.07%
168	37151	57	8.9s	1341.4s	0.05%
168	40157	82	8.6s	3055.0s	0.12%
168	43791	19	17.2s	>3h	0.04%
168	49323	36	16.1s	8508.6s	0.07%
168	54964	66	15.0s	3593.6s	0.05%
168	57829	95	21.7s	5992.7s	0.06%
744	4451	47	1.1s	8.5s	0.30%
744	5888	53	1.4s	17.8s	0.20%
744	7955	91	1.6s	13.5s	-0.00%
744	10101	96	2.0s	35.1s	0.42%
744	10977	59	2.5s	59.3s	0.27%
744	14379	171	2.1s	25.0s	0.53%
744	17634	178	3.1s	45.6s	0.28%
744	18274	104	4.4s	124.4s	0.34%
744	19785	255	2.8s	53.2s	0.67%
744	23846	66	5.8s	696.0s	0.84%
744	24360	274	3.5s	110.4s	0.29%
744	30625	206	5.0s	184.9s	0.41%
744	37594	127	11.0s	623.1s	0.40%
744	39179	303	6.1s	367.6s	0.88%

continue next page...

#### 4. Decomposition with Recombining Scenario Trees

... continued from previous page					
$T$	#nodes	#sub	rough	all	diff
744	55582	230	11.2s	276.7s	0.82%
744	56089	70	25.1s	2230.7s	0.33%
744	67999	330	12.3s	311.5s	0.66%
744	76915	150	18.6s	1691.8s	0.33%
744	102057	258	21.6s	1235.8s	0.99%
744	116573	378	32.3s	1070.6s	0.57%
744	117410	83	53.4s	>3h	1.38%
744	142840	153	83.6s	5690.6s	2.08%
744	169662	290	56.9s	2414.9s	1.33%
744	184754	421	54.6s	1827.4s	1.00%
2232	12417	139	2.6s	22.8s	0.38%
2232	16205	153	2.7s	39.7s	0.64%
2232	23136	283	3.7s	32.0s	0.53%
2232	29511	292	4.4s	40.1s	0.62%
2232	31785	186	5.3s	240.8s	0.32%
2232	42637	528	5.8s	48.3s	0.21%
2232	53441	328	9.5s	149.4s	0.67%
2232	53732	564	7.1s	56.3s	0.41%
2232	59530	784	7.7s	42.3s	0.45%
2232	70059	202	13.4s	2722.8s	0.49%
2232	73468	818	9.2s	91.1s	0.55%
2232	92926	629	12.8s	285.5s	0.84%
2232	111134	392	27.0s	578.8s	1.95%
2232	119832	927	19.3s	178.2s	0.69%
2232	166667	210	37.7s	2339.6s	0.77%
2232	170703	715	26.4s	812.6s	0.42%
2232	208071	1040	33.3s	365.3s	0.40%
2232	230640	450	127.4s	1944.0s	1.52%
2232	307644	803	79.8s	1380.3s	0.92%
2232	356472	1159	64.2s	1141.1s	0.80%
2232	510919	887	155.8s	3369.5s	1.18%
2232	560952	1300	130.1s	2655.9s	1.29%
8760	48175	569	14.7s	55.0s	0.07%
8760	63928	613	23.3s	64.4s	0.24%
8760	115752	1164	33.8s	121.0s	0.03%
8760	123030	719	27.3s	228.4s	0.43%
8760	167784	2100	49.7s	128.7s	0.03%
8760	211845	2234	47.2s	160.9s	0.06%
8760	212768	1297	48.0s	226.7s	0.06%
8760	234958	3082	65.2s	201.7s	0.11%
8760	276564	780	56.1s	2039.3s	0.63%
8760	290481	3289	74.7s	316.7s	0.54%
8760	360912	2465	68.4s	480.5s	0.10%
8760	473378	3646	104.3s	502.8s	0.06%
8760	819600	4097	157.6s	1273.3s	0.01%

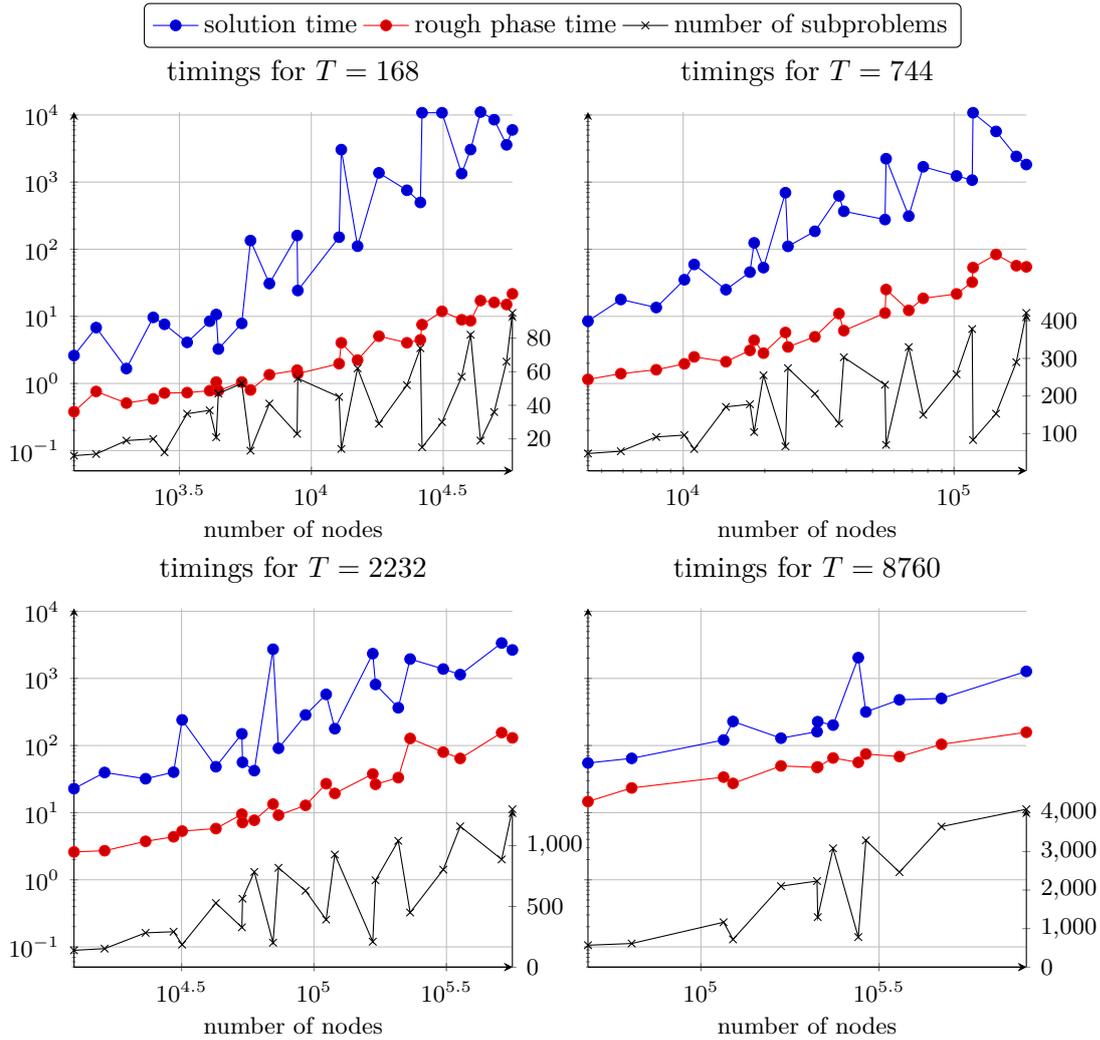


Figure 4.6.: Solution and rough phase time when applying Nested Benders Decomposition to the power scheduling problem (4.28) for different recombining scenario trees and time horizons of one week, one month, three months, and one year.

**Aggregation Parameter**

Finally, we investigated the gain of starting with a large value of the aggregation parameter  $\rho$  and decreasing it over time, instead of using the final value of  $\rho = 10^{-4}$  from the beginning on. Table 4.4 shows the number of fast-forward-fast-backward passes (column ‘#passes’) and solving times for several scenario trees and increasing time horizon. The last column shows the ratio between solving with a stationary value of  $\rho = 10^{-4}$  and a decreasing sequence. Further, for each time horizon, we print the geometric mean of the solution times and their ratio with respect to all considered scenario trees<sup>11</sup>. It is

<sup>11</sup>Computations that hit the timelimit (3 hours) are accounted with 6 hours in geom. means and ratios.

#### 4. Decomposition with Recombining Scenario Trees

seen, that even though the number of passes increases, the benefit of starting with a rough value  $\rho = 0.1$  and decreasing it gradually (via multiplication by 0.3) grows with an increasing time horizon. This is not surprising, since the effect of growing approximation solution sets increases with the number of time stages.

Table 4.4.: Performance for stationary and decreasing sequences for aggr. parameter  $\rho$ .

$T$	instance		stationary $\rho$		decreasing $\rho$		time ratio
	#nodes	#sub	#passes	time	#passes	time	
48	572	2	3	0.24s	12	0.26s	0.90
48	692	3	3	0.24s	12	0.16s	1.52
48	745	3	3	0.55s	13	0.44s	1.25
48	948	4	3	0.42s	12	0.48s	0.87
48	1008	5	3	0.33s	13	0.40s	0.83
48	1225	8	3	0.38s	13	0.44s	0.86
48	1353	6	3	0.53s	13	0.59s	0.90
48	1357	3	3	1.01s	14	0.85s	1.18
48	1606	8	3	0.50s	11	0.54s	0.92
48	1762	4	3	0.90s	13	0.93s	0.98
48	2395	6	3	0.94s	14	0.89s	1.06
48	2504	3	3	2.52s	14	2.08s	1.22
48	2660	10	4	1.02s	13	0.96s	1.06
48	3014	4	3	1.69s	12	1.66s	1.02
48	3853	7	3	1.53s	13	1.73s	0.89
48	4507	11	3	1.48s	14	1.90s	0.78
48	4992	3	3	5.76s	12	5.84s	0.99
48	5625	5	3	3.92s	13	4.10s	0.96
48	6616	8	3	3.19s	14	4.07s	0.78
48	7069	11	3	2.50s	15	3.31s	0.75
48	8193	4	3	13.40s	17	13.43s	1.00
48	9099	6	3	7.93s	12	9.96s	0.80
48	9757	10	3	5.31s	13	7.34s	0.72
48	10294	14	3	4.81s	16	6.81s	0.71
48	12205	4	3	21.14s	17	23.60s	0.90
48	12641	7	3	11.98s	13	14.07s	0.85
48	13518	11	3	8.76s	13	11.31s	0.77
48	13920	17	3	7.77s	14	9.41s	0.83
<b>geometric mean</b>				<b>1.85s</b>		<b>2.00s</b>	<b>0.92</b>
72	682	3	3	0.70s	13	0.66s	1.06
72	844	3	3	0.83s	14	0.72s	1.16
72	919	6	3	0.69s	13	0.67s	1.03
72	1188	6	3	1.18s	13	1.06s	1.11
72	1362	9	3	0.95s	13	0.79s	1.20
72	1524	4	3	4.31s	13	2.20s	1.96
72	1718	14	3	0.71s	13	0.80s	0.89
72	1777	9	3	1.25s	14	1.27s	0.98

continue next page...

## 4.6. Numerical Experiment

... continued from previous page

$T$	instance		stationary $\rho$		decreasing $\rho$		time ratio
	#nodes	#sub	#passes	time	#passes	time	
72	1942	6	4	2.88s	15	2.08s	1.39
72	2185	14	3	1.20s	13	1.06s	1.13
72	3028	10	3	4.45s	13	2.45s	1.82
72	3106	4	3	12.99s	15	5.67s	2.29
72	3445	15	6	3.76s	15	2.53s	1.48
72	3974	6	4	11.07s	13	7.17s	1.55
72	5632	12	3	12.64s	15	7.98s	1.58
72	6038	4	4	50.15s	14	26.21s	1.91
72	6373	18	4	12.48s	16	7.59s	1.64
72	7646	9	4	32.58s	16	21.70s	1.50
72	9118	16	4	34.17s	16	20.01s	1.71
72	10003	20	4	23.96s	15	18.39s	1.30
72	12211	6	5	386.95s	15	175.22s	2.21
72	13804	9	4	98.10s	16	68.16s	1.44
72	14607	18	4	66.27s	17	51.50s	1.29
72	16014	23	4	65.41s	14	40.86s	1.60
72	18432	6	6	1088.08s	17	457.74s	2.38
72	20139	10	4	257.08s	14	191.77s	1.34
72	21246	20	5	150.94s	14	88.88s	1.70
72	22227	27	5	121.93s	14	66.91s	1.82
<b>geometric mean</b>				<b>12.18s</b>		<b>8.30s</b>	<b>1.46</b>
96	795	5	3	1.20s	13	0.74s	1.61
96	1075	5	3	2.56s	11	1.35s	1.90
96	1202	9	3	1.47s	13	0.95s	1.55
96	1548	10	3	3.19s	13	1.60s	2.00
96	1810	18	4	2.20s	14	1.21s	1.81
96	1883	7	3	14.06s	15	5.23s	2.69
96	2342	18	3	3.05s	13	1.98s	1.54
96	2436	26	3	2.06s	13	1.36s	1.51
96	2599	11	3	11.75s	13	5.18s	2.27
96	2980	27	3	3.32s	14	2.76s	1.21
96	3728	7	4	132.20s	15	51.69s	2.56
96	3884	19	3	13.94s	12	6.35s	2.20
96	4901	30	3	13.73s	13	6.36s	2.16
96	4987	12	3	60.27s	14	28.57s	2.11
96	7031	23	3	84.81s	13	58.68s	1.45
96	7838	7	3	945.29s	15	286.53s	3.30
96	8177	34	3	63.95s	14	28.27s	2.26
96	9920	15	3	683.60s	14	287.98s	2.37
96	12121	26	3	229.28s	13	150.51s	1.52
96	13783	36	3	215.01s	14	181.40s	1.19
96	14528	8	3	>3h	15	7406.12s	2.92
96	17472	15	3	3144.57s	15	1088.49s	2.89
96	20086	28	3	1177.58s	14	771.39s	1.53
96	21101	41	3	656.90s	14	523.17s	1.26
96	23807	10	2	>3h	16	>3h	1.00
96	26781	19	3	>3h	16	>3h	1.00

continue next page...

#### 4. Decomposition with Recombining Scenario Trees

... continued from previous page								
$T$	instance		stationary $\rho$		decreasing $\rho$		time ratio	
	#nodes	#sub	#passes	time	#passes	time		
96	28247	32	3	3910.66s	14	1433.31s	2.73	
96	30109	49	3	2197.38s	15	1050.43s	2.09	
<b>geometric mean</b>				<b>83.99s</b>		<b>46.44s</b>	<b>1.85</b>	
168	1260	10	3	6.90s	11	2.61s	2.65	
168	1529	11	3	9.07s	14	6.80s	1.33	
168	1991	19	4	7.48s	11	1.67s	4.49	
168	2517	20	4	36.19s	11	9.65s	3.75	
168	2776	12	3	35.97s	12	7.61s	4.72	
168	3383	35	5	18.72s	11	4.10s	4.57	
168	4120	37	4	37.84s	12	8.47s	4.47	
168	4362	21	4	159.63s	12	10.66s	14.97	
168	4444	47	5	17.29s	9	3.26s	5.31	
168	5450	53	4	51.08s	14	7.85s	6.51	
168	5883	13	3	270.74s	12	134.70s	2.01	
168	6936	41	5	196.70s	14	30.79s	6.39	
168	8834	23	3	226.66s	15	160.08s	1.42	
168	8886	56	5	208.12s	14	24.28s	8.57	
168	12742	45	3	228.29s	12	151.18s	1.51	
168	12991	14	3	2019.46s	11	3047.86s	0.66	
168	14969	62	3	286.78s	12	111.02s	2.58	
168	18044	29	3	1685.39s	16	1373.28s	1.23	
168	23039	52	3	1213.51s	13	756.06s	1.61	
168	25912	74	3	2012.16s	11	497.43s	4.05	
168	26294	15	2	>3h	12	>3h	1.00	
168	31292	30	3	8091.45s	16	>3h	0.37	
168	37151	57	3	6163.44s	13	1341.37s	4.59	
168	40157	82	3	4985.09s	13	3054.97s	1.63	
168	43791	19	1	>3h	15	>3h	1.00	
168	49323	36	2	>3h	11	8508.62s	2.54	
168	54964	66	3	>3h	12	3593.60s	6.01	
168	57829	95	3	7705.34s	19	5992.68s	1.29	
<b>geometric mean</b>				<b>383.41s</b>		<b>149.97s</b>	<b>2.62</b>	
744	4451	47	7	63.71s	14	8.48s	7.52	
744	5888	53	4	19.78s	15	17.83s	1.11	
744	7955	91	8	49.95s	11	13.53s	3.69	
744	10101	96	9	228.79s	15	35.13s	6.51	
744	10977	59	7	465.38s	12	59.30s	7.85	
744	14379	171	9	69.04s	13	24.97s	2.76	
744	17634	178	8	121.29s	17	45.56s	2.66	
744	18274	104	6	277.83s	12	124.42s	2.23	
744	19785	255	7	168.27s	14	53.20s	3.16	
744	23846	66	4	821.77s	12	696.04s	1.18	
744	24360	274	6	140.80s	13	110.43s	1.28	
744	30625	206	5	306.49s	18	184.92s	1.66	
744	37594	127	7	3460.61s	12	623.10s	5.55	
744	39179	303	5	356.27s	17	367.60s	0.97	
744	55582	230	5	1015.17s	15	276.67s	3.67	

continue next page...

... continued from previous page							
$T$	instance		stationary $\rho$		decreasing $\rho$		time ratio
	#nodes	#sub	#passes	time	#passes	time	
744	56089	70	5	>3h	14	2230.68s	9.68
744	67999	330	4	796.65s	12	311.46s	2.56
744	76915	150	4	3178.55s	11	1691.81s	1.88
744	102057	258	4	2684.72s	17	1235.78s	2.17
744	116573	378	5	4219.65s	14	1070.59s	3.94
744	117410	83	2	>3h	15	>3h	1.00
744	142840	153	3	>3h	14	5690.63s	3.80
744	169662	290	5	>3h	15	2414.91s	8.94
744	184754	421	4	10147.00s	14	1827.42s	5.55
<b>geometric mean</b>			<b>726.09s</b>		<b>263.17s</b>		<b>3.00</b>

## 4.7. Out-Of-Sample Evaluation

As discussed in Section 1.3, different techniques have been developed to approximate random variables or stochastic processes by a limited number of scenarios or finite scenario trees, respectively. Further, stability analysis of stochastic programs has been conducted (cf., e.g., Römisch [2003]), convergence of optimal values and/or solution sets has been proven for specific techniques, and properties of statistical estimates and bounds have been established (cf., e.g., Shapiro [2003a] and Eichhorn and Römisch [2007]).

Unfortunately, on the one hand, these theoretical results may require the optimization problems and underlying random variables to fulfill specific regularity assumptions that may be hard to verify in some cases of practical interest. On the other hand, quantitative error bounds and statistical properties are not available for all problem classes. Furthermore, due to the numerical complexity of stochastic programming models, it is sometimes necessary to use approximations that are too rough to obtain meaningful error bounds or confidence intervals via asymptotic results.

In such cases, one has to resort to numerical methods to measure the performance and quality of approximation and solution methods. Since a main task of stochastic programming is to provide decision strategies that are robust enough to be applicable in real-world scenarios, it suggests itself to measure the quality of an approximation method by evaluating the (optimal) solutions obtained from solving the approximate problem. This can be done, e.g., by evaluating these solutions along *out-of-sample scenarios*, cf., e.g., Kaut and Wallace [2007] and Chiralaksanakul and Morton [2004], Hilli and Pennanen [2008] for one- and multistage problems, respectively.

In this section, we study how out-of-sample testing may be used to study the behavior of approximations to multistage stochastic linear programs. Thereby, we aim for problems with many stages, where, due to numerical complexity, the thorough construction of out-of-sample strategies as in Hilli and Pennanen [2008] and the second method of Chiralaksanakul and Morton [2004] do not apply. Furthermore, our framework differs since we abstain from a (relatively) complete recourse assumption. Then, in particular, optimal solutions of an approximate problem are not necessarily feasible along out-of-sample scenarios. Therefore, the generation of feasible solutions out of solutions of an

#### 4. Decomposition with Recombining Scenario Trees

approximate problem is an important issue. Furthermore, this question may be of interest whenever one is interested in obtaining practically applicable solutions. For this *feasibility restoration* we adopt different projection approaches.

Finally, we apply the proposed feasibility restoration approach to the power scheduling model from Section 4.6.2 and study the quality of solutions obtained by the adapted Nested Benders Decomposition approach from Section 4.2.1, based on recombining scenario trees, and solutions induced by non-recombining trees.

##### 4.7.1. Problem Setting

We consider the multistage stochastic linear program (1.9) and denote its optimal value for a specific stochastic process  $\xi$  by  $v(\xi)$ . For notational simplicity, we assume non-random technology and recourse matrices ( $A_{t,k}(\cdot) \equiv A_{t,k}$ ,  $t \in [2 : T]$ ,  $k = 1, 2$ ). Further, let  $\tilde{\xi}$  be a discrete approximation of  $\xi$ , such that the corresponding approximate multistage stochastic linear program, see also (3.4), can be solved by some numerical method. The optimal value  $v(\tilde{\xi})$  is often considered as an approximation of  $v(\xi)$ . However, specific regularity assumptions on (1.9) and the processes  $\xi$  and  $\tilde{\xi}$  are necessary to ensure certain approximation qualities, cf. Heitsch et al. [2006] and Küchler [2009]. Indeed, without such conditions,  $\tilde{\xi}$  may be close to  $\xi$  in some sense, but passing from the original problem to the approximate one may lead to significant changes in the optimal value, e.g., by providing arbitrage possibilities, see Example 2.5.1 in Küchler [2009].

Being interested in a good approximation of the unknown value  $v(\xi)$ , it is thus reasonable rather to evaluate an approximate solution  $\tilde{x} = \{\tilde{x}_t(\cdot)\}_{t \in [T]}$  of (1.9) w.r.t.  $\tilde{\xi}$  with regard to the original data process  $\xi$ , that is, to consider  $\mathbb{E}[\varphi(\xi, \tilde{x}(\xi))]$  as an approximation for  $v(\xi)$ , where  $\varphi(\xi, x(\xi))$  denotes the cost function of (1.9),

$$\varphi(\xi, x(\xi)) := \sum_{t=1}^T \langle b_t(\xi_t), x_t(\xi_{[t]}) \rangle.$$

However, the approximate solution  $\tilde{x}$  may not be feasible (or even not defined) along the original process  $\xi$ . Thus, it may be appropriate to modify  $\tilde{x}$  to a solution  $\hat{x}$  that is feasible for (1.9) w.r.t.  $\xi$ . Then the value

$$\mathbb{E}[\varphi(\xi, \hat{x}(\xi))] \tag{4.29}$$

provides an upper bound on  $v(\xi)$  that can be realized by implementing the strategy  $\hat{x}$ . The value (4.29) appears to be a more reliable approximation of  $v(\xi)$  than  $\mathbb{E}[\varphi(\xi, \tilde{x}(\xi))]$ .

To evaluate the integral (4.29), the law of large numbers suggests to draw independent samples  $\xi^j$ ,  $j \in J$ , from the distribution of  $\xi$  and to consider the *out-of-sample value*

$$\hat{v}(\tilde{\xi}) := \frac{1}{|J|} \sum_{j \in J} \varphi(\xi^j, \hat{x}(\xi^j)). \tag{4.30}$$

The value  $\hat{v}(\tilde{\xi})$  can be seen as the *real-world performance* of the approximate solution  $\tilde{x}(\cdot)$ .

Consequently, approximations  $\tilde{\xi}'$  and  $\tilde{\xi}''$  of different accuracy or constructed by different algorithms may be compared by means of their out-of-sample values  $\hat{v}(\tilde{\xi}')$  and  $\hat{v}(\tilde{\xi}'')$ . Similarly, solution algorithms can be compared by evaluating the resulting solutions.

#### 4.7.2. Adaptation of Approximate Solutions to Out-Of-Sample Scenarios

Let  $\{\tilde{\xi}^i\}_{i \in I}$  denote the finite set of scenarios of  $\tilde{\xi}$  and consider a solution  $\tilde{x}$  to the approximate problem ((1.9) with  $\tilde{\xi}$ ). Starting from  $\tilde{x}$ , we aim to construct a strategy  $\hat{x}$  that is feasible along a set of out-of-sample scenarios  $\{\xi^j\}_{j \in J} \subset \text{supp } P_\xi$ . In order to ensure that  $\hat{x}$  is implementable by a decision maker without perfect information, this *feasibility restoration* has to be nonanticipative.

To this end, we consider a *nonanticipative* mapping  $\pi : \{\xi^j\}_{j \in J} \rightarrow \{\tilde{\xi}^i\}_{i \in I}$  that assigns every out-of-sample scenario  $\xi^j$  to some scenario  $\tilde{\xi}^i$  of the approximated process that is close to  $\xi^j$ , in some sense. We say that  $\pi$  is nonanticipative, if it can be written as  $\pi(\xi^j) = (\pi_1(\xi_{[1]}^j), \dots, \pi_T(\xi_{[T]}^j))$ , where  $\pi_t : \mathbb{R}^{s_1 + \dots + s_t} \rightarrow \{\tilde{\xi}_t^i\}_{i \in I}$  are Borel measurable mappings. Assuming a decision maker who has observed  $\xi_{[t]} = \xi_{[t]}^j$  until time  $t$ , the rule  $\pi$  suggests him a scenario  $(\pi_1(\xi_{[1]}^j), \dots, \pi_t(\xi_{[t]}^j))$  of the approximate model (and the corresponding strategy) that is close to his observation. The mapping  $\pi$  can be defined as a (conditional) projection as follows. Set  $\pi_1(\xi_{[1]}^j) := \tilde{\xi}_1^i$  for some  $i \in I$  (well defined, because first stage is deterministic). For  $t \in [2 : T]$ , let

$$\pi_t(\xi_{[t]}^j) := \underset{\tilde{\xi}_t^i}{\operatorname{argmin}} \left\{ \|\tilde{\xi}_t^i - \xi_t^j\| : i \in I \text{ with } \tilde{\xi}_\tau^i = \pi_\tau(\xi_{[\tau]}^j), \tau \in [t-1] \right\},$$

that is, if  $\xi_{[t-1]}^j$  is assigned by  $(\pi_1, \dots, \pi_{t-1})$  to some node  $\tilde{\xi}_{[t-1]}^i$  in the scenario tree  $\tilde{\xi}$ , then  $\xi_{[t]}^j$  is assigned to a successor of  $\tilde{\xi}_{[t-1]}^i$  which  $t$ -stage value is closest to  $\xi_t^j$ .

The distance between the set of out-of-sample scenarios  $\{\xi^j\}_{j \in J}$  and their associated tree scenarios  $\pi(\xi^j)$  can be measured by the term

$$d_\pi(J, I) := \frac{1}{|J|} \sum_{j \in J} \frac{\sum_{t=1}^T \|\xi_t^j - \pi_t(\xi_{[t]}^j)\|}{\sum_{t=1}^T \|\xi_t^j\|}, \quad (4.31)$$

which is the relative euclidean distance between a scenario  $\xi^j$  and its assigned scenario  $\pi(\xi^j)$ , averaged over all scenarios  $j \in J$ .

Having related the out-of-sample scenarios  $\xi^j$  to the approximation scenarios  $\tilde{\xi}^i$  by the mapping  $\pi$ , we obtain that  $\tilde{x}_t(\pi(\cdot)) \in \mathcal{F}_t$ ,  $t \in [T]$ , i.e.,  $\tilde{x}(\pi(\xi))$  is nonanticipative w.r.t. the process  $\xi$  and thus indeed a potential solution to the initial problem. Unfortunately,  $\tilde{x}(\pi(\xi^j))$  does not need to be *feasible* along the scenario  $\xi^j$  of the initial process  $\xi$ , in general. In order to achieve this feasibility, different projection-based approaches to modify  $\tilde{x}(\pi(\cdot))$  are proposed in the following.

In the following, we denote the decision  $\tilde{x}(\cdot)$  along the scenario  $\pi(\xi^j)$  by  $\tilde{x}^j$  and refer to it as the *reference solution*. The modification of  $\tilde{x}^j$  along the out-of-sample scenario  $\xi^j$  is denoted by  $\hat{x}^j$ .

#### 4. Decomposition with Recombining Scenario Trees

##### Feasibility Restoration

Aiming for a (nonanticipative) solution  $\hat{x}^j$  that is feasible along the scenario  $\xi^j$ , we propose the following straightforward approach. Let  $\hat{x}_1^j := \tilde{x}_1^j$ . For  $t = 2, \dots, T$  and given  $\hat{x}_{t-1}^j$ , we search for a feasible point  $\hat{x}_t^j$  that is *close to*  $\tilde{x}_t^j$ .

Such a point  $\hat{x}_t^j$  may be found by projecting  $\tilde{x}_t^j$  on the feasible set at time stage  $t$ , i.e., on the set  $\{x_t \in X_t : A_{t,0}x_t + A_{t,1}\hat{x}_{t-1}^j = h_t(\xi_t^j)\}$ . However, in order to cope with possible future infeasibilities in models without relatively complete recourse, we further restrict the feasible set by incorporating information about future constraints. More precisely, we consider the value

$$\Delta_t^j := \min \left\{ \begin{array}{l} A_{t,0}x_t + A_{t,1}\hat{x}_{t-1}^j = h_t(\xi_t^j), \\ \|x_t - \tilde{x}_t^j\|_\infty : A_{\tau,0}x_\tau + A_{\tau,1}x_{\tau-1} \in [\underline{h}_\tau, \overline{h}_\tau], \quad \tau \in [t+1 : T], \\ x_\tau \in X_\tau, \quad \tau \in [t : T], \end{array} \right\} \quad (4.32)$$

being the minimal distance from  $\tilde{x}_t^j$  onto the (reduced) feasible set at time stage  $t \in [2 : T]$ . The vectors  $\underline{h}_\tau$  and  $\overline{h}_\tau$  are chosen such that  $\underline{h}_\tau \leq h_\tau(\xi_\tau^j) \leq \overline{h}_\tau$  holds true for all  $j \in J$ . The corresponding conditions in (4.32) are added to avoid making decisions at time  $t$  that are easily seen to lead to future infeasibilities. In particular, we set  $(\underline{h}_\tau)_k = (\overline{h}_\tau)_k$  for those components of  $h_\tau(\cdot)$  that do not depend on  $\xi$ . Observe, that this simple approach to avoid future infeasibilities relies on the assumption of non-random matrices  $A_{\tau,0}$  and  $A_{\tau,1}$ . However, the approach can be extended, e.g., by demanding the existence of feasible decisions  $x_\tau, \tau \geq t$ , along all possible future realizations of the process  $\tilde{\xi}$ .

**Naive Restoration.** One may think about several techniques for determining a feasible point  $\hat{x}_t^j$  based on previously computed values for  $\hat{x}_1^j, \dots, \hat{x}_{t-1}^j$ . A *naive* method is to just stay as close as possible to the reference solution  $\tilde{x}_t^j$  and to set  $\hat{x}_t^j$  to the  $x_t$  component of an optimal solution of problem (4.32):

$$\hat{x}_t^j := \operatorname{argmin}_{x_t} \left\{ \begin{array}{l} A_{t,0}x_t + A_{t,1}\hat{x}_{t-1}^j = h_t(\xi_t^j), \\ \|x_t - \tilde{x}_t^j\|_\infty : A_{\tau,0}x_\tau + A_{\tau,1}x_{\tau-1} \in [\underline{h}_\tau, \overline{h}_\tau], \quad \tau \in [t+1 : T], \\ x_\tau \in X_\tau, \quad \tau \in [t : T]. \end{array} \right\}$$

**Myopic Restoration.** However, sometimes it may be reasonable to exchange some closeness to  $\tilde{x}_t^j$  by cost minimality along the out-of-sample scenario  $\xi^j$ . That is, we allow the decision  $\hat{x}_t^j$  to deviate from the set of closest feasible solutions by a relative fraction  $\varepsilon \geq 0$  in order to minimize the costs along  $\xi^j$ . Doing so in a *myopic* way means to minimize  $\langle b_t(\xi_t^j), x_t \rangle$ , i.e.,

$$\hat{x}_t^j := \operatorname{argmin}_{x_t} \left\{ \begin{array}{l} A_{t,0}x_t + A_{t,1}\hat{x}_{t-1}^j = h_t(\xi_t^j), \\ \langle b_t(\xi_t^j), x_t \rangle + \rho_t \|x_t - \tilde{x}_t^j\|_\infty : A_{\tau,0}x_\tau + A_{\tau,1}x_{\tau-1} \in [\underline{h}_\tau, \overline{h}_\tau], \tau \in [t+1 : T], \\ x_\tau \in X_\tau, \quad \tau \in [t : T], \\ \|x_t - \tilde{x}_t^j\|_\infty \leq (1 + \varepsilon)\Delta_t^j, \end{array} \right\}$$

where the term  $\rho_t \|x_t - \tilde{x}_t^j\|_\infty$  has been added to ensure that among solutions with the same cost  $\langle b_t(\xi_t^j), x_t \rangle$ , one is chosen that minimizes the distance to  $\tilde{x}_t^j$ . Thus, the value  $\rho_t \geq 0$  should be such that it does not dominates the objective function, e.g.,  $\rho_t = 10^{-4} \|b_t(\xi_t^j)\|_\infty$ .

Note that for  $\rho_t \rightarrow 0$  and  $\varepsilon \rightarrow \infty$ , the emphasis on cost minimality along  $\xi^j$  increases on the price of a larger deviation from the reference solution  $\tilde{x}_t^j$ .

**Farsighted Restoration.** Due to the time-coupling constraints, a decision  $x_t$  at time  $t$  impacts the feasible sets for future decisions and thus the future costs. These future costs can be taken into account within the feasibility restoration by considering the *shadow prices* associated to the time-coupling constraints. For this purpose, we recall the dual problem (4.21) to (1.9) w.r.t.  $\tilde{\xi}$ :

$$\max \left\{ \mathbb{E} \left[ \sum_{t=2}^T \langle h_t(\tilde{\xi}_t), \mu_t \rangle \right] : \begin{array}{l} \mu_t(\tilde{\xi}) \in \mathbb{R}^{d_t}, \mu_t \in \tilde{\mathcal{F}}_t, \quad t \in [2 : T], \\ b_1(\tilde{\xi}_1) - \mathbb{E}[\mu_2^\top A_{2,1}] \in X_1^*, \\ b_t(\tilde{\xi}_t) - \mu_t^\top A_{t,0} - \mathbb{E}[\mu_{t+1}^\top A_{t+1,1} | \tilde{\xi}_{[t]}] \in X_t^*, \quad t \in [2 : T-1], \\ b_T(\tilde{\xi}_T) - \mu_T^\top A_{T,0} \in X_T^*, \end{array} \right\}$$

and denote by  $\tilde{\mu}$  an optimal solution of (4.21). The dual variable corresponding to the primal decision  $x_t$  is then equal to  $\mathbb{E}[\tilde{\mu}_{t+1}(\tilde{\xi}_{[t+1]})^\top A_{t+1,1} | \tilde{\xi}_t]$ . In particular,  $-\tilde{\mu}_{t+1}(\tilde{\xi}_{[t+1]})^\top A_{t+1,1}$  is a subgradient of the *cost-to-go function*  $\mathcal{Q}_{t+1}(\cdot, \tilde{\xi}_{[t+1]})$  as defined in (1.10), refer also to Section 3.1.1. In order to maintain the nonanticipativity of the feasibility restoration along the out-of-sample scenario  $\xi^j$ , we resort to the subgradient of the *expected* cost-to-go function, i.e.,

$$\eta_{t+1} := -\mathbb{E}[\tilde{\mu}_{t+1}(\tilde{\xi}_{[t+1]})^\top A_{t+1,1} | \tilde{\xi}_{[t]} = \pi_t(\xi_{[t]}^j)].$$

Using this dual information about future costs, we choose  $\hat{x}_t^j$  in a *farsighted* way by setting

$$\hat{x}_t^j := \operatorname{argmin}_{x_t} \left\{ \langle b_t(\xi_t^j) + \eta_{t+1}, x_t \rangle + \rho_t \|x_t - \tilde{x}_t^j\|_\infty : \begin{array}{l} A_{t,0}x_t + A_{t,1}\hat{x}_{t-1}^j = h_t(\xi_t^j), \\ A_{\tau,0}x_\tau + A_{\tau,1}x_{\tau-1} \in [h_\tau, \bar{h}_\tau], \\ \tau \in [t+1 : T], \\ x_\tau \in X_\tau, \quad \tau \in [t : T], \\ \|x_t - \tilde{x}_t^j\|_\infty \leq (1 + \varepsilon)\Delta_t^j, \end{array} \right\}$$

**Extensive Restoration.** An even more farsighted method is to use not only a single subgradient, but several, which may be available when (1.9) is solved by a Nested Benders Decomposition algorithm, c.f. Sections 3.1.2 and 4.2. Such a method is closely related to the first approach in Chiralaksanakul and Morton [2004], which has been proposed for multistage stochastic linear programs with interstage independence or a weak type of interstage dependence.



value  $v(\tilde{\xi})$  of the scenario tree based problem is computed by solving the deterministic equivalent with CPLEX<sup>13</sup>.

The recombining scenario trees were constructed as described in Section 4.5. Recombination takes place every six time stages, the number of different subtrees per time period (the time between two recombination points) is bounded by six, and the SCENRED parameter `red_percentage` was varied to construct trees of different size. The tree based problem is solved by the adapted Nested Benders Decomposition, cf. Algorithm 4.2, where  $\rho = 0.001$  has been used as final value for the aggregation parameter.

Recall, that model (4.28) does not possess relatively complete recourse due to the condition on the minimal final fill level of the water storage, the restriction on the power gradient, and the reserve requirement. However, without the latter, the capacity of the thermal units is sufficient to cover the maximal load. Thus, since a violation of the reserve requirement does not prohibit the power plant to operate inside its operational bounds, we selected constraint (4.28e) as soft constraints during feasibility restoration, i.e., weakly infeasible solutions are allowed to violate the reserve requirement.

### Projection distance

Figure 4.7 shows the average distance  $d_\pi(J, I)$  of 1000 out-of-sample scenarios to various scenario trees. For an initial set of 1000, 2000, 3000, or 4000 wind energy scenarios, SCENRED's forward tree construction algorithm was applied with varying values for the parameter `red_percentage` to construct non-recombining scenario trees of different size. At first, the average distance between the out-of-sample scenarios and the assigned scenarios in the non-recombining scenario tree reduces as expected when the number of nodes in the scenario trees are increased. However, when the number of nodes is increased too much, and thus the corresponding scenario trees become "more similar" to the initial fan, the average distance  $d_\pi(J, I)$  increases. This is not surprising, since both the mean scenario (having only 48 nodes) and the initial fan (with the maximal possible number of nodes) are expected to yield bad approximations of the actual stochastic process  $\xi$ . Even though the initial fan may approximate the distribution  $\xi$  well, it completely ignores the filtration  $\mathcal{F}_t$  defined by the conditional distributions (the "growth of information over time"). Since the projection  $\pi$  is nonanticipative, the decision which scenario of the initial fan is assigned to an out-of-sample scenario depends only on the value at the second stage. However, it is rather unlikely, that selecting a scenario  $\tilde{\xi}^i$  which value at the second stage is similar to  $\xi_2^j$  yields a small distance  $\|\xi^j - \tilde{\xi}^i\|$  w.r.t. the whole time horizon. Similarly, when constructing a scenario tree from a low number of initial scenarios, an early branching during tree construction leads to having only a very low number of scenarios available to approximate conditional distributions in later time stages<sup>14</sup>. Accordingly, the average distances  $d_\pi(J, I)$  reduce somewhat when the number of initial scenarios is increased.

<sup>13</sup><http://www.cplex.com>

<sup>14</sup>A binary branching tree, which may offer a good representation of the growth of information over time, has  $\approx 10^{14}$  scenarios for a time horizon of 48 stages. On the contrary, 1000 scenarios on 48 stages correspond to only  $\approx 1.15$  branches per node.

#### 4. Decomposition with Recombining Scenario Trees

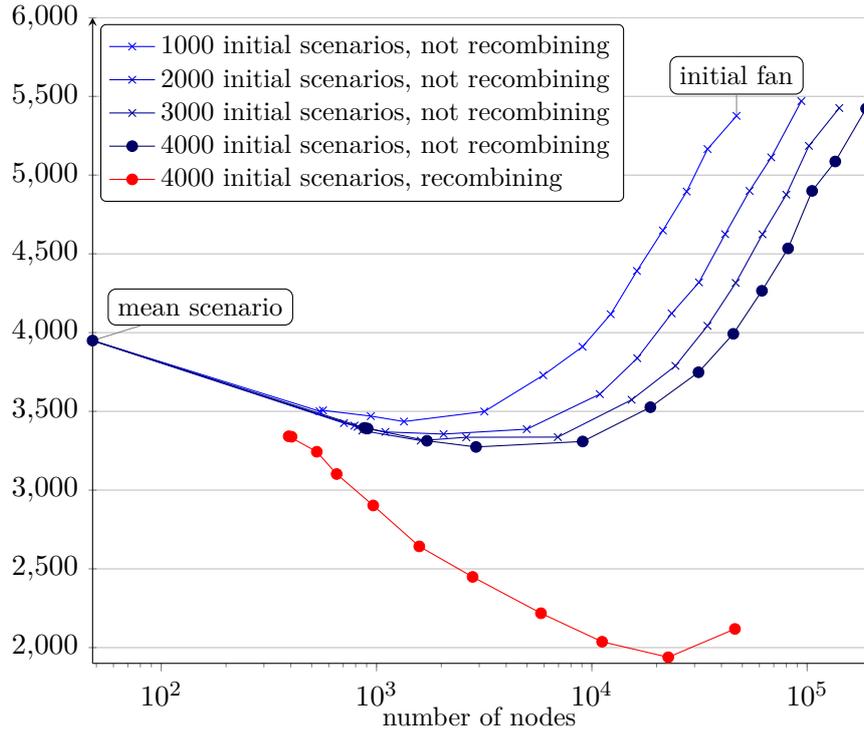


Figure 4.7.: Average distance  $d_{\pi}(J, I)$  (cf. (4.31)) of 1000 out-of-sample scenarios from non-recombining and recombining scenarios trees of varying size and constructed from initial scenario fans of different size.

When using non-recombining scenario trees, much smaller distances from the out-of-sample scenarios to the scenarios in the recombining scenario tree can be observed. This is clearly due to the much larger number of scenarios that can be represented by a recombining scenario tree and the better approximation of conditional distributions, since at each recombining stage the full set of initial scenarios is available.

#### Out-of-sample value

Figure 4.8 presents the results of the out-of-sample evaluation for non-recombining and recombining scenario trees. Both types of trees were constructed from an initial fan of 4000 scenarios. For now, consider only the out-of-sample value obtained with naive, myopic, or farsighted feasibility restoration with  $\varepsilon \leq 0.1$  (the remaining ones are discussed below). As one can observe, the out-of-sample values  $\hat{v}(\tilde{\xi})$  are higher than the minimal costs  $v(\tilde{\xi})$  of the tree based stochastic programs. While the optimal values  $v(\tilde{\xi})$  of the approximate problems do not significantly differ for scenario trees having different numbers of nodes, the out-of-sample values first decrease and then increase with an increasing number of nodes in the same manner as the distance  $d_{\pi}(J, I)$  between the scenario trees and the set of out-of-sample scenarios is changing. This indicates that the

quality of the tree based solutions with regard to their usefulness for real-world decision making improves with increasing accuracy of the scenario tree approximation (if one regards  $d_\pi(J, I)$  as a measure for the approximation quality).

Next, we observe that the out-of-sample values  $\hat{v}(\xi)$  computed from solutions of *recombining* scenario tree based stochastic programs are better than when non-recombining scenario trees on the same number of nodes and with the same feasibility restoration method are used. This is probably due to the much higher number of scenarios that can be used within recombining trees.

### Feasibility restoration methods

When comparing different types of feasibility restoration, we see that the naive approach of only minimizing the distance between a feasible out-of-sample solution and a reference solution yields out-of-sample values that are considerably worse than when also costs are taken into account (myopic or farsighted restoration). When comparing the myopic and farsighted feasibility restoration with  $\varepsilon = 0.05$ , we observe that the out-of-sample values from the myopic approach are even slightly better than those from the farsighted one. However, not visible in the graphics is the number of weakly and strongly infeasible out-of-sample scenarios. While the number of strongly infeasible out-of-sample scenarios is at most 1% for both feasibility restoration methods, the number of weakly infeasible scenarios increases by between 32% (for trees with small  $d_\pi(J, I)$ ) and 110% (for trees with large  $d_\pi(J, I)$ ) when replacing a farsighted by a myopic feasibility restoration. Hence, the additional information on (future) costs related to changes in the fill level of the water reservoir has a rather adverse effect on the out-of-sample values, but increases the chance that feasibility restoration can modify the reference solution without violating the reserve requirement.

Further, we compare the out-of-sample values that are obtained with farsighted feasibility restoration and varying values of  $\varepsilon$ , i.e., allowing the modified solution to differ more or less from the reference solution in favor of cost minimality. The choice  $\varepsilon = 0$  is similar to the naive method, but chooses a solution with minimal costs among all solutions that are closest to the reference solution. This additional step improves the out-of-sample value considerably (likely, because it gives preference to the cheap coal power plant over the more expensive gas-and-steam or steam power plants when deciding on how to satisfy the additional demand). When increasing the value of  $\varepsilon$  and thereby the freedom to move solutions towards optimality, we can observe a decrease in the out-of-sample values, until their behavior appears chaotic for  $\varepsilon \geq 0.5$ . An explanation is found by recalling that the out-of-sample values are defined w.r.t. all out-of-sample scenarios where a feasible solution could be constructed. Relaxing closeness to the reference solution by increasing  $\varepsilon$  leads in more and more cases to a failure of the feasibility restoration phase, and thus to decreasing reliability of the out-of-sample value. Figure 4.9 shows the amount of feasible, weakly infeasible, and strongly infeasible scenarios for a single non-recombining scenario tree when using farsighted feasibility restoration with different values of  $\varepsilon$ .

Finally, Figure 4.10 visualizes the solutions for one out-of-sample scenario as it has been computed by the naive and farsighted feasibility restoration methods. The upper left

#### 4. Decomposition with Recombining Scenario Trees

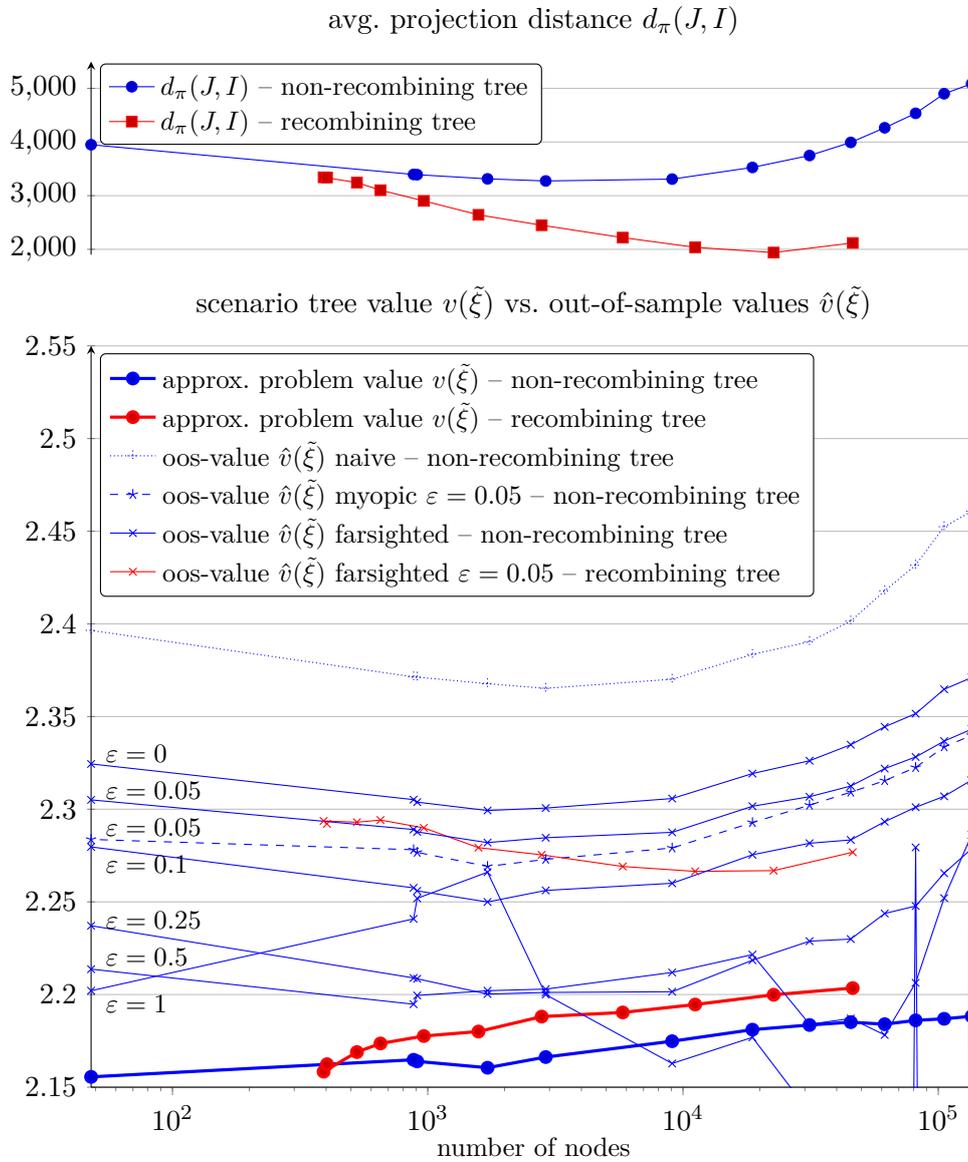


Figure 4.8.: Comparison of out-of-sample (oos) value  $\hat{v}(\tilde{\xi})$  with value  $v(\tilde{\xi})$  of approximation problem for scenario trees with varying number of nodes and different feasibility restoration methods. For comparison, also the projection distances of the considered scenario trees are shown again (top).

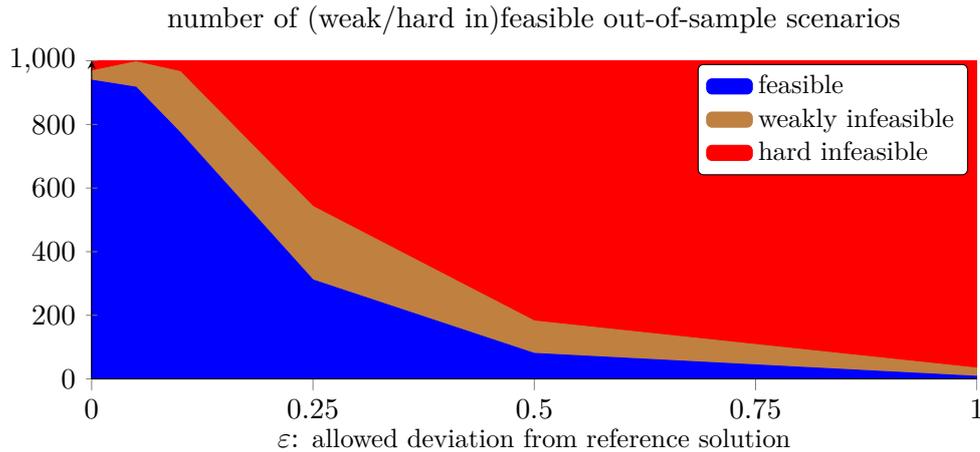


Figure 4.9.: The number of feasible, weakly infeasible, and hard infeasible out-of-sample scenarios when using farsighted feasibility restoration with increasing parameter  $\varepsilon$  for a fixed non-recombining scenario tree (2899 nodes).

picture in Figure 4.10 shows the out-of-sample scenario and the corresponding reference scenario. When comparing the solutions computed by the naive and the farsighted approach with  $\varepsilon = 0$ , one observes that around time stages 30 and 47, the naive approach uses the gas power plant to compensate a lack of wind energy, while the farsighted approach (which takes costs into account) chooses the cheaper gas & steam power plant. Further, with increasing  $\varepsilon$ , the farsighted approach chooses to make less use of the water pumps around time stage 30, even though it also increases the use of the water turbine around time stage 36 (where high demand and low wind energy occur at the same time, see also the demand curve in Figure 4.2). Increasing deviation from the reference solution finally leads to a failure in the feasibility restoration with  $\varepsilon = 1$ . While emptying the water storage in the late time stages allows to satisfy the demand in these stages at very low costs (even reducing the operation of the coal power plant), refilling the water storage to satisfy the restriction on the fill level in the final stage fails, partly also because the wind energy input in the out-of-sample scenario reduces considerable in the last time stage.

#### 4. Decomposition with Recombining Scenario Trees

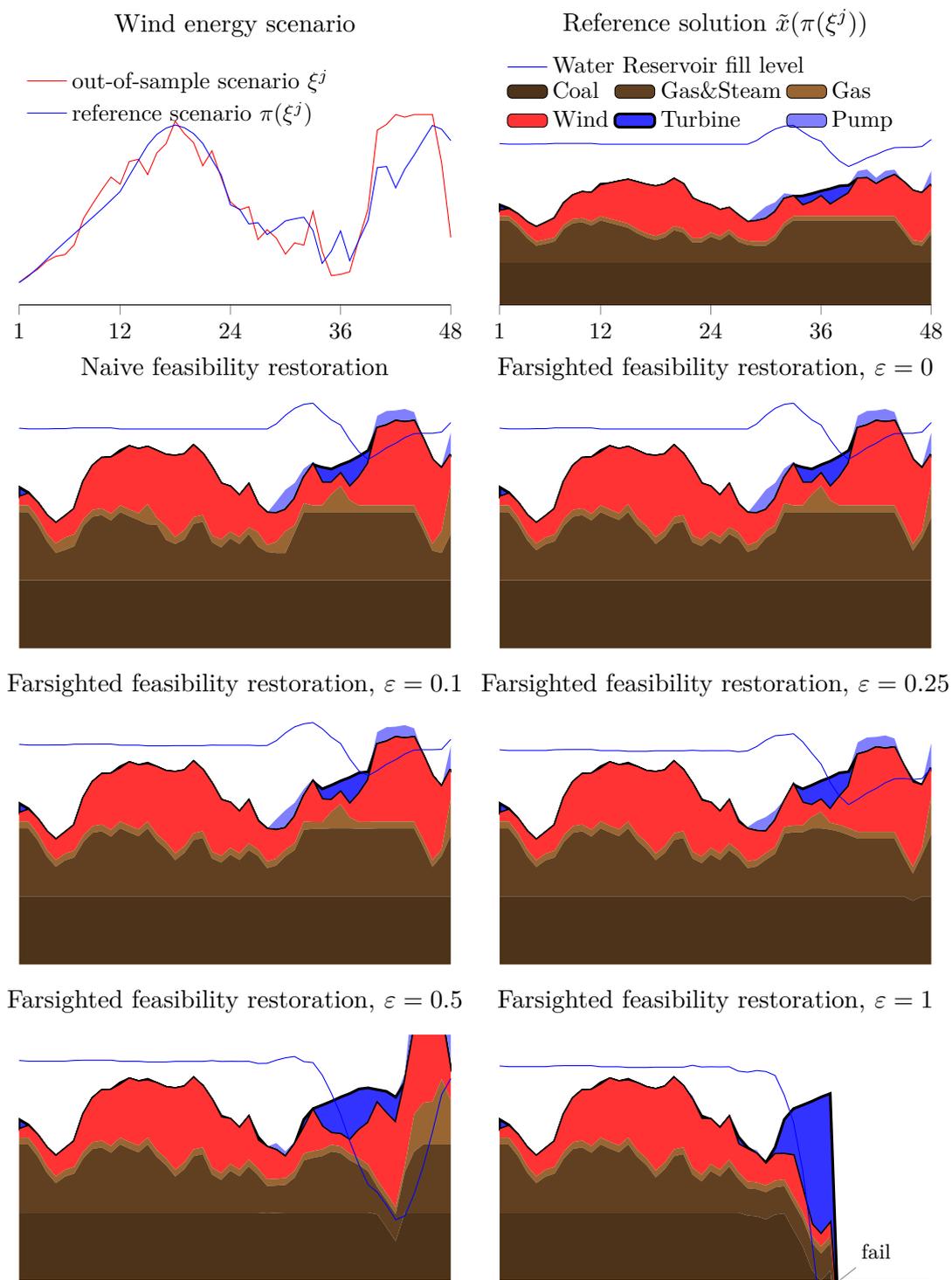


Figure 4.10.: Visualization of out-of-sample solution when using naive and farsighted feasibility restoration for various values of  $\varepsilon$ .

## 5. Optimization of Dispersed Energy Supply

The steadily increasing share of wind energy within many power generating systems leads to strong and unpredictable fluctuations of the electricity supply and is thus a challenge with regard to power generation and transmission. In this chapter, we investigate the potential of energy storages to contribute to a cost optimal electricity supply by decoupling the supply and the demand. For this purpose we study a stochastic programming model of a regional power generating system, where uncertainties are due to the availability of wind energy and the development of EEX prices. The identification of a cost optimal operation plan allows to evaluate the economical possibilities and optimal capacities of the considered storage technologies.

On the one hand, the optimization of energy storages and investment decisions requires the consideration of long-term planning horizons. On the other hand, the highly fluctuating wind energy input requires a detailed temporal resolution. Consequently, the resulting optimization problem can, due to its dimension, not be tackled by standard solution approaches, but is well suited for the decomposition algorithm developed in the previous chapter.

The presentation is taken from the publications Epe, Küchler, Römisch, Vigerske, Wagner, Weber, and Woll [2007, 2009a,b].

### 5.1. Introduction

Electric power, one of the most important fields within energy supply, has two main characteristics: on the one hand, supply and demand have to be balanced at every time, but on the other hand, it is storable at only small rates. For these reasons, power plants have to regulate any imbalances between supply and demand, and, in particular, need to cope with unpredictable changes in the customer load. For that purpose, regulating power plants are used, which mostly run in part load and with reduced efficiency. Alternatively, fast power plants such as gas turbines may be used, which can start up within short time. Beyond the cover of the fluctuating load of the customer side, these power plants must also adjust to the increasing share of time-varying power production on the supply side, mostly caused from fluctuating renewables, notably wind.

Germany is the country with the third-highest installed wind power capacities worldwide [World Wind Energy Association, 2011]. In the year 2010, 27.2 GW had been installed, which contributed 9.4% to Germany's total energy supply [European Wind Energy Association, 2011]. With the planned offshore development in the course of the German "Energiewende" (aiming at an increase of the share of energy supply from renewable sources from today 17% to at least 35% in 2050), it could be much more. Thereby, the sometimes strong and rapid fluctuations of the wind energy fed into the electrical

## 5. Optimization of Dispersed Energy Supply

network as well as the regional concentration in the north of the country increasingly pose problems to the network operators and power suppliers [Handschin et al., 2006, Wagner, 2002]. Conventional fuel consumption may be saved by down-regulating conventional (back-up) power plants, but investments in the power plant park can hardly be saved.

In this context, electrical energy storages offer a possibility to decouple supply and demand and to achieve a better capacity utilization as well as a higher efficiency of existing power plants. The changing context has led to an increased interest in such possibilities over the last few years. Yet with the liberalization of the electricity markets, the economics of storages have to be valued against market prices as established at the energy exchanges. Also the operation of storages will mostly not follow local imbalances of demand and supply, but rather try to benefit from market price variations. In particular, the (partial) unpredictability of market prices as well as of wind energy supply have to be taken into account. Things are complicated further through daily, weekly, seasonal, and other cyclic patterns in demand, supply, and prices. This requires a valuation of storages (and other options) over periods as long as one year.

Cost optimal operation planning under uncertainty for such long time periods poses a huge challenge to conventional stochastic programming methods. However, the short-term memory of wind energy input allows to reduce complexity by applying recombining scenario trees.

In the following, we describe a regional energy system model for cost optimal operation planning and analyze the use of storages on a case study. Afterwards, the model is extended by allowing for investments in storages and power plants, and we investigate the optimal allocation of storage sizes on a case study.

## 5.2. Cost Optimal Operation Planning

### 5.2.1. Model

To study the economics of storages, we use a model that describes the fundamental connections between energy supply and storage technologies. Combining technical and economical aspects, the model describes the energy supply of a large city, the available technologies for electricity generation, and the demand. An optimal load dispatch has to consider the marginal generation costs as well as the impact of other system restrictions such as start up costs, etc. Most important restriction of the model is the covering of the demand according to a given profile. For this purpose, energy can be produced by conventional power plants, procured as wind energy, and purchased on the spot market. Additionally, two types of storages are available to decouple availability and demand.

Uncertainty in the amount of available wind energy and electricity prices is modeled by a multivariate stochastic process that can be represented by a recombining scenario tree, c.f. Chapter 4. Thus, the proposed model combines many features of generation scheduling models (unit commitment and load dispatch) as found typically in energy system models, see, e.g., Nowak and Römisch [2000], Swider et al. [2004], and Swider and Weber [2007]. In the following, the model is discussed in detail. Table 5.1 gives an overview of the notation used.

Variables	$Q$	production	$H$	storage level
	$Q^{\text{imp}}$	imported energy	$IC$	import costs
	$Q^{\text{wind}}$	used wind energy	$SC$	start-up costs
	$L^{\text{st}}$	start-up capacity	$OC$	operating costs
	$L^{\text{onl}}$	capacity online	$TC$	total costs
Parameters	$D$	demand	$c^{\text{stu}}$	start-up costs
	$W$	available wind energy	$c^{\text{imp}}$	import costs
	$\bar{Q}$	maximal capacity	$c^{\text{oth}}$	other variable costs
	$\ell$	load factor	$c^{\text{fuel}}$	fuel price
	$\eta^0$	efficiency at minimal load	$\underline{H}$	minimal storage level
	$\eta^m$	marginal efficiency	$\bar{H}$	maximal storage level

Table 5.1.: Notation used by operation planning model.

Under the assumption of power markets with efficient information treatment and without market power, the market results correspond to the outcomes of an optimization carried out by a fully informed central planner. If electricity demand is assumed to be price inelastic, welfare maximization is equivalent to cost minimization within the considered power network. Thereby, the total costs  $TC$  are given as the sum of import costs  $IC_t$ , operating costs  $OC_{t,u}$ , and startup costs  $SC_{t,u}$  over all time steps  $t \in [T]$ , power plant unit types  $u \in U_{\text{pow}}$  and storage types  $u \in U_{\text{st}}$ :

$$TC = \sum_{t=1}^T \left( IC_t + \sum_{u \in U} (OC_{t,u} + SC_{t,u}) \right), \quad (5.1)$$

where  $U := U_{\text{pow}} \cup U_{\text{st}}$ .

The costs for power import at time  $t$  are given by

$$IC_t = c_t^{\text{imp}} Q_t^{\text{imp}} \quad (t \in [T]). \quad (5.2)$$

For the operating costs  $OC_{t,u}$ , an affine function of the plant output  $Q_{t,u}$  is assumed. An exact description of the plant operation costs requires a mixed-binary nonlinear formulation due to the dependency of the plant efficiency on the power output and the startup behavior. This is hardly feasible here due to the high level of time detail. An appropriate linearization can be done by defining an additional decision variable for each plant type, the *capacity currently online*  $L_{t,u}^{\text{onl}}$  [Weber, 2005]. The capacity online forms an upper bound on the actual output. Multiplied with the minimum load factor, it is also a lower bound on the output for each power plant. Hence, operating costs can be decomposed in fuel costs for operation at minimum load, fuel costs for incremental output, and other variable costs (e.g., for CO<sub>2</sub> production):

$$OC_{t,u} = \frac{c_{u,t}^{\text{fuel}}}{\eta_u^0} \ell_u L_{t,u}^{\text{onl}} + \frac{c_{u,t}^{\text{fuel}}}{\eta_u^m} (Q_{t,u} - \ell_u L_{t,u}^{\text{onl}}) + c_u^{\text{oth}} Q_{t,u} \quad (u \in U, t \in [T]). \quad (5.3)$$

## 5. Optimization of Dispersed Energy Supply

Here,  $\eta_u^m$  denotes the marginal efficiency for an operating unit and  $\eta_u^0$  the efficiency at the minimum load factor  $\ell_u$ . With  $\eta_u^m > \eta_u^0$ , the operators have an incentive to reduce the capacity online (for details see Weber [2005]).

Besides operating costs, start-up costs may influence the power scheduling decisions considerably. The start-up costs of a unit are given by

$$SC_{t,u} = c_u^{\text{stu}} L_{t,u}^{\text{stu}} \quad (u \in U, t \in [T]), \quad (5.4)$$

where  $L_{t,u}^{\text{stu}}$  is the start-up capacity given by

$$L_{t,u}^{\text{stu}} = \max(0, L_{t,u}^{\text{onl}} - L_{t-1,u}^{\text{onl}}) \quad (u \in U, t \in [T]) \quad (5.5)$$

and  $L_{0,u}^{\text{stu}}$  is a fixed parameter that decides the initial status of unit  $u \in U$ .

Covering the demand at any time is ensured by

$$\sum_{u \in U_{\text{pow}}} Q_{t,u} + Q_t^{\text{wind}} + Q_t^{\text{imp}} \geq D_t + \sum_{u \in U_{\text{st}}} Q_{t,u} \quad (t \in [T]), \quad (5.6)$$

i.e., the supply at time  $t$  is given by the sum of the power production  $Q_{t,u}$ ,  $u \in U_{\text{pow}}$ , the imported energy  $Q_t^{\text{imp}}$ , and the wind energy supply  $Q_t^{\text{wind}}$ . The total demand equals the sum of the exogenously given domestic demand  $D_t$  and the power  $Q_{t,u}$ ,  $u \in U_{\text{st}}$ , used to fill the energy storages.

The operation levels of all units are constrained by the available capacity,

$$Q_{t,u} \leq \bar{Q}_{t,u} \quad (u \in U, t \in [T]), \quad (5.7)$$

whereas the wind energy supply is bounded by the available wind energy,

$$Q_t^{\text{wind}} \leq W_t \quad (t \in [T]). \quad (5.8)$$

For a storage plant  $u \in U_{\text{st}}$ , storage constraints need to be considered and the filling and discharging has to be described. This leads to the storage level equation

$$H_{t,u} = H_{t-1,u} - \frac{1}{\eta_{\bar{u}}^m} Q_{t,\bar{u}} - \left( \frac{1}{\eta_{\bar{u}}^0} - \frac{1}{\eta_{\bar{u}}^m} \right) \ell_{\bar{u}} L_{t,\bar{u}}^{\text{onl}} + \eta_u^m Q_{t,u} + (\eta_u^0 - \eta_u^m) \ell_u L_{t,u}^{\text{onl}} \quad (5.9)$$

linking the storage level  $H_{t,u}$  at time  $t \in [T]$  with the level  $H_{t-1,u}$  at time  $t-1$ , both expressed in energy units. Here,  $\bar{u} \in U_{\text{pow}}$  denotes the turbine powered by the content of storage  $u \in U_{\text{st}}$ . Additionally, an adequate terminal condition for the reservoirs has to be included. One attractive formulation is to require that the final and initial reservoir levels are identical, but in view of the temporal decomposition algorithm that is applied to solve the model, we decided to use a simpler formulation that does not couple the first and last time stage. That is, we fix the initial fill level  $H_{0,u}$  at the minimum fill level  $\underline{H}_u$ , while the storage level at any time is limited by the minimum and maximum storage levels,

$$\underline{H}_u \leq H_{t,u} \leq \bar{H}_u \quad (u \in U_{\text{st}}, t \in [T]). \quad (5.10)$$

Finally, all variables have to fulfill non-negativity conditions.

The objective of the optimization is to find a decision process satisfying the constraints (5.5)–(5.10), being nonanticipative with respect to the stochastic process  $(W_t, c_t^{\text{imp}})_{t \in [T]}$ , and minimizing the expected total costs  $\mathbb{E}[TC]$ .

### 5.2.2. Case study

We study a power generating system, consisting of a hard coal power plant to cover the minimum and medium load, and two fast gas turbines on different power levels to cover the peaks. The operating parameters of these units rely on real data. Furthermore, the model contains an offshore wind park, a pump-storage power plant (PSW) with the basic data of the PSW Geesthacht, Germany, and a compressed-air energy storage (CAES) with the operating parameters of the CAES Huntorf, Germany. Another source for power supply is the EEX spot market. The time horizon considered for the optimization is one year and a hourly discretization is used, i.e., the model contains  $T = 8760$  time stages.

The stochastic wind power process is represented by a time series model fitted to historical data and scaled to the size of the offshore wind park regarded. To take into account the interdependency between wind power and spot price behavior, the *expected* spot market prices have been calculated from a fundamental model that is based on the existing power plants in Germany and their reliability, prices for fuels and CO<sub>2</sub>, the German load, and the wind power process above. Fluctuation of the spot prices around their expected value are modeled by a further time series model. This hybrid approach was used to generate 1000 scenarios, containing hourly values of wind power and spot prices in the course of one year. These trajectories were used to generate a recombining scenario tree as sketched in Section 4.5. The tree branches three times per day in a binary way and recombination takes place once a day into three different subtrees.

### 5.2.3. Numerical Results

The optimization problem was solved with varying model parameters. To this end, a *base setting* was defined, where the maximal available wind power corresponds to approximately 50% of the demand. The storage sizes correspond to the aforementioned CAES and PSW units in Huntorf and Geesthacht, respectively. Coming from this setting, variations with higher and lower levels of installed wind power and different storage dimensions were calculated. In the following some results are presented.

The optimal operation levels along a randomly chosen scenario from the base setting during a winter week are depicted in Figure 5.1. Whenever the power production exceeds the demand curve, energy is put into the storages, whereas the white spaces under the demand curve represent the output of the storage plants. The operation levels of the thermal units show the usual characteristics, i.e., the hard coal power plant covers the base load and the gas turbine delivers energy at times of high demand on weekdays. The availability of wind power obviously reduces imports from the spot market. The storage units are mainly used to cover the peaks and are only marginally used during the weekend. In this model, the contribution of the operating costs to the power supply

## 5. Optimization of Dispersed Energy Supply

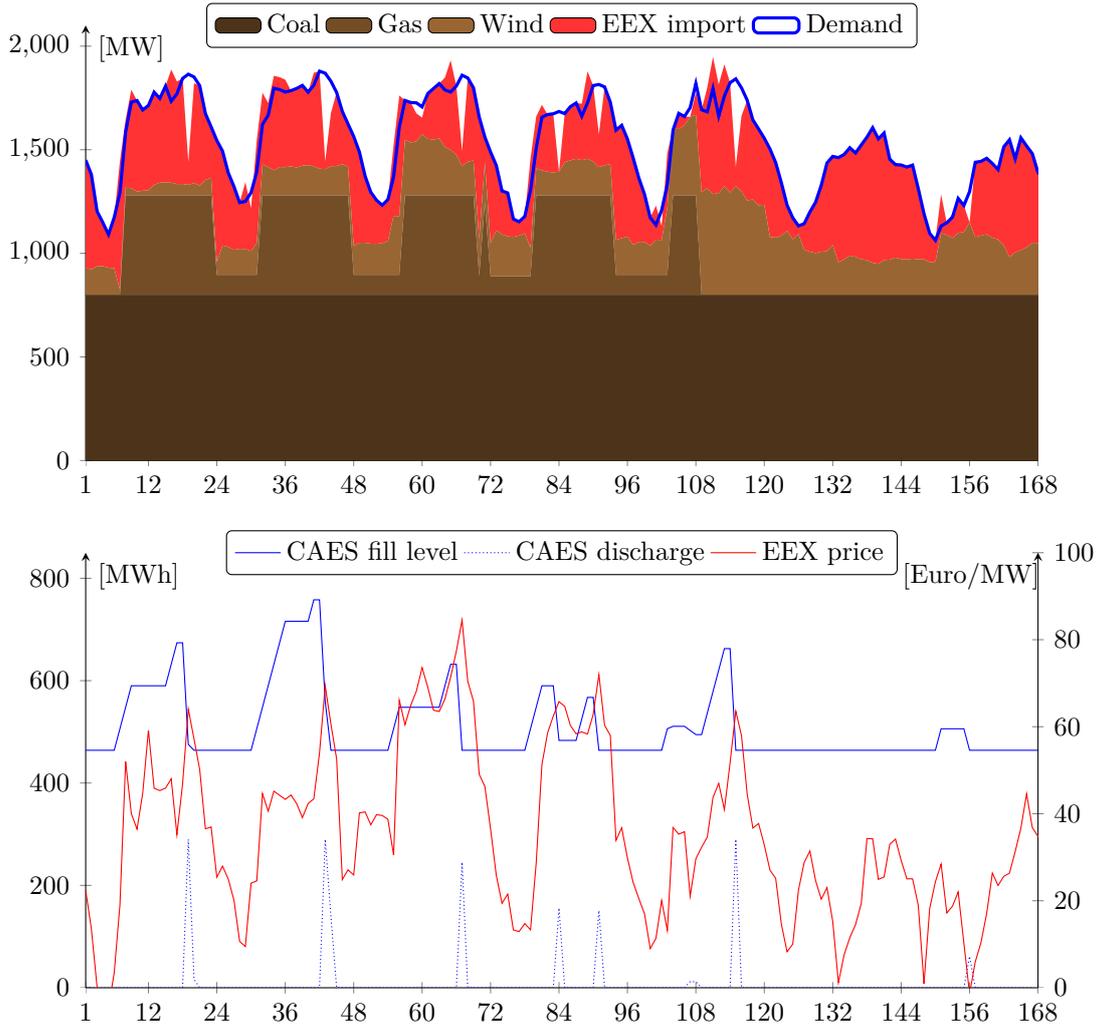


Figure 5.1.: Optimal power scheduling and CAES operation in a winter week.

costs amount to 2.08 Eurocents/kWh with using storage plants and 2.10 Eurocents/kWh without using storage plants.

Figure 5.1 also shows the optimal output and fill level of the CAES in comparison to the actual power price. The minimum fill level of the CAES is 60%. Obviously, the storage plant discharges in times of high spot prices on weekdays. The aforementioned marginal usage of storage plants during the weekend coincides with lower power prices over this period.

To study the impact of the share of wind power on the system, the optimization problem was solved again with doubled wind power capacity. The results along the same scenario and for the same winter week are depicted in Figure 5.2. While this extension does not lead to significant changes of the thermal units, it allows to largely reduce the amount of energy bought at the spot market.

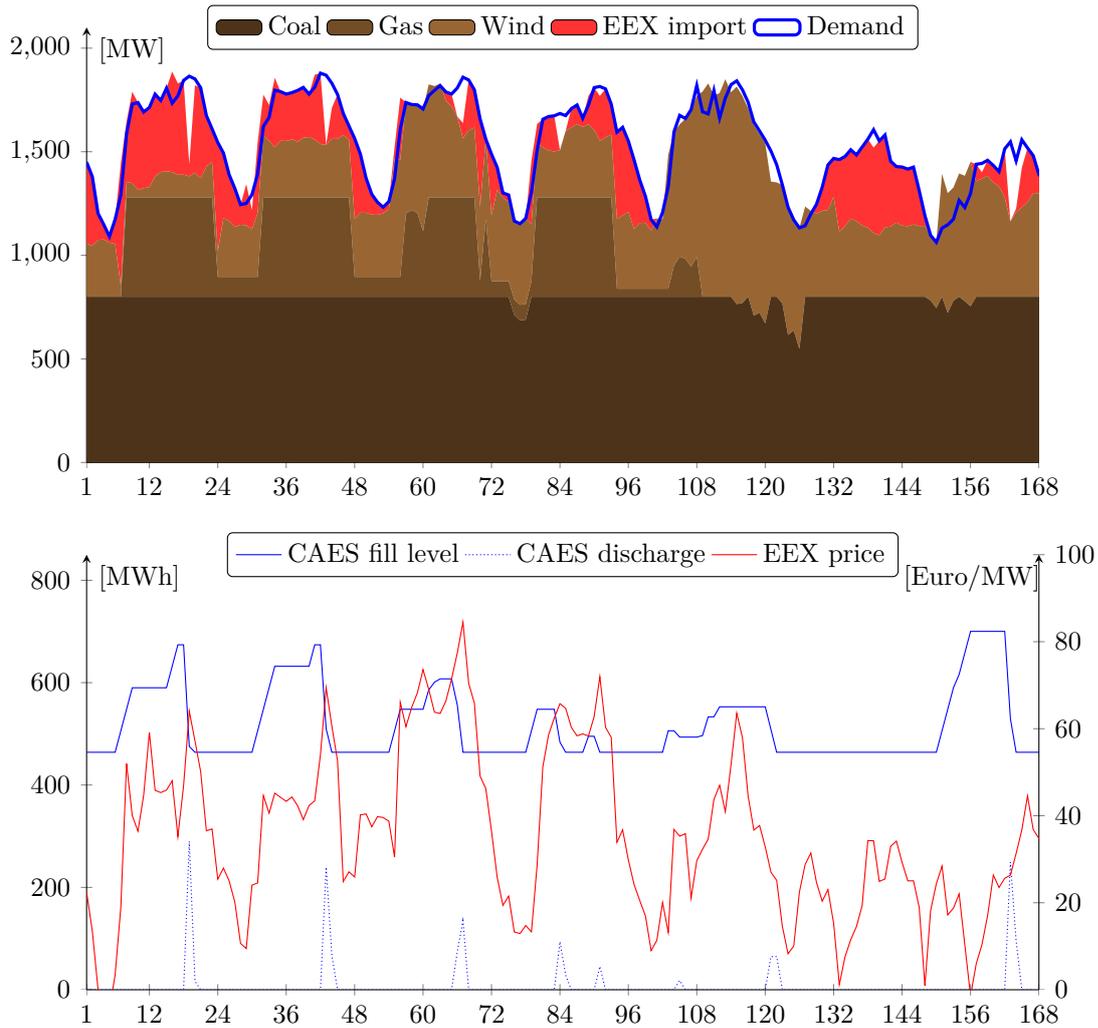


Figure 5.2.: Optimal power scheduling and CAES operation in a winter week with doubled wind power capacity.

Figure 5.2 also shows the operation of the CAES storage in the course of the week. Again, the storage is mainly used at peak times to avoid expensive imports from the spot market. Even though the wind energy input has been doubled, the storage levels are comparable with those in the reference scenario. Thus, the use of storages seems to depend mainly on the development of the electricity price at the EEX and only secondarily on the available wind energy.

The optimization problem was solved further times with varying quantities of installed wind power and storage capacities. Figure 5.3 shows the relative reduction of costs that can be achieved by the use of storage systems of different dimensions, where a model without storages generates operating costs of 100%. A storage system dimension of  $y$  corresponds to  $y$  times the dimension of the base setting. The results clearly show, that

## 5. Optimization of Dispersed Energy Supply

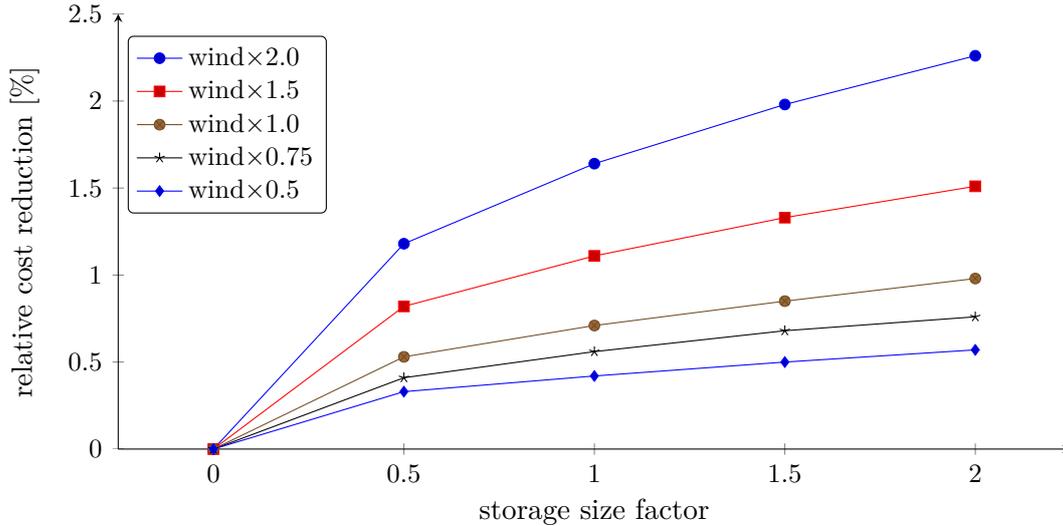


Figure 5.3.: Reduction of minimal expected costs depending on the storage capacity installed for different installations of wind power capacities.

the relative cost reduction due to storage use is the highest in the twice-wind-setting, and in all settings the most prevailing gradient is between no use and the use of the half dimension of storage sizes.

Hence, there obviously exist optimal storage sizes for this application. However, calculations that include the investment costs for storages have shown that using storages solely for costs minimization is not economical. But notice that in this model, no export of electrical energy was allowed. Thus, in the next section, we will discuss an extended model for determining the optimal size of an energy storage, which includes both operation and investment costs and includes the possibility for selling energy at the EEX.

## 5.3. Expansion Planning

### 5.3.1. Model

To investigate the economics of storage technologies over a long time horizon, the model from Section 5.2.1 has been extended by the possibility to invest in storage technologies and power plants. Since for the evaluation of investment decisions it is not sufficient to consider only a single year as optimization horizon, a longer time horizon of several years is represented by selected years  $Y$  and typical months  $M$ . Therefore, in the following, the set  $[T]$  corresponds to the hourly discretization of one month and additional indices  $y$  and  $m$  are used to indicate the year and month.

Main differences to the operation model are in the objective function and the capacity restrictions. Next to the operation costs, the objective function now also includes the investment costs for additional units. Since for a longer time horizon, operational costs from different time periods have to be considered, they have to be scaled to a uniform

pricing point by using discounting factors<sup>1</sup>. Additional notation used for the expansion model is summarized in Table 5.2. The new objective function has the form

$$TC = \sum_{y \in Y} DF_y CF_y^{\text{op}} \left[ \sum_{m \in M} f_m \sum_{t=1}^T \left( IC_{y,m,t} + \sum_{u \in U} (OC_{y,m,t,u} + SC_{y,m,t,u}) \right) \right] \\ + \sum_{y \in Y} DF_y CF_y^{\text{exp}} AF \left[ \sum_{u \in U} c_u^{\text{inv}} Z_{y,u} + \sum_{u \in U_{\text{st}}} c_u^{\text{inv, st}} Z_{y,u}^{\text{st}} \right],$$

where the variable  $Z_{y,u}$  denotes additionally installed capacity for unit  $u \in U$  in year  $y \in Y$  and  $Z_{y,u}^{\text{st}}$  denotes additionally installed storage capacity for storage unit  $u \in U_{\text{st}}$  in year  $y \in Y$ . While we allow a continuous increase for storage capacities ( $Z_{y,u}^{\text{st}} \in \mathbb{R}_+$ ), power plants are allowed to be build only as a whole, i.e.,  $Z_{y,u}$  is restricted to a given set of discrete values<sup>2</sup>. To reflect newly build capacities in the operational part of the model, the restrictions (5.7) and (5.10) are modified to

$$Q_{y,m,t,u} \leq \bar{Q}_{t,u} + \sum_{y' \in Y: y' \leq y} Z_{y',u} \quad (u \in U, t \in [T], m \in M, y \in Y)$$

and

$$\underline{H}_u \leq H_{y,m,t,u} \leq \bar{H}_u + \sum_{y' \in Y: y' \leq y} Z_{y',u}^{\text{st}} \quad (u \in U_{\text{st}}, t \in [T], m \in M, y \in Y).$$

The amount of exported (sold) energy is modeled by an additional variable  $Q_{y,m,t}^{\text{exp}}$ , so that Equation (5.2) now takes the form

$$IC_{y,m,t} = c_{y,m,t}^{\text{imp}} Q_{y,m,t}^{\text{imp}} - c_{y,m,t}^{\text{exp}} Q_{y,m,t}^{\text{exp}} \quad (t \in [T], m \in M, y \in Y).$$

The gain from exporting energy depends on the EEX price, so that we have chosen

<sup>1</sup> The *discount factor* scales the costs of a future investment to the current time. That is, assuming a yearly interest rate of  $r$ , the cost of an investment in  $n$  years has to be divided by  $(1+r)^n$  to yield its current value. The cash value computes the value of yearly investments  $w$  over a time horizon of  $n$  years, which is  $\sum_{j=0}^{n-1} (1+r)^j w = w \frac{(1+r)^n - 1}{r}$ , scaled to the current value by multiplication with the discount factor. The corresponding factor  $\frac{(1+r)^n - 1}{r} \frac{1}{(1+r)^n} = \frac{1}{r} \left(1 - \frac{1}{(1+r)^n}\right)$  is called the *cash value factor*. Finally, the *annuity factor* allows to calculate the cost of an investment with lifetime  $n$ , if the cost is distributed over the whole lifetime and interest rates are taken into account. Thus, the annuity factor is given as the reverse of the cash value factor. Therefore, for the parameters in the objective function, we obtain  $DF_y = \frac{1}{(1+r)^{y-y_0}}$  to scale a value from year  $y$  to the current year  $y_0$ ,  $CF_y^{\text{op}} = \frac{1}{r} \left(1 - \frac{1}{(1+r)^{f_y}}\right)$  to scale the operation costs for a time horizon of  $f_y$  years to the beginning of year  $y$ ,  $CF_y^{\text{exp}} = \frac{1}{r} \left(1 - \frac{1}{(1+r)^{y_e-y}}\right)$  to scale the investment costs for a unit that is build in year  $y$  and which costs are considered until the end of the time horizon, denoted by  $y_e$ , and  $AF = \frac{r}{1 - \frac{1}{(1+r)^{\ell}}}$ .

<sup>2</sup> The discreteness restriction on the variables  $Z_{y,u}$  (modeled by integer variables) are no problem for the applied decomposition algorithm, since these variables appear only on the first stage, which renders the corresponding master problem to be a mixed-integer linear program. Due to the small number of discrete variables, this mixed-integer linear program is still easy to solve.

## 5. Optimization of Dispersed Energy Supply

Variables	$Z$	plant capacity expansion
	$Z^{\text{st}}$	storage capacity expansion
	$Q^{\text{exp}}$	exported energy
Parameters	$c^{\text{inv}}$	plant capacity expansion costs
	$c^{\text{inv,st}}$	storage capacity expansion costs
	$c^{\text{exp}}$	export price
	$DF$	discounting factor
	$CF^{\text{op}}$	cash value factor for operation
	$CF^{\text{exp}}$	cash value factor for expansion
	$AF$	annuity factor
	$f$	frequency of month in year or year in time horizon
	$\ell$	life time of power unit

Table 5.2.: Additional notation used by expansion planning model.

$c^{\text{exp}} = 0.99c^{\text{imp}}$ , which takes transaction costs into account. Finally, Equation (5.6) for covering the demand is modified to

$$\sum_{u \in U_{\text{pow}}} Q_{y,m,t,u} + Q_t^{\text{wind}} + Q_{y,m,t}^{\text{imp}} \geq D_{y,m,t} + \sum_{u \in U_{\text{st}}} Q_{y,m,t,u} + Q_{y,m,t}^{\text{exp}} \quad (t \in [T], m \in M, y \in Y).$$

### 5.3.2. Case Study

As time horizon for the expansion model we consider the years 2010–2025, where every fifth year and four typical months are selected for optimization ( $Y = \{2010, 2015, 2020\}$ ,  $M = \{\text{march, july, october, december}\}$ ). A more detailed representation of the time horizon seemed to be numerically too demanding. For expansion, three gas turbines with different capacity and the storage types CAES and PSW are available. The investments in storages are divided in those for storage capacity  $Z_u^{\text{st}}$ , i.e., the size of a cavern or basin, and those for the units to operate the storages  $Z_u$ ,  $u \in U_{\text{st}}$ .

In the first year, all power plants from the operation model (c.f. Section 5.2.2) are available, i.e., one hard coal power plant, two gas turbines, and wind energy, but no storages. The expected future development of primary energy costs, EEX prices, and demand are incorporated via progression rates according to International Energy Agency [2006]. The development of wind energy is modeled such that in the first year, onshore wind energy contributes only 17% to the power supply. In the year 2015, on- and offshore wind energy contribute together approximately 28%, which increases to approximately 38% for 2020. Both types of wind energy (on- and offshore) as well as the correlated EEX price contribute for the simulation of an initial set of scenarios, see also Section 5.2.2. With the algorithm sketched in Section 4.5, one recombining scenario tree was constructed for each typical month of each selected year. These trees branch three times a day and recombine into two different subtrees once a day.

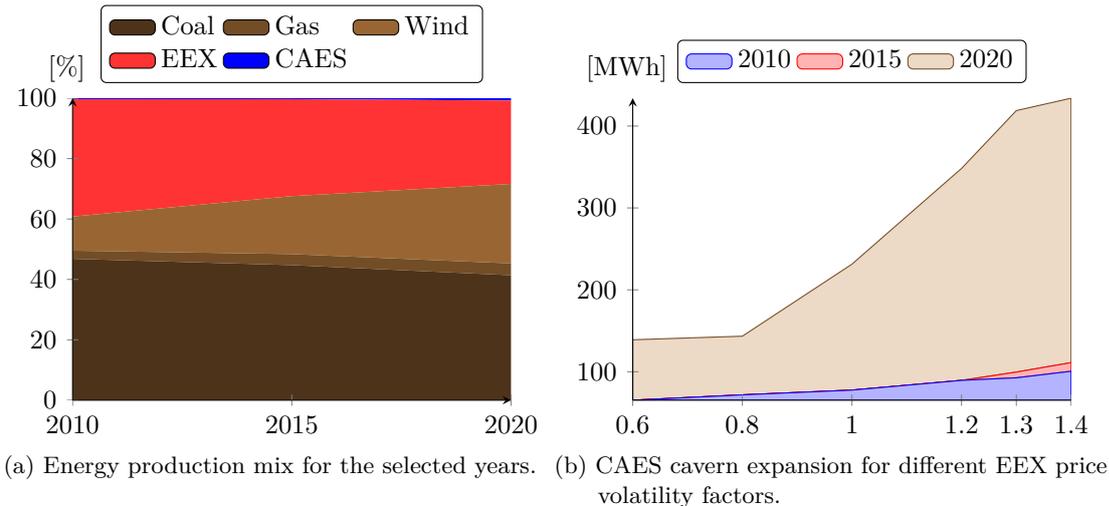


Figure 5.4.: Illustration of numerical results for expansion model.

### 5.3.3. Numerical Results

The development of the energy generation mix for the selected years is shown in Figure 5.4a. The increase in wind energy over the years results in a considerable reduction of importing energy from the EEX. At the same time, the production of the existing gas power plants increases by two third from 2010 to 2020, but no new gas turbines are build over the whole time horizon. The share of the compressed air storage on the overall production is below 1%, even though the use of storages is more than doubled from 2010 to 2020. After two expansion steps, the cavern has in 2020 approximately half of the size of the compressed air storage in Huntorf, which was used in the operation model from Section 5.2.2. Compressors and turbine have approximately two third of the capacity of the corresponding units in Huntorf. Note further, that investments are only done for a compressed air storage, but not for a pumped hydro storage.

The results indicate, that volatility of EEX prices are a deciding factor for the use and economic efficiency of storage technologies and thus also influences investments into these. Hence, to investigate the relationship between build storages and EEX price volatility, we varied the spread of the price scenarios around the yearly average value by factors between 0.6 and 1.4 and computed corresponding optimal expansions and operations. Figure 5.4b shows the expansions of the CAES cavern for the different volatility factors.

It is clearly seen, that an increase of energy price volatility makes investments in larger storage capacities economically more attractive. The tendency for substantial expansions of storage capacity not before 2020 is evident for all volatility factors. For volatility factors larger than 1.4, the investments into storage capacity increase even more, but also additional gas turbines are build. Thus, a high volatility in the EEX prices makes storages and fast power plants necessary for a market oriented energy production. In the considered model, for each variation a reduction of the overall costs by approximately 1% could be achieved by building and operation energy storages.

## 5.4. Conclusions

We investigated the economic use of energy storages in the context of increasing generation of wind energy. On the one hand, the additional benefit by temporal decoupling supply and demand of electrical energy was analyzed. It has been shown, that the optimal use of energy storages depends on the market price and does not primarily avoid a partial load operation of conventional power plants. Furthermore, storage technologies were analyzed as investment possibility for future energy systems that include energy trading. It has been seen, that an increasing amount of wind energy is advantageous for investments into energy storages, but a larger influence is made by varying volatility of energy prices, i.e., an expected larger volatility of energy prices (due to increasing use of wind energy) benefits investments in storage technologies additionally.

Finally, we note that in this chapter only a small energy supply system (consisting of only three power plants) has been considered. Thus, additional investigations are necessary to see whether the obtained results can be carried over to larger systems. For example, Epe [2011] has shown that for an extension of the model from this chapter to an aggregated system of all existing power plants in Germany, investments in both storage technologies take only place if the share of electricity from wind energy reaches 70% of the electricity demand.

**Part II.**

**Mixed Integer Nonlinear  
Programming**



## 6. Introduction

Nonlinear optimization problems containing both discrete and continuous variables are called *mixed-integer nonlinear programs* (MINLPs). Such problems arise in many fields, such as energy production and distribution, logistics, engineering design, manufacturing, and the chemical and biological sciences [Floudas, 1995, Grossmann and Kravanja, 1997, Tawarmalani and Sahinidis, 2002b, Pintér, 2006, Ahadi-Oskui et al., 2010].

A general MINLP can be formulated as

$$\min\{f(x) : x \in X\} \quad (6.1a)$$

with

$$X := \{x \in [\underline{x}, \bar{x}] : Ax \leq b, g(x) \leq 0, x_i \in \mathbb{Z}, i \in I\}, \quad (6.1b)$$

where  $\underline{x}, \bar{x} \in \overline{\mathbb{R}}^n$  determine the *lower and upper bounds* on the variables ( $\overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ ), the matrix  $A \in \mathbb{R}^{m' \times n}$  and the vector  $b \in \mathbb{R}^{m'}$  specify the *linear constraints*,  $I \subseteq [n]$ <sup>1</sup> denotes the set of variables with *integrality requirement*,  $f : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}$  is the *objective function*, and  $g : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}^m$  are the *constraint functions*. Here and in the following, we denote by  $[\underline{x}, \bar{x}] := \{x \in \mathbb{R}^n : \underline{x}_i \leq x_i \leq \bar{x}_i, i \in [n]\}$  the *box* for the variables and by  $x_J := (x_j)_{j \in J}$  the subvector of  $x$  for some index set  $J \subseteq [n]$ . The set  $X$  is called *feasible set* of (6.1). The restriction to inequality constraints is only for notational simplicity.

In (6.1), we assumed that  $[\underline{x}, \bar{x}]$  is contained in the domain<sup>2</sup> of the functions  $f(x)$  and  $g(x)$ . Certain solution algorithms allow to relax this assumption by requiring  $f(x)$  and  $g(x)$  to be defined only for points that additionally satisfy the linear constraints, i.e.,  $\text{dom } f(x), \text{dom } g(x) \subseteq \{x \in [\underline{x}, \bar{x}] : Ax \leq b\}$ . Further, we assume  $f(x)$  and  $g(x)$  to be at least continuous. Efficient solution algorithm often require continuous differentiability.

A point  $x \in X$  is called *local optimum*, if there exists an  $\varepsilon > 0$  such that for all  $y \in X$  with  $\|x - y\| < \varepsilon$  we have  $f(x) \leq f(y)$ . A local optimum which objective function value equals the optimal value is called a *global optimum*. Note, that due to continuity of  $f(x)$  and  $g(x)$ , there always exists a global optimum of (6.1), if its optimal value is finite.

If the functions  $f(x)$  and  $g_j(x)$ ,  $j \in [m]$ , are convex on  $[\underline{x}, \bar{x}]$ , we say that (6.1) is a *convex* MINLP. Note, that convexity of  $g_j(x)$ ,  $j \in [m]$ , implies convexity of the feasible set, but not vice versa<sup>3</sup>. A convex MINLP has the important property, that for fixed discrete variables  $x_i$ ,  $i \in I$ , every local minimum of (6.1) is also a global minimum.

<sup>1</sup>As in Part I, we let  $[n : m] := \{n, n+1, \dots, m\}$  for  $n, m \in \mathbb{N}$  with  $n \leq m$  and  $[m] := [1 : m]$  for  $m \in \mathbb{N}$ .

<sup>2</sup>The domain of a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is the set of point on which  $f(x)$  is defined and takes a finite value,  $\text{dom } f := \{x \in \mathbb{R}^n : f(x) \in \mathbb{R}\}$ .

<sup>3</sup>Sometimes, also MINLPs with convex objective function and convex feasible set are denoted as convex MINLP. However, within this thesis, we use the more restrictive formulation that requires convexity of the objective and all constraint functions.

## 6. Introduction

abbrev.	full name	restriction of (6.1)
LP	linear program	$m = 0$ , $f(x)$ linear, $I = \emptyset$
QP	quadratic program	$m = 0$ , $f(x)$ quadratic, $I = \emptyset$
NLP	nonlinear program	$I = \emptyset$
MIP	mixed-integer linear program	$m = 0$ , $f(x)$ linear
MIQP	mixed-integer quadratic program	$m = 0$ , $f(x)$ quadratic
MIQCP	mixed-integer quadratically constrained program	$f(x)$ linear, $g_j(x)$ quadratic, $j \in [m]$
MIQQP	mixed-integer quadratically constrained quadratic program	$f(x)$ and $g_j(x)$ quadratic, $j \in [m]$

Table 6.1.: Subclasses of MINLP, defined as restriction of (6.1).

By restricting nonlinearity of  $f(x)$  and  $g(x)$  or integrality requirement in (6.1), one obtains several important subclasses of MINLP, which are summarized in Table 6.1. We call a function  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  *linear*, if it can be written as  $h(x) = \langle c, x \rangle$  for some  $c \in \mathbb{R}^n$ , and call it *quadratic*, if it can be written as  $h(x) = \langle x, Qx \rangle + \langle q, x \rangle + \bar{q}$  for some  $Q \in \mathbb{R}^{n \times n}$ ,  $q \in \mathbb{R}^n$ , and  $\bar{q} \in \mathbb{R}$ . W.l.o.g., we assume  $Q$  to be symmetric. Note, that a quadratic function is convex if and only if  $Q$  is positive-semidefinite, i.e.,  $\langle x, Qx \rangle \geq 0$  for all  $x \in \mathbb{R}^n$ .

The combination of discrete decisions, nonlinearity, and possible nonconvexity of the nonlinear functions in MINLP combines the areas of mixed-integer linear programming, nonlinear programming, and global optimization into a single problem class. While linear and convex nonlinear programs are solvable in polynomial time<sup>4</sup> in theory [Khachiyan, 1979, Vavasis, 1995] and very efficiently in practice [Bixby, 2002, Nocedal and Wright, 2006], nonconvexities as imposed by discrete variables or nonconvex nonlinear functions easily lead to problems that are NP hard in theory and computationally demanding in practice. However, substantial progress has been made in the solvability of mixed-integer linear programs [Bixby et al., 2000]. As a consequence, state-of-the-art MIP solvers are nowadays capable of solving a variety of MIP instances arising from real-world applications within reasonable time [Koch et al., 2011]. On the other side, also global optimization has been a field of active research and development, see, e.g., the textbooks Horst and Tuy [1990], Horst and Pardalos [1995], Floudas [2000], and Tawarmalani and Sahinidis [2002b] and the survey papers Neumaier [2004], Floudas et al. [2005], and Gounaris and Floudas [2009].

Since its beginning in the mid 1970's [Beale, 1980, Forrest and Tomlin, 2007], the integration of MIP and global optimization of NLPs and the development of new algorithms unique for MINLP has done a remarkable progress, see also the recent book edited by Lee

<sup>4</sup> Polynomial time solvability for LP and convex NLP can be proven via the Ellipsoid method, which encloses the minimizer of a convex function by a sequence of ellipsoids of uniformly decreasing volume, see also Yudin and Nemirovski [1976]. For the case of minimizing a differentiable convex function  $f(x)$  over the unit cube in  $\mathbb{R}^n$ , Theorem 2 in Vavasis [1995] states that for a given  $\varepsilon > 0$  a point  $x \in [0, 1]^n$  can be found in  $2n(n+1)(\ln(p/\varepsilon) + \ln(n)) + 2$  iterations such that  $f(x) \leq f(x^*) + \varepsilon$ , where  $p$  is an upper bound on the range of  $f(x)$  on  $[0, 1]^n$  and  $x^*$  is a global optimum.

and Leyffer [2012] and the survey papers by Burer and Letchford [2012] and Belotti et al. [2012]. While the integration of nonlinear aspects into a MIP solver often accounts at first only to the easier case of convex nonlinearities [Bonami et al., 2008, Abhishek et al., 2010, IBM, 2012, FICO, 2008], discrete decision variables are integrated into a global optimization solver often by a simple extension of an already existing branch-and-bound algorithm. The latter is then gradually extended by more advanced MIP machinery (presolving, cutting planes, branching rules, ...). Even though not competitive with MIP, yet, there exists nowadays a variety of general purpose software packages for the solution of medium-size MINLPs, see Bussieck and Vigerske [2010] and Section 6.2.

In the following, we introduce the main ideas for solving MINLPs or important subclasses and give an overview on available solver software. Afterwards, we discuss the implementation of a MINLP solver within the constraint integer programming framework SCIP and present a computational study on the performance of our implementation.

## 6.1. Algorithmic Concepts

In the following, we review some established solution techniques for solving convex and nonconvex MINLPs. We only consider deterministic and complete methods that either proof infeasibility of a MINLP or compute within finite time a feasible solution which objective function value is at most a given epsilon worse than the optimal value. When terminated before completion, these algorithms usually provide a lower bound on the optimal value, which also certifies the quality of a feasible solution, if any has been found.

A common methodology for handling nonconvexities is branch-and-bound [Land and Doig, 1960], where an optimization problem is successively divided into smaller subproblems until the individual subproblems are easy to solve. Further, bounding is used to decide early whether improving solutions can be found in a subproblem. Thereby, bounds are computed from some relaxation of the current subproblem. All algorithms presented combine an outer-approximation obtained by relaxing nonconvex nonlinear constraints, all nonlinear constraints, and/or integrality requirements with a branch-and-bound search. The main differences are in the choice of the type of constraints to be relaxed and in the interaction between updating an outer-approximation and employing a branch-and-bound method. A further common characteristic of all methods is that the solution of a LP, a convex NLP, and sometimes also a MIP, is considered to be “cheap” in comparison to the effort for solving a MINLP or nonconvex NLP. Thus, LP, convex NLP, or MIP solving is used as a subroutine that is called occasionally for the solution of a single MINLP.

The following survey reviews the basic algorithms that have already been employed in state-of-the-art solver software (which will be reviewed in Section 6.2) or have been developed recently. Some of the methods not discussed here are, e.g., the use of inner-approximations via column generation [Nowak, 2005], the use of conic relaxations other than LPs [Drewes, 2009, Bao et al., 2011], a-priori approximation of a MINLP by a “similar” MIP [Geißler et al., 2012], or methods that apply only to pure integer nonlinear programs ( $I = [n]$ ) [Li and Sun, 2005, Hemmecke et al., 2010].

Before discussing algorithms for general MINLPs, we first restrict ourselves to the

## 6. Introduction

convex case. Some of these methods build the foundation of algorithms for general MINLPs that will be presented subsequently.

### 6.1.1. Convex MINLP

Since MIP and convex NLP can be solved efficiently, the development of MINLP algorithm has first focused on the case of convex MINLPs. It is thus not surprising, that many solution algorithms decompose a convex MINLP into a MIP and a convex NLP. The following presentation is based on the excellent surveys Grossmann [2002], and Bonami, Kılınç, and Linderoth [2010]. We assume that the nonlinear functions  $f(x)$  and  $g(x)$  are continuously differentiable on their domain and that (6.1) is bounded from below.

Consider the following NLP problems related to (6.1): a *NLP relaxation* obtained by replacing the integrality requirement  $x_I \in \mathbb{Z}^{|I|}$  by a restriction of  $x_I$  to some box  $[\underline{x}'_I, \bar{x}'_I]$ ,

$$\min\{f(x) : x \in [\underline{x}, \bar{x}], x_I \in [\underline{x}', \bar{x}'], Ax \leq b, g(x) \leq 0\}, \quad (6.2)$$

a *NLP subproblem* obtained by fixing the integer variables  $x_I$  to a given vector  $\hat{x}_I \in \mathbb{R}^{|I|}$ ,

$$\min\{f(x) : x \in [\underline{x}, \bar{x}], x_I = \hat{x}_I, Ax \leq b, g(x) \leq 0\}, \quad (6.3)$$

and a *NLP infeasibility minimization problem* obtained by fixing the integer variables  $x_I$  to a given vector  $\hat{x}_I \in \mathbb{Z}^{|I|}$  and adding slack variables for the nonlinear constraints,

$$\min\{u \in \mathbb{R}_+ : x \in [\underline{x}, \bar{x}], x_I = \hat{x}_I, Ax \leq b, g_j(x) \leq u, j \in [m]\}. \quad (6.4)$$

Note, that (6.2) is a convex NLP. Its optimal value yields a lower bound on the optimal value of (6.1) for  $\underline{x}'_I = \underline{x}_I$  and  $\bar{x}'_I = \bar{x}_I$ . Further, the feasible solutions of the convex NLP (6.3) are also feasible for (6.1) and thus yield an upper bound on the optimal value of (6.1). If (6.3) has no feasible solution, but all linear and box constraints can be satisfied, then (6.4) gives a solution with minimal maximal violation of the nonlinear constraints.

Further, we consider a MIP relaxation of (6.1) obtained by linearizing nonlinear functions. Note, that due to convexity and differentiability assumptions, it holds that

$$\begin{aligned} f(x) &\geq f(\hat{x}) + \langle \nabla f(\hat{x}), x - \hat{x} \rangle && (\hat{x} \in \text{dom } f(x)), \\ g_j(x) &\geq g_j(\hat{x}) + \langle \nabla g_j(\hat{x}), x - \hat{x} \rangle && (\hat{x} \in \text{dom } g_j(x)). \end{aligned}$$

Thus, linearization of  $f(x)$  yields an underestimation of  $f(x)$ , while replacing  $g_j(x)$  in  $X$  by linearizations yields an overestimation of  $X$ , as illustrated in Figure 6.1. Hence, for a given set of points  $K \subseteq \mathbb{R}^n$ , the following *MIP relaxation* of (6.1) can be constructed:

$$\min \left\{ \alpha \in \mathbb{R} : \begin{array}{l} Ax \leq b, \\ f(\hat{x}) + \langle \nabla f(\hat{x}), x - \hat{x} \rangle \leq \alpha, \quad \hat{x} \in K, \\ g_j(\hat{x}) + \langle \nabla g_j(\hat{x}), x - \hat{x} \rangle \leq 0, \quad j \in [m], \hat{x} \in K, \\ x \in [\underline{x}, \bar{x}], x_I \in \mathbb{Z}^{|I|}. \end{array} \right\} \quad (6.5)$$

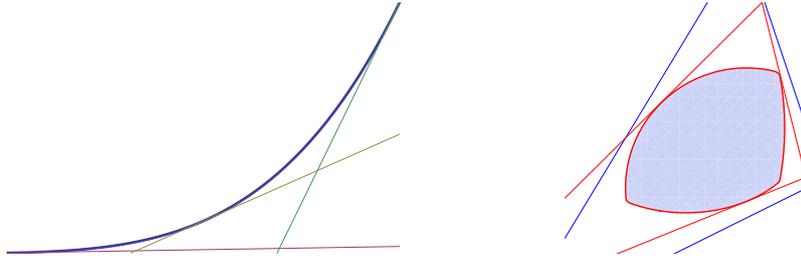


Figure 6.1.: Underestimation of convex function and overestimation of convex region as in MIP relaxation (6.5).

Varying the strategy when to solve the NLP problems (6.2)–(6.4) and the MIP relaxation (6.5), one obtains several algorithms for solving (6.1).

### NLP-based Branch-and-Bound

Observe, that the NLP relaxation (6.2) is analogous to the LP relaxation employed in branch-and-bound algorithms for MIP [Land and Doig, 1960]. That is, the optimal value of (6.2) yields a lower bound on (6.1) if  $x_I$  is restricted to be within  $[\underline{x}'_I, \bar{x}'_I]$ . If an optimal solution of (6.2) satisfies the integrality requirement  $x_I \in \mathbb{Z}^{|I|}$ , then this solution is also feasible for (6.1) and optimal for (6.1) with  $x_I$  restricted to  $[\underline{x}'_I, \bar{x}'_I]$ . These bounding properties of (6.2) can be used to develop an analog of MIP branch-and-bound algorithms for convex MINLP [Gupta and Ravindran, 1985, Borchers and Mitchell, 1994, Leyffer, 2001], see Algorithm 6.1.

As an efficient branch-and-bound algorithm for MIP relies on a fast solver for LP relaxations, so does the NLP-based branch-and-bound for MINLP relies on the availability of a fast NLP solver. Especially warm-starting capabilities are important, since successive NLPs typically differ only in the bound of a variable that has been branched on. Leyffer [2001] and Bonami et al. [2011] have shown that sequential quadratic programming solvers are well-suited, but also for interior point solvers, warm-starting capabilities are currently improved [Benson, 2011]. The practical performance of Algorithm (6.1) further relies on good strategies for selecting the next node  $[\underline{x}'_I, \bar{x}'_I]$  to process and the fractional variable to branch on, see also Section 6.1.4.

### Outer-Approximation

Outer-approximation algorithms for (6.1) rely on the following property of the MIP relaxation (6.5):

**Theorem 6.1** (Duran and Grossmann [1986], Fletcher and Leyffer [1994], Bonami et al. [2010]). *Let  $X_L := \{x \in [\underline{x}, \bar{x}] : Ax \leq b\}$ . Assume  $X_L \neq \emptyset$ ,  $f(x)$  and  $g(x)$  are convex and continuously differentiable, and that a constraint qualification holds. Let  $K \subseteq X_L$  be such that for any  $\hat{x} \in K$ , it is  $\hat{x}_I \in \mathbb{Z}^{|I|}$  and  $\hat{x}$  is an optimal solution of (6.3), if feasible, or an optimal solution of (6.4), otherwise. Then the optimal value of (6.5) equals the optimal value of (6.1) and every optimal solution of (6.1) is also optimal for (6.5).*

---

**Algorithm 6.1:** Basic NLP-based branch-and-bound algorithm for solving (6.1)

---

```

 $\mathcal{L} \leftarrow \{[\underline{x}_I, \bar{x}_I]\};$ 
 $\bar{v} \leftarrow \infty;$ 
while  $\mathcal{L} \neq \emptyset$  do
    select  $[\underline{x}'_I, \bar{x}'_I]$  from  $\mathcal{L}$ ;  $\mathcal{L} \leftarrow \mathcal{L} \setminus \{[\underline{x}'_I, \bar{x}'_I]\};$ 
    solve relaxation (6.2) w.r.t.  $[\underline{x}'_I, \bar{x}'_I]$ ;
    if (6.2) is infeasible then continue ; /* node is infeasible */
     $\hat{x} \leftarrow$  optimal solution of (6.2);
    if  $f(\hat{x}) \geq \bar{v}$  then continue ; /* no better solution within  $[\underline{x}'_I, \bar{x}'_I]$  */
    if  $\hat{x}$  is feasible for (6.1) then
         $\bar{v} \leftarrow f(\hat{x})$  ; /* update upper bound (new incumbent) */
        continue;
    end
    select  $i^* \in I$  with  $\hat{x}_{i^*} \notin \mathbb{Z}$ ;
     $\bar{x}''_i \leftarrow \begin{cases} \bar{x}'_i, & i \neq i^* \\ \lfloor \hat{x}_{i^*} \rfloor, & i = i^* \end{cases}$ ,  $\underline{x}''_i \leftarrow \begin{cases} \underline{x}'_i, & i \neq i^* \\ \lceil \hat{x}_{i^*} \rceil, & i = i^* \end{cases}$ ;
     $\mathcal{L} \leftarrow \mathcal{L} \cup \{[\underline{x}'_I, \bar{x}''_I], [\underline{x}''_I, \bar{x}'_I]\}$  ; /* branch on  $x_{i^*}$  */
end
output : optimal value of (6.1) is  $\bar{v}$ 

```

---

Thus, the idea of the outer-approximation algorithm by Duran and Grossmann [1986] is to start with a simple MIP relaxation (6.5) that contains linearizations for only a few points (i.e.,  $K$  small). Then, we alternate between solving the MIP relaxation (6.5) to obtain points that satisfy the integrality requirements and solving the NLP subproblems (6.3) and (6.4) that yield new linearizations<sup>5</sup> (i.e., augment  $K$ ) and solutions that are feasible for (6.1). Note, that (6.4) is always feasible for a fixation of  $x_I$  to a point in the projection of  $X_L$  onto the  $x_I$ -coordinates. The method is summarized in Algorithm 6.2. If  $X_L$  is bounded, then Theorem 6.1 ensures that Algorithm 6.2 terminates in a finite number of steps with proving either infeasibility of (6.1) or computing its optimal value.

Note, that solving the MIP outer-approximation to optimality in every iteration of Algorithm 6.2 may computationally be too expensive. However, computing some feasible solution with objective function value below the current upper bound  $\bar{v}$  is often sufficient.

**Generalized Benders Decomposition.** Note, that the outer-approximation algorithm is similar to a Benders Decomposition algorithm [Benders, 1962], where the MIP approximation takes the role of the master problem and the NLP (6.3) takes the role of the subproblem. The main difference to a Benders Decomposition is the presence of continuous variables in the MIP approximation (6.5). A variant that works with a master problem consisting of discrete variables only has been suggested by Geoffrion [1972]. As

---

<sup>5</sup>In the right picture of Figure 6.1, the red hyperplanes correspond to the linearizations created by the algorithm of Duran and Grossmann [1986], i.e., they are always tight w.r.t. the nonlinear constraints.

**Algorithm 6.2:** Outer-Approximation algorithm for solving (6.1)

---

```

 $v \leftarrow -\infty; \bar{v} \leftarrow \infty;$ 
solve NLP relaxation (6.2);
if (6.2) is infeasible then STOP: (6.1) is infeasible;
 $K \leftarrow \{\hat{x}\}$ , where  $\hat{x}$  is an optimal solution of (6.2);
while  $v < \bar{v}$  do
  solve MIP relaxation (6.5);
  if (6.5) is infeasible then STOP: (6.1) is infeasible;
   $\hat{x} \leftarrow$  optimal solution of (6.5);
   $v \leftarrow$  optimal value of (6.5); /* update lower bound */
  solve NLP (6.3) w.r.t. fixation  $\hat{x}_I$ ;
  if (6.3) is feasible then
     $\tilde{x} \leftarrow$  optimal solution of (6.3);
     $\bar{v} \leftarrow \min(\bar{v}, f(\tilde{x}))$ ; /* update upper bound (incumbent) */
  else
    solve NLP (6.4) w.r.t. fixation  $\hat{x}_I$ ;
     $\tilde{x} \leftarrow$  optimal solution of (6.4);
  end
   $K \leftarrow K \cup \{\tilde{x}\}$ ; /* improve outer-approximation */
end
output: optimal value of (6.1) is  $\bar{v}$ 

```

---

in Benders Decomposition, see also Section 3.1.1, the idea is to construct optimality and feasibility cuts from a dual solution of (6.3) and (6.4), respectively. The cuts are used to build a MIP outer-approximation that involves only the integer variables  $x_I$ , see also Geoffrion [1972] and Bonami et al. [2010]. The reduction in the number of variables, when compared to the outer-approximation algorithm of Duran and Grossmann [1986], comes with usually weaker lower bounds, since the constructed optimality and feasibility cuts are aggregations of linearizations that are added in the outer-approximation algorithm.

**Extended Cutting Plane Algorithm.** The solution of NLP subproblems in the outer-approximation algorithm (Algorithm 6.2) can be omitted, if one uses the solution of the MIP relaxation as reference point for linearization of  $f(x)$  and  $g(x)$  instead<sup>6</sup>. This method, as proposed by Westerlund and Petterson [1995] and summarized in Algorithm 6.3, is similar to Kelley’s cutting plane method for convex NLPs [Kelley, 1960]. Since it totally relies on an outer-approximation, it guarantees to find only an  $\varepsilon$ -optimal solution within finite time, that is, a solution that satisfies the nonlinear constraints within an  $\varepsilon$ -tolerance and has an objective function value that is at most  $\varepsilon$  below the optimal value. Note, that it is not necessary to add linearizations for all nonlinear functions in each

---

<sup>6</sup>In the right picture of Figure 6.1, the blue hyperplanes correspond to the linearizations created by the extended cutting plane algorithm. That is, they may not be as close to the feasible set as those from the outer-approximation algorithm.

---

**Algorithm 6.3:** Extended Cutting Plane algorithm for solving (6.1)

---

```

input : convergence tolerance  $\varepsilon$ 
 $K \leftarrow \emptyset$ ;
loop
  solve MIP relaxation (6.5);
  if (6.5) is infeasible then STOP: (6.1) is infeasible;
  let  $(\hat{x}, \hat{\alpha})$  be an optimal solution of (6.5);
  if  $f(\hat{x}) - \hat{\alpha} < \varepsilon$  and  $g_j(\hat{x}) < \varepsilon, j \in [m]$  then STOP:  $\hat{x}$  is  $\varepsilon$ -optimal for (6.1);
   $K \leftarrow K \cup \{\hat{x}\}$ ; /* improve outer-approximation */
end;
```

---

iteration. Instead, only linearizations for functions that are (most) violated in the current solution of the MIP relaxation are added.

The reduced computational effort due to omitting solution of NLP subproblems in Algorithm 6.3 usually comes with a larger number of iterations, since the generated cuts are weaker than those in Algorithm 6.2. Further, the absence of an upper bound or incumbent solution requires to run the algorithm until it terminates, even if one is only interested in a good feasible solution of the MINLP. Thus, implementations also solve NLP subproblems from time to time to obtain a feasible solution and generate some tighter linearizations [Westerlund and Lundquist, 2003].

### LP/NLP-based Branch-and-Bound

Algorithms 6.2 and 6.3 may suffer from the computational costs to solve the MIP relaxation (6.5). While LP and NLP solvers can often be warmstarted, for a MIP solver the only warmstarting capability consists usually of providing a known feasible solution. On the other hand, the NLP-based branch-and-bound algorithm (Algorithm 6.1) may suffer from the computational cost to solve NLP relaxations. One combination of both algorithms consists of using only a single branch-and-bound tree for the MIP relaxation, but improving the relaxation during the tree search by adding linearizations in solutions of NLP subproblems in nodes where the LP relaxation satisfies the integrality requirements [Quesada and Grossmann, 1992]. Algorithm 6.4 summarizes this method.

As for the extended cutting plane variant of the outer-approximation algorithm, it is also possible to omit the solution of NLP subproblems in Algorithm (6.4). Instead, linearizations are added in a solution of the LP relaxation (6.6) that satisfies the integrality requirements. Solving the LP relaxation and improving it by new linearizations is then alternated until it becomes infeasible, its optimal value reaches the current upper bound, or its solution violates one of the integrality requirements.

Further variations are obtained by sometimes using the NLP relaxation (6.2) instead of the LP relaxation (6.6) for bounding [Bonami et al., 2008] and by adding linearizations already in nodes with fractional  $\hat{x}_I$  [Abhishek et al., 2010, Nowak and Vigerske, 2008]. The latter has the advantage of a tight LP relaxation early in the tree search.

---

**Algorithm 6.4:** LP/NLP-based branch-and-bound algorithm for solving (6.1)
 

---

```

 $\mathcal{L} \leftarrow \{[\underline{x}_I, \bar{x}_I]\};$ 
 $\bar{v} \leftarrow \infty;$ 
solve NLP relaxation (6.2);
if (6.2) is infeasible then STOP: (6.1) is infeasible;
 $K \leftarrow \{\hat{x}\}$ , where  $\hat{x}$  is an optimal solution of (6.2);
while  $\mathcal{L} \neq \emptyset$  do
  select  $[\underline{x}'_I, \bar{x}'_I]$  from  $\mathcal{L}$ ;  $\mathcal{L} \leftarrow \mathcal{L} \setminus \{[\underline{x}'_I, \bar{x}'_I]\}$ ;
  solve the linear relaxation
  
$$\min \left\{ \alpha \in \mathbb{R} : \begin{array}{ll} Ax \leq b, & \hat{x} \in K, \\ f(\hat{x}) + \langle \nabla f(\hat{x}), x - \hat{x} \rangle \leq \alpha, & \hat{x} \in K, \\ g_j(\hat{x}) + \langle \nabla g_j(\hat{x}), x - \hat{x} \rangle \leq 0, & j \in [m], \hat{x} \in K, \\ x \in [\underline{x}'_I, \bar{x}'_I] \end{array} \right\} \quad (6.6)$$

  if (6.6) is infeasible then continue; /* node is infeasible */
   $(\hat{x}, \hat{\alpha}) \leftarrow$  optimal solution of (6.6);
  if  $\hat{\alpha} \geq \bar{v}$  then continue; /* no better solution within  $[\underline{x}'_I, \bar{x}'_I]$  */
  if  $\hat{x}_I \in \mathbb{Z}^{|I|}$  then /* process NLP subproblems */
    solve NLP subproblem (6.3) w.r.t. fixation  $\hat{x}_I$ ;
    if (6.3) is feasible then
       $\tilde{x} \leftarrow$  optimal solution of (6.3);
      if  $\tilde{x}$  is feasible for (6.1) then
         $\bar{v} \leftarrow f(\tilde{x})$ ; /* update upper bound (new incumbent) */
      end
    else
      solve NLP subproblem (6.4) w.r.t. fixation  $\hat{x}_I$ ;
       $\tilde{x} \leftarrow$  optimal solution of (6.4);
    end
     $K \leftarrow K \cup \{\tilde{x}\}$ ; /* improve outer-approximation */
  else /* branching */
    select  $i^* \in I$  with  $\hat{x}_{i^*} \notin \mathbb{Z}$ ;
     $\bar{x}''_i \leftarrow \begin{cases} \bar{x}'_i, & i \neq i^* \\ \lfloor \hat{x}_{i^*} \rfloor, & i = i^* \end{cases}, \quad \underline{x}''_i \leftarrow \begin{cases} \underline{x}'_i, & i \neq i^* \\ \lceil \hat{x}_{i^*} \rceil, & i = i^* \end{cases};$ 
     $\mathcal{L} \leftarrow \mathcal{L} \cup \{[\underline{x}'_I, \bar{x}''_I], [\underline{x}''_I, \bar{x}'_I]\}$ ;
  end
end
output: optimal value of (6.1) is  $\bar{v}$ 

```

---

### Reducing the Effect of Nonconvexity

The completeness of the algorithms presented so far rely on the convexity assumption for  $f(x)$  and  $g(x)$ . However, often it is possible to extend them in a way that allows to compute good feasible solutions also in the presence of nonconvex functions, even though one loses the guarantee of finding a global optimum.

For the NLP-based branch-and-bound algorithm, the convexity assumption is made to allow the efficient computation of a global optimum of the NLP subproblems. Usually, one employs a solver that guarantees only local optimal solutions for NLPs. Thus, in the presence of a nonconvex NLP relaxation, the solver may return a non-global solution of a node's relaxation, which may result in a wrong lower bound and an erroneous pruning of the node. Strategies to reduce the likelihood of this undesirable event are to solve the NLP from different starting points, to employ different NLP solvers, or to require a defined safety margin on how much the value of the NLP relaxation has to exceed the current upper bound before pruning takes place, see Section 5.3 in Bonami and Lee [2011] and the chapter on SBB in GAMS Development Corp. [2012].

For the algorithms relying on the MIP relaxation (6.5) or the LP relaxation (6.6), convexity is important for the underestimating property of the generated linearizations. In case of a nonconvex  $f(x)$ , linearizations may not underestimate  $f(x)$  on the whole  $X$ , and in case of a nonconvex  $g(x)$ , linearizations may cut off parts of the feasible set  $X$ , see Figure 6.2a. To reduce the effect of the latter in the outer-approximation algorithm (Alg. 6.2), Viswanathan and Grossmann [1990] propose to relax the inequalities from linearization by adding a penalty term to the objective function that minimizes violation of these inequalities, see Figure 6.2b. Hence, the MIP relaxation (6.5) is replaced by

$$\min \left\{ \alpha + \sum_{\hat{x} \in K} w^{\hat{x}} p^{\hat{x}} : \begin{array}{ll} Ax \leq b, & \hat{x} \in K, \\ f(\hat{x}) + \langle \nabla f(\hat{x}), x - \hat{x} \rangle \leq \alpha, & \hat{x} \in K, \\ g_j(\hat{x}) + \langle \nabla g_j(\hat{x}), x - \hat{x} \rangle \leq p^{\hat{x}}, & j \in [m], \hat{x} \in K, \\ x \in [\underline{x}, \bar{x}], x_I \in \mathbb{Z}^{|I|}, \alpha \in \mathbb{R}, p^{\hat{x}} \in \mathbb{R}_+, & \hat{x} \in K, \end{array} \right\} \quad (6.7)$$

where the weights  $w^{\hat{x}} > 0$  are derived from the dual solution of the NLP subproblem that yield the linearization point  $\hat{x}$ . Even though (6.7) does not yield a lower bound on the optimal value of the MINLP in general, it may provide a good fixation for the integer variables in the NLP subproblem (6.3) for computing a feasible solution to (6.1).

Kocis and Grossmann [1987] proposed a further modification of the outer-approximation algorithm to allow for nonlinear equality constraints. The idea is to use the dual variable associated with the equality constraint in the NLP subproblem as indicator on how to relax the equality into an inequality (i.e., whether to replace  $=$  by  $\leq$  or  $\geq$ ). A linearization is then only generated for the relaxed inequality. The resulting inequality may further be relaxed by penalizing its violation as in (6.7) [Viswanathan and Grossmann, 1990].

Finally, for pseudo-convex functions<sup>7</sup>, Westerlund and Pörn [2002] proposed an extension of the extended cutting plane algorithm (Alg. 6.3) that guarantees global optimality.

<sup>7</sup>A differentiable function  $h : X \rightarrow \mathbb{R}$  is *pseudo-convex* on a convex set  $X \subseteq \mathbb{R}^n$ , if for every  $x, y \in X$  with  $h(x) < h(y)$  it follows that  $\langle \nabla h(y), x - y \rangle < 0$ . An important property of a pseudo-convex function is the convexity of its level-sets.

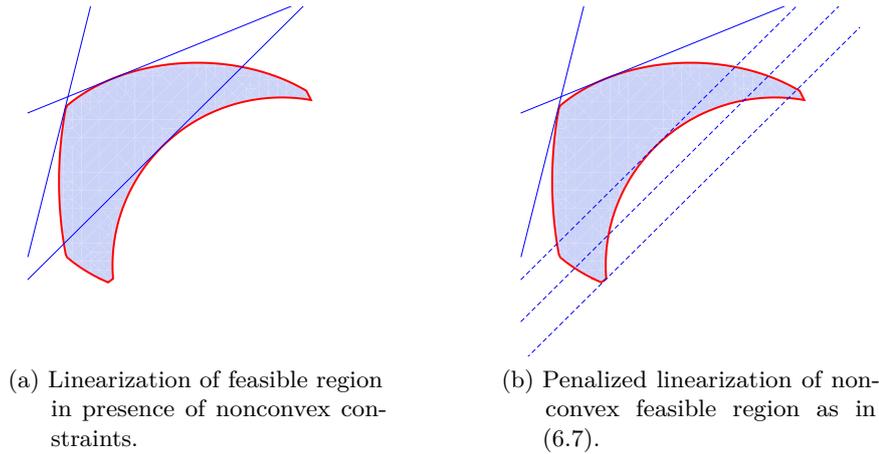


Figure 6.2.: Effect of linearizing nonconvex functions.

### 6.1.2. Convexification for Nonconvex MINLPs

In the following, we drop the assumption that  $f(x)$  and  $g(x)$  in (6.1) are convex. Thus, as discussed above, the algorithms for convex MINLP from the previous section may not yield global optimal solutions if the NLP relaxation in Algorithm 6.1 is not solved to global optimality or the linearizations in the outer-approximation algorithms cut off optimal solutions, see Figure 6.2a.

To be able to compute valid lower bounds on the optimal value of a general MINLP, usually convexification techniques that construct a convex outer-approximation of (6.1) are employed. Such a convex relaxation can be linear or nonlinear, but for computational reasons, most solvers employ a linear relaxation.

While analytic formulas for the convexification of the feasible set  $X$  itself are rarely available, one often settles with a convex relaxation that is derived by underestimating each of the functions  $g_j(x)$ ,  $j \in J$ , separately. In the following, we present some basic techniques to construct a convex relaxation of a general MINLP. W.l.o.g., we assume a linear objective function,  $f(x) = \langle c, x \rangle$ . Further, we assume that the bounds  $\underline{x}_i$ ,  $\bar{x}_i$  are finite for all variables  $i \in [n]$  that occur nonlinearly in some  $g_j(x)$ ,  $j \in J$ .

Given a function  $h : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}$ , we seek for a function  $h^c(x)$  that is convex, underestimates  $h(x)$ , and is “as close as possible” to  $h(x)$ . The following definitions introduce some basic terminology, see also Figure 6.3 for an illustration.

**Definition 6.2.** Let  $h : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}$ . A *convex underestimator* of  $h(x)$  is a function  $h^c : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}$  that is convex on  $[\underline{x}, \bar{x}]$  and that satisfies  $h^c(x) \leq h(x)$  for all  $x \in [\underline{x}, \bar{x}]$ . A convex underestimator  $h_1^c(x)$  of  $h(x)$  is said to be *tighter* than a convex underestimator  $h_2^c(x)$ , if  $\text{epi } h_1 \subsetneq \text{epi } h_2$ . A *convex envelope* of  $h(x)$  is a convex underestimator  $h^e : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}$  such that  $h^c(x) \leq h^e(x)$ ,  $x \in [\underline{x}, \bar{x}]$ , for all convex underestimators  $h^c(x)$  of  $h(x)$ . That is, a convex envelope is a tightest convex underestimator of  $h(x)$ .

## 6. Introduction

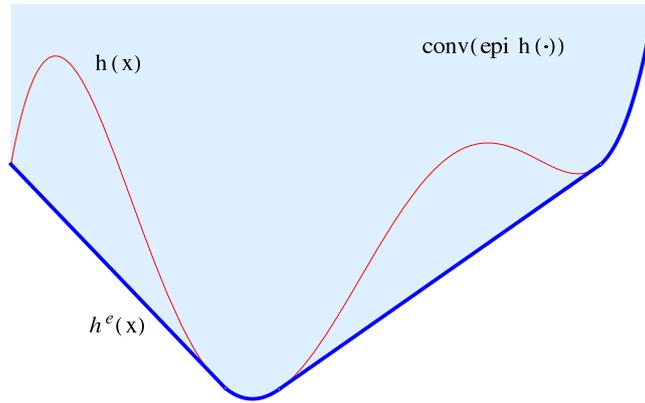


Figure 6.3.: Nonconvex function  $h(x)$ , its convex envelope  $h^e(x)$ , and  $\text{epi } h^e(x) = \text{conv}(\text{epi } h(x))$ .

**Proposition 6.3** (Chapter 2.E in Rockafellar and Wets [1998]). *Let  $h : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}$ .  $h^e(x)$  is the pointwise supremum of all convex underestimators of  $h(x)$ . It is  $\text{epi } h^e(x) = \text{conv}(\text{epi } h(x))$ , where  $\text{conv}(A)$  denotes the convex hull of a set  $A \subset \mathbb{R}^m$ . For  $x \in [\underline{x}, \bar{x}]$ ,*

$$h^e(x) = \inf \left\{ \sum_{i=0}^n \lambda_i h(x^i) : \sum_{i=0}^n \lambda_i x^i = x, \sum_{i=0}^n \lambda_i = 1, \lambda_i \geq 0, x^i \in [\underline{x}, \bar{x}], i \in [0 : n] \right\}. \quad (6.8)$$

Constructing the convex envelope of an arbitrary given function is in general not possible. Thus, much research has been done to derive convex envelopes or at least tight convex underestimators for special classes of nonconvex functions, see, e.g., McCormick [1976] and Al-Khayyal and Falk [1983] for early references and Rockafellar and Wets [1998] and Tawarmalani and Sahinidis [2002a] for fundamental theory. In the following, we introduce the convex envelopes for some simple but common cases. Together with a reformulation scheme, they allow to construct convex relaxations for a broad class of optimization problems.

### Convex Functions

The convex envelope of a convex function is the function itself.

### Concave Functions

Let  $h(x)$  be a concave function on  $[\underline{x}, \bar{x}]$ , i.e.,  $-h(x)$  is convex. In this case, Equation (6.8) can be simplified by considering only the extreme points of  $[\underline{x}, \bar{x}]$  as candidates for the points  $x^i$  [Falk and Hoffman, 1976]. Thus, for  $\{x^i\}_{i \in [2^n]}$  such that  $\text{conv}(\{x^i\}_{i \in [2^n]}) = [\underline{x}, \bar{x}]$ ,

$$h^e(x) = \inf \left\{ \sum_{i=1}^{2^n} \lambda_i h(x^i) : \sum_{i=1}^{2^n} \lambda_i x^i = x, \sum_{i=1}^{2^n} \lambda_i = 1, \lambda_i \geq 0 \right\} \quad (x \in [\underline{x}, \bar{x}]). \quad (6.9)$$

For a univariate function ( $n = 1$ ), (6.9) yields the secant between  $(\underline{x}, h(\underline{x}))$  and  $(\bar{x}, h(\bar{x}))$  as convex envelope, see also Figure 6.4a:

$$h^e(x) = h(\underline{x}) + \frac{h(\bar{x}) - h(\underline{x})}{\bar{x} - \underline{x}}(x - \underline{x}) \quad (x \in [\underline{x}, \bar{x}]). \quad (6.10)$$

### Product

Let  $h(x) = x_1x_2$ . McCormick [1976] and Al-Khayyal and Falk [1983] have shown, that the convex envelope of  $h(x)$  w.r.t.  $[\underline{x}, \bar{x}]$  is given by

$$h^e(x) = \max\{\bar{x}_1x_2 + \bar{x}_2x_1 - \bar{x}_1\bar{x}_2, \underline{x}_1x_2 + \underline{x}_2x_1 - \underline{x}_1\underline{x}_2\}, \quad (6.11)$$

see also Figure 6.4c. The fact that  $h^e(x)$  is an underestimator of  $h(x)$  is easily derived from the valid inequalities  $(x_1 - \underline{x}_1)(x_2 - \underline{x}_2) \geq 0$  and  $(\bar{x}_1 - x_1)(\bar{x}_2 - x_2) \geq 0$ .

For a product of three variables, facets of the convex envelope are also known [Meyer and Floudas, 2004]. The recursive application of (6.11) yields convex underestimators for products of three or four variables [Maranas and Floudas, 1995, Meyer and Floudas, 2004, Cafieri, Lee, and Liberti, 2010].

### Quotient

Let  $h(x) = x_1/x_2$ . Assume  $\underline{x}_1 \geq 0$  and  $\underline{x}_2 > 0$ . Note, that  $h(x)$  is linear in  $x_1$  (for fixed  $x_2$ ) and convex in  $x_2$  (for fixed  $x_1$ ). Zamora and Grossmann [1998] have derived the convex underestimator

$$h^c(x) = \frac{1}{x_2} \left( \frac{x_1 + \sqrt{\underline{x}_1 \bar{x}_1}}{\sqrt{\underline{x}_1} + \sqrt{\bar{x}_1}} \right)^2, \quad (6.12)$$

and Theorem 3.4 in Tawarmalani and Sahinidis [2001] shows, that  $h^c(x)$  is the convex envelope of  $h(x)$  w.r.t.  $[\underline{x}_1, \bar{x}_1] \times (0, \infty)$ . For finite bounds on  $x_2$ , more complex formulas for the convex envelope have been derived in Theorem 2.27 in Tawarmalani and Sahinidis [2002b] and in Jach et al. [2008], the latter being

$$h^e(x) = \frac{\bar{x} - x}{\bar{x} - \underline{x}} \frac{\underline{x}}{\max\left(\underline{y}, \frac{\bar{y} - \underline{y}}{\bar{x} - \underline{x}}(\underline{x} - x) + \underline{y}, \frac{y\sqrt{\bar{x}(\bar{x} - x)}}{(\bar{x} - x)\sqrt{\bar{x}} + (x - \underline{x})\sqrt{\bar{x}}}\right)} + \frac{x - \underline{x}}{\bar{x} - \underline{x}} \frac{\bar{x}}{\min\left(\bar{y}, \frac{y - \underline{y}}{x - \underline{x}}(\bar{x} - x) + \bar{y}, \frac{y\sqrt{\bar{x}(\bar{x} - x)}}{(\bar{x} - x)\sqrt{\bar{x}} + (x - \underline{x})\sqrt{\bar{x}}}\right)}, \quad (6.13)$$

see also Figure 6.4d.

Next, consider  $h(x) = -x_1/x_2$  and assume  $\underline{x}_1 \geq 0$  and  $\underline{x}_2 > 0$ . Thus,  $h(x)$  is linear in  $x_1$  and concave in  $x_2$ . It can be shown, that the convex envelope of  $h(x)$  is given by (6.9)

## 6. Introduction

[Tawarmalani and Sahinidis, 2001] and has the analytic formula

$$h^e(x) = -\frac{1}{\underline{x}_2\bar{x}_2} \min\{\bar{x}_2x_1 - \underline{x}_1x_2 + \underline{x}_1\underline{x}_2, \underline{x}_2x_1 - \bar{x}_1x_2 + \bar{x}_1\bar{x}_2\}, \quad (6.14)$$

which was first derived by Zamora and Grossmann [1999], see also Figure 6.4d. The fact that  $h^e(x)$  is an underestimator of  $h(x)$  is easily seen from the valid inequalities

$$\left(\frac{x_1}{x_2} - \frac{\underline{x}_1}{\bar{x}_2}\right) \left(\frac{x_2}{\underline{x}_2} - 1\right) \geq 0, \quad \left(\frac{\bar{x}_1}{\underline{x}_2} - \frac{x_1}{x_2}\right) \left(1 - \frac{x_2}{\bar{x}_2}\right) \geq 0.$$

Convex envelopes have also been derived for quotients  $\pm x_1/x_2$  with  $\underline{x}_1 < 0 < \bar{x}_1$ ,  $\frac{ax_1+bx_2}{cx_1+dx_2}$ , and  $f(x_1)/x_2$  with  $f$  a univariate concave function, see Tawarmalani and Sahinidis [2001] and Tawarmalani and Sahinidis [2002b].

### Monomials of odd degree

Let  $h(x) = x^{2k+1}$  for some  $k \in \mathbb{N}$ . Note, that  $h(x)$  is concave for  $x \leq 0$  and convex for  $x \geq 0$ . Assume  $\underline{x} < 0$  and  $\bar{x} > 0$ . Let  $x^* > 0$  be such that the slope of the line between  $(\underline{x}, h(\underline{x}))$  and  $(x^*, h(x^*))$  equals the gradient of  $h(x)$  at  $x^*$ , i.e.,

$$\frac{h(x^*) - h(\underline{x})}{x^* - \underline{x}} = (2k+1)(x^*)^{2k}.$$

Liberti and Pantelides [2003] have shown, that the convex envelope of  $h(x)$  is given by the secant (6.10) for  $x \in [\underline{x}, x^*]$  and by  $h(x)$  for  $x > x^*$ , see also Figure 6.4b:

$$h^e(x) = \begin{cases} h(\underline{x}) + h'(x^*)(x - \underline{x}), & \text{if } x \leq x^*, \\ h(x), & \text{if } x > x^*. \end{cases} \quad (6.15)$$

The point  $x^*$  is given as  $\underline{x}c$ , where  $c$  is the real root of the polynomial  $1 + \sum_{i=2}^{2k} ix^{i-1}$ . As shown in Liberti and Pantelides [2003], this polynomial has exactly one real root, which lies in the interval  $[-1 + 1/2k, -1/2]$ .

### Reformulation

We are now able to state a convex relaxation for the class of *factorable* MINLPs [McCormick, 1976, Tawarmalani and Sahinidis, 2002b].

**Definition 6.4** (McCormick [1976]). A MINLP (6.1) with linear objective function  $f(x) = \langle c, x \rangle$  is in *factorable form*, if the nonlinear functions  $g_j(x)$ ,  $j \in [m]$ , have the form

$$g_j(x) = \sum_{p=1}^n t_p^j(x_p) + \sum_{p=1}^{j-1} \tilde{t}_p^j(g_p(x)) + \sum_{p=1}^n \sum_{q=1}^p v_{p,q}^j(x_p) u_{p,q}^j(x_q) + \sum_{p=1}^{j-1} \sum_{q=1}^p \tilde{v}_{p,q}^j(g_p(x)) \tilde{u}_{p,q}^j(g_q(x)),$$

where  $t_p^j(\cdot)$ ,  $\tilde{t}_p^j(\cdot)$ ,  $v_{p,q}^j(\cdot)$ ,  $\tilde{v}_{p,q}^j(\cdot)$ ,  $u_{p,q}^j(\cdot)$ , and  $\tilde{u}_{p,q}^j(\cdot)$  are univariate functions.

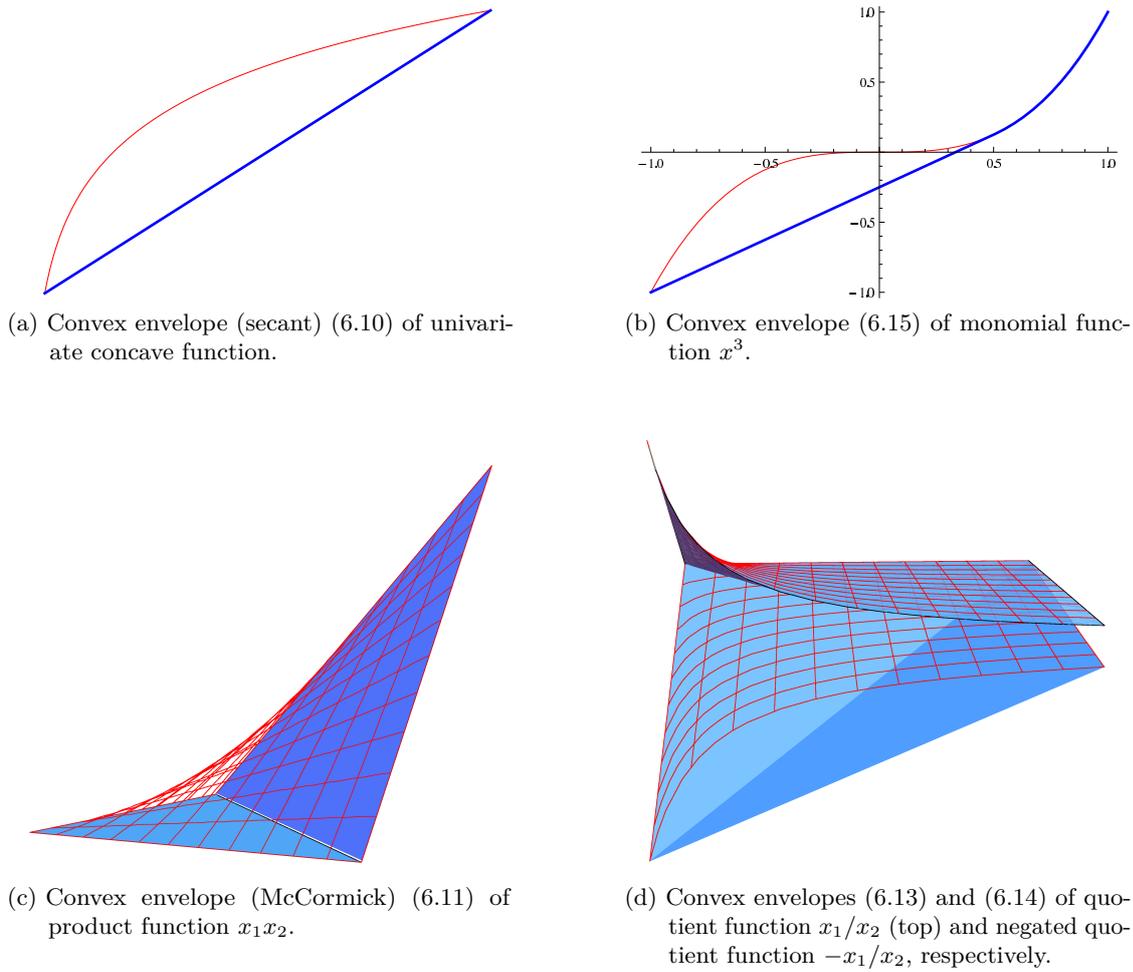


Figure 6.4.: Convex underestimators for some simple nonconvex functions.

Note, that every MINLP with nonlinear functions given as recursive sums and products of univariate functions can be stated in factorable form. McCormick [1976] suggested to derive a convex relaxation for MINLPs in factorable form via the following property:

**Theorem 6.5** (McCormick [1976], Theorem 4.1 in Tawarmalani and Sahinidis [2002b]). *Let  $h(x) = f(g(x))$  where  $f : \mathbb{R} \rightarrow \mathbb{R}$  and  $g : \mathbb{R}^n \rightarrow \mathbb{R}$ . Let  $S \subseteq \mathbb{R}^n$  be a convex domain of  $g(\cdot)$  and assume  $g(x) \in [\underline{g}, \bar{g}]$  over  $S$ . Let  $g_+^c(x)$  and  $g_-^c(x)$  be convex underestimators of  $g(x)$  and  $-g(x)$  over  $S$ , respectively, and let  $f^e(\cdot)$  be the convex envelope of  $f(g)$  for  $g \in [\underline{g}, \bar{g}]$ . Let  $g_{\min} = \operatorname{argmin}_{g \in [\underline{g}, \bar{g}]} f(g)$ . Then,*

$$h^c(x) = f^e(\operatorname{mid}(g_+^c(x), -g_-^c(x), g_{\min}))$$

*is a convex underestimator of  $h(x)$ , where  $\operatorname{mid}(\cdot, \cdot, \cdot)$  is the function that selects the middle value of three scalars.*

## 6. Introduction

However, most algorithms that work on factorable MINLPs introduce additional variables<sup>8</sup> for functions  $g_j(x)$  that occur in some  $g_{j'}(x)$  with  $j' > j$ . Such a reformulation is easier to handle and may be tighter if functions occur several times. A standard form is given by

$$\min \left\{ \begin{array}{l} Ax \leq b, \\ x_i x_j \leq x_k, \quad (i, j, k) \in \mathcal{T}_p, \\ \langle c, x \rangle : g_k(x_i) \leq x_k, \quad (i, k) \in \mathcal{T}_u, \\ x \in [\underline{x}, \bar{x}], \\ x_I \in \mathbb{Z}^{|I|}, \end{array} \right\} \quad (6.16)$$

where  $g_k(\cdot)$  are univariate functions and the sets  $\mathcal{T}_p$  and  $\mathcal{T}_u$  describe the bilinear products and univariate nonlinear terms, respectively [Smith and Pantelides, 1997, 1999]<sup>9</sup>. Assuming finite bounds and known convex underestimators for the univariate functions  $g_k(\cdot)$ , a convex relaxation of a MINLP in standard form (6.16) is given by

$$\min \left\{ \begin{array}{l} Ax \leq b, \\ \bar{x}_i x_j + \bar{x}_j x_i - \bar{x}_i \bar{x}_j \leq x_k, \quad (i, j, k) \in \mathcal{T}_p, \\ \langle c, x \rangle : \underline{x}_i x_j + \underline{x}_j x_i - \underline{x}_i \underline{x}_j \leq x_k, \quad (i, j, k) \in \mathcal{T}_p, \\ g_k^c(x_i) \leq x_k, \quad (i, k) \in \mathcal{T}_u, \\ x \in [\underline{x}, \bar{x}]. \end{array} \right\} \quad (6.17)$$

For the products, the convex envelope given by (6.11) is used. The convex envelope of a function  $g_k(\cdot)$  is given by  $g_k(\cdot)$  itself if classified as convex, by (6.10) if classified as concave, and by (6.15) if of the form  $x^{2p+1}$ . Further, note that we dropped the integrality restriction on the variables  $x_I$  in (6.17). For notational simplicity, we assumed that also convex constraints in (6.1) are reformulated into standard form (6.16). Of course, in practice these constraints can be added in their original form to (6.17).

**Example 6.6.** Let  $h(x) = \sqrt{e^{x_1^2} \ln(x_2)}$  and  $[\underline{x}, \bar{x}] = [0, 2] \times [1, 2]$ . A reformulation of  $h(x)$  into standard form is

$$\begin{aligned} h &= \sqrt{y_1} \\ y_1 &= y_2 y_3, & y_1 &\in [0, \ln(2)e^4], \\ y_2 &= e^{y_4}, & y_2 &\in [1, e^4], \\ y_3 &= \ln(x_2), & y_3 &\in [0, \ln(2)], \\ y_4 &= x_1^2, & y_4 &\in [0, 4], \end{aligned} \quad (6.18)$$

where the bounds on the new variables  $y_1, \dots, y_4$  are obtained from the bounds on  $x$ , see also the paragraph on constraint-based bound tightening in Section 6.1.5 below.

<sup>8</sup>An implementation that works directly on McCormick's factorable form is LIBMC [Scott et al., 2011].

<sup>9</sup>In the standard form of Smith and Pantelides [1999], also quotient terms  $x_k = x_i/x_j$  appear and for nonlinearities, equalities  $x_i x_j = x_k$  and  $g_k(x_i) = x_k$  are used. For simplicity, we use here the equivalent form (6.16), where quotients  $x_k = x_i/x_j$  are reformulated as bilinear terms  $x_k x_j = x_i$  and equalities are replaced by two inequalities.

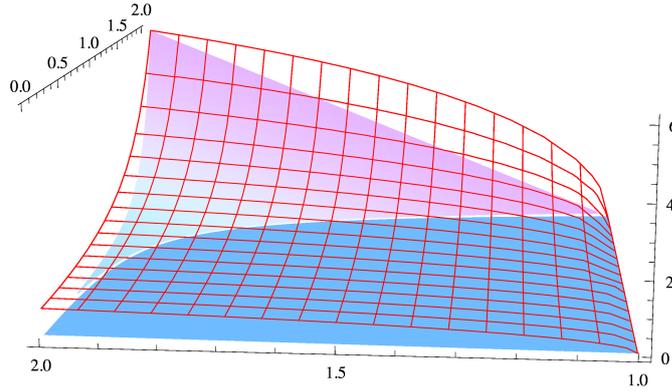


Figure 6.5.: Nonconvex function  $h(x) = \sqrt{e^{x_1^2} \ln(x_2)}$  from Example 6.6 and convex underestimator (6.19) of reformulation (6.18).

Relaxing (6.18) by using convex envelopes (6.10) and (6.11) yields the following convex underestimator of  $h(x)$  for  $x \in [\underline{x}, \bar{x}]$ , see also Figure 6.5:

$$\min \left\{ \begin{array}{l} \sqrt{\underline{y}_1} + \frac{\sqrt{\bar{y}_1} - \sqrt{\underline{y}_1}}{\bar{y}_1 - \underline{y}_1} (y_1 - \underline{y}_1) : \\ \ln(\underline{x}_2) + \frac{\ln(\bar{x}_2) - \ln(\underline{x}_2)}{\bar{x}_2 - \underline{x}_2} (x_2 - \underline{x}_2) \leq y_3 \\ \ln(x_2) \geq y_3 \\ x_1^2 \leq y_4 \\ \underline{x}_1^2 + \frac{\bar{x}_1^2 - \underline{x}_1^2}{\bar{x}_1 - \underline{x}_1} (x_1 - \underline{x}_1) \geq y_4 \\ y \in [\underline{y}, \bar{y}] \end{array} \right. \quad \left. \begin{array}{l} \bar{y}_2 y_3 + \bar{y}_3 y_2 - \bar{y}_2 \bar{y}_3 \leq y_1 \\ \underline{y}_2 y_3 + \underline{y}_3 y_2 - \underline{y}_2 \underline{y}_3 \leq y_1 \\ \bar{y}_2 y_3 + \underline{y}_3 y_2 - \bar{y}_2 \underline{y}_3 \geq y_1 \\ \underline{y}_2 y_3 + \bar{y}_3 y_2 - \underline{y}_2 \bar{y}_3 \geq y_1 \\ e^{y_4} \leq y_2 \\ e^{y_4} + \frac{e^{\bar{y}_4} - e^{\underline{y}_4}}{\bar{y}_4 - \underline{y}_4} (y - \underline{y}_4) \geq y_2 \end{array} \right. \quad (6.19)$$

The reformulation of a factorable MINLP into the standard form (6.16) yields the convex relaxation (6.17). However, a tighter relaxation may be obtained if functions are not decomposed into single products, quotients, and univariate functions, but can be underestimated directly. Thus, research has recently focused on deriving convex envelopes on more complex functions and on the development of new techniques for deriving convex envelopes of functions or convex hulls of sets. Some examples are convex envelopes for  $(n-1)$ -convex functions<sup>10</sup> [Jach et al., 2008], convex hulls for orthogonal disjunctive sets<sup>11</sup> [Tawarmalani et al., 2010], convex envelopes derived from polyhedral

<sup>10</sup>A function  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  is  $(n-1)$ -convex, if  $h|_{x_i = \hat{x}_i} : \mathbb{R}^{n-1} \rightarrow \mathbb{R}$  is convex for any  $\hat{x}_i \in \mathbb{R}$ ,  $i \in [n]$ .

<sup>11</sup>A set  $S \subseteq \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_n}$  is an *orthogonal disjunctive set*, if it can be written as disjunctive union of a finite number of sets defined over subspaces that are orthogonal to each other. That is, if there exists

## 6. Introduction

subdivisions of the box [Tawarmalani et al., 2013], or convex envelopes of polynomials over general polytopes [Locatelli and Schoen, 2009, Locatelli, 2010]. In Section 6.1.6, we review additional techniques to tighten a convex relaxation of a MIQCP or a MINLP with quadratic constraints.

### Twice continuously differentiable functions

Techniques that allow to derive a convex relaxation without reformulation into factorable form or standard form (6.16), have been developed by Maranas and Floudas [1992]. Given a twice continuously differentiable function  $h : [\underline{x}, \bar{x}] \rightarrow \mathbb{R}$  and a diagonal matrix  $D \in \mathbb{R}^{n \times n}$  with nonnegative entries, an underestimator of  $h(x)$  w.r.t.  $[\underline{x}, \bar{x}]$  is given by

$$h^c(x) = h(x) + \frac{1}{2} \langle x - \underline{x}, D(x - \bar{x}) \rangle.$$

If additionally  $\nabla^2 h(x) + D \succeq 0$  for all  $x \in [\underline{x}, \bar{x}]$ , then  $h^c(x)$  is convex. A matrix  $D$  that satisfies this criterion can be constructed by using Gershgorin's circle theorem to estimate the minimal eigenvalue from the vertex matrices of an interval evaluation of the Hessian [Adjiman and Floudas, 1996].

Improvements of this techniques for computing tighter convex underestimators are discussed in Akrotirianakis and Floudas [2004], Meyer and Floudas [2005] and Gounaris and Floudas [2008b,c].

### 6.1.3. Spatial Branch-and-Bound

Assume convex envelopes are used to derive a convex relaxation (6.17) of (6.16). If an optimal solution for (6.17) is not feasible for (6.16), then either due to a fractional value for some discrete variable or due to the gap between the convex envelope of a nonconvex functions and the function itself. For both types of nonconvexity, a method to eliminate the infeasible solution from the relaxation (and thereby hopefully improving the lower bound) is to consider subproblems corresponding to a partition of the box  $[\underline{x}, \bar{x}]$ . When partitioning  $[\underline{x}, \bar{x}]$  along the coordinate of a discrete variable with fractional value, the solution is eliminated from both subproblems by reducing the bounds of the corresponding variable to integral values, cf. Algorithm 6.1. When partitioning  $[\underline{x}, \bar{x}]$  along the coordinate of a variable that is involved in a nonconvex function  $g_k(x)$ , new convex envelopes can be constructed in both subproblems. The new envelopes on the reduced boxes are usually tighter than the underestimator given by the convex envelope over  $[\underline{x}, \bar{x}]$ , see also Figure 6.6.

Recursively repeating the partitioning scheme and computing bounds for the convex relaxation of the MINLP w.r.t. the partition elements yields a branch-and-bound algorithm similar to Algorithm 6.1, see Algorithm 6.5. Since branching is now performed also w.r.t. continuous variables, it is called *spatial branch-and-bound*. Under mild assumptions,

---

$S_i \subseteq S$ ,  $i \in [n]$ , such that for all  $(z_1, \dots, z_n) \in S_i$ ,  $z_j = 0$  for all  $j \neq i$ , and for any  $z \in S$  there exists an  $I \subseteq [n]$  and  $\chi_i \in \text{conv}(S_i)$ ,  $i \in I$ , such that  $z \in \text{conv}\{\chi_i\}_{i \in I}$ .

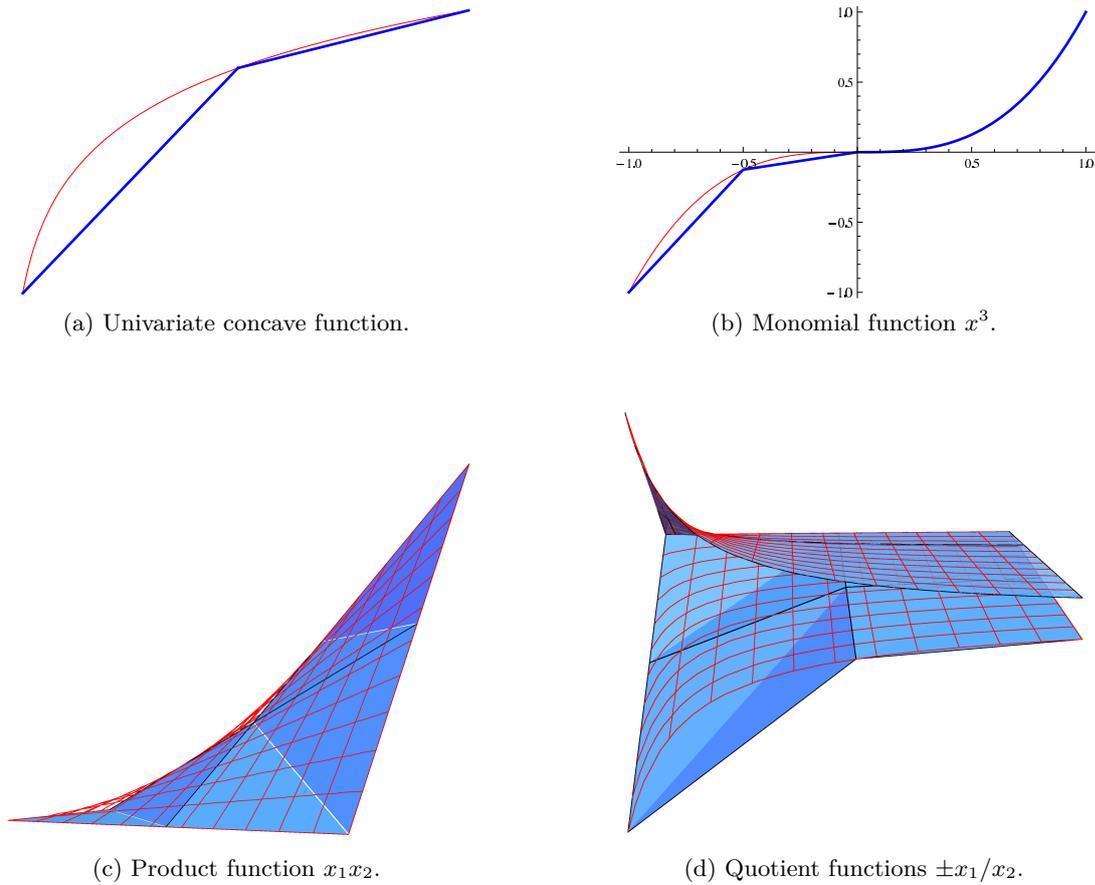


Figure 6.6.: Convex underestimators for some simple nonconvex functions after branching, see also Figure 6.4.

Algorithm 6.5 computes a sequence of improving global lower bounds  $\min_{[\underline{x}', \bar{x}'] \in \mathcal{L}} \underline{v}([\underline{x}', \bar{x}'])$  that converges to the optimal value of the MINLP:

**Theorem 6.7** (Corollary IV.5 in Horst and Tuy [1990]). *Let (6.16) be feasible. Assume that the functions  $g_k(\cdot)$  in (6.16) are continuous on  $[\underline{x}, \bar{x}]$  and that  $[\underline{x}, \bar{x}]$  is bounded. Further, assume that  $g_k^c(\cdot) = g_k^e(\cdot)$  when constructing a convex relaxation (6.17) in Line 5 of Algorithm 6.5. Let  $\{\hat{x}^j\}_j$  be the sequence of optimal solutions of the convex relaxations in Line 7. Then every accumulation point of  $\{\hat{x}^j\}_j$  is an optimal solution to (6.16).*

**Remark 6.8.** The following remarks to Algorithm 6.5 and Theorem 6.7 are in order.

- Algorithm 6.5 is in general not finite, i.e.,  $\mathcal{L} = \emptyset$  may never happen. Also, the upper bound  $\bar{v}$  may remain at  $\infty$ , if optimal solutions of the relaxed problem are never feasible for the original problem (and no other methods to find feasible solutions are applied). However, implementations often allow for small constraint violations,

**Algorithm 6.5:** Spatial branch-and-bound algorithm for solving (6.16)

---

```

1  $\mathcal{L} \leftarrow \{[\underline{x}, \bar{x}]\};$ 
2  $\underline{v}([\underline{x}, \bar{x}]) \leftarrow -\infty;$ 
3  $\bar{v} \leftarrow +\infty;$ 
4 while  $\mathcal{L} \neq \emptyset$  do
5   select  $[\underline{x}', \bar{x}']$  from  $\mathcal{L}$  with minimal lower bound  $\underline{v}([\underline{x}', \bar{x}'])$ ;  $\mathcal{L} \leftarrow \mathcal{L} \setminus \{[\underline{x}', \bar{x}']\};$ 
6   construct and solve convex relaxation (6.17) for (6.16) restricted to  $[\underline{x}', \bar{x}']$ ;
7   if (6.17) is infeasible then continue; /* node is infeasible */
8    $\hat{x} \leftarrow$  optimal solution of (6.17);
9    $\underline{v}([\underline{x}', \bar{x}']) \leftarrow f(\hat{x});$ 
10  if  $\underline{v}([\underline{x}', \bar{x}']) \geq \bar{v}$  then continue; /* no better solution within  $[\underline{x}', \bar{x}']$  */
11  if  $\hat{x}$  is feasible for (6.16) then
12     $\bar{v} \leftarrow \underline{v}([\underline{x}', \bar{x}'])$ ; /* update upper bound (new incumbent) */
13    continue;
14  end
15   $i^* \leftarrow \operatorname{argmax}_{i \in [n]} (\bar{x}'_i - \underline{x}'_i)$ ; /* branch on var. with largest interval */
16   $\bar{x}''_i \leftarrow \begin{cases} \bar{x}'_i, & i \neq i^* \\ \lfloor (\underline{x}'_i + \bar{x}'_i)/2 \rfloor, & i = i^* \in I \\ (\underline{x}'_i + \bar{x}'_i)/2, & i = i^* \notin I \end{cases}$ ,  $\underline{x}''_i \leftarrow \begin{cases} \underline{x}'_i, & i \neq i^* \\ \lceil (\underline{x}'_i + \bar{x}'_i)/2 \rceil, & i = i^* \in I \\ (\underline{x}'_i + \bar{x}'_i)/2, & i = i^* \notin I \end{cases}$ 
17   $\mathcal{L} \leftarrow \mathcal{L} \cup \{[\underline{x}', \bar{x}''], [\underline{x}'', \bar{x}']\};$ 
18   $\underline{v}([\underline{x}', \bar{x}'']) \leftarrow \underline{v}([\underline{x}', \bar{x}'])$ ;  $\underline{v}([\underline{x}'', \bar{x}']) \leftarrow \underline{v}([\underline{x}', \bar{x}'])$ ;
19 end
output: optimal value of (6.1) is  $\bar{v}$ 

```

---

which guarantees that once the box  $[\underline{x}', \bar{x}']$  is sufficiently small, solutions that are feasible for the relaxation are also “almost feasible” for the original problem.

Further, the algorithm is usually interrupted once the gap between the global lower bound and the upper bound is sufficiently small.

Finitely terminating spatial branch-and-bound algorithms have been developed for certain special classes of global optimization problems, see Shectman and Sahinidis [1998] and Al-Khayyal and Serali [2000].

- The selection of a box  $[\underline{x}', \bar{x}']$  with lowest lower bound in Line 4 is to ensure a *bound improving selection scheme* as required by Corollary IV.5 in Horst and Tuy [1990]. Also a scheme where at each time a box with the lowest lower bound is chosen after a finite number of steps would be sufficient.
- The partitioning of  $[\underline{x}', \bar{x}']$  w.r.t. a variable with largest interval  $[\underline{x}'_i, \bar{x}'_i]$  in Line 14 is far from being efficient, but is a simple way to ensure an *exhaustive subdivision*<sup>12</sup> of  $[\underline{x}, \bar{x}]$  as required by Corollary IV.5 in Horst and Tuy [1990]. In actual

<sup>12</sup>The branching scheme used to partition  $[\underline{x}, \bar{x}]$  is called *exhaustive*, if for all subsequences  $\{[\underline{x}^k, \bar{x}^k]\}_k$  of

implementations, only variables from violated constraints in (6.1) are considered for branching. That is, either a discrete variable  $x_i$ ,  $i \in I$ , with fractional value  $\hat{x}_i \notin \mathbb{Z}$  or a variable  $x_i$  or  $x_j$  with  $\hat{x}_i \hat{x}_j > \hat{x}_k$ ,  $(i, j, k) \in \mathcal{T}_p$ , or a variable  $x_i$  with  $g_k(\hat{x}_i) > \hat{x}_k$ ,  $(i, k) \in \mathcal{T}_u$ . See also Section 6.1.4 for a short discussion of branching rules for MINLP.

- The assumption on using convex envelopes in (6.17) can be relaxed by requiring the use of underestimators  $g_k^c(\cdot)$  that yield a *strongly consistent* bounding scheme. That is, the optimal value of the relaxations corresponding to a sequence  $\{[\underline{x}^k, \bar{x}^k]\}_k$  of successively refined partition elements ( $[\underline{x}^{k+1}, \bar{x}^{k+1}] \subset [\underline{x}^k, \bar{x}^k]$ ) must converge to the optimal value of the original problem when restricted to the limit of the sequence,  $\bigcap_k [\underline{x}^k, \bar{x}^k]$ .

Since convex envelopes can have quite involved formulas and may not be smooth enough to employ an efficient algorithm for solving the relaxation (6.17), underestimators that are not as tight as the envelope may be used. Especially polyhedral underestimators are commonly used, since they ensure a *linear relaxation*. Such polyhedral underestimators can be obtained by linearization of a known convex envelope as for the MIP relaxation (6.5) for convex MINLPs or by separation of facets of an implicitly given relaxation, see also the second part of Section 6.1.6.

### 6.1.4. Branching

Branching in the Algorithms 6.1, 6.4, and 6.5 has been motivated with the need to eliminate solutions from a relaxation that violate certain nonconvex constraints like integrality restrictions or nonlinear nonconvex constraints. Thus, an obvious strategy to select a variable for branching is to choose one with largest fractionality or one that occurs in a nonlinear constraint that is mostly violated. However, once a global optimal solution has been found, if not earlier, the sole purpose of branching is to improve the lower bound given by the relaxation. Therefore, in the setting of MIP, strategies have been developed that try to select variables for branching that not only resolve infeasibility, but also yield a hopefully large improvement in the lower bound. In this context, branching rules like pseudo cost branching, strong branching, or the hybrid reliability branching have been shown to supersede simple strategies like branching on a variable with largest fractionality [Achterberg et al., 2005]. The studies Belotti et al. [2009] and Bonami et al. [2011] show, that these strategies can successfully be adapted for MINLP, too. In the following, we give a short overview of these branching strategies in the context of MIP and their adaptation for MINLP.

#### Pseudo Cost Branching

The idea of pseudo costs is to estimate how much one can improve the lower bound of the node's relaxation by branching on a variable [Bénichou et al., 1971].

---

the sequence of generated partition elements with  $[\underline{x}^{k+1}, \bar{x}^{k+1}] \subset [\underline{x}^k, \bar{x}^k]$ , the diameter of the partition elements converges to 0,  $\lim_{k \rightarrow \infty} \|\bar{x}^k - \underline{x}^k\| \rightarrow 0$ .

## 6. Introduction

**Fractional Variables.** In MIP, such an estimate is computed for a variable  $x_i$ ,  $i \in I$ , from an average of the bound improvements that have been obtained by previous branchings on  $x_i$ . That is, assume  $x_i$  has fractional value  $\hat{x}_i$  in the relaxation's solution and estimates  $\psi_i^-$  and  $\psi_i^+$  for the improvement in the lower bound per unit change of  $x_i$  are available. Then the expected improvement in the lower bound for the child node with  $x_i \leq \lfloor \hat{x}_i \rfloor$  (the down-branch) is  $\psi_i^-(\hat{x}_i - \lfloor \hat{x}_i \rfloor)$ . Analog, for the child node with  $x_i \geq \lceil \hat{x}_i \rceil$  (the up-branch), the expected improvement is  $\psi_i^+(\lceil \hat{x}_i \rceil - \hat{x}_i)$ . A *branching score* of the variable  $x_i$  is then computed by combining these two values into a single one, e.g., by taking

$$(1 - \mu) \min\{\psi_i^-(\hat{x}_i - \lfloor \hat{x}_i \rfloor), \psi_i^+(\lceil \hat{x}_i \rceil - \hat{x}_i)\} + \mu \max\{\psi_i^-(\hat{x}_i - \lfloor \hat{x}_i \rfloor), \psi_i^+(\lceil \hat{x}_i \rceil - \hat{x}_i)\} \quad (6.20)$$

for a parameter  $\mu \in [0, 1]$  [Linderoth and Savelsbergh, 1999] or

$$\max\{\psi_i^-(\hat{x}_i - \lfloor \hat{x}_i \rfloor), \varepsilon\} \cdot \max\{\psi_i^+(\lceil \hat{x}_i \rceil - \hat{x}_i), \varepsilon\} \quad (6.21)$$

for a small value  $\varepsilon > 0$  [Achterberg, 2007]. A fractional variable with largest score is then chosen for branching.

The pseudo costs  $\psi_i^-$  and  $\psi_i^+$  are computed by averaging the change of the lower bound in previous branchings on  $x_i$ . Thus, in the beginning, no pseudo costs  $\psi_i^-$  or  $\psi_i^+$  are available and an average of the available pseudo costs for all variables is used, if available, see also Section 5.3 in Achterberg [2007].

**Nonlinear Variables.** In MIP, a variable  $x_i$  with value  $\hat{x}_i \notin \mathbb{Z}$  in the solution of a node's relaxation will have a value of at most  $\lfloor \hat{x}_i \rfloor$  in the down-branch and a value of at least  $\lceil \hat{x}_i \rceil$  in the up-branch after branching on  $x_i$ . However, when branching on a continuous variable, it is not sure that  $x_i$  will have a value different from  $\hat{x}_i$  in the relaxation's solution of both child nodes. Indeed, if one partitions the interval for  $x_i$  exactly at  $\hat{x}_i$ , then the solution  $\hat{x}$  is still feasible for both child nodes. Only a subsequent tightening of the convex relaxation in both nodes may cut off  $\hat{x}$ . Thus, pseudo cost estimates and branching scores cannot be computed in the same way as for fractional variables.

As a consequence, Belotti et al. [2009] have developed several alternatives. Interpreting the quantities  $\hat{x}_i - \lfloor \hat{x}_i \rfloor$  and  $\lceil \hat{x}_i \rceil - \hat{x}_i$  as the amount of violation of the integrality constraint on  $x_i$ ,  $i \in I$ , Belotti et al. [2009] suggest to assign to each nonlinear variable  $x_i$  an average of the (scaled) violation of those nonconvex constraints that have  $x_i$  involved. Such a *variable infeasibility* is then used as multiplier for the values  $\psi_i^-$  and  $\psi_i^+$ . Further, the values  $\psi_i^-$  and  $\psi_i^+$  are not computed by averaging the bound changes w.r.t. the fractionality of the variable, but w.r.t. the variable infeasibility.

An alternative method is motivated by the observation that the gap between a function and its convex envelope w.r.t.  $[\underline{x}, \bar{x}]$  is usually proportional to the size of the box  $[\underline{x}, \bar{x}]$ . Thus, one may expect that the improvement in the lower bound of a relaxation can be estimated by the amount of reduction of the interval  $[\underline{x}_i, \bar{x}_i]$ . This leads to the estimates  $\psi_i^-(\bar{x}_i - \hat{x}_i)$  and  $\psi_i^+(\hat{x}_i - \underline{x}_i)$  for the improvement in the down- and up-branches if the interval for  $x_i$  is partitioned at  $\hat{x}_i$ .

Similarly, one may multiply the pseudo costs  $\psi_i^-$  and  $\psi_i^+$  with the remaining interval length in each child, i.e., use  $\psi_i^-(\hat{x}_i - \underline{x}_i)$  and  $\psi_i^+(\bar{x}_i - \hat{x}_i)$  as estimates.

### Strong Branching

The idea of strong branching is to select a variable that is known to give the best improvement in the lower bound after branching [Applegate et al., 1998]. Thus, instead of estimating bound improvements, the actual change in the lower bound when branching on a variable is computed for all branching variable candidates, the improvement w.r.t. both branches is combined into a single value (using a formula similar to (6.20) or (6.21)), and a variable with largest aggregated improvement is selected for branching.

Since computing the bound improvements w.r.t. all branching candidates is very expensive, one may first sort the list of candidates according to pseudo costs and evaluate only a fixed number of candidates or as long as the best candidate changes within a certain number of evaluations, see Section 5.4 in Achterberg [2007] for details.

Belotti et al. [2009] proposed to apply strong branching also in the context of spatial branch-and-bound for MINLP. While in MIP, the relaxations solved for the various branches are very similar and thus warm- and hot-starting facilities of the LP solver can be employed, more effort is necessary in the context of MINLP. Here, when evaluating a branching on a nonlinear variable, the change of the bounds of this variable does not necessarily also change the lower bound given by the relaxation, since the same solution may still be feasible in both branches. Thus, also the convex underestimators need to be tightened in both branches, which makes the method computationally even more expensive than it is already for MIP.

### Reliability Branching

The idea of reliability branching is to combine the strength of pseudo cost branching and strong branching to solve each methods drawbacks [Achterberg et al., 2005]. Strong branching usually gives the smallest search trees, but is computationally very expensive. Evaluating branching candidates according to pseudo costs is cheap, but reliable pseudo costs are not available early in the tree search, which is especially dangerous, since bad branching decisions at the top of the search tree are most harmful.

Thus, a combination consists in scoring a branching variable candidate by pseudo costs if the pseudo costs are deemed trustworthy and, otherwise, to evaluate both branches as in strong branching. Thereby, the availability of eight recorded lower bound changes for both branching directions of a variable has found to be a useful indication for reliability of the corresponding pseudo costs [Achterberg et al., 2005].

### Branching Point

Finally, we discuss shortly strategies on how partitioning the interval  $[\underline{x}_i, \bar{x}_i]$  of a variable  $x_i$ , once it has been selected for branching. If one branches on a discrete variable  $x_i$ ,  $i \in I$ , because of a fractional value  $\hat{x}_i \notin \mathbb{Z}$  in the relaxation's solution, the natural choice is to branch w.r.t.  $\hat{x}_i$ , i.e., to use the intervals  $[\underline{x}_i, \lfloor \hat{x}_i \rfloor]$  and  $[\lceil \hat{x}_i \rceil, \bar{x}_i]$  for the child nodes.

## 6. Introduction

For a nonlinear variable, the choice  $\hat{x}_i$  also seems natural, since convex envelopes are tight at the interval ends and thus after a partitioning w.r.t.  $\hat{x}_i$ , a tighter relaxation can be expected around  $\hat{x}$  in both child nodes. However, care must be taken if  $\hat{x}_i$  is close to one of the bounds of  $x_i$ . In this case, branching on  $\hat{x}_i$  may have almost no effect on one branch, but (almost) fixes the variable in the other branch. To ensure better balanced search trees, the branching point is moved inside the interval, e.g., such that the distance to both interval bounds is at least 20% of the interval length. Other choices are to take convex combinations of  $\hat{x}_i$  and the interval midpoint  $(\underline{x}_i + \bar{x}_i)/2$  [Tawarmalani and Sahinidis, 2002b] or the value of  $x_i$  in an (local) optimal solution of the current node [Shectman and Sahinidis, 1998].

A further strategy, suggested by Belotti et al. [2009], is to branch on a point that minimizes the convexification gap for a nonconvex constraint that involves  $x_i$ . The convexification gap is measured either by  $\int_{[\underline{x}, \bar{x}]} h(x) - h^e(x) dx$  or by  $\max_{x, x' \in [\underline{x}, \bar{x}]} \|(x - x', h(x) - h^e(x'))\|$ , the latter being the maximal distance between the graphs of  $h(x)$  and  $h^e(x)$ . However, computational experiments indicate, that the strategies that take the relaxation's solution  $\hat{x}$  into account yield better results [Belotti et al., 2009].

### 6.1.5. Bound Tightening

Since the convex envelopes depend crucially on the bounds of the variables, reducing the box  $[\underline{x}, \bar{x}]$  can help to tighten the relaxation without branching. The tightest possible bounds are given as interval enclosure of the feasible set  $X$  of the MINLP, cf. (6.1b), see also Figure 6.7, possibly also restricted by an objective cutoff  $\langle c, x \rangle \leq \bar{v}$ , where  $\bar{v}$  is an upper bound on the optimal value (possibly  $+\infty$ ). That is, for a variable  $x_i$ ,  $i \in [n]$ , one is interested in finding<sup>13</sup>

$$\inf\{x_i : x \in X, \langle c, x \rangle \leq \bar{v}\} \quad \text{and} \quad \sup\{x_i : x \in X, \langle c, x \rangle \leq \bar{v}\}. \quad (6.22)$$

Since solving (6.22) is usually not easier than solving the original MINLP (6.1), relaxations of (6.22) are used to compute bounds on the bounds of  $x_i$  efficiently. In MIP, bound tightening is also known as a part of node preprocessing, see also Savelsbergh [1994]. In the following, we give a short introduction to some bound tightening techniques that are applied for MINLP.

### Constraint-Based Bound Tightening

Constraint-based bound tightening (also referred as feasibility-based range reduction in Tawarmalani and Sahinidis [2004]) tries to infer variables bounds from each constraint of (6.16) independently, see also Chapter 5 in Blicik et al. [2001] and the references therein. That is, (6.22), if applied to the standard form (6.16), is relaxed to the following (usually

---

<sup>13</sup>(6.22) states, that a best possible bounding box is sought that contains all feasible points with objective value at most  $\bar{v}$ . However, even tighter bounds may be obtained by requiring a box that contains only at least one optimal solution. This leads to so called *dual reduction techniques*, which, in the simplest case, fix variables that do not occur anywhere in the constraints. For simplicity, we content us here with the weaker form (6.22).

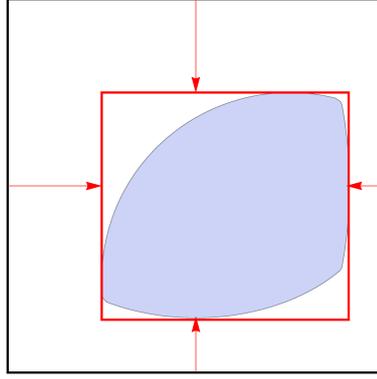


Figure 6.7.: Symbolic visualization of bound tightening (6.22).

much simpler) optimization problems:

$$(\inf | \sup)\{x_i : (Ax)_j \leq b_j, x \in [\underline{x}, \bar{x}]\} \quad (j \in [m'], i \in [n]), \quad (6.23a)$$

$$(\inf | \sup)\{x_i : \langle c, x \rangle \leq \bar{v}, x \in [\underline{x}, \bar{x}]\} \quad (i \in [n]), \quad (6.23b)$$

$$(\inf | \sup)\{x_i : x_i x_j \leq x_k, x \in [\underline{x}, \bar{x}]\} \quad ((i, j, k) \in \mathcal{T}_p), \quad (6.23c)$$

$$(\inf | \sup)\{x_j : x_i x_j \leq x_k, x \in [\underline{x}, \bar{x}]\} \quad ((i, j, k) \in \mathcal{T}_p), \quad (6.23d)$$

$$(\inf | \sup)\{x_k : x_i x_j \leq x_k, x \in [\underline{x}, \bar{x}]\} \quad ((i, j, k) \in \mathcal{T}_p), \quad (6.23e)$$

$$(\inf | \sup)\{x_i : g_k(x_i) \leq x_k, x \in [\underline{x}, \bar{x}]\} \quad ((i, k) \in \mathcal{T}_u), \quad (6.23f)$$

$$(\inf | \sup)\{x_k : g_k(x_i) \leq x_k, x \in [\underline{x}, \bar{x}]\} \quad ((i, k) \in \mathcal{T}_u), \quad (6.23g)$$

$$(\inf | \sup)\{x_i : x_i \in \mathbb{Z}, x \in [\underline{x}, \bar{x}]\} \quad (i \in I). \quad (6.23h)$$

Problems (6.23) are solved consecutively to obtain tightenings for the  $[\underline{x}, \bar{x}]$ . If a problem is infeasible, then infeasibility of the MINLP w.r.t.  $[\underline{x}, \bar{x}]$  is recognized. If a bound tightening is found, then further bound reductions may be inferred from other constraints, so that the procedure can be restarted as long as sufficiently good reductions are found.

Constraint-based bound tightening is usually employed at all or many nodes of the branching tree, since it can efficiently propagate the bound change that is determined by the branching decision in short time.

First, consider a linear constraint  $\sum_j a_j x_j \leq \bar{b}$  from (6.23a)–(6.23b). A new bound for variable  $x_i$  is computed from the estimate

$$a_i x_i \leq \bar{b} - \sum_{j: a_j > 0, j \neq i} a_j \underline{x}_j - \sum_{j: a_j < 0, j \neq i} a_j \bar{x}_j, \quad (6.24)$$

If  $\underline{x}_j > -\infty$  for  $j \in [n]$  with  $j \neq i$  and  $a_j > 0$ , and  $\bar{x}_j < \infty$  for  $j \in [n]$  with  $j \neq i$  and  $a_j < 0$ , then the term on the right-hand-side, divided by  $a_i$ , yields an upper bound on  $x_i$  if  $a_i > 0$  and a lower bound if  $a_i < 0$ . An efficient implementation of the bound tightening technique for linear constraints is discussed in Section 7.1 in Achterberg [2007]. Note, that the same method can also be applied to the objective cutoff inequality  $\langle c, x \rangle \leq \bar{v}$ .

## 6. Introduction

For a constraint  $x_i x_j \leq x_k$  from (6.23c)–(6.23e), the following inequalities are valid:

$$\begin{aligned}
 x_k &\geq \min\{\underline{x}_i \underline{x}_j, \underline{x}_i \bar{x}_j, \bar{x}_i \underline{x}_j, \bar{x}_i \bar{x}_j\}, \\
 x_i &\leq \bar{x}_k / \underline{x}_j, & \text{if } \bar{x}_k > 0, \underline{x}_j > 0, \\
 x_i &\leq \bar{x}_k / \bar{x}_j, & \text{if } \bar{x}_k < 0, \underline{x}_j > 0, \\
 x_i &\geq \bar{x}_k / \bar{x}_j, & \text{if } \bar{x}_k > 0, \underline{x}_j < 0, \\
 x_i &\geq \bar{x}_k / \underline{x}_j, & \text{if } \bar{x}_k < 0, \underline{x}_j < 0, \\
 x_j &\leq \bar{x}_k / \underline{x}_i, & \text{if } \bar{x}_k > 0, \underline{x}_i > 0, \\
 x_j &\leq \bar{x}_k / \bar{x}_i, & \text{if } \bar{x}_k < 0, \underline{x}_i > 0, \\
 x_j &\geq \bar{x}_k / \bar{x}_i, & \text{if } \bar{x}_k > 0, \underline{x}_i < 0, \\
 x_j &\geq \bar{x}_k / \underline{x}_i, & \text{if } \bar{x}_k < 0, \underline{x}_i < 0.
 \end{aligned}$$

For a univariate constraint  $g_k(x_i) \leq x_k$  from (6.23f)–(6.23g), a lower bound on  $x_k$  can be computed from a lower bound on  $\{g_k(x_i) : x_i \in [\underline{x}_i, \bar{x}_i]\}$ , if available. Especially if  $g_k(\cdot)$  is monotonic, then a valid bound is given by  $g_k(\underline{x}_i)$  or  $g_k(\bar{x}_i)$ . Further, bounds on  $x_i$  are often available by inverting the function  $g_k$ , i.e., by computing an interval that contains  $g_k^{-1}([-\infty, \bar{x}_k]) \cap [\underline{x}_i, \bar{x}_i]$ .

Finally, (6.23h) is easily seen to yield the lower bound  $\lceil \underline{x}_i \rceil$  and the upper bound  $\lfloor \bar{x}_i \rfloor$  for a discrete variable  $x_i$ ,  $i \in I$ .

If a MINLP (6.1) is not reformulated into the form (6.16), more involved techniques may be applied to compute tight bounds for

$$(\inf | \sup)\{x_i : g_j(x) \leq 0, x \in [\underline{x}, \bar{x}]\}, \quad (i \in [n], j \in [m]),$$

see, e.g., Section 7.3.2 and Schichl and Neumaier [2005], Domes and Neumaier [2010].

### Probing

Probing is a technique where the effect of constraint-based bound tightening is tested when applied to a restricted version of  $[\underline{x}, \bar{x}]$ . That is, given a partitioning  $[\underline{x}, \bar{x}] = [\underline{x}, \tilde{x}] \cup [\tilde{x}, \bar{x}]$ , (6.22) is reformulated as

$$\begin{aligned}
 &\min(\inf\{x_i : x \in X \cap [\underline{x}, \tilde{x}], \langle c, x \rangle \leq \bar{v}\}, \inf\{x_i : x \in X \cap [\tilde{x}, \bar{x}], \langle c, x \rangle \leq \bar{v}\}) \\
 \text{and} \quad &\max(\sup\{x_i : x \in X \cap [\underline{x}, \tilde{x}], \langle c, x \rangle \leq \bar{v}\}, \sup\{x_i : x \in X \cap [\tilde{x}, \bar{x}], \langle c, x \rangle \leq \bar{v}\})
 \end{aligned}$$

and bound tightening is applied to each subproblem independently. If the box  $[\underline{x}, \tilde{x}]$  is found infeasible, then  $[\underline{x}, \bar{x}]$  can be tightened to  $[\tilde{x}, \bar{x}]$ . Usually, partitionings along a single variable are used.

In MIP,  $[\underline{x}, \bar{x}]$  is partitioned along binary variables only [Savelsbergh, 1994]. Thus, if constraint-based bound tightening finds that the box  $[\underline{x}, \bar{x}] \cap \{x_i = 0\}$  is infeasible, then  $x_i$  can be fixed to 1, and analogously for probing on  $x_i = 1$ . Further, implications like  $x_i = 0 \rightarrow x_j = 1$  can be found, which are useful for cut generation and heuristics, or equalities like  $x_j = x_k$  if  $x_j$  and  $x_k$  are fixed to the same value when probing on  $x_i = 0$

and  $x_i = 1$ . Since probing is computationally intensive, it is usually applied only in preprocessing (or during root node processing), see also Section 10.6 in Achterberg [2007].

For MINLP, Tawarmalani and Sahinidis [2002b] suggested to apply probing also for continuous (nonlinear) variables. If a feasible solution  $\hat{x}$  is available, then  $\tilde{x}_i$  is chosen below  $\hat{x}_i$  when probing on  $x_i \leq \tilde{x}_i$  and above  $\hat{x}_i$  when probing on  $x_i \geq \tilde{x}_i$ , since infeasibility cannot be expected for a box that contains  $\hat{x}_i$ . Even though computationally intensive, probing on nonlinear variables may also be applied (in a restricted form) during the branch-and-bound search [Tawarmalani and Sahinidis, 2002b, Belotti et al., 2009].

### Relaxation-Based Bound Tightening

While constraint-based bound tightening considers each constraint independently, relaxation-based bound tightening (also referred as optimality-based bound tightening in Belotti et al. [2009]) tries to better capture the interaction of the constraints. The idea is to replace the feasible set  $X$  in (6.22) by some relaxation  $\tilde{X}$  and to solve the corresponding optimization problems

$$\inf\{x_i : x \in \tilde{X}, \langle c, x \rangle \leq \bar{v}\} \quad \text{and} \quad \sup\{x_i : x \in \tilde{X}, \langle c, x \rangle \leq \bar{v}\}. \quad (6.25)$$

See Quesada and Grossmann [1993, 1995], Maranas and Floudas [1997], and Smith and Pantelides [1999] for early references on this idea. Often, the linear relaxation used to compute a lower bound on the optimal value of the MINLP is also used for  $\tilde{X}$ . Huang [2011] reported good results for water network operative planning problems when solving the root node relaxation of the original (6.22) by a MINLP solver.

Since the computational costs of solving up to  $2n$  problems (6.25) may be high (also if  $\tilde{X}$  is polyhedral), it is normally applied only at the root node, but may also be used during tree search in very limited form [Nowak and Vigerske, 2008, Belotti et al., 2009]. Further, filtering algorithms may allow a considerable reduction in the number of subproblems to be solved. For example, if it is known that  $\tilde{X} \cap \{x \in \mathbb{R}^n : x_i = \underline{x}_i\} \neq \emptyset$ , then the left variant of (6.25) does not need to be solved. Recently, Gleixner and Weltge [2013] proposed an algorithm that derives valid linear inequalities from a dual solution of (6.25) with polyhedral  $\tilde{X}$  at the root node of a branch-and-bound search and reuses these inequalities during tree search for constraint-based bound tightening, thereby allowing for a cheap approximation of relaxation-based bound tightening.

### Reduced Cost Bound Tightening

Reduced cost bound tightening [Nemhauser and Wolsey, 1988] uses the dual solution of a convex relaxation (e.g., (6.17)) of the MINLP and an available upper bound to infer new bounds. Assume a convex relaxation has been solved and yields a lower  $\underline{v}$  on the optimal value. Further, let variable  $x_i$  take value  $\underline{x}_i$  in the relaxations solution and let the dual variable  $\mu$  that corresponds to the constraint  $x_i \geq \underline{x}_i$  be nonzero (and thus positive). By duality,  $\mu$  allows to bound the optimal value of the relaxation for different values of  $\underline{x}_i$ . That is, one obtains the inequality  $\underline{v}(\ell) \geq \underline{v} + \mu(\ell - \underline{x}_i)$ , where  $\underline{v}(\ell)$  denotes the optimal value of the relaxation if the lower bound of  $x_i$  is replaced by  $\ell$ , see also Section 3.1.1.

## 6. Introduction

Since  $x_i \geq \underline{x}_i$  is active for  $\ell = \underline{x}_i$ , one can argue that  $x_i \geq \ell$  is also active when solving the relaxation for  $\ell \geq \underline{x}_i$ . Thus,  $v(x_i) \geq \underline{v} + \mu(x_i - \underline{x}_i)$  for  $x_i \geq \underline{x}_i$ . For optimal solutions of the MINLP, we have  $\underline{v}(x_i) \leq \bar{v}$ , which yields the inequality  $\bar{v} \geq \underline{v} + \mu(x_i - \underline{x}_i)$ . Hence,

$$x_i \leq \underline{x}_i + \frac{\bar{v} - \underline{v}}{\mu}$$

can be used to tighten the upper bound on  $x_i$ . Similarly, if  $x_i \leq \bar{x}_i$  is active with dual variable  $\mu < 0$ , then

$$x_i \geq \bar{x}_i + \frac{\bar{v} - \underline{v}}{\mu}.$$

### 6.1.6. Cutting Planes

#### Separating fractional solutions

As in MIP, the branch-and-bound-based algorithms (Algorithms 6.1, 6.4, and 6.5) can profit from cutting planes that tighten the NLP or LP relaxation by cutting off fractional solutions. For the LP relaxation (6.6), established techniques for the generation of, e.g., Gomory cutting planes, mixed-integer rounding cuts, or flowcover cuts [Marchand et al., 2002, Wolter, 2006] from the linear and linearized nonlinear inequalities can be applied. Extensions that allow to explicitly take the nonlinear constraints of a MINLP into account are a generalization of the procedure to generate Gomory cutting planes for nonlinear conic constraints [Çezik and Iyengar, 2005] and an extension of mixed-integer rounding cuts for second-order cone programs [Atamtürk and Narayanan, 2010].

Further, disjunctive programming techniques from Balas [1998] have been extended for convex MINLP [Stubbs and Mehrotra, 1999]. For a variable  $x_i$ ,  $i \in I$ , with fractional value  $\tilde{x}_i$  in the current relaxation, the method derives a cut that is valid for the union of the convex relaxations that correspond to the child nodes from a branching on  $x_i$ . This method has been applied to mixed-integer second-order cone programs by Drewes [2009] and to convex MINLPs by Kiliç et al. [2010]. To improve performance, Kiliç et al. [2010] replace the nonlinear separation problem by a linear outer-approximation that is successively improved by further cuts. Further, Kiliç et al. [2011] discuss how to reuse information that is collected from relaxations solved during strong branching for cut generation. For nonconvex MINLPs, Belotti [2012a] applied disjunctive programming to the union of linear relaxations in sibling nodes of the branch-and-bound tree.

#### Reducing the convexification gap

Next to separation techniques for fractional solutions, also methods that tighten the convex relaxation of a nonconvex MINLP have been developed. Even though the convex envelopes in (6.17) are already the best convex underestimator for each constraint function  $g_k(\cdot)$  or  $x_i x_j$ , a further tightening of the relaxation may be possible if one finds cuts that are valid for the feasible set defined by one or several constraints, e.g., for  $\{x \in [\underline{x}, \bar{x}] : x_i x_j \leq x_k\}$  where  $(i, j, k) \in \mathcal{T}_p$  is fixed, or  $\{x \in [\underline{x}, \bar{x}] : x_i x_j \leq x_k, (i, j, k) \in \mathcal{T}_p\}$ . In the following, we review some of the separation techniques that have been developed for MIQCPs.

Consider (6.1) with linear objective function and only linear or quadratic constraints,

$$\min\{\langle c, x \rangle : Ax \leq b, \langle x, Q_j x \rangle + \langle q_j, x \rangle + \bar{q}_j \leq 0, x \in [\underline{x}, \bar{x}], x_I \in \mathbb{Z}^{|I|}\}. \quad (6.26)$$

A common idea to many methods for solving (6.26) is to introduce auxiliary variables  $X_{i,j}$ , which model the quadratic terms  $x_i x_j$  via the equation  $X_{i,j} = x_i x_j$ ,  $i, j \in [n]$ . The MIQCP (6.26) is then reformulated to

$$\min\{\langle c, x \rangle : Ax \leq b, \langle Q_j, X \rangle + \langle q_j, x \rangle + \bar{q}_j \leq 0, x \in [\underline{x}, \bar{x}], x_I \in \mathbb{Z}^{|I|}, X = xx^\top\}, \quad (6.27)$$

where  $\langle A, B \rangle := \sum_{i,j} a_{i,j} b_{i,j}$  denotes the inner product of two square matrices  $A = (a_{i,j})_{i,j}$  and  $B = (b_{i,j})_{i,j}$ . The only nonlinear constraint in (6.27) is the equation  $X = xx^\top$ . Dropping this constraint yields a mixed-integer linear relaxation. Several techniques to tighten it have been developed, see Burer and Saxena [2012] for a recent survey.

The reformulation linearization technique (RLT) by Sherali and Adams [1999] suggests to multiply two linear constraints of (6.27) and to replace the products  $x_i x_j$  by  $X_{i,j}$ . When doing so for  $x_i \geq \underline{x}_i$  and  $x_j \geq \underline{x}_j$  or for  $x_i \leq \bar{x}_i$  and  $x_j \leq \bar{x}_j$ , one obtains the McCormick inequalities (6.11).

A semidefinite programming (SDP) relaxation of (6.27) is obtained by relaxing the constraint  $X = xx^\top$  to  $X - xx^\top \succeq 0$ , which is convex and can equivalently be written as

$$\tilde{X} := \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0, \quad (6.28)$$

see Shor [1987]. Anstreicher [2009] has shown for the continuous case, that combining RLT and SDP relaxations gives substantially better bounds than the individual ones.

However, the computational costs to solve a SDP relaxation are sometimes too high for real world problems. To convey some of the strength of a SDP relaxation into a linear relaxation, Sherali and Fraticelli [2002] suggest to write the constraint (6.28) as  $\langle v, \tilde{X} v \rangle \geq 0$  for all  $v \in \mathbb{R}^{n+1}$  and to adopt a cutting plane approach that separates inequalities  $\langle v, \tilde{X} v \rangle \geq 0$  for good choices of  $v \in \mathbb{R}^{n+1}$ . The method starts by computing a spectral decomposition for a given  $\tilde{X}^*$ . Letting  $v$  correspond to a orthonormal eigenvector of  $\tilde{X}^*$  with negative eigenvalue  $\lambda$ , one has  $\langle v, \tilde{X}^* v \rangle = \langle v, \lambda v \rangle = \lambda < 0$ . Thus,  $\langle v, \tilde{X} v \rangle \geq 0$  is a valid inequality that cuts off  $\tilde{X}^*$ . Unfortunately, these cuts may be dense, i.e., have nonzero entries for all or most variables, which can deteriorate the performance of the used LP solver. Thus, Qualizza et al. [2009] discuss sparsification schemes for this cut. If all eigenvalues of  $\tilde{X}^*$  are nonnegative, then  $\tilde{X}^*$  is already positive-semidefinite.

Considering again the original form  $X - xx^\top = 0$ , valid equations  $\langle v, (X - xx^\top) v \rangle = 0$  are obtained for any  $v \in \mathbb{R}^n$ . Rearranging yields the two inequalities  $\langle v, x \rangle^2 \leq \langle v, X v \rangle$  and  $-\langle v, x \rangle^2 \leq -\langle v, X v \rangle$ . The first inequality, similar to the one from Sherali and Fraticelli [2002], is convex and thus may be added as it or in linearized form to a relaxation. To utilize also the second inequality, Saxena, Bonami, and Lee [2010] suggest to underestimate  $-\langle v, x \rangle^2$  by a secant for both cases of the disjunction

$$\langle v, x \rangle \leq \langle v, x^* \rangle \quad \vee \quad \langle v, x \rangle \geq \langle v, x^* \rangle,$$

## 6. Introduction

(assuming that lower and upper bounds on  $\langle v, x \rangle$  are available), cf. (6.10), and to use disjunctive programming techniques [Balas, 1998] to derive a cut that is valid for both cases. Subsequently, Saxena, Bonami, and Lee [2011] used lift-and-project [Balas, 2001] to compute cuts for the original formulation (6.26), i.e., without requiring the additional variables  $X_{i,j}$ , by projecting from the RLT relaxation of (6.27).

Recently, techniques have also been developed to compute outer-approximations of the feasible set given by a single quadratic constraint that are tighter than those obtained by decomposing the constraint into single square and bilinear terms and using convex envelopes for each term separately. For example, for  $h(x) = \prod_{i=1}^n x_i$ , a recursive McCormick relaxation, obtained by introducing variables  $y_1, \dots, y_n$  that satisfy  $y_i = x_i y_{i+1}$ ,  $i \in [n-1]$ , and  $y_n = x_n$  and applying formula (6.11) to each term  $x_i y_{i+1}$ , is not necessarily also a convex envelope of  $h(x)$ . In fact, Ryoo and Sahinidis [2001], Luedtke et al. [2012] show, that the recursive McCormick relaxation is the convex envelope of  $h^e(x)$  if  $[\underline{x}, \bar{x}] = [0, \bar{x}]$  or  $[\underline{x}, \bar{x}] = [-\bar{x}, \bar{x}]$  for some  $\bar{x} \in \mathbb{R}_+^n$ , but also can be arbitrarily worse than the convex envelope in general.

For multilinear functions, Rikun [1997] has shown, that the convex envelopes over a cartesian product of polytopes is polyhedral and is completely determined by the vertices of the product set. Bao et al. [2009] used this observation to develop a separation algorithm for facets of the convex envelope of bilinear quadratic functions. Further, Jach et al. [2008] characterized the convex envelope of bivariate quadratic functions.

For the bilinear covering set  $\{(x, y) \in \mathbb{R}_+^{2n} : \sum_{i=1}^n (a_i x_i y_i + b_i x_i + c_i y_i) \geq r\}$ , where  $a_i, b_i, c_i \geq 0$ ,  $i \in [n]$ , and  $r > 0$ , Tawarmalani, Richard, and Chung [2010] have shown, that the convex hull can be written as  $\{(x, y) \in \mathbb{R}_+^{2n} : \sum_{i=1}^n \mu_i(x, y) \geq 2r\}$ , where  $\mu_i(x, y) = b_i x_i + c_i y_i + \sqrt{(b_i x_i + c_i y_i)^2 + 4a_i r x_i y_i}$  is a concave function. Further characterizations are available for bilinear knapsack sets ( $a_i = 1$ ,  $b_i = 0$ ,  $c_i = 0$ ) where  $x$  and/or  $y$  is restricted to  $\mathbb{Z}_+^n$ .

Finally, Belotti, Miller, and Namazifar [2011] developed a technique to compute valid cuts for the set  $\{x \in \mathbb{R}^3 : x_1 x_2 = x_3, x \in [\underline{x}, \bar{x}]\}$  with  $\underline{x} \geq 0$  by lifting a tangent on the curve  $x_1 x_2 = \underline{x}_3$  to be valid for all  $x_3 \in [\underline{x}_3, \bar{x}_3]$ . If  $\bar{x}_3 < \bar{x}_1 \bar{x}_2$ , then the resulting cut is tighter than the one obtained by using a McCormick underestimator (6.11), since it explicitly takes the bound on  $x_3$  into account.

### 6.1.7. Primal Heuristics

The aim of primal heuristics is to find feasible solutions early in the solving process. For branch-and-bound algorithms, the presence of good feasible solutions allows to guide the search process, to prune nodes with lower bound above the upper bound, and to fix variables by propagating the objective function (cf. Section 6.1.5). Further, they allow for an earlier interruption of the solving process in case one is only interested in a solution whose objective function value is within a given percentage of the optimal value. Many MIP heuristics can roughly be categorized as *rounding* heuristic, *diving* heuristic, or *large neighborhood search* heuristic. The concepts that were originally developed for MIP are often also applicable for convex MINLPs. However, also extensions that treat the nonlinear constraints different than the linear ones have been developed.

### Rounding Heuristics

Rounding heuristics for MIP try to round fractional values of discrete variables in a solution of the LP relaxation such that it becomes feasible for the MIP. The same methodology can be applied to NLP-based branch-and-bound algorithms for MINLPs.

### Diving Heuristics

Bonami and Gonçalves [2012] adapted diving heuristics from MIP [Berthold, 2006] for use within a NLP-based branch-and-bound algorithm for MINLPs. These heuristics simulate a depth-first-search in the branch-and-bound tree by iteratively resolving the NLP relaxation and restricting the bounds of fractional variables to integral values. Bonami and Gonçalves [2012] also propose a scheme where diving is only performed with respect to the nonlinear discrete variables and the sub-MIP that is obtained from the MINLP by fixing all nonlinear variables to their value in the NLP relaxation in the last diving problem is solved by a MIP solver.

### Large Neighborhood Search

Large neighborhood search is a widely applicable metaheuristic concept. The main idea is to restrict the search for “good” solutions to a neighborhood of specific points which are usually close to optimal or feasible solutions. The hope is that such a restriction makes the subproblem much easier to solve, while still providing solutions of high quality.

For MIP, large neighborhood search has been realized in a series of primal heuristics, see Berthold [2006] for an overview. Berthold et al. [2011] have shown that many large neighborhood search heuristics originally developed for MIP (in particular LOCAL BRANCHING, RENS, RINS, DINS, and CROSSOVER) can generically be extended to the much broader classes, including MINLPs, see also Section 7.8.

While the large neighborhood search heuristics from the MIP context obtain their subproblem by restricting the search space for the discrete variables, a new heuristic specific for MINLP that restricts the search space for the nonlinear variables has been developed by Berthold and Gleixner [2009, 2012]. Here, the idea is to fix a minimal number of nonlinear variables in a way that the resulting subproblem becomes a MIP, which is hopefully easier to solve than the original MINLP. In difference to the specialized diving heuristic of Bonami and Gonçalves [2012] (see above), only a small subset of the nonlinear variables may need to be fixed and the diving procedure that fixes the variables applies fast domain propagation instead of repeated solutions of the relaxation.

Finally, a MINLP specific extension of the local branching heuristic has been developed in Nannicini et al. [2009] and a combination of variable neighborhood search and local branching with other MINLP heuristics has been developed in Liberti et al. [2011].

### Feasibility Pump

The Feasibility Pump for MIP [Fischetti, Glover, and Lodi, 2005] has been extended to MINLP by Bonami et al. [2009]. The idea is to compute a sequence of points by alternated

## 6. Introduction

projection on the feasible sets of the MIP relaxation (6.5) and the NLP relaxation (6.2). In the case of a convex MINLP, the method can be shown to converge to a feasible solution or to prove infeasibility of the problem. A variant that replaces the projection onto the MIP relaxation by a rounding step is discussed in Bonami and Gonçalves [2012].

Extensions of the Feasibility Pump heuristic that account for nonconvex nonlinear constraints have been discussed in D’Ambrosio, Frangioni, Liberti, and Lodi [2010, 2012] and Nannicini and Belotti [2012].

## 6.2. Solvers

In the following, we give a brief overview of the state-of-the-art in software for the solution of MINLPs. We establish several groupings with respect to various features and give concise individual descriptions for each solver. The presentation is taken from Bussieck and Vigerske [2010].

We consider the MINLP (6.1) and the subclasses MIQCP and MIQQP stated in Table 6.1. Additionally to integer requirements on variables, other kinds of discrete constraints are sometimes supported by MINLP solvers. These are, e.g., special-ordered-set constraints (only one (SOS type 1) or two consecutive (SOS type 2) variables in an (ordered) set are allowed to be nonzero) [Beale and Tomlin, 1970], semicontinuous variables (the variable is allowed to take either the value zero or a value above some bound), semiinteger variables (like semicontinuous variables, but with an additional integer restriction), and indicator variables (a binary variable indicates whether a certain set of constraints must hold). In all cases it is possible to reformulate such constraints into a standard form by introducing additional variables and linear constraints.

### 6.2.1. History

To the best of our knowledge, the earliest commercial software package that could solve MINLP problems was SCICONIC in the mid 1970’s [Beale, 1980, Forrest and Tomlin, 2007, SCICON Ltd., 1989]. Rather than handling nonlinearities directly, linked Special-Ordered-Set variables provided a mechanism to represent low dimensional nonlinear terms by a piecewise linear approximation and thus allowed to use a MIP solver to obtain approximate solutions for the MINLP. In the mid 1980’s Grossmann and Kocis developed DICOPT, a general purpose algorithm for convex MINLPs based on the outer approximation method, see Algorithm 6.2. Since then, a number of academic and commercial codes for convex MINLP have emerged, either based on outer approximation using MIP relaxations (see Algorithms 6.2 and 6.3), an integration of outer approximation into a LP-based branch-and-cut (see Algorithm 6.4), or NLP-based branch-and-bound algorithms (see Algorithm 6.1). For the global solution of nonconvex MINLP, the first general purpose solvers were ALPHABB, BARON, and GLOP, all based on convexification techniques for nonconvex constraints and spatial branch-and-bound, see Sections 6.1.2 and 6.1.3.

### 6.2.2. Groupings

#### Embedded vs. independent

Due to the high complexity of MINLP and the wide range of applications that can be modeled as MINLPs, it is sometimes desirable to customize the MINLP solver for a specific application in order to achieve good computational performance [Bragalli et al., 2012, Bussieck, 2003, Farkas et al., 2008]. Further, MINLP solvers are often built by combining LP, MIP, and NLP solvers. These are two main reasons for tightly integrating some MINLP solvers into modeling systems (general systems like AIMMS [Roelofs and Bisschop, 2009], AMPL [Fourer et al., 1993], and GAMS [Brooke et al., 2012] or vendor specific systems like FICO XPRESS-MOSEL [FICO, 2009b], LINGO [Schrage, 2008], and OPL [Dong, 2009]). For example, the AIMMS Outer Approximation solver AOA allows modifications of its algorithm by the user. Further, the solvers DICOPT and SBB are exclusively available for GAMS users since they revert to MIP and NLP solvers in the GAMS system for the solution of subproblems. Also for an efficient use of the solver OQNLP it is preferable to use one of the GAMS NLP solvers.

On the other side, there are many solvers that can be used independently of a modeling system, even though they may still require the presence of a LP, MIP, or NLP solver plugin. However, often also these “independent” solvers are used within a modeling system, since the modeling system typically provides evaluators for nonlinear functions, gradients, and Hessians and gives easy access to algebraic information about the problem.

#### Extending MIP vs. extending NLP vs. starting from scratch

MINLP solvers are not always developed completely from scratch. In many cases, a MIP or NLP solver builds the basis for an extension towards MINLP. Solvers that can be categorized as extending a MIP solver with capabilities for nonlinear objectives and constraints are BONMIN, COUENNE, CPLEX, FICO XPRESS-OPTIMIZER, FILMINT, LINDOAPI without global option, MOSEK, and SCIP<sup>14</sup>. On the other hand, solvers where a NLP solver was extended to handle discrete variables are BNB, FICO XPRESS-SLP, FMINCONSET, KNITRO, MILANO, MINLPBB, MISQP, OQNLP, and SBB.

Finally, there is a group of solvers which were more-or-less developed from scratch, but which may solve LP, MIP, NLP, or MINLP subproblems. In this category we have ALPHABB, ALPHA ECP, AOA, BARON, DICOPT, GLOMIQO, LAGO, LINDOAPI, MIDACO, and MINOTAUR.

#### Algorithms

Most solvers implement one or several of three algorithmic ideas to tackle MINLPs, see also Section 6.1. First, there are branch-and-bound solvers that use NLP relaxations (see Algorithm 6.1): ALPHABB, BNB, BONMIN (in B-BB mode), CPLEX, FICO XPRESS-OPTIMIZER, FICO XPRESS-SLP (in “SLP within MIP” mode), FMINCONSET, KNITRO, LINDOAPI without global option, MILANO, MINLPBB, MINOTAUR, MOSEK,

<sup>14</sup>We note, that SCIP is in fact a CIP framework, which includes a MIP solver, cf. Chapter 7.

## 6. Introduction

and SBB. Except for ALPHABB, all of them obtain the NLP relaxation by relaxing the integrality restriction in (6.1). Since the NLP solver used to solve this possibly nonconvex NLP relaxation usually ensures only local optimal solutions, these solvers work as heuristics in case of a nonconvex MINLP. The solver ALPHABB, however, generates a convex NLP relaxation by using convex underestimators for the nonlinear functions in (6.1), cf. Section 6.1.2. This solver can therefore be applied to nonconvex MINLPs, too.

As an alternative to relaxing integrality restrictions and keeping nonlinear constraints, some solvers keep the integrality constraints and instead replace the nonlinear functions by a linear relaxation. The resulting MIP relaxation is then solved by a MIP solver. Solvers that implement the outer-approximation algorithm (see Algorithm 6.2) are AOA, BONMIN (in B-OA mode), DICOPT, MISQP (with OA extension), and FICO XPRESS-SLP (in “MIP within SLP” mode). Since gradient-based linearizations yield an outer-approximation only for convex MINLPs, these solvers ensure global optima only for convex MINLPs (solvers like DICOPT also try to compensate the effect of nonconvexity, see the corresponding paragraph in Section 6.1.1). The extended cutting plane algorithm (see Algorithm 6.3) is implemented by the solver ALPHAECF, which can be applied to convex as well as pseudo-convex MINLPs.

A third class of solvers are those which integrate the linearization of the nonlinear functions into the branch-and-cut process, see Algorithms 6.4 and 6.5. Thus, a LP relaxation is successively solved, new linearizations are generated to improve the relaxation, and integrality constraints are enforced by branching on the discrete variables. Solvers which use gradient-based linearizations are AOA, BONMIN (in B-QG mode) and FILMINT.

Since the use of gradient-based linearizations in a branch-and-cut algorithm ensures global solutions only for convex MINLPs, solvers for nonconvex MINLPs use convexification techniques to compute linear underestimators of a nonconvex function, cf. Section 6.1.2. However, the additional convexification step may require to branch also on continuous variables in nonconvex terms (so called *spatial* branching, cf. Section 6.1.3). Such a branch-and-cut algorithm is implemented by BARON, COUENNE, GLOMIQO, LAGO, LINDOAPI, and SCIP. In difference to the other solvers, GLOMIQO uses a MIP relaxation, i.e., does not relax integrality requirements.

The remaining solvers implement a different methodology. BONMIN (in B-Hyb mode) alternates between LP and NLP relaxations during one branch-and-bound process. MISQP integrates the handling of integrality restrictions into the solution of a nonlinear program via sequential quadratic programming, i.e., it ensures that  $f(x)$  and  $g(x)$  are only evaluated at points where  $x_I$  is integral. MIDACO applies an extended ant colony optimization method and can use MISQP as a local solver. Finally, OQNLP applies a randomized approach by sampling starting points and fixings of integer variables for the solution of NLP subproblems.

### Capabilities

Not every solver accepts general MINLPs as input. Solvers that currently handle only MIQCPs or second order cone (SOC) programs are CPLEX, GLOMIQO, FICO XPRESS-OPTIMIZER, and MOSEK. All solvers support convex quadratic functions. Further,

nonconvex quadratic functions that involve only binary variables are supported by CPLEX and FICO XPRESS-OPTIMIZER. Quadratic constraints that permit a SOC representation are supported by CPLEX. SOC constraints are supported by MOSEK. CPLEX and GLOMIQO support quadratic constraints in any form, but CPLEX ensures global optimality only for the cases mentioned before.

Solvers that guarantee global optimal solutions for general convex MINLPs but not for general nonconvex MINLPs are ALPHAIECP, AOA, BNB, BONMIN, DICOPT, FICO XPRESS-SLP, FILMINT, FMINCONSET, KNITRO, LAGO, LINDOAPI without global option, MILANO, MINLPBB, MINOTAUR, MISQP with OA extension, and SBB. In case of a nonconvex MINLP, these solvers can still be used as a heuristic. Especially branch-and-bound-based algorithms that use NLPs for bounding often find good solutions also for nonconvex problems, while pure outer-approximation-based algorithms may easily run into infeasible LP or MIP relaxations due to wrong cutting planes. Note, that ALPHAIECP ensures global optimal solutions also for pseudo-convex MINLPs.

Solvers that also guarantee global optimality for nonconvex general MINLPs require an algebraic representation of the nonlinear functions for the computation of convex envelopes and underestimators, cf. Section 6.1.2. That is, each function need to be provided as a composition of basic arithmetic operations and functions (addition, multiplication, power, exponential, trigonometric, ...) on constants and variables. The solvers ALPHABB, BARON, COUENNE, LINDOAPI, and SCIP belong into this category.

MIDACO, MISQP, and OQNLP can handle general MINLPs, but do not guarantee global optimality even on convex problems.

### 6.2.3. List of solvers

In the following we briefly discuss individual solvers for MINLPs. We have excluded solvers from this list that are clearly no longer available (e.g., SCICONIC). The solvers listed below have different levels of reliability and activity with respect to development and maintenance. Wide availability through modeling systems and other popular software indicates that a solver has reached a decent level of maturity. Hence, in this list, we mention availability (e.g., open source, standalone binary, interfaces to general modeling systems) in addition to a solver's developer, capability, and algorithmic details. Table 6.2 summarizes the list of solvers and indicates for each solver the availability via AIMMS, AMPL, GAMS, and the NEOS server [Czyzyk et al., 1998].

#### **alphaBB ( $\alpha$ -Branch-and-Bound) [Adjiman et al., 2000, Androulakis et al., 1995].**

This solver has been developed by the research group of C. A. Floudas at the Computer-Aided Systems Laboratory of Princeton University. It is available to their collaborators.

ALPHABB can be applied for convex and nonconvex MINLPs. It implements a branch-and-bound algorithm that utilizes convex NLPs for bounding. Convex envelopes and tight convexifications are obtained for specially structured nonconvex terms (e.g., bilinear, trilinear, multilinear, univariate concave, edge concave, generalized polynomials, fractional), and convex  $\alpha$ -underestimators for general twice continuously differentiable functions, see also Section 6.1.2.

## 6. Introduction

**AlphaECP ( $\alpha$ -Extended Cutting Plane) [Westerlund and Lundquist, 2003, Westerlund and Pörn, 2002].** This solver has been developed by the research group of T. Westerlund at the Process Design and Systems Engineering Laboratory of the Åbo Akademi University, Finland. It is available as a commercial solver within GAMS.

ALPHAECP ensures global optimal solutions for convex and pseudo-convex MINLPs. It generates and successively improves a MIP outer approximation of a neighborhood of the set of optimal solutions of the MINLP and can solve NLP subproblems to find feasible solutions early. The MIP is here refined by linearizing nonlinear constraints at solutions of the MIP outer approximation, cf. Algorithm 6.3. By shifting hyperplanes, pseudo-convex functions can also be handled.

**AOA (AIMMS Outer Approximation) [Roelofs and Bisschop, 2009].** This solver has been developed by Paragon Decision Technology. AOA is available as an “open solver” inside AIMMS. The open solver approach allows the user to customize the algorithm for a specific application.

AOA ensures global optimal solutions only for convex MINLPs. It generates and successively improves a MIP outer approximation of the MINLP and can solve NLP subproblems to find feasible solutions early. In contrast to ALPHAECP, AOA constructs a MIP outer approximation of the feasible region of the MINLP by linearizing nonlinear functions in solutions of NLP subproblems, cf. Algorithm 6.2. Since for a nonconvex constraint such a linearization may not be valid, the MIP relaxation is modified such that the corresponding hyperplane is allowed to move away from its support point, see also Section 6.1.1. Recently, also a branch-and-bound algorithm that utilizes LPs for bounding, cf. Algorithm 6.4 has been added to AOA.

**BARON (Branch-And-Reduce Optimization Navigator) [Tawarmalani and Sahinidis, 2002a, 2005].** This solver was originally developed by the group of N.V. Sahinidis at the University of Illinois at Urbana-Champaign and is currently developed by N.V. Sahinidis at Carnegie Mellon University and M. Tawarmalani at Purdue University. It is available as a commercial solver within AIMMS and GAMS.

BARON can be applied to convex and nonconvex MINLPs. It implements a spatial branch-and-bound algorithm, cf. Algorithm 6.5, that utilizes LPs for bounding. The algorithm is enhanced by using advanced box reduction techniques and new convexification techniques for quadratic functions [Bao et al., 2009]. Further, BARON is able to use NLP relaxations for bounding [Ghildyal and Sahinidis, 2001], even though this option is not encouraged.

**bnb (Branch 'n Bound) [Kuipers, 2003].** This solver has been developed by K. Kuipers of the Department of Applied Physics at the University of Groningen. It is available as MATLAB [MathWorks, 2009] source.

BNB ensures global optimal solutions for convex MINLPs. It implements a branch-and-bound algorithm utilizing nonlinear relaxations for the bounding step, cf. Algorithm 6.1. The NLPs are solved by the MATLAB Optimization Toolbox routine FMINCON.

**Bonmin (Basic Open-source Nonlinear Mixed Integer Programming) [Bonami et al., 2008].** This open-source solver has been developed primarily by P. Bonami in a cooperation of Carnegie Mellon University and IBM Research, now at University Marseille. It is available in source code and as standalone binaries from COIN-OR (Computational Infrastructure for Operations Research) [Lougee-Heimer, 2003], has an AMPL interface, and is distributed as a free solver within GAMS.

BONMIN ensures global optimal solutions only for convex MINLPs. Among others, it implements the Algorithms 6.2 (B-OA), 6.4 (B-QG), 6.1 (B-BB), and a hybrid of B-QG and B-BB which alternates between LP and NLP relaxations for bounding (B-Hyb). BONMIN includes several diving heuristics, RINS, and a feasibility pump (see Section 6.1.7 and Bonami and Gonçalves [2012], Bonami et al. [2009]). BONMIN has been implemented on top of the MIP solver CBC<sup>15</sup> and can use FILTERSQP [Fletcher and Leyffer, 2002] and IPOPT [Wächter and Biegler, 2006] as NLP solvers.

**Couenne (Convex Over and Under Envelopes for Nonlinear Estimation) [Belotti et al., 2009].** This open-source solver has been developed primarily by P. Belotti, originally in a cooperation of Carnegie Mellon University and IBM Research, and now at Clemson University. It is available in source code and as standalone binaries from COIN-OR, has an AMPL interface, and is distributed as a free solver within GAMS.

COUENNE ensures global optimal solutions for convex and nonconvex MINLPs. It implements a spatial branch-and-bound algorithm that utilizes LPs for bounding. Similar to BARON, the linear outer-approximation is generated from a reformulation of the MINLP, see Section 6.1.2. The algorithm is enhanced by bound tightening techniques (see Section 6.1.5, Belotti et al. [2010], and Belotti [2012b]), disjunctive cuts [Belotti, 2012a], MINLP heuristics (see Section 6.1.7, Nannicini et al. [2009] and Nannicini and Belotti [2012]), and symmetry handling [Liberti, 2012]. COUENNE has been implemented on top of BONMIN.

**CPLEX [IBM, 2012].** This solver has been developed by CPLEX Optimization, Inc. (later acquired by ILOG and recently acquired by IBM). It is available as standalone binaries and as a component in many modeling systems.

CPLEX can solve convex MIQCPs. For models that only have binary variables in the potentially indefinite quadratic matrices, CPLEX automatically reformulates the problem to an equivalent MIQCP with positive-semidefinite matrices. It implements a branch-and-bound algorithm that utilizes LPs or quadratically constraint quadratic programs for bounding. Recently, also an option to solve general nonconvex MIQCPs by Algorithm 6.1 has been added, but global optimality is not guaranteed for this case.

**DICOPT (Discrete and Continuous Optimizer) [GAMS Development Corp., 2012, Kocis and Grossmann, 1989].** This solver has been developed by the research group of I. E. Grossmann at the Engineering Research Design Center at Carnegie Mellon University. It is available as a commercial solver within GAMS.

<sup>15</sup><http://projects.coin-or.org/Cbc>

## 6. Introduction

DICOPT implements the outer-approximation algorithm 6.2 and thus ensures global optimal solutions for convex MINLPs. To accommodate also nonconvex MINLPs, inequality-relaxation of nonlinear equality constraints and the augmented penalty function relaxation for linearizations of nonlinear functions, see Section 6.1.1, are available. Since for this case valid lower bounds cannot be obtained, the termination criterion is based on lack of improvement in the objective of the NLP subproblem.

**FICO Xpress-Optimizer [FICO, 2009a].** This solver has been developed by Dash Optimization (later acquired by FICO). It is available as standalone binaries and as a component in many modeling systems.

FICO XPRESS-OPTIMIZER can solve convex MIQCPs. For models that only have binary variables in the potentially indefinite quadratic matrices, FICO XPRESS-OPTIMIZER automatically reformulates the problem to an equivalent MIQCP with positive-semidefinite matrices. It implements a branch-and-bound algorithm that utilizes quadratically constrained quadratic programs for bounding, cf. Algorithm 6.1.

**FICO Xpress-SLP [FICO, 2008].** This solver has been developed by Dash Optimization (later acquired by FICO). It is available as standalone binaries and as a FICO XPRESS-MOSEL module [FICO, 2009b].

FICO XPRESS-SLP ensures global optimal solutions for convex MINLPs. It implements three algorithms: The (default) “SLP within MIP” variant is a branch-and-bound algorithm that utilizes NLPs for bounding, cf. Algorithm 6.1. The NLP subproblems are solved by Successive Linear Programming (SLP). Solving MIPs as subproblems of the SLP algorithm leads to the “MIP within SLP” variant, which is comparable with the outer-approximation algorithm, cf. Algorithm 6.2. A third variant (“SLP then MIP”) solves first a NLP relaxation (by SLP), then a MIP relaxation, and finally a NLP subproblem to obtain a feasible solution to the MINLP [FICO, 2008]. To accommodate also nonconvex constraints, in all variants, the hyperplanes obtained from gradient-based linearizations in SLP can move away from their support point, cf. Section 6.1.1.

**FilMINT (Filter-Mixed Integer Optimizer) [Abhishek et al., 2010].** This solver has been developed by the research groups of S. Leyffer at the Laboratory for Advanced Numerical Simulations of Argonne National Laboratory and J. Linderoth at the Department of Industrial and Systems Engineering of Lehigh University. It has an AMPL interface.

FILMINT ensures global optimal solutions only for convex MINLPs. It implements a branch-and-bound algorithm that utilizes LPs for bounding, cf. Algorithm 6.4, where different strategies for choosing the linearization point for the nonlinear functions are available. Further, FILMINT includes several variants of disjunctive cutting planes for convex MINLP [Kiliç et al., 2010, 2011] and a feasibility pump. FILMINT has been implemented on top of the MIP solver MINTO [Nemhauser et al., 1994] and the NLP solver FILTERSQP [Fletcher and Leyffer, 2002].

**fminconset [Solberg, 2000].** This solver had been developed by I. Solberg at the Department of Engineering Cybernetics of the University of Trondheim (now NTNU). It is available as MATLAB source.

FMINCONSET ensures global optimal solutions for convex MINLPs. It implements the NLP-based branch-and-bound algorithm, cf. Algorithm 6.1. The NLPs are solved by the MATLAB Optimization Toolbox routine FMINCON.

**GloMIQO (Global Mixed-Integer Quadratic Optimizer) [Misener and Floudas, 2012b].** This solver has been developed by R. Misener and C. A. Floudas at the Computer-Aided Systems Laboratory of Princeton University. It is available as a commercial solver within GAMS.

GLOMIQO ensures global optimal solutions for convex and nonconvex MIQCPs. It implements a MIP-relaxation-based spatial branch-and-bound algorithm, cf. Algorithm 6.5, and employs a large collection of convexification and bound tightening techniques for quadratic constraints, see also Misener and Floudas [2012a]. An extension of GLOMIQO for general MINLPs is about to be released under the name ANTIGONE in the near future.

**Knitro [Byrd et al., 2006].** This solver has been developed by Ziena Optimization, Inc. It is available as standalone binary and as a component in many modeling systems.

KNITRO ensures global optimal solutions for convex MINLPs. MINLPs are solved by branch-and-bound, where both linear or nonlinear problems can be used for the bounding step, cf. Algorithms 6.1 and 6.4.

**LaGO (Lagrangian Global Optimizer) [Nowak and Vigerske, 2008].** This open-source solver had been developed by the research group of I. Nowak at the Department of Mathematics of Humboldt University Berlin. It is available in source code from COIN-OR and provides AMPL and GAMS interfaces.

LAGO ensures global optimal solutions for convex MINLPs and nonconvex MIQCPs. It implements a spatial branch-and-bound algorithm utilizing a linear relaxation for the bounding step. The relaxation is obtained by linearizing convex functions, underestimating quadratic nonconvex functions, and approximating nonconvex nonquadratic functions by quadratic ones.

**LindoAPI [Lindo Systems, Inc., 2010, Lin and Schrage, 2009].** This solver library has been developed by LINDO Systems, Inc. It is available within the LINDO environment [Lindo Systems, Inc., 2010], LINGO [Schrage, 2008], *What'sBest!* [Lindo Systems, Inc., 2009], and as a commercial solver within GAMS.

LINDOAPI ensures global optimal solutions for convex and nonconvex MINLPs. It implements a branch-and-cut algorithm that utilizes LPs for bounding [Gau and Schrage, 2003, Lin and Schrage, 2009]. Branching is performed for subproblems that are not provably infeasible and where nonconvex constraints are present or the LP relaxation has a fractional solution. LINDOAPI can also handle some nonsmooth or discontinuous functions like  $\text{abs}(\mathbf{x})$ ,  $\text{floor}(\mathbf{x})$ , and  $\text{max}(\mathbf{x}, \mathbf{y})$ .

## 6. Introduction

Additionally, LINDOAPI allows to disable the global solver components, by what the MIP solver is used together with nonlinear relaxations for the bounding step, cf. Algorithm 6.1. This option still ensures global optimal solutions for convex MINLPs. It was the first commercially available solver implementing a branch-and-bound algorithm utilizing nonlinear relaxations for bounding. The NLP relaxations are solved by CONOPT [Drud, 1994, GAMS Development Corp., 2012].

**MIDACO (Mixed Integer Distributed Ant Colony Optimization) [Schlüter and Gerdts, 2009, Schlüter et al., 2012].** This solver has been developed by M. Schlüter at the Theoretical & Computational Optimization Group of the University of Birmingham. It works as a library with Matlab, C/C++, and Fortran interfaces and is available from the author on request.

MIDACO can be applied to convex and nonconvex MINLPs. It implements an extended ant colony search method based on an oracle penalty function and can be combined with MISQP as solver for local searches. It targets applications where the problem formulation is unknown ( $f(x)$  and  $g(x)$  are black-box functions) or involves critical properties like nonconvexities, discontinuities, flat spots, or stochastic distortions. Further, MIDACO can exploit distributed computer architectures by parallelizing function evaluation calls.

**MILANO (Mixed-Integer Linear and Nonlinear Optimizer) [Benson, 2011].** This solver is developed by H. Y. Benson at the Department of Decision Sciences of Drexel University. It is still in development and available as MATLAB source.

MILANO ensures global optimal solutions for convex MINLPs. It implements a NLP-based branch-and-bound algorithm, cf. Algorithm 6.1. The NLPs are solved by LOQO [Vanderbei and Shanno, 1999], where special emphasis is put on how to warmstart this interior-point solver.

**MINLPBB (Mixed Integer Nonlinear Programming Branch-and-Bound) [Leyffer, 2001].** This solver had been developed by R. Fletcher and S. Leyffer at the University of Dundee. It provides an AMPL interface and is available for MATLAB via the TOMLAB Optimization Environment [Holmström, 1999].

MINLPBB ensures global optimal solutions for convex MINLPs. It implements a NLP-based branch-and-bound algorithm, cf. Algorithm 6.1, where NLP relaxations can be solved from several starting points in order to achieve higher robustness for nonconvex MINLPs. The NLPs are solved by FILTERSQP [Fletcher and Leyffer, 2002].

**MINOTAUR (Mixed-Integer Nonconvex Optimization Toolbox – Algorithms, Underestimators, Relaxations) [Mahajan and Munson, 2010, Mahajan et al., 2012].** This solver is developed by S. Leyffer, J. Linderoth, J. Luedtke, A. Mahajan, and T. Munson at Argonne National Laboratory and the University of Wisconsin-Madison. It provides an AMPL interface.

MINOTAUR ensures global optimal solutions for convex MINLPs. It implements a NLP-based branch-and-bound algorithm, cf. Algorithm 6.1, where the NLPs are

solved by IPOPT [Wächter and Biegler, 2006] or FILTERSQP [Fletcher and Leyffer, 2002]. Additionally, it offers to replace the NLP relaxations by faster to solve QP approximations, can recognize unions of second-order cones, and is continuously extended towards a solver for nonconvex MINLPs.

**MISQP (Mixed Integer Sequential Quadratic Programming) [Exler et al., 2012, Exler and Schittkowski, 2007].** This solver has been developed by the research group of K. Schittkowski at the Department of Computer Science of the University of Bayreuth. It works as a standalone library with a Fortran interface.

MISQP can be applied to convex and nonconvex MINLPs, but assumes that the values of the nonlinear functions  $f(x)$  and  $g(x)$  do not change drastically as a function of  $x_I$ . MISQP implements a modified sequential quadratic programming (SQP) method, where functions are only evaluated at points  $x$  with  $x_I \in \mathbb{Z}^{|I|}$ . It targets applications where the evaluation of  $f(x)$  or  $g(x)$  may be expensive. Additionally, a combination with outer-approximation that guarantees convergence for convex MINLPs is available [Lehmann, in preparation].

**MOSEK [MOSEK Corporation, 2009].** This solver has been developed by MOSEK ApS. It is available as a standalone binary, has AMPL and MATLAB interfaces, and is distributed as a commercial solver within AIMMS and GAMS.

MOSEK can be applied to convex MIQCPs and to mixed-integer conic programs. It implements a branch-and-bound method that utilizes quadratically constraint programs or SOC programs for bounding, cf. Algorithm 6.1.

**OQNLP (OptQuest Nonlinear Programming) [GAMS Development Corp., 2012, Ugray et al., 2007].** This solver has been jointly developed by OptTek Systems, Inc. and Optimal Methods, Inc. It is available as a standalone library, for MATLAB via the TOMLAB Optimization Environment [Holmström, 1999], and is distributed as a commercial solver within GAMS.

OQNLP is a heuristic that can be applied to any MINLP. It implements a multistart scatter search algorithm which solves NLP subproblems with fixed discrete variables.

**SBB (Simple Branch-and-Bound) [GAMS Development Corp., 2012].** This solver has been developed by ARKI Consulting and Development A/S. It is available as a commercial solver within GAMS.

SBB ensures global optimal solutions for convex MINLPs. It implements the NLP-based branch-and-bound algorithm 6.1. The NLP relaxations are solved by one (or several) of the NLP solvers available with GAMS. Using the GAMS Branch-Cut-and-Heuristic facility [Bussieck, 2003], SBB allows the user to implement a model-specify heuristic in the GAMS language.

**SCIP (Solving Constraint Integer Programs) [Achterberg, 2007, Berthold et al., 2009b].** This solver has been developed by the Optimization Department at the Zuse

## 6. Introduction

Institute Berlin and its collaborators. For academic institutions, it is available in source code and as standalone binary and is distributed within GAMS.

SCIP ensures global optimal solutions for convex and nonconvex MINLPs. It implements a spatial branch-and-bound algorithm that utilizes LPs for bounding. Similar to BARON, the outer-approximation is generated from a reformulation of the MINLP, see Section 6.1.2. The algorithm is discussed in more detail in Chapter 7.

### 6.2.4. Outlook

While state-of-the-art MIP solvers typically implement advanced automatic reformulation and preprocessing algorithms, such techniques are less commonly available in MINLP solvers, and in a limited form. Therefore, the modeler's choice of a problem formulation is still very important when solving a MINLP. However, software for guided automatic model reformulations and relaxations has recently been developed. LOGMIP [Vecchietti and Grossmann, 1999], one of the first systems available, translates a MINLP with disjunctions into a standard MINLP by applying bigM and convex hull reformulations [Grossmann and Lee, 2003]. More recently, frameworks like GAMS/EMP (Extended Mathematical Programming) [Ferris et al., 2009] and ROSE (Reformulation/Optimization Software Engine) [Liberti et al., 2009] provide a growing toolbox for reformulating MINLPs. Other recent activities like LIBMC [Mitsos et al., 2009] focus on (convex) relaxations for (nonconvex) MINLP.

Another important area is the collection and dissemination of MINLP models. Instance collections like MACMINLP<sup>16</sup> and MINLPLIB [Bussieck et al., 2003] provide valuable test cases for solver developers. The new Cyber-Infrastructure for MINLP [Grossmann and Lee, 2011] features a growing library of problems with high level model descriptions, reformulations, and problem instances.

---

<sup>16</sup><http://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP>

solver	AIMMS	AMPL	GAMS	NEOS	URL
<b>MIQP</b>					
CPLEX	✓	✓	✓	-	<a href="http://www.cplex.com">http://www.cplex.com</a>
FICO XPRESS-OPTIMIZER	✓	✓	✓	-	<a href="http://www.fico.com/xpress">http://www.fico.com/xpress</a>
GLOMIQO	-	-	✓	-	<a href="http://helios.princeton.edu/GLOMIQO">http://helios.princeton.edu/GLOMIQO</a>
MOSEK	✓	✓	✓	-	<a href="http://www.mosek.com">http://www.mosek.com</a>
ALPHABB	-	-	-	-	<a href="http://titan.princeton.edu">http://titan.princeton.edu</a>
ALPHAIECP	-	-	✓	✓	<a href="http://www.abo.fi/~twesterl">http://www.abo.fi/~twesterl</a>
AOA	✓	-	-	-	<a href="http://www.aimms.com">http://www.aimms.com</a>
BARON	✓	-	✓	✓	<a href="http://archimedes.cheme.cmu.edu/baron/baron.html">http://archimedes.cheme.cmu.edu/baron/baron.html</a>
BNB	-	-	-	-	<a href="http://www.mathworks.com/matlabcentral/fileexchange/95">http://www.mathworks.com/matlabcentral/fileexchange/95</a>
BONMIN	-	✓	✓	✓	<a href="https://projects.coin-or.org/Bonmin">https://projects.coin-or.org/Bonmin</a>
COUENNE	-	✓	✓	✓	<a href="https://projects.coin-or.org/Couenne">https://projects.coin-or.org/Couenne</a>
DICOPT	-	-	✓	✓	<a href="http://www.gams.com/solvers">http://www.gams.com/solvers</a>
FICO XPRESS-SLP	-	-	-	-	<a href="http://www.fico.com/xpress">http://www.fico.com/xpress</a>
FILMINT	-	✓	-	✓	<a href="http://www.neos-server.org/neos/solvers/minco:FiLMINT/AMPL.html">http://www.neos-server.org/neos/solvers/minco:FiLMINT/AMPL.html</a>
FMINCONSET	-	-	-	-	<a href="http://www.mathworks.com/matlabcentral/fileexchange/96">http://www.mathworks.com/matlabcentral/fileexchange/96</a>
KNITRO	✓	✓	✓	✓	<a href="http://www.ziena.com">http://www.ziena.com</a>
LAGO	-	✓	✓ <sup>a</sup>	-	<a href="https://projects.coin-or.org/LaGO">https://projects.coin-or.org/LaGO</a>
LINDOAPI	-	-	✓	✓	<a href="http://www.lindo.com">http://www.lindo.com</a>
LOGMIP	-	-	✓	-	<a href="http://www.logmip.ceride.gov.ar">http://www.logmip.ceride.gov.ar</a>
MIDACO	-	-	-	-	<a href="http://www.midaco-solver.com">http://www.midaco-solver.com</a>
MILANO	-	-	-	-	<a href="http://www.pages.drexel.edu/~hvb22/milano">http://www.pages.drexel.edu/~hvb22/milano</a>
MINLPBB	-	✓	-	✓	<a href="http://wiki.mcs.anl.gov/Sven_Leyffer's_Software">http://wiki.mcs.anl.gov/Sven_Leyffer's_Software</a>
MINOTAUR	-	✓	-	-	<a href="http://wiki.mcs.anl.gov/minotaur">http://wiki.mcs.anl.gov/minotaur</a>
MISQP	-	-	-	-	<a href="http://www.klaus-schittkowski.de/misqp.htm">http://www.klaus-schittkowski.de/misqp.htm</a>
OQNLP	-	-	✓	✓	<a href="http://www.gams.com/solvers">http://www.gams.com/solvers</a>
SBB	-	-	✓	✓	<a href="http://www.gams.com/solvers">http://www.gams.com/solvers</a>
SCIP	-	✓	✓	✓	<a href="http://scip.zib.de">http://scip.zib.de</a>

Table 6.2.: An overview on solvers for MINLP. The first row indicates whether a solver accepts only problems with quadratic functions (MIQPPs) or general nonlinear functions (MINLPs).

<sup>a</sup>interface for MINLPs available, but not included in GAMS distribution



# 7. A Constraint Integer Programming Approach to MINLP

As seen in the previous chapter, MIP solving techniques can often be extended for solving MINLPs. Analogously, several authors have shown that an integrated approach of *constraint programming (CP)* and MIP can help to solve optimization problems that were intractable with either of the two methods alone, for an overview see Hooker [2007].

The paradigm of *constraint integer programming (CIP)* [Achterberg, 2007, Achterberg et al., 2008a] combines modeling and solving techniques from the fields of constraint programming, mixed-integer linear programming, and *satisfiability testing (SAT)*. In this chapter, we show how nonlinear constraints can be incorporated into a framework for CIPs to obtain a competitive solver for MINLPs. Thereby, the power of already existing MIP and CP technologies is utilized to handle the linear and discrete parts of the problem.

The idea of CIP has been implemented in the branch-cut-and-price framework SCIP (Solving Constraint Integer Programs) [Achterberg, 2007, 2009], which also implements state-of-the-art techniques for MIP solving. Due to its plugin-based design, it can be easily customized and extended, e.g., by adding problem specific separation, presolving, or domain propagation algorithms.

In the following, we formally define CIP and explain the concept of SCIP. Afterwards, we discuss how nonlinear constraints are handled in SCIP. Chapter 8 studies the computational performance of our implementation. Parts of the presentation in this chapter are taken from Berthold, Heinz, and Vigerske [2009b].

## 7.1. Constraint Integer Programming

Before explaining the concept of constraint integer programming, we first have to define constraint programs and satisfiability programs and recall the definition of mixed-integer linear programs. For that purpose, we follow the presentation of Achterberg [2007].

**Definition 7.1** (constraint program). A *constraint program* CP is a triple  $(\mathfrak{C}, \mathfrak{D}, f)$ , where  $\mathfrak{D} = \mathcal{D}_1 \times \dots \times \mathcal{D}_n$  represents the domain of the (finitely many) variables,  $\mathfrak{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\}$  with  $\mathcal{C}_j : \mathfrak{D} \rightarrow \{0, 1\}$ ,  $j \in [m]$ , denotes the (finitely many) constraints, and  $f : \mathfrak{D} \rightarrow \mathbb{R}$  is the objective function. The task is to solve

$$\min\{f(x) : x \in \mathfrak{D}, \mathfrak{C}(x)\},$$

where  $\mathfrak{C}(x)$  holds if and only if  $\mathcal{C}_j(x) = 1$  for all  $j \in [m]$ .

## 7. A Constraint Integer Programming Approach to MINLP

A CP where all domains  $\mathcal{D}_i$ ,  $i \in [n]$ , are finite is called a *finite domain constraint program*. A CP with constant objective function is also called *constraint satisfaction problem* and consists in finding an  $x \in \mathfrak{D}$  with  $\mathfrak{C}(x)$ , or proving that no such solution exists.

A satisfiability program is a special case of a constraint satisfaction problem where the variable domains are  $\{\mathbf{true}, \mathbf{false}\}$  and the constraints are logical clauses.

**Definition 7.2** (satisfiability program). Let  $\mathfrak{C} = \mathcal{C}_1 \wedge \dots \wedge \mathcal{C}_m$  be a logic formula in conjunctive normal form on Boolean variables  $x_i$ ,  $i \in [n]$ . Each clause  $\mathcal{C}_j = \ell_1^j \vee \dots \vee \ell_{k_j}^j$ ,  $j \in [m]$  is a disjunction of literals. A literal  $\ell \in \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$  is either a variable  $x_i$  or the negation of a variable  $\neg x_i$ . The task of the *satisfiability problem* (SAT) is to either find an assignment  $x^* \in \{\mathbf{true}, \mathbf{false}\}^n$  such that  $\mathfrak{C}$  is satisfied, i.e., each clause  $\mathcal{C}_i$  evaluates to  $\mathbf{true}$ , or to conclude that  $\mathfrak{C}$  is unsatisfiable, i.e., for all  $x \in \{\mathbf{true}, \mathbf{false}\}^n$  at least one  $\mathcal{C}_j$ ,  $j \in [m]$ , evaluates to  $\mathbf{false}$ .

Finally, recall the definition of a mixed-integer linear program from Table 6.1.

**Definition 7.3** (mixed-integer linear program). A *mixed-integer linear program* (MIP) is given by a tuple  $(A, b, c, I)$  with matrix  $A \in \mathbb{R}^{m \times n}$ , vectors  $b \in \mathbb{R}^m$  and  $\underline{x}, \bar{x}, c \in \mathbb{R}^n$ , and a subset  $I \subseteq [n]$ . The task is to solve

$$\min\{\langle c, x \rangle : Ax \leq b, x \in [\underline{x}, \bar{x}], x_I \in \mathbb{Z}^{|I|}\}.$$

We note, that MIP is a special case of CP where the variable domains are given as continuous or discrete subsets of  $\mathbb{R}$  and the constraints and objective function are specified via linear functions. Further, SAT is a special case of MIP as can be seen when identifying the values  $\mathbf{false}$  and  $\mathbf{true}$  with 0 and 1 and a clause  $\mathcal{C}_j = \ell_1^j \vee \dots \vee \ell_{k_j}^j$  with the linear constraint  $\ell_1^j + \dots + \ell_{k_j}^j \geq 1$ , where the negation  $\neg x_i$  is replaced by  $(1 - x_i)$ . SAT was the first problem shown to be  $\mathcal{NP}$ -complete [Cook, 1971]. Thus, it follows that also problems like CP, MIP, and MINLP are  $\mathcal{NP}$ -complete.

A common methodology for solving CP with finite domain, SAT, and MIP is to recursively split the problem into smaller subproblems, thereby creating a search tree and implicitly enumerating all potential solutions. However, algorithms differ in the way how subproblems are processed.

For MIP, as a very specific case of CP, algorithms that operate on the subproblem as a whole can be applied. In particular, the optimal value on a MIP can be bounded by the linear programming relaxation that is obtained by dropping the integrality requirement on  $x_I$ . The bound can be used to prune subproblems from the search tree by proving that they contain no feasible solution with objective function value better than the best solution found so far, see also Section 6.1.

Also for SAT, specific techniques have been developed<sup>1</sup>, see Section 1.2 in Achterberg [2007] and the references therein: Boolean constraint propagation that analyzes subsets of

<sup>1</sup>The stated reformulation of a SAT as a MIP is useless from a computational point of view, since the LP relaxation of a clause with at least two unfixed literals can always be satisfied. Hence, for SAT solving, other techniques than reformulating as MIP have been developed.

clauses and the variable fixations in a subproblem to deduce fixations of further variables; analysis of infeasible subproblems to produce additional clauses (called conflict clauses) that can help to recognize infeasibility of other subproblems earlier during the remaining search; periodic restarts of the search to revise initial branching decisions after having learned information about the problem instance in form of conflict clauses.

Due to the generality of constraint programming, CP solvers often cannot take a global perspective the way MIP and SAT solvers can. Instead, they have to rely on efficient domain propagators for each constraint class. The task of domain propagation is to infer reductions on the variables' domains from the current domains of the variables and the definition of the constraint (primal reductions) or from the objective function and a known feasible solution (dual reductions)<sup>2</sup>. Thus, communication between individual constraint classes usually takes place only via the variables' domains.

The strength of CP is the strong modeling potential. While a MIP formulation of a constraint may require a large set of linear constraints and additional variables, in CP very expressive constraints that contain a lot of structure can be used. The latter can often be exploited directly by the domain propagation routines. The concept of constraint integer programming aims at restricting the generality of CP modeling as little as needed while still retaining the full performance of MIP and SAT solving techniques. This paradigm allows to address a wide range of optimization problems.

**Definition 7.4** (constraint integer program). A *constraint integer program* (CIP) is a tuple  $(\mathfrak{C}, I, c)$  that encodes the task of solving

$$\min\{\langle c, x \rangle : \mathfrak{C}(x), x \in \mathbb{R}^n, x_I \in \mathbb{Z}^{|I|}\},$$

where  $c \in \mathbb{R}^n$  is the objective function vector,  $\mathfrak{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\}$  specifies the constraints  $\mathcal{C}_j : \mathbb{R}^n \rightarrow \{0, 1\}$ ,  $j \in [m]$ , and  $I \subseteq [n]$  specifies the set of variables that have to take integral values. Further, a CIP has to fulfill the condition<sup>3</sup>

$$\forall \hat{x}_I \in \mathbb{Z}^{|I|} \exists (A', b') \in \mathbb{R}^{k \times |C|} \times \mathbb{R}^k : \text{Pr}_{x_C} \{x \in \mathbb{R} : \mathfrak{C}(x), x_I = \hat{x}_I\} = \{y \in \mathbb{R}^{|C|} : A'y \leq b'\}, \quad (7.1)$$

where  $C := [n] \setminus I$  and  $k \in \mathbb{N}$ .

Condition (7.1) states that the problem becomes a linear program when all integer variables are fixed. Thus, if the discrete variables are bounded, a CIP can be solved, in principle, by enumerating all values of the integral variables and solving the corresponding LPs. In Proposition 1.7 of Achterberg [2007] it is shown that CIP includes MIP and CP over finite domains as special cases<sup>4</sup>.

<sup>2</sup>Domain propagation for domains that are given as intervals in  $\mathbb{R}$  and that reduce the interval bounds are also called bound tightening, see also Section 6.1.5.

<sup>3</sup> $\text{Pr}_{x_J} S := \{x_J \in \mathbb{R}^{|J|} : \exists \hat{x} \in S : \hat{x}_J = x_J\}$  denotes the  $x_J$ -components of the points in  $S$ .

<sup>4</sup>That MIP is a special case of CIP is easily seen. That finite domain CP is a special case of CIP is seen by recognizing that finite domains can equivalently be represented by integers (thus  $I = [n]$ ), the restrictions  $x_i \in \mathcal{D}_i$ ,  $i \in [n]$ , can be represented as additional constraints, and a general objective function  $f(x)$  can be replaced by an auxiliary variable  $z$  and a constraint  $z = f(x)$ .

## 7. A Constraint Integer Programming Approach to MINLP

While MINLP is a special case of CP, it is in general not a special case of CIP, since the nonlinear constraint  $g(x) \leq 0$  may forbid a linear representation of the MINLP after fixing the integer variables, i.e., (7.1) would be violated (unless  $I = [n]$ ). However, the main purpose of condition (7.1) is to ensure that the problem that remains after fixing all integer variables in the CIP is efficiently solvable. For practical applications, a spatial branch-and-bound algorithm that can solve the remaining NLP up to a given precision within finite time, c.f. Theorem 6.7, is sufficient. Fortunately, the framework SCIP, as discussed in more detail in the next section, does not necessarily require the remaining problem to be a linear one, but only that an algorithm for solving it is available.

### 7.2. The CIP Framework SCIP

SCIP is a framework for constraint integer programs that has originally been developed by Tobias Achterberg [Achterberg, 2007, 2009] since 2002 and is now continuously extended by a group of researches centered at Zuse Institute Berlin (ZIB). SCIP is the successor of the MIP solver SIP developed by Martin [1999], from which it adopts several ideas and algorithms. However, SCIP offers a much more flexible design that allows to support constraint integer programming techniques and a variety of user plugins.

SCIP 2.1.0, released at Halloween 2011, consists of more than 400,000 lines of C code (25% of it serving documentary purposes), can read and/or write 11 different input formats, interfaces 7 external LP solvers, and provides more than 1,000 parameters. SCIP is available free for academic use in source code and binary form<sup>5</sup>.

SCIP solves CIPs by a branch-and-bound algorithm. At each subproblem, domain propagation is performed to exclude further values from the variables' domains, and a relaxation may be solved to obtain a local lower bound. The relaxation may be strengthened by adding further valid constraints (e.g., linear inequalities), which cut off the optimal solution of the relaxation. In case a subproblem is found to be infeasible, conflict analysis is performed to learn additional valid constraints (conflict clauses). Primal heuristics are used as supplementary methods to improve the upper bound. Figure 7.1 illustrates the main algorithmic components of SCIP. In the context of this chapter, the relaxation employed in SCIP is a linear program.

Flexibility of the framework SCIP is guaranteed by its plugin-oriented design. While the SCIP core provides the basic infrastructure (search tree, LP relaxation, bound changes, ...), the actual algorithms that control the search are implemented as external plugins. SCIP merely organizes the order in which they are called. The "outsourced" algorithms communicate with the core via a well-defined interface, which avoids unclear and error-prone interactions between single plugins (there are exceptions, of course) and allows for easy customization for a specific application.

The most important plugin type is the *constraint handler*. It defines the semantics and the algorithms to process constraints of a certain class (e.g., linear constraints  $\ell \leq \langle a, x \rangle \leq u$ ). A single constraint handler is responsible for all the constraints belonging to its constraint class. Each constraint handler has to implement an enforcement

---

<sup>5</sup><http://scip.zib.de>

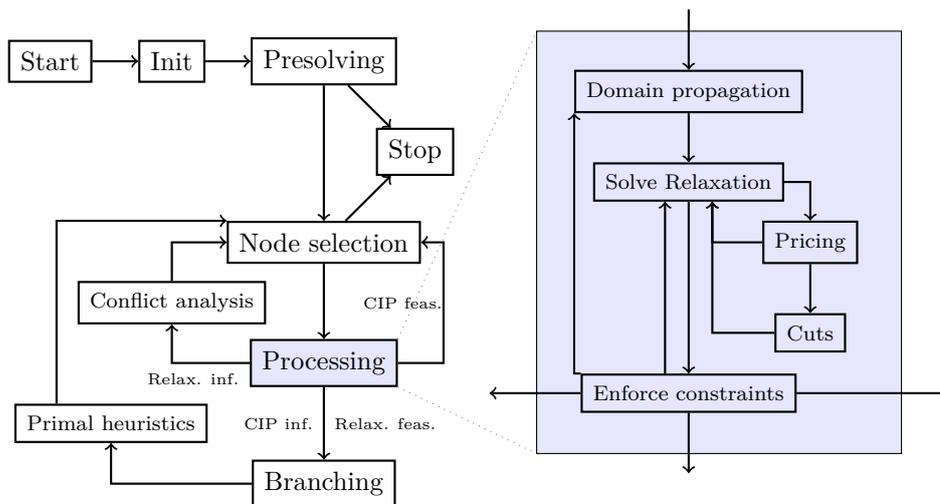


Figure 7.1.: Flowchart of the main solving loop of SCIP.

method. In enforcement, the handler has to decide whether a given solution, e.g., the optimum of the LP relaxation<sup>6</sup>, satisfies all of its constraints. If the solution violates one or more constraints, the handler may resolve the infeasibility by adding another constraint, performing a domain reduction, or a branching. For speeding up computation, a constraint handler may further implement additional features like presolving, cutting plane separation, and domain propagation for its particular class of constraints. Besides that, a constraint handler can add valid linear rows to the initial LP relaxation and nonlinear rows to the initial NLP relaxation. For example, all constraint handlers for (general or specialized) linear constraints add their constraints to the initial LP relaxation and all MINLP constraint handlers for (general or specialized) nonlinear constraints add their constraints to the initial NLP relaxation.

Other plugins types are (see Achterberg [2007] for a thorough description)

- *presolvers* to deduce problem reductions that are not implemented in the presolving algorithms of a constraint handler (e.g., fixing unrestricted variables by taking the objective function into account),
- *cut separators* to strengthen the LP relaxation by additional constraints that cut off fractional solutions (e.g., Gomory, flowcover, or mixed-integer rounding cuts),
- *domain propagators* to derive bound tightenings that are not seen by the propagation algorithms of the constraint handlers (e.g., propagation of the objective function and a known upper bound or reduced cost bound tightening, see also Section 6.1.5),
- *variable pricers* to dynamically create variables in a branch-cut-and-price fashion (e.g., corresponding to paths in a graph),

<sup>6</sup>We assume here that the relaxation is bounded. In the implementation, the so-called pseudo solution, see Achterberg [2007] for details, is used in the case of an unbounded LP relaxation.

## 7. A Constraint Integer Programming Approach to MINLP

- *branching rules* to select a variable for branching from a list of candidates (e.g., strong branching or reliability branching, see also Section 6.1.4),
- *node selectors* to select the next node to process in the tree search (e.g., to follow strategies like depth-first-search or best-first-search),
- *primal heuristics* to find feasible solutions early in the tree search (e.g., by rounding, diving, or large neighborhood search, see also Section 6.1.7),
- *relaxation handlers* to compute bounds from relaxations other than the default LP relaxation (e.g., NLP, SDP, or Lagrange relaxations),
- *event handlers* to inform plugins about events that occurred during the solving process (e.g., bound tightenings, variable additions, new solutions),
- *conflict handlers* to formulate the information learned from infeasible subproblems (conflict clauses) as new constraints (e.g., as disjunction of variable bounds),
- *NLP solver interfaces* to solve NLP subproblems (e.g., to IPOPT),
- *file readers* to parse an input file and to create (or augment) a problem instance or to write a given CIP in a specific format (e.g., reading and writing of MIQCPs in MPS or CPLEX' LP format, reading and parsing SCIP's own CIP file format),
- *display columns* to print solution progress information in a clear and structured way (e.g., solving time, number of processed nodes, dual and primal bound),
- *dialog handlers* to extend SCIP's interactive command line interface (e.g., by dialogs to modify parameter settings),
- *message handlers* to redirect SCIP's output (e.g., to a file).

Figure 7.2 visualizes the design of SCIP and the plugins available with SCIP 2.1.0.

In its beginning, SCIP contained a library of plugins necessary for MIP solving [Achterberg, 2007, Berthold, 2006, Wolter, 2006]. The first CP-like application was the verification of chip designs [Achterberg, 2007, Achterberg, Brinkmann, and Wedler, 2007]. Next to improvements in SCIP's MIP solving capabilities [Achterberg and Berthold, 2007, Berthold, 2007, Berthold and Pfetsch, 2009, Achterberg, Berthold, and Hendel, 2012], SCIP has been extended to count solutions [Achterberg, Heinz, and Koch, 2008b] and to handle pseudo-boolean optimization problems [Berthold, Heinz, and Pfetsch, 2009a], scheduling problems [Berthold, Heinz, Lübbecke, Möhring, and Schulz, 2010b, Heinz and Schulz, 2011, Heinz and Beck, 2011], MIQCPs [Berthold, Heinz, and Vigerske, 2009b, Berthold and Gleixner, 2009], and MINLPs, the latter two being the topic of this chapter. Further, extensions for solving MIPs exactly (i.e., without numerical inaccuracies) [Cook, Koch, Steffy, and Wolter, 2011], applying generic column generation to a MIP [Gamrath, 2010, Gamrath and Lübbecke, 2010], and parallelizing the tree search for a CIP [Shinano, Achterberg, Berthold, Heinz, and Koch, 2012] are in development.

To handle MINLPs, SCIP has been extended by



## 7. A Constraint Integer Programming Approach to MINLP

- constraint handlers for *general nonlinear constraints*, *quadratic constraints*, *second-order-cone constraints*, and *signed power constraints*,
- data structures and algorithms to work with *expression trees* and *expression graphs*,
- data structures and interfaces to branch on non-fractional (e.g., nonlinear) variables,
- an extended interval arithmetic for domain propagation,
- heuristics that solve sub-NLPs, sub-MIPs, and sub-MINLPs of a MINLP,
- a central NLP similar to (but not yet as far developed as) the LP relaxation in SCIP,
- a new plugin type for interfaces to NLP solvers and an interface to IPOPT [Wächter and Biegler, 2006], and
- possibilities to read and write a quadratic objective and quadratic constraints in MPS and CPLEX's LP format, to read polynomial constraints in ZIMPL files [Koch, 2004], to read and write polynomially constrained mixed-integer programs in PIP format<sup>7</sup>, and to write MINLPs in GAMS format.

The data structures for expressions trees and general NLP solver interfaces were designed together with Thorsten Gellermann. The infrastructure to collect branching candidate lists of non-fractional solutions was implemented by Gerald Gamrath and Timo Berthold. Interval arithmetic for addition, subtraction, and multiplication was implemented by Tobias Achterberg and Kati Wolter. MINLP heuristics other than the sub-NLP heuristics are due to Timo Berthold and Ambros Gleixner. Writing of MINLPs in GAMS and PIP format is due to Ambros Gleixner and Marc Pfetsch, respectively.

In the following sections, the algorithms in these extensions are described in more detail. We note, that no modifications on already existing plugins were necessary. Thus, when SCIP solves a MINLP, the already existing MIP and CP technologies are fully utilized for handling the linear and the discrete parts of the problem<sup>8</sup>.

The new MINLP facilities of SCIP are already used by other researchers<sup>9</sup> for the topology optimization of gas networks [Fügenschuh et al., 2010, Pfetsch et al., 2012], the operative planning of water supply networks [Gleixner et al., 2012, Huang, 2011], and the improved convex underestimation of bivariate constraints [Ballerstein et al., 2013].

---

<sup>7</sup><http://polip.zib.de>

<sup>8</sup>We note, that in difference to a pure MIP solver, the generality of CIP imposes certain restrictions on the MIP techniques that can be implemented in SCIP. These restrictions apply to techniques that require knowledge of the whole problem, which cannot be offered by a constraint-based system like SCIP. However, a partial view is available due to certain global data structures and it can often be utilized to extend MIP techniques to general CIPs. For example, dual fixing [Achterberg, 2007, Section 10.8] can be implemented based on the variable locks, which specify the number of constraints that may become violated if the value of a variable is increased or decreased.

<sup>9</sup>Excitement about new features was also expressed on the SCIP mailing list: “I use Visual Studio 2010 on Windows 7 and have all Scip files sitting in different projects. The pub\_expression.h files then cause linkage errors. I don't want to use the nlp\_i at all. Isn't there a way to get rid of it?”

### 7.3. The Expression Graph

The most general form of MINLPs handled by SCIP are those that can be written in factorable form (cf. Definition 6.4) with a special selection for the univariate functions (see  $\Omega_1$  in Definition (7.5) below). Thereby, the nonlinear constraints in SCIP that are handled by the most general MINLP constraint handler (`cons_nonlinear`) have the form

$$\ell \leq \langle a, x \rangle + h(x) \leq u \quad (7.2)$$

with  $\ell, u \in \bar{\mathbb{R}}$ ,  $a \in \mathbb{R}^n$ , and  $h : \mathbb{R}^n \rightarrow \mathbb{R}$ . All functions  $h(x)$  that occur in nonlinear constraints are stored in a single so-called *expression graph*, which is a directed acyclic graph that has variables and constants as sources and the functions  $h(x)$  as sinks, see also Schichl and Neumaier [2005] and Vu et al. [2009].

**Definition 7.5** (expression graph, expression tree). An expression graph  $G = (V, E, o)$  is a directed acyclic graph with a finite set of vertices  $V$ , a finite set of edges  $E \subseteq V \times V$ , and an operator mapping  $o : V \rightarrow \bigcup_{m \geq 0} (V^m \times \Omega_m)$ , where  $V^m$  are  $m$ -tuples of vertices and  $\Omega_m$  are  $m$ -variate functions given by

$$\begin{aligned} \Omega_0 &:= \mathbb{R} \cup \{x_1, \dots, x_n\}, \\ \Omega_1 &:= \left\{ y \mapsto \sum_{j=1}^k \alpha_j y^{\beta_j} : \alpha_j, \beta_j \in \mathbb{R}, j \in [k], k \in \mathbb{N} \right\} \cup \\ &\quad \{y \mapsto \text{sign}(y)|y|^a : a \in \mathbb{R}_{>1}\} \cup \{y \mapsto \exp(y)\} \cup \{y \mapsto \log(y)\} \cup \{y \mapsto |y|\}, \\ \Omega_m &:= \left\{ y \mapsto \sum_{j=1}^k \alpha_j y_1^{\beta_{j,1}} \cdots y_m^{\beta_{j,m}} : \alpha_j \in \mathbb{R}, \beta_j \in \mathbb{R}^m, j \in [k], k \in \mathbb{N} \right\} \quad (m \geq 2). \end{aligned}$$

In  $\Omega_0$ , we identify with  $\mathbb{R}$  the set of constant functions and with  $x_i$ ,  $i \in [n]$ , a parametrized constant function<sup>10</sup>. We require that for every  $f \in \Omega_0$ , there is at most one vertex  $v \in V$  with  $o(v) = (\emptyset; f)$ .

For a vertex  $v \in V$ , we denote by  $c(v) := \{w \in V : (w, v) \in E\}$  the *children* of  $v$  and by  $p(v) := \{w \in V : (v, w) \in E\}$  the *parents* of  $v$ . Vertices  $v \in V$  with  $c(v) = \emptyset$  are called *sources*. Vertices  $v \in V$  with  $p(v) = \emptyset$  are called *sinks*.

For the mapping  $o(\cdot)$ , we require that for  $v \in V$  with  $o(v) = (w_1, \dots, w_m; f)$ , the vertex list  $(w_1, \dots, w_m)$  specifies an order of the children of  $v$  and the function  $f$  is  $m$ -variate, formally  $\{w_1, \dots, w_m\} = c(v)$  and  $f \in \Omega_m$  for  $m = |c(v)|$ .

Further, we define the *depth*  $d(v) \in \mathbb{N}$  of a vertex  $v \in V$  recursively as

$$d(v) := \begin{cases} 0, & \text{if } c(v) = \emptyset, \\ \max\{d(w) : w \in c(v)\} + 1, & \text{otherwise.} \end{cases}$$

An expression graph with exactly one sink and no vertices with more than one parent is called *expression tree*.

<sup>10</sup>As the name indicates,  $x_i$  stands for the  $i$ -th variable of a MINLP.

## 7. A Constraint Integer Programming Approach to MINLP

The *domains* of the univariate functions in  $\Omega_1$  are given by

$$\begin{aligned} \text{dom} \left( y \in \mathbb{R} \mapsto \sum_{j=1}^k \alpha_j y^{\beta_j} \right) &:= \bigcap_{j=1}^k \begin{cases} \mathbb{R}, & \text{if } \beta_j \in \mathbb{N}, \\ \mathbb{R}_{\neq 0}, & \text{if } \beta_j \in \mathbb{Z}_{<0}, \\ \mathbb{R}_{\geq 0}, & \text{if } \beta_j \in \mathbb{R}_{\geq 0} \setminus \mathbb{N}, \\ \mathbb{R}_{>0}, & \text{if } \beta_j \in \mathbb{R}_{<0} \setminus \mathbb{Z}_{<0} \end{cases}, \\ \text{dom} (y \mapsto \text{sign}(y)|y|^a) &:= \mathbb{R} \quad (a > 1), \\ \text{dom} (y \mapsto \exp(y)) &:= \mathbb{R}, \\ \text{dom} (y \mapsto \log(y)) &:= \mathbb{R}_{>0}, \\ \text{dom} (y \mapsto |y|) &:= \mathbb{R}, \end{aligned}$$

The domains of the multivariate signomial functions in  $\Omega_m$ ,  $m \geq 2$ , are given by

$$\begin{aligned} \text{dom} \left( y \mapsto \sum_{j=1}^k \alpha_j y_1^{\beta_{j,1}} \cdots y_m^{\beta_{j,m}} \right) &:= \\ &\text{dom} \left( y_1 \mapsto \sum_{j=1}^k \alpha_j y_1^{\beta_{j,1}} \right) \times \cdots \times \text{dom} \left( y_m \mapsto \sum_{j=1}^k \alpha_j y_m^{\beta_{j,m}} \right). \end{aligned}$$

Each vertex  $v$  of the expression graph is associated with an algebraic expression due to the function given by  $o(v)$  and the expressions associated with the children of  $v$ . The first argument of  $o(v)$  (the one in  $V^m$ ) specifies an order for the children, while the second argument (the one in  $\Omega_m$ ) specifies the operation to be applied to the children's expressions. Thus, given values  $\hat{x}$  for the parameters  $\{x_1, \dots, x_n\}$ , we can evaluate the expression graph by assigning values to its vertices with increasing depth.

**Definition 7.6** (expression graph evaluation). Let  $G = (V, E, o)$  be an expression graph and  $\hat{x} \in \mathbb{R}^n$ . With each vertex  $v \in V$ , we associate a value  $v(\hat{x}) \in \mathbb{R} \cup \{\text{DOMERR}\}$  as follows. Let  $o(v) = (w_1, \dots, w_m; f)$ . Then

$$v(\hat{x}) := \begin{cases} f, & \text{if } f \in \mathbb{R}, \\ \hat{x}_i, & \text{if } f = x_i, \\ \text{DOMERR}, & \text{if } \exists j \in [m] : w_j(\hat{x}) = \text{DOMERR}, \\ \text{DOMERR}, & \text{if } (w_1(\hat{x}), \dots, w_m(\hat{x})) \notin \text{dom } f, \\ f(w_1(\hat{x}), \dots, w_m(\hat{x})), & \text{otherwise.} \end{cases}$$

In SCIP, an expression graph is used to store all nonlinear functions that occur in general nonlinear constraints (those that are handled by the constraint handler `cons_nonlinear`). It is ensured, that for each variable  $x_i$  at most one vertex  $v$  with  $o(v) = x_i$  exists in the graph. For each depth  $d \in \{0, \dots, d_{\max}\}$ ,  $d_{\max} = \max\{d(v) : v \in V\}$ , the vertices at depth  $d$  are stored in a set  $V_d$ . For each vertex  $v$ , we store the list of children  $c(v)$  and the list of parents  $p(v)$ . This design allows easy traversing of the graph.

An expression tree is used in SCIP to store single nonlinear functions, e.g., when specifying a nonlinear constraint. When an expression tree is added to an existing expression graph, then the nodes of the tree are added with increasing depth to the graph, whereby SCIP tries to omit new vertices for subexpressions that are already stored in the graph. That is, given vertices  $(w_1, \dots, w_m) \in V^m$  and a function  $f \in \Omega_m$ , a new vertex for  $f(w_1, \dots, w_m)$  is only added if there exists no  $v \in V$  with  $o(v) = (w_1, \dots, w_m; f)$ , which is easily<sup>11</sup> checked by inspecting common parents of  $w_1, \dots, w_m$ .

For expression trees, SCIP also provides functionality to compute first and second derivatives of the associated function. Their computation is delegated to the *expression interpreter* plugin, which computes function value, gradient, Hessian, and sparsity pattern of the Hessian<sup>12</sup> for a function given in form of an expression tree. So far, only one expression interpreter has been implemented, which uses the algorithmic differentiation (also referred to as *automatic differentiation* [Griewank and Walther, 2008]) code CPPAD<sup>13</sup>.

### 7.3.1. Simplification

The expression graph implementation in SCIP includes methods to shrink a given expression graph by applying some simple transformations. These transformations include the replacement of vertices which have only constants as children,

$$\left\{ \begin{array}{l} o(v) = (w_1, \dots, w_m; f) \\ o(w_j) = (\emptyset; a_j), a_j \in \mathbb{R}, j \in [m] \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} E \leftarrow E \setminus \{(w_j, v) : j \in [m]\} \\ o(v) \leftarrow (\emptyset; f(a_1, \dots, a_m)) \end{array} \right\}$$

and the substitution of arguments  $y_i$  with positive integral exponent ( $\beta_{j,i} \in \mathbb{N}$ ) in a signomial function by a signomial function in the child vertex  $w_i$ , for  $i = m$  this is

$$\left\{ \begin{array}{l} o(v) = \left( w_1, \dots, w_m; y \mapsto \sum_{j=1}^k \alpha_j y_1^{\beta_{j,1}} \dots y_m^{\beta_{j,m}} \right), \\ \beta_{j,m} \in \mathbb{N}, j \in [k], \\ o(w_m) = \left( w'_1, \dots, w'_{m'}; y \mapsto \sum_{j=1}^{k'} \alpha'_j y_1^{\beta'_{j,1}} \dots y_{m'}^{\beta'_{j,m'}} \right) \end{array} \right\} \\ \Rightarrow \left\{ \begin{array}{l} E \leftarrow E \setminus \{(w_m, v)\} \\ o(v) \leftarrow \left( (w_1, \dots, w_{m-1}, w'_1, \dots, w'_{m'}), \right. \\ \left. y \mapsto \sum_{j=1}^k \alpha_j y_1^{\beta_{j,1}} \dots y_{m-1}^{\beta_{j,m-1}} \left( \sum_{j'=1}^{k'} \alpha'_{j'} y_m^{\beta'_{j',1}} \dots y_{m+m'-1}^{\beta'_{j',m'}} \right)^{\beta_{j,m}} \right) \end{array} \right\}.$$

<sup>11</sup>Whether two functions  $f, f' \in \Omega_m$  are indeed equivalent is more involved for signomial functions. Here, SCIP sorts the summands  $\alpha_j y_1^{\beta_{j,1}} \dots y_m^{\beta_{j,m}}$  by applying a lexicographical ordering w.r.t. the exponent vectors  $\beta$ . Two summands with equal  $\beta$  are merged into a single one by adding up the coefficients  $\alpha_j$ .

<sup>12</sup>The sparsity pattern of the Hessian  $\nabla^2 h(x)$  of a twice continuously differentiable function  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  is a matrix  $S \in \{0, 1\}^{n \times n}$  such that  $S_{i,j} = 1$  if there exists an  $\hat{x} \in \text{dom } h(\cdot)$  such that  $\frac{\partial^2}{\partial x_i \partial x_j} h(\hat{x}) \neq 0$ .

<sup>13</sup><http://www.coin-or.org/CppAD>

## 7. A Constraint Integer Programming Approach to MINLP

Since convex underestimators are computed for each summand of a nonquadratic signomial function separately, a  $y_i$  in a term  $y_i^{\beta_{j,i}}$  is only substituted by a new signomial if the corresponding expanded signomial function does not have more nonlinear and nonquadratic summands than the original one. The check has been implemented as applying a substitution for  $y_m^{\beta_{j,m}}$  only if  $\beta_{j,1} = 0, \dots, \beta_{j,m-1} = 0, \beta_{j,m} = 1$  or  $\beta_{j,i} \geq 0, \beta'_{j',i} \geq 0, i \in [m], j' \in [k']$ , and  $\beta_{j,1} + \dots + \beta_{j,m-1} + \beta_{j,m} \max_{j' \in [k']} (\beta'_{j',1} + \dots + \beta'_{j',m'}) \leq 2$ .

Other reformulations that use equivalences like  $\log(\prod_i y_i) = \sum_i \log(y_i)$ , see also Schichl and Neumaier [2005] and Liberti et al. [2009], are not implemented, yet.

**Example 7.7.** Consider the constraints

$$\begin{aligned} 420.169\sqrt{900 + x_1^2} - x_3x_1x_2 &= 0 \\ \frac{2960.88 + 296088 \cdot 0.0625x_2^2}{7200 + x_1^2} - x_3 &\geq 0 \\ x_{\text{obj}} - 0.047x_2\sqrt{900 + x_1^2} &\geq 0 \end{aligned}$$

from the instance `nvs01` of MINLPLib [Bussieck et al., 2003]. Initial and simplified expression graphs that represent the three nonlinear functions from these constraints are shown in Figure 7.3. The nonlinear expressions on the left-hand-side of the constraints are identified with the sinks (top vertices) and the variables and constants with the sources (bottom vertices).

### 7.3.2. Bound Tightening

For nonlinear constraints, a constraint-based bound tightening method, see also Section 6.1.5, has been implemented in SCIP. For a nonlinear constraint (7.2), bounds on the variables in the linear term  $\langle a, x \rangle$  can be computed from the constraint sides  $\ell$  and  $u$  and bounds on  $\{h(x) : x \in [\underline{x}, \bar{x}]\}$  with a formula similar to (6.24). The bounds on  $\{h(x) : x \in [\underline{x}, \bar{x}]\}$  are obtained by an *interval evaluation* of the expression graph.

#### Interval Arithmetic

Interval arithmetic extends the arithmetic defined on real numbers to the set of intervals in  $\mathbb{R}$  [Moore, 1966, Neumaier, 1990, Kearfott, 1996, Moore et al., 2009]. Ideally, interval arithmetic for the elementary operations  $\diamond \in \{+, -, *, \div\}$  obey

$$[\underline{x}, \bar{x}] \diamond [\underline{y}, \bar{y}] = \{x \diamond y : x \in [\underline{x}, \bar{x}], y \in [\underline{y}, \bar{y}]\}. \quad (7.3)$$

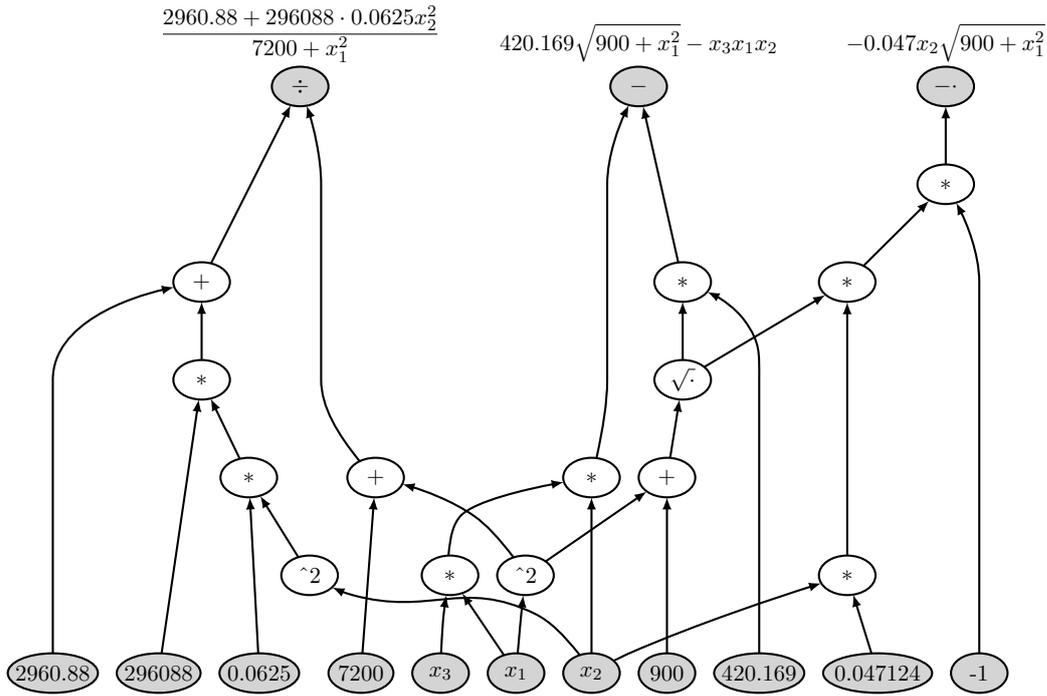
Their usefulness from a practical point of view is due to the property that the resultant interval can be expressed by the bounds of the initial interval. That is,

$$[\underline{x}, \bar{x}] + [\underline{y}, \bar{y}] = [\underline{x} + \underline{y}, \bar{x} + \bar{y}], \quad (7.4a)$$

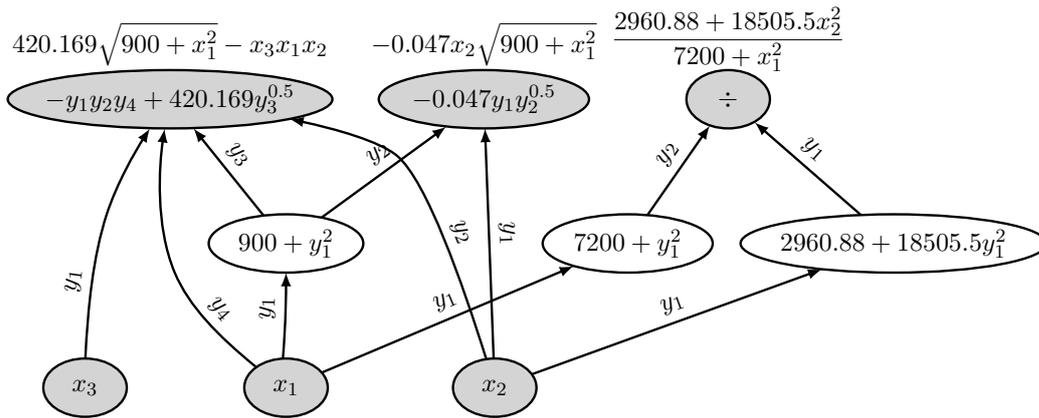
$$[\underline{x}, \bar{x}] - [\underline{y}, \bar{y}] = [\underline{x} - \bar{y}, \bar{x} - \underline{y}], \quad (7.4b)$$

$$[\underline{x}, \bar{x}] * [\underline{y}, \bar{y}] = [\min(\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y}), \max(\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y})], \quad (7.4c)$$

$$[\underline{x}, \bar{x}] \div [\underline{y}, \bar{y}] = [\underline{x}, \bar{x}] * [1/\bar{y}, 1/\underline{y}], \quad \text{if } 0 \notin [\underline{y}, \bar{y}]. \quad (7.4d)$$



(a) Initial expression graph as given to SCIP.



(b) Equivalent simplified expression graph.

Figure 7.3.: Expression graph for three nonlinear functions.

Further, for elementary functions  $\varphi : \mathbb{R} \rightarrow \mathbb{R}$  (like  $\exp$ ,  $\log$ ,  $x^p$ ), interval arithmetic extensions are usually defined by

$$\varphi([\underline{x}, \bar{x}]) := [\inf\{\varphi(x) : x \in [\underline{x}, \bar{x}]\}, \sup\{\varphi(x) : x \in [\underline{x}, \bar{x}]\}] \quad ([\underline{x}, \bar{x}] \subseteq \text{dom } \varphi), \quad (7.5)$$

where the infima and suprema can often be expressed in terms of the interval bounds  $\underline{x}$  and  $\bar{x}$ , too.

## 7. A Constraint Integer Programming Approach to MINLP

Composition of operations and elementary functions allows to compute *bounds on the ranges* of factorable real functions, which makes interval arithmetic a commonly used tool in global optimization [Hansen, 1992, Jaulin et al., 2001, Moore et al., 2009]. Such bounds are usually easy and fast to compute, but are not always best possible due to the *dependency problem* in interval arithmetic: If a variable appears several times in an arithmetic expressions, interval arithmetic based on elementary operations treats each occurrence of the variable independently. Simple examples for this problem are the subtraction of an interval from itself ( $[\underline{x}, \bar{x}] - [\underline{x}, \bar{x}] = [\underline{x} - \bar{x}, \bar{x} - \underline{x}]$ ) or the square of an interval (computing  $[-1, 2]^2 (= [0, 4])$  as  $[-1, 2] * [-1, 2] = [-2, 4]$ ).

While we have explicitly excluded the case  $0 \in [y, \bar{y}]$  in (7.4d), it is also possible to define interval division for nominators that contain 0 by using an *extended interval arithmetic* for unbounded intervals. For elementary operations and functions, also extended interval arithmetic has to satisfy the rules (7.3) and (7.5), but the formulas (7.4) need to be extended to deal with infinite bounds, see, e.g., Vu et al. [2009].

Interval arithmetic is also used often for rigorous computing [Jaulin et al., 2001, Moore et al., 2009, Cook et al., 2011], since outwardly rounding interval arithmetic always computes intervals for the outcome of an arithmetic operation in which the exact result must lie, while floating-point arithmetic gives only approximate results.

SCIP includes an implementation of rounding-safe extended interval arithmetic for elementary operations and functions. These methods are used by the MINLP constraint handlers to compute bounds on nonlinear functions from bounds on the involved variables and to compute bounds on variables from bounds on expressions in which these variables occur. For general nonlinear constraints, these computations are done along the vertices of the expression graph and described in more detail in the following.

### Forward Propagation

Given an expression graph, bounds on the expressions associated with the vertices of the graph can be computed from bounds on the variables analogously to the expression graph evaluation from Definition 7.6 by means of (extended) interval arithmetic. Domain violations are easily taken into account by allowing empty intervals, that is, for a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , we let  $f([\underline{x}, \bar{x}]) = \emptyset$  if  $[\underline{x}, \bar{x}] \cap \text{dom } f = \emptyset$ .

**Definition 7.8** (expression graph interval evaluation). Let  $G = (V, E, o)$  be an expression graph and  $[\underline{x}, \bar{x}] \subseteq \mathbb{R}^n$  be a box (a Cartesian product of intervals). With each  $v \in V$ , we associate an interval  $v([\underline{x}, \bar{x}]) \subseteq \mathbb{R}$  as follows. Let  $o(v) = (w_1, \dots, w_m; f)$ . Then

$$v([\underline{x}, \bar{x}]) := \begin{cases} [f, f], & \text{if } f \in \mathbb{R}, \\ [\underline{x}_i, \bar{x}_i], & \text{if } f = x_i, \\ f(w_1([\underline{x}, \bar{x}]), \dots, w_m([\underline{x}, \bar{x}])), & \text{otherwise,} \end{cases}$$

where  $f(w_1([\underline{x}, \bar{x}]), \dots, w_m([\underline{x}, \bar{x}]))$  denotes an interval that contains the range of the function  $f$  on the box  $w_1([\underline{x}, \bar{x}]) \times \dots \times w_m([\underline{x}, \bar{x}])$ .

Thus, for given bounds on the variables, intervals for all vertices in the expression

graph can be computed with increasing depth. This process is called *forward propagation* of variable bounds [Schichl and Neumaier, 2005, Vu et al., 2009], see also Algorithm 7.1. Bounds on a function  $h(x)$  are then obtained from the interval of the corresponding sink in the expression graph. If forward propagation yields the empty interval, then  $h(x)$  is not defined for any point within the current bounds. During branch-and-bound, one can then prune the corresponding subproblem.

---

**Algorithm 7.1:** Forward Propagation of variable bounds in expression graph
 

---

```

input : expression graph  $G = (V, E, o)$ 
input : box  $[\underline{x}, \bar{x}]$ 
/* assign intervals to sources */
foreach  $v \in V$  with  $c(v) = \emptyset$  do  $[\underline{v}, \bar{v}] \leftarrow \begin{cases} [f, f], & \text{if } o(v) = (\emptyset; f) \text{ with } f \in \mathbb{R}, \\ [\underline{x}_i, \bar{x}_i], & \text{if } o(v) = (\emptyset; x_i) \end{cases}$ 
/* propagate intervals through expression graph */
for  $d = 1$  to  $d_{\max}$  do
  foreach  $v \in V$  with  $d(v) = d$  do
    let  $o(v) = (w_1, \dots, w_m; f);$ 
     $[\underline{v}, \bar{v}] \leftarrow f([\underline{w}_1, \bar{w}_1], \dots, [\underline{w}_m, \bar{w}_m]);$ 
  end
end
output :  $[\underline{v}, \bar{v}], v \in V$ 

```

---

**Example 7.9.** Consider the simplified expression graph from Example 7.7 and let  $[\underline{x}, \bar{x}] = [200, 200] \times [200, 200] \times [0, 100]$ . Figure 7.4 shows the intervals for all vertices as computed by the Forward Propagation Algorithm 7.1.

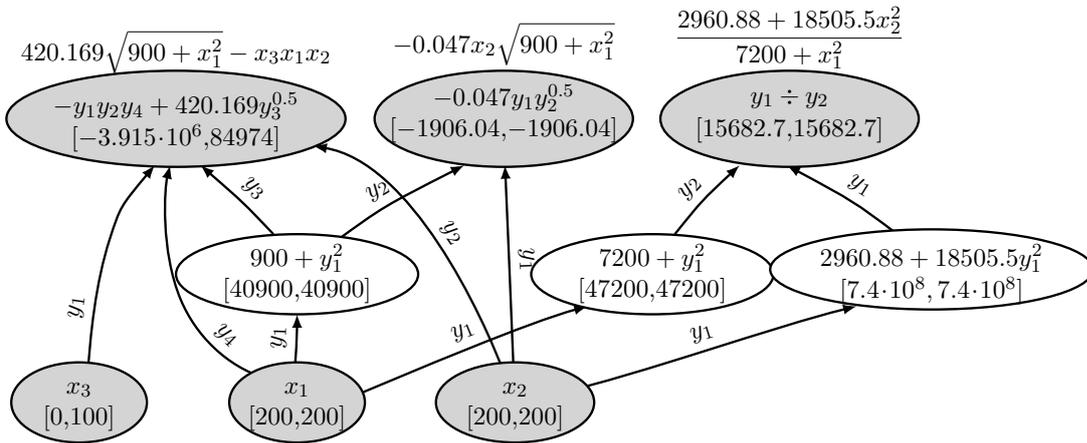


Figure 7.4.: Forward Propagation of variable bounds in expression graph.

### Backward Propagation

The purpose of *backward propagation* is to tighten the intervals for the vertices in an expression graph based on given intervals for the sources and sinks such that no feasible assignment to the sources is lost. By a feasible assignment we mean a vector  $\hat{x}$  such that  $v(\hat{x})$  lies within the given intervals for the sources and sinks. Formally, let  $[\underline{v}, \bar{v}]$  be given intervals for the sources and sinks  $v$  in an expression graph. For a variable  $x_i$ , let  $[\underline{x}_i, \bar{x}_i] = [\underline{v}, \bar{v}]$  for  $v \in V$  with  $o(v) = (\emptyset; x_i)$  be the corresponding interval<sup>14</sup>. Then we seek for a box  $[\underline{x}', \bar{x}'] \subseteq \mathbb{R}^n$  (as small as possible) such that

$$x \in [\underline{x}, \bar{x}], v(x) \in [\underline{v}, \bar{v}] \quad \Rightarrow \quad x \in [\underline{x}', \bar{x}'] \quad (v \in V \text{ with } p(v) = \emptyset).$$

The tightened intervals can be computed by inverting the functions associated with vertices in an expression graph and applying the interval evaluation of an expression graph in a backward manner, thereby propagating intervals at the sinks to the sources. That is, for non-sink vertices  $v \in V$ , let  $[\underline{v}, \bar{v}]$  be the interval computed by forward propagation from the given source intervals. For any non-source  $v \in V$  with  $o(v) = (w_1, \dots, w_m; f)$ , assume that arithmetic functions  $f_i : \mathbb{R} \times \mathbb{R}^{m-1} \rightarrow \mathbb{R}$ ,  $i \in [m]$ , are available such that

$$z = f(y_1, \dots, y_m) \quad \Leftrightarrow \quad y_i = f_i(z; y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_m). \quad (7.6)$$

Then an interval  $[\underline{w}'_i, \bar{w}'_i]$  that contains the set

$$\text{Pr}_{y_i} \left\{ y : f(y_1, \dots, y_m) \in [\underline{v}, \bar{v}], y_j \in [\underline{w}_j, \bar{w}_j], j \in [m] \right\} \quad (7.7)$$

can be computed by an interval arithmetic evaluation of  $f_i$  as

$$[\underline{w}'_i, \bar{w}'_i] := [\underline{w}_i, \bar{w}_i] \cap f_i([\underline{v}, \bar{v}]; [\underline{w}_1, \bar{w}_1], \dots, [\underline{w}_{i-1}, \bar{w}_{i-1}], [\underline{w}_{i+1}, \bar{w}_{i+1}], \dots, [\underline{w}_n, \bar{w}_n]). \quad (7.8)$$

The new interval  $[\underline{w}'_i, \bar{w}'_i]$  can be used to replace  $[\underline{w}_i, \bar{w}_i]$ . If  $[\underline{w}'_i, \bar{w}'_i]$  is empty, then there exists no feasible assignment to the sources with respect to the given intervals for sources and sinks and the corresponding branch-and-bound subproblem can be pruned. See also Vu, Schichl, and Sam-Haroud [2009] for a proof of correctness.

A function  $f_i$  that satisfies (7.6) is often available for simple monotonic functions. In general, any algorithm that computes bounds on (7.7) can be employed. The rules that SCIP applies for backward propagation of the functions in  $\Omega$  are given in Table 7.1. For better readability, we omitted the intersection of the computed intervals with the existing ones and the convexification step for a union of intervals. For univariate and multivariate signomial functions, the formulas in Table 7.1 compute intervals for a variable  $y_i$  that appears with an exponent  $\beta$  by reduction to the simple form  $[\underline{f}, \bar{f}] = [\underline{g}, \bar{g}]y_i^\beta$  and application of the rules for the inversion of a power function. Tighter intervals for univariate and bivariate quadratic functions are discussed in Section 7.4.

The backward propagation algorithm is summarized in Algorithm 7.2.

<sup>14</sup>We assume here, that for every variable there exists a vertex in the graph. By Definition 7.5, there is at most one vertex  $v \in V$  with  $o(v) = (\emptyset; x_i)$ .

Function $f$	Backward Propagation formula
$z = y_1 + y_2$	$[y_1, \bar{y}_1] \leftarrow [z, \bar{z}] - [y_2, \bar{y}_2]$ $[y_2, \bar{y}_2] \leftarrow [z, \bar{z}] - [y_1, \bar{y}_1]$
$z = y_1 - y_2$	$[y_1, \bar{y}_1] \leftarrow [z, \bar{z}] + [y_2, \bar{y}_2]$ $[y_2, \bar{y}_2] \leftarrow [y_1, \bar{y}_1] - [z, \bar{z}]$
$z = y_1 * y_2$	$[y_1, \bar{y}_1] \leftarrow [z, \bar{z}] \div [y_2, \bar{y}_2]$ $[y_2, \bar{y}_2] \leftarrow [z, \bar{z}] \div [y_1, \bar{y}_1]$
$z = y_1 \div y_2$	$[y_1, \bar{y}_1] \leftarrow [z, \bar{z}] * [y_2, \bar{y}_2]$ $[y_2, \bar{y}_2] \leftarrow [y_1, \bar{y}_1] \div [z, \bar{z}]$
$z = \exp(y)$	$[y, \bar{y}] \leftarrow \log([\max(z, 0), \bar{z}])$
$z = \log(y)$	$[y, \bar{y}] \leftarrow \exp([z, \bar{z}])$
$z = y^a, a \in 2\mathbb{Z}$	$[y, \bar{y}] \leftarrow (-[\max(z, 0), \bar{z}]^{1/a} \cup [\max(z, 0), \bar{z}]^{1/a}$
$z = y^a, a \in 2\mathbb{Z} + 1$	$[y, \bar{y}] \leftarrow (-[\max(-\bar{z}, 0), -z]^{1/a} \cup [\max(z, 0), \bar{z}]^{1/a}$
$z = y^a, a \in \mathbb{R} \setminus \mathbb{Z}$	$[y, \bar{y}] \leftarrow [\max(z, 0), \bar{z}]^{1/a}$
$z = \text{sign}(y) y ^a, a > 1$	$[y, \bar{y}] \leftarrow (-[\max(-\bar{z}, 0), -z]^{1/a} \cup [\max(z, 0), \bar{z}]^{1/a}$
$z =  y $	$[y, \bar{y}] \leftarrow [-\bar{z}, -\max(z, 0)] \cup [\max(z, 0), \bar{z}]$
$z = \sum_{j \in [k]} \alpha_j y^{\beta_j}$	get $[\underline{u}_j, \bar{u}_j]$ such that $[z, \bar{z}] - \sum_{\substack{j' \in [k] \\ j' \neq j}} \alpha_{j'} [y, \bar{y}]^{\beta_{j'}} = \alpha_j [\underline{u}_j, \bar{u}_j]^{\beta_j}$ $[y, \bar{y}] \leftarrow \bigcap_{j \in [k]} [\underline{u}_j, \bar{u}_j]$
$z = \sum_{j \in [k]} \alpha_j \prod_{i \in [m]} y_i^{\beta_{j,i}}$	for $i \in [m]$ : for all $b \in \{\beta_{1,i}, \dots, \beta_{k,i}\}$ , get $[\underline{u}_b, \bar{u}_b]$ s.t. $[\underline{u}_b, \bar{u}_b]^b =$ $\left( [z, \bar{z}] - \sum_{\substack{j \in [k] \\ \beta_{j,i} \neq b}} \alpha_j \prod_{i' \in [m]} [y_{i'}, \bar{y}_{i'}]^{\beta_{j,i'}} \right) \div \left( \sum_{\substack{j \in [k] \\ \beta_{j,i} = b}} \alpha_j \prod_{\substack{i' \in [m] \\ i' \neq i}} [y_{i'}, \bar{y}_{i'}]^{\beta_{j,i'}} \right)$ $[y_i, \bar{y}_i] \leftarrow \bigcap_{b \in \{\beta_{1,i}, \dots, \beta_{k,i}\}} [\underline{u}_b, \bar{u}_b]$

Table 7.1.: Backward Propagation for functions in  $\Omega$ . We assume that extended interval arithmetic formulas are available for the elementary operations  $+$ ,  $-$ ,  $*$ ,  $\div$  and the elementary functions  $y \mapsto y^a$  ( $a \in \mathbb{R}$ ),  $\exp(\cdot)$ , and  $\log(\cdot)$ .

**Algorithm 7.2:** Backward Propagation in expression graph

---

```

input : expression graph  $G = (V, E, o)$ 
input : intervals  $[\underline{v}^{\text{orig}}, \bar{v}^{\text{orig}}]$  for sources and sinks  $v$ 
/* initialize intervals in vertices by forward propagation */
foreach  $i \in [n]$  do  $[\underline{x}_i, \bar{x}_i] \leftarrow [\underline{v}^{\text{orig}}, \bar{v}^{\text{orig}}]$  for  $v \in V$  with  $o(v) = (\emptyset; x_i)$ ;
let  $[\underline{v}, \bar{v}]$ ,  $v \in V$ , be the intervals computed by Algorithm 7.1 w.r.t.  $[\underline{x}, \bar{x}]$ ;
/* tighten intervals in sinks w.r.t. given bounds */
foreach  $v \in V$  with  $p(v) = \emptyset$  do
     $[\underline{v}, \bar{v}] \leftarrow [\underline{v}, \bar{v}] \cap [\underline{v}^{\text{orig}}, \bar{v}^{\text{orig}}]$ ;
    if  $[\underline{v}, \bar{v}] = \emptyset$  then STOP: no feasible assignment exists
end
/* propagate intervals in sinks backward through expression graph */
for  $d = d_{\max}$  to 1 do
    foreach  $v \in V$  with  $d(v) = d$  do
        let  $o(v) = (w_1, \dots, w_m; f)$ ;
        for  $i = 1$  to  $m$  do
            compute interval  $[\underline{w}'_i, \bar{w}'_i]$  that contains (7.7), see Table 7.1;
            if  $[\underline{w}'_i, \bar{w}'_i] = \emptyset$  then STOP: no feasible assignment exists;
             $[\underline{w}_i, \bar{w}_i] \leftarrow [\underline{w}'_i, \bar{w}'_i]$ ;
        end
    end
end
output :  $[\underline{v}, \bar{v}]$  for  $v \in V$  with  $o(v) = (\emptyset; x_i)$ 

```

---

**Constraint-Based Domain Propagation**

Backward propagation enables SCIP to compute bounds on variables that occur in nonlinear constraints. Let  $[\underline{x}, \bar{x}]$  be the variable bounds in a branch-and-bound node and

$$\ell_j \leq \langle a^j, x \rangle + h_j(x) \leq u_j \quad (j \in [m])$$

be a set of constraints  $(\ell_j, u_j \in \bar{\mathbb{R}}, a^j \in \mathbb{R}^n)$ , where each function  $h_j(x)$  is associated with a sink in an expression graph  $G = (V, E, o)$ . SCIP first applies the Forward Propagation Algorithm 7.1 to obtain an interval for  $h_j([\underline{x}, \bar{x}])$ . If  $h_j([\underline{x}, \bar{x}]) = \emptyset$  or  $\langle a^j, [\underline{x}, \bar{x}] \rangle + h_j([\underline{x}, \bar{x}]) \cap [\ell_j, u_j] = \emptyset$  for some  $j \in [m]$ , then infeasibility of the current subproblem is recognized (in the former case, because  $[\underline{x}, \bar{x}] \cap \text{dom } h_j = \emptyset$ ; in the latter case, because the  $j$ -th constraint cannot be satisfied on  $[\underline{x}, \bar{x}]$ ). Otherwise, SCIP applies bound tightening to the variables  $x_i$  with  $a_i^j \neq 0$  by a formula like (6.24). If no infeasibility is recognized, the Backward Propagation Algorithm 7.2 is called with the current bounds  $[\underline{x}, \bar{x}]$  for the sources and  $[\ell_j, u_j] - \langle a^j, [\underline{x}, \bar{x}] \rangle$  for the sink that is associated with  $h_j(x)$ .

Forward and Backward Propagation may be repeated as long as bound tightenings are found. However, to avoid an endless loop with very little improvements in the bounds, interval tightenings are only propagated if the amount of tightening, relative to the interval

length, is above a certain threshold, see also Section 7.1 in Achterberg [2007]. Further, to avoid recomputing intervals for all vertices in an expression graph in each iteration of the bound tightening algorithm, a flag in each vertex of the expression graph indicates whether the currently stored interval has been tightened by backward propagation and whether the interval of a child vertex has recently been relaxed or tightened otherwise, e.g., due to traversal of the branch-and-bound tree. Only in these cases, forward propagation needs to refresh the interval associated with the vertex. Similarly, backward propagation marks the vertices for which it found an interval tightening (either by propagation from a parent node or due to the given intervals for a sink). Subsequently, unmarked vertices do not need to be processed, since no interval tightening for a child can be expected.

**Example 7.10.** Consider the simplified expression graph from Example 7.7 and the intervals computed by Forward Propagation for  $[x, \bar{x}] = [200, 200] \times [200, 200] \times [0, 100]$  (Example 7.9, Figure 7.4). Note, that the left-most sink corresponds to the nonlinear function in the constraint

$$420.169\sqrt{900 + x_1^2} - x_3x_1x_2 = 0.$$

Thus, the interval for this sink can be tightened to  $[0, 0]$ . Backward Propagation of the tightened interval in the sink then yields the fixation of variable  $x_3$  to 2.12435, see Figure 7.5.

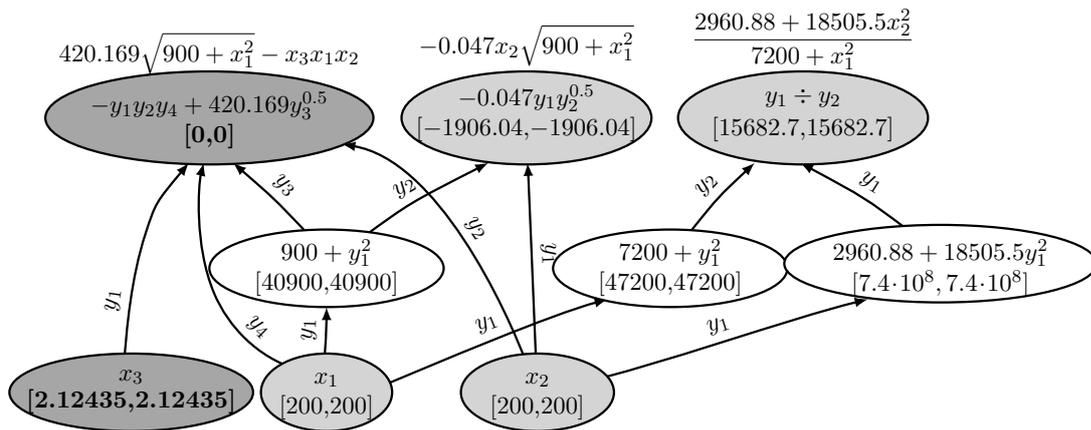


Figure 7.5.: Backward Propagation in expression graph.

### 7.3.3. Convexity Detection

Recall, that for the construction of a convex relaxation of a factorable MINLP, the problem may be reformulated into the standard form (6.16) by introducing additional variables with the purpose of constructing a convex relaxation (6.17) via known convex underestimators for elementary univariate functions and McCormick’s convex hull description for the products of two variables. However, if a function  $g_j(\cdot)$  in a constraint  $g_j(x) \leq 0$  in the original formulation is already known to be convex (w.r.t., e.g.,  $[x, \bar{x}]$ ), then no reformulation into univariate functions is necessary. Similar, if  $g_j(\cdot)$  is concave, then its convex envelope is described by (6.9) and thus reformulating  $g_j(\cdot)$  can be omitted.

## 7. A Constraint Integer Programming Approach to MINLP

Existing deterministic methods for proving or disproving the convexity of a function (given as composition of elementary expressions) with respect to bounds on its variables are based on walking an expression tree and applying convexity rules for function compositions [Bao, 2007, Fourer et al., 2009], estimating the spectra of the Hessian matrix or its sign in case of a univariate function [Mönnigmann, 2008, Nenov et al., 2004], or deciding positive-semidefiniteness of the interval Hessian [Nenov et al., 2004]. All these approaches may give inconclusive results, because they either take only a “local look” at the function (when walking the tree), or compute overestimates of Hessian matrices<sup>15</sup>. A novel symbolic method that proves or disproves convexity of *rational functions* over polyhedral sets has been suggested by Neun, Sturm, and Vigerske [2010]. Here, the idea is to reduce the problem of deciding convexity to a real quantifier elimination problem and to apply methods from computer logic to decide these problems. This method, implemented via the REDLOG package [Dolzmann and Sturm, 1997] of the computer algebra system REDUCE [Hearn, 1967, 2005], has been interfaced to SCIP experimentally, but is not included in the publicly available version of SCIP which has been used for the computational study in Chapter 8.

Nevertheless, SCIP implements a method that recognizes evidently convex/concave expressions in an expression graph by applying simple rules as in [Bao, 2007, Fourer et al., 2009]. Consider a composition  $f(g(x))$  of twice-differentiable functions  $f : \mathbb{R} \rightarrow \mathbb{R}$  and  $g : \mathbb{R}^n \rightarrow \mathbb{R}$ . Then

$$(f \circ g)''(x) = f''(g(x))\nabla g(x)(\nabla g(x))^\top + f'(g(x))\nabla^2 g(x). \quad (7.9)$$

Thus, for convexity of  $f(g(x))$ , it is sufficient that  $f(x)$  is convex and monotonically increasing on  $[g, \bar{g}]$  and  $g(x)$  is convex on  $[x, \bar{x}]$ , where we denote by  $[g, \bar{g}]$  an interval that contains  $\{g(x) : x \in [x, \bar{x}]\}$ . Analogously, if  $f(x)$  is concave and monotonically increasing on  $[g, \bar{g}]$  and  $g(x)$  is concave on  $[x, \bar{x}]$ , then  $f(g(x))$  is concave on  $[x, \bar{x}]$ . Similar conclusions can be drawn if  $f(x)$  is monotonically decreasing.

These observations allow to propagate convexity and concavity properties from the sources to the sinks in an expression graph. With every vertex, we associate two flags

<sup>15</sup> For example, the method from Fourer et al. [2009] – even though very fast – may fail to detect convexity of the function  $f(x) = -x/(1+x)$  on the set  $X = [0, 1]$ , since it includes no rules for concluding convexity of a quotient of two non-constant functions. Formulating the function as  $f(x) = 1/(1+x) - 1$ , however, convexity is proven, since the numerator of  $1/(1+x)$  is a positive constant, and the denominator  $1+x$  is concave and positive. The methods from [Mönnigmann, 2008, Nenov et al., 2004], in contrast, have no problem in proving convexity for  $f(x)$ , since they only need to prove positivity of the second derivative  $f''(x) = 2/(1+x)^3$ .

Nevertheless, for the function  $f(x) = 2x^7 - 7x^4 + 84x^2 + 42$  the method of Nenov et al. [2004] may fail to prove convexity of  $f(x)$  on the interval  $[-1, 2]$ . In this example the second derivative is  $f''(x) = 84(x^5 - x^2 + 2)$ . Replacing each occurrence of  $x$  by  $[-1, 2]$  and applying interval arithmetic yields  $f''(x) \in 84 \cdot ([-1, 32] - [0, 4] + [2, 2]) = 84 \cdot [-3, 34]$ , which allows no conclusive result.

Finally, when proving or disproving convexity of a function over a set, all these methods can consider only *simple bound constraints* on the variables  $x_i$ . For instance, the function  $f(x) = (x_1 - x_2)^3$  is obviously convex on the set  $X = \{x \in [0, 1]^2 : x_1 \geq x_2\}$ . However, the method of Fourer et al. [2009] would fail to prove convexity of  $f(x)$  since it only considers the simple bounds  $x \in [0, 1]^2$  and thus does not “see” that  $x_1 - x_2 \geq 0$  on  $X$ .

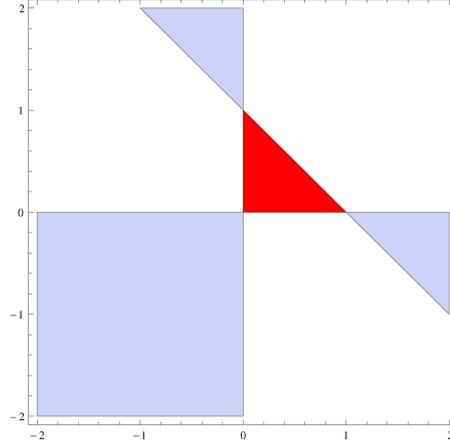


Figure 7.6.: Regions for  $(\beta_1, \beta_2) \in [-2, 2]^2$  where  $x_1^{\beta_1} x_2^{\beta_2}$  is convex (blue) or concave (red) with respect to  $x \in \mathbb{R}_+^2$ .

which indicate whether the associated expression is convex or concave w.r.t. the current variables domain. If both flags are set, the corresponding expression is obviously linear. If a flag is not set, this does not necessarily mean that also the corresponding expression is nonconvex or nonconcave, since the applied rules only check sufficient conditions.

For sources  $v \in V$ , we set both  $\text{convex}(v)$  and  $\text{concave}(v)$  to **true**. For a non-source vertex  $v \in V$  with  $o(v) = (w_1, \dots, w_m; f)$ , we set  $\text{convex}(v)$  and  $\text{concave}(v)$  to the outcomes of the rules defined in Table 7.2 when applied to the function  $f$  with flags  $\text{convex}(w_i)$  and  $\text{concave}(w_i)$  and intervals  $w_i([\underline{x}, \bar{x}])$  from the child expressions,  $i \in [m]$ . The rule for signomials  $\prod_{j \in [m]} g_j^{\beta_j}$  is proven in Maranas and Floudas [1995] and Chen and Huang [2009]<sup>16</sup>, see also Figure 7.6. For the other functions, the rules are easily shown by checking the signs of the terms appearing on the right-hand-side of (7.9).

**Example 7.11.** Consider the function

$$(\log(x + y - 3z^2))^{-2}$$

and the bounds  $x \in [2, 3]$ ,  $y \in [2, 3]$ ,  $z \in [0, 1]$ . The function is recognized as convex, because from the linearity of  $x$ ,  $y$ , and  $z$  it follows that

- $z^2$  is convex (by the rule for  $f \circ g = g^a$ ,  $a = 2$ ) and  $z^2 \in [0, 1]$
- $\Rightarrow -3z^2$  is concave (by the rule for  $f \circ g = ag$ ,  $a = -3$ ) and  $-3z^2 \in [-3, 0]$
- $\Rightarrow x + y - 3z^2$  is concave (by the rule for  $f \circ g = f + g$ ) and  $x + y - 3z^2 \in [1, 6]$

<sup>16</sup>Conjecture: The convexity condition for  $\prod_{j \in [m]} g_j^{\beta_j}$  can be relaxed to require only convexity of  $g_j$  if  $\beta_j > 0$  and concavity if  $\beta_j < 0$ , instead of linearity. Similar, the conditions on concavity of  $\prod_{j \in [m]} g_j^{\beta_j}$  can be relaxed to require only concavity of  $g_j$  instead of linearity.

For  $m = 2$  and univariate functions  $g_j : \mathbb{R} \rightarrow \mathbb{R}$ , the correctness of this conjecture is easily seen by checking conditions on positive-semidefiniteness of the Hessian of  $g_1^{\beta_1} g_2^{\beta_2}$ . Also for arbitrary  $m$  and univariate  $g_j$ , convexity of  $\prod_{j \in [m]} g_j^{\beta_j}$  with  $\beta_j \leq 0$  and concave  $g_j$ ,  $j \in [m]$ , can be shown by applying Theorem 5.1 from Gounaris and Floudas [2008a] for the convexity of a product of univariate functions.

## 7. A Constraint Integer Programming Approach to MINLP

$f \circ g$	condition for <b>convexity</b> of $(f \circ g)(x)$
$g_1 + g_2$	$\text{convex}(g_1) \wedge \text{convex}(g_2)$
$ag, a \in \mathbb{R}$	$(a \geq 0 \wedge \text{convex}(g)) \vee (a \leq 0 \wedge \text{concave}(g))$
$\exp(g)$	$\text{convex}(g)$
$\log(g)$	<b>false</b>
$g^a, a \in \mathbb{R}$	$(\bar{g} \geq 0 \Rightarrow ((a \geq 1 \wedge \text{convex}(g)) \vee (a \leq 0 \wedge \text{concave}(g)))) \wedge$ $(g \leq 0 \Rightarrow ((a \in -2\mathbb{N} \wedge \text{convex}(g)) \vee (a \in 2\mathbb{N} \wedge \text{concave}(g)))) \wedge$ $((g < 0 \wedge \bar{g} > 0) \Rightarrow a \geq 0)$
$ g $	$(\bar{g} \leq 0 \wedge \text{concave}(g)) \vee (g \geq 0 \wedge \text{convex}(g)) \vee (\text{convex}(g) \wedge \text{concave}(g))$
$\prod_{j \in [m]} g_j^{\beta_j}$	$\left( \exists j^* \in [m] : \left( (j \in [m] \setminus \{j^*\} \Rightarrow \beta_j \leq 0) \wedge \left( \beta_{j^*} \leq 0 \vee \sum_{j \in [m]} \beta_j \geq 1 \right) \right) \right) \wedge$ $(j \in [m] \Rightarrow \underline{g}_j \geq 0 \wedge \text{convex}(g_j) \wedge \text{concave}(g_j))$
$f \circ g$	condition for <b>concavity</b> of $(f \circ g)(x)$
$g_1 + g_2$	$\text{concave}(g_1) \wedge \text{concave}(g_2)$
$ag, a \in \mathbb{R}$	$(a \geq 0 \wedge \text{concave}(g)) \vee (a \leq 0 \wedge \text{convex}(g))$
$\exp(g)$	<b>false</b>
$\log(g)$	$\text{concave}(g)$
$g^a, a \in \mathbb{R}$	$(\bar{g} \geq 0 \Rightarrow ((a \in [0, 1] \wedge \text{concave}(g)))) \wedge$ $(g \leq 0 \Rightarrow ((a \in -2\mathbb{N} - 1 \wedge \text{convex}(g)) \vee (a \in 2\mathbb{N} + 1 \wedge \text{concave}(g)))) \wedge$ $((g < 0 \wedge \bar{g} > 0) \Rightarrow a \geq 0)$
$ g $	$(\bar{g} \leq 0 \wedge \text{convex}(g)) \vee (g \geq 0 \wedge \text{concave}(g))$
$\prod_{j \in [m]} g_j^{\beta_j}$	$(j \in [m] \Rightarrow (\beta_j \geq 0 \wedge \underline{g}_j \geq 0 \wedge \text{convex}(g_j) \wedge \text{concave}(g_j)) \wedge \sum_{j \in [m]} \beta_j \leq 1)$

Table 7.2.: Sufficient conditions for convexity or concavity of a function composition  $(f \circ g)(x)$  with  $f : \mathbb{R}^m \rightarrow \mathbb{R}$  and  $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$  for given convexity/concavity and bounds for  $g_i(x)$ ,  $i \in [m]$ . For a product  $\prod_{j=1}^m g_j^{\beta_j}$  where some  $g_j$  are negative ( $\bar{g}_j \leq 0$  and  $\beta_j \in \mathbb{Z}$ ), the given rules can be applied after replacing  $g_j$  by  $-g_j$ :

$$\prod_{j \in [m]} g_j^{\beta_j} = (-1)^{|\{j \in [m] : \bar{g}_j \leq 0, \beta_j \in 2\mathbb{Z} + 1\}|} \left( \prod_{j \in [m], \underline{g}_j \geq 0} g_j^{\beta_j} \right) \left( \prod_{j \in [m], \bar{g}_j \leq 0} (-g_j)^{\beta_j} \right).$$

For the signed power  $\text{sign}(g)|g|^a$  ( $a > 1$ ), no convexity or concavity is concluded, except if  $\bar{g} \leq 0$  or  $\underline{g} \geq 0$  (i.e., sign is fixed), in which case the rules for  $-(-g)^a$  (if  $\bar{g} \leq 0$ ) or  $g^a$  (if  $\underline{g} \geq 0$ ) apply.

## 7.4. Bound Tightening for Quadratic Functions

$\Rightarrow \log(x + y - 3z^2)$  is concave (by the rule for  $f \circ g = \log(g)$ ) and  $\log(x + y - 3z^2) \in [0, \log(6)]$   
 $\Rightarrow \log(x + y - 3z^2)^{-2}$  is convex (by the rule for  $f \circ g = g^a$ ,  $a = -2$ , with  $g \geq 0$ ).

Unfortunately, convexity of the function

$$\left( \left( \frac{x}{\varepsilon + z} \right)^2 - 97 \frac{x}{\varepsilon + z} + 2352.25z \right) (\varepsilon + z) \quad (7.10)$$

with  $\varepsilon = 10^{-6}$ ,  $x \geq 0$ , and  $z \in [0, 1]$  is not recognized by SCIP (however, the method in Neun et al. [2010] recognizes convexity). This function is part of a convex hull description [Grossmann and Lee, 2003] for an indicator constraint  $z = 1 \Rightarrow (x - \frac{97}{2})^2 + (y - \frac{157}{2})^2 \leq 0$ .

Additionally to the convexity detection for expressions in an expression graph, SCIP detects convexity of quadratic constraints

$$\langle x, Qx \rangle + \langle q, x \rangle + \bar{q} \leq 0$$

by checking whether the minimal eigenvalue of the matrix  $Q$  is nonnegative.

## 7.4. Bound Tightening for Quadratic Functions

Recall, that due to the dependency problem in interval arithmetic, the bounds computed for an algebraic expression may not be best possible if a variable appears several times in the expression. In this section, we discuss how to obtain tighter bounds for univariate and bivariate quadratic expressions (forward propagation). Further, we discuss how to compute tight bounds on variables involved in such quadratic expression for given bounds on the expression itself (backward propagation). The univariate case has been discussed in Domes and Neumaier [2010] and we only summarize the results here. For the multivariate (non-separable) case, Domes and Neumaier [2010] discuss estimations for the bilinear terms such that hopefully good bounds can be computed for the resulting multivariate separable quadratic form. For the multivariate strictly convex case, Domes and Neumaier [2011] show how to compute tight bounds. However, the formulas for the bivariate case that we derive in this section allow to compute best bounds for bivariate quadratic expression and best bounds on the solution of bivariate quadratic equations, for both the convex and the nonconvex case.

### 7.4.1. Univariate Quadratic Expressions

#### Bounds for Univariate Quadratic Expressions

We seek for bounds on a quadratic form

$$ax^2 + bx \quad (7.11)$$

for given bounds on  $x$ . Assume  $a \neq 0$ .

## 7. A Constraint Integer Programming Approach to MINLP

For  $a > 0$  ( $a < 0$ ), (7.11) attains its global (unrestricted) maximum (minimum) at  $\hat{x} = -\frac{b}{2a}$ , with  $a\hat{x}^2 + b\hat{x} = -\frac{b^2}{4a}$ . Thus, if  $[\underline{x}, \bar{x}]$  is bounded,

$$\{ax^2 + bx : x \in [\underline{x}, \bar{x}]\} = \begin{cases} \text{conv}\{a\underline{x}^2 + b\underline{x}, a\bar{x}^2 + b\bar{x}, -\frac{b^2}{4a}\}, & \text{if } -\frac{b}{2a} \in [\underline{x}, \bar{x}], \\ \text{conv}\{a\underline{x}^2 + b\underline{x}, a\bar{x}^2 + b\bar{x}\}, & \text{otherwise.} \end{cases} \quad (7.12)$$

If  $[\underline{x}, \bar{x}]$  is unbounded, the same formula is used with  $a(\pm\infty)^2 \pm \infty$  replaced by  $\text{sign}(a)\infty$ .

The case where also the coefficients  $a$  and  $b$  can vary within intervals  $[\underline{a}, \bar{a}]$  and  $[\underline{b}, \bar{b}]$  can be reduced to find

$$\begin{aligned} & [\min\{\underline{a}x^2 + \underline{b}x : x \in [0, \bar{x}]\}, \max\{\bar{a}x^2 + \bar{b}x : x \in [0, \bar{x}]\}] \\ & \cup [\min\{\underline{a}x^2 - \underline{b}x : x \in [0, -\underline{x}]\}, \max\{\bar{a}x^2 - \bar{b}x : x \in [0, -\underline{x}]\}], \end{aligned}$$

where the minima and maxima are given by (7.12).

### Solving Univariate Quadratic Expressions

Consider an equation

$$ax^2 + bx \geq c \quad (7.13)$$

with  $a \neq 0$ . To find the smallest interval containing all solutions of (7.13), we rewrite as

$$\begin{aligned} \left(x + \frac{b}{2a}\right)^2 &\geq \frac{1}{a} \left(c + \frac{b^2}{4a}\right) && \text{for } a > 0, \\ \left(x + \frac{b}{2a}\right)^2 &\leq \frac{1}{a} \left(c + \frac{b^2}{4a}\right) && \text{for } a < 0, \end{aligned}$$

If  $c + \frac{b^2}{4a} \leq 0$  and  $a > 0$ , then obviously  $[\underline{x}, \bar{x}] = \mathbb{R}$ . If  $c + \frac{b^2}{4a} > 0$  and  $a < 0$ , then  $[\underline{x}, \bar{x}] = \emptyset$ . Otherwise, i.e.,  $\frac{1}{a}(c + \frac{b^2}{4a}) \geq 0$ , we obtain

$$x \in \left[-\infty, -\sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}\right] \cup \left[\sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}, \infty\right] \quad \text{for } a > 0, \quad (7.14a)$$

$$x \in \left[-\sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}, \sqrt{\frac{c}{a} + \frac{b^2}{4a^2}} - \frac{b}{2a}\right] \quad \text{for } a < 0. \quad (7.14b)$$

The case where also an upper bound  $ax^2 + b$  is given and the coefficients  $a$  and  $b$  can vary within intervals  $[\underline{a}, \bar{a}]$  and  $[\underline{b}, \bar{b}]$  can be reduced to (7.13) by noting that

$$\begin{aligned} \{x : ax^2 + bx \in [\underline{c}, \bar{c}] \text{ for some } a \in [\underline{a}, \bar{a}], b \in [\underline{b}, \bar{b}]\} = \\ (\{x : \underline{a}x^2 + \underline{b}x \leq \bar{c}\} \cap \{x : \bar{a}x^2 + \bar{b}x \geq \underline{c}\} \cap \mathbb{R}_+) \cup \\ (\{-x : \underline{a}x^2 - \underline{b}x \leq \bar{c}\} \cap \{-x : \bar{a}x^2 - \bar{b}x \geq \underline{c}\} \cap \mathbb{R}_-). \end{aligned} \quad (7.15)$$

### 7.4.2. Bivariate Quadratic Expressions

We now consider a bivariate quadratic form

$$q(x, y) := a_x x^2 + a_y y^2 + a_{xy} xy + b_x x + b_y y \quad (7.16)$$

with  $a_{xy} \neq 0$ . If  $a_{xy} = 0$ , the formulas for the univariate case can be applied.

#### Bounds for Bivariate Quadratic Expressions

For given bounds  $[\underline{x}, \bar{x}]$  on  $x$  and  $[\underline{y}, \bar{y}]$  on  $y$ , we aim to find the interval

$$\text{conv}\{q(x, y) : x \in [\underline{x}, \bar{x}], y \in [\underline{y}, \bar{y}]\}.$$

Thus, we compute the minima and maxima of  $q(x, y)$  in the interior and on the boundary of  $[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$ . It is  $\frac{\partial q(x, y)}{\partial x} = 2a_x x + b_x + a_{xy} y$  and  $\frac{\partial q(x, y)}{\partial y} = 2a_y y + b_y + a_{xy} x$ , which gives the following (unrestricted) minima/maxima of  $q(x, y)$ : If  $4a_x a_y \neq a_{xy}^2$ , then

$$\hat{x} = \frac{a_{xy} b_y - 2a_y b_x}{4a_x a_y - a_{xy}^2}, \quad \hat{y} = \frac{a_{xy} b_x - 2a_x b_y}{4a_x a_y - a_{xy}^2} \quad \Rightarrow \quad q(\hat{x}, \hat{y}) = \frac{a_{xy} b_x b_y - a_y b_x^2 - a_x b_y^2}{4a_x a_y - a_{xy}^2}.$$

If  $4a_x a_y = a_{xy}^2$  and  $2a_y b_x = a_{xy} b_y$ , then

$$\hat{x} \in \mathbb{R}, \quad \hat{y} = -\frac{b_x}{a_{xy}} - \frac{a_{xy}}{2a_y} \hat{x} \quad \Rightarrow \quad q(x, y) = -\frac{a_y b_x^2}{a_{xy}^2}.$$

Otherwise ( $4a_x a_y = a_{xy}^2$  and  $2a_y b_x \neq a_{xy} b_y$ ), there is no unrestricted minimum/maximum.

For  $x$  or  $y$  fixed to  $\underline{x}$ ,  $\bar{x}$ , or  $\underline{y}$ ,  $\bar{y}$ , respectively, the minimum/maximum of  $q(x, y)$  can be derived for the corresponding univariate quadratic form via (7.12).

Bounds on  $q(x, y)$  are then obtained by comparing the minimal/maximal values at the boundary of  $[\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$  with the value at  $(\hat{x}, \hat{y})$ , if  $(\hat{x}, \hat{y}) \in [\underline{x}, \bar{x}] \times [\underline{y}, \bar{y}]$ . The latter need to be computed only if  $4a_x a_y \neq a_{xy}^2$ , since otherwise the unrestricted minimum/maximum, if any, is also attained for  $x \in \{\underline{x}, \bar{x}\}$ .

#### Solving Bivariate Quadratic Expressions ( $a_x \neq 0$ )

Consider the bivariate quadratic equation

$$q(x, y) \in [\underline{c}, \bar{c}] \quad (7.17)$$

for some interval  $[\underline{c}, \bar{c}]$ . For given  $[\underline{x}, \bar{x}]$  and  $[\underline{y}, \bar{y}]$  (possibly unbounded), we aim to find the interval

$$[\underline{x}', \bar{x}'] := \text{conv}\{x \in [\underline{x}, \bar{x}] : q(x, y) \in [\underline{c}, \bar{c}] \text{ for some } y \in [\underline{y}, \bar{y}]\}.$$

The case for finding bounds on  $y$  is analog.

## 7. A Constraint Integer Programming Approach to MINLP

First, consider the case  $a_x \neq 0$ . Thus, w.l.o.g., assume  $a_x > 0$ . We rewrite (7.17) as

$$a_x x^2 + (b_x + a_{xy}y)x \in [\underline{c}, \bar{c}] - a_y y^2 - b_y y,$$

which is equivalent to

$$(\sqrt{a_x}x + b(y))^2 \in r([\underline{c}, \bar{c}], y), \quad (7.18)$$

where

$$b(y) := \frac{b_x + a_{xy}y}{2\sqrt{a_x}} \quad \text{and} \quad r(c, y) := c - a_y y^2 - b_y y + b(y)^2,$$

that is,

$$r(c, y) = \frac{b_x^2}{4a_x} + c + \left( \frac{a_{xy}b_x}{2a_x} - b_y \right) y + \left( \frac{a_{xy}^2}{4a_x} - a_y \right) y^2.$$

By (7.12), we can compute the interval  $r([\underline{c}, \bar{c}], [y, \bar{y}])$ . If  $r([\underline{c}, \bar{c}], [y, \bar{y}]) \cap \mathbb{R}_+ = \emptyset$ , then (7.17) has no solution for any  $y \in [y, \bar{y}]$ , i.e.,  $[x', \bar{x}'] = \emptyset$ . Otherwise, the set of  $x$  that satisfies (7.18) is

$$\left\{ x : \exists y \in [y, \bar{y}] : \sqrt{a_x}x + b(y) \in -\sqrt{r([\underline{c}, \bar{c}], y)} \cup \sqrt{r([\underline{c}, \bar{c}], y)} \right\}. \quad (7.19)$$

It thus remains to compute minimal and maximal values for  $\pm\sqrt{r([\underline{c}, \bar{c}], y)} - b(y)$ .

**Minimum of  $\sqrt{r(c, y)} - b(y)$ .** Since  $r(c, y)$  is monotonically increasing in  $c$ ,

$$\begin{aligned} & \min_{c \in [\underline{c}, \bar{c}], y \in [y, \bar{y}], r(c, y) \geq 0} \left( \sqrt{r(c, y)} - b(y) \right) \\ &= \min \left\{ \min_{c \in [\underline{c}, \bar{c}], y \in [y, \bar{y}], r(c, y) > 0} \left( \sqrt{r(c, y)} - b(y) \right), \min_{c \in [\underline{c}, \bar{c}], y \in [y, \bar{y}], r(c, y) = 0} -b(y) \right\} \\ &= \min \left\{ \min_{y \in [y, \bar{y}], r(\underline{c}, y) > 0} \left( \sqrt{r(\underline{c}, y)} - b(y) \right), \min_{c \in [\underline{c}, \bar{c}], y \in [y_0, \bar{y}_0], r(c, y) = 0} -b(y) \right\} \quad (7.20) \end{aligned}$$

The second equality holds, since for any  $(c^*, y^*)$  with  $r(c^*, y^*) > 0$ , it is  $\sqrt{r(c^*, y^*)} - b(y^*) > \sqrt{r(c^* - r(c^*, y^*), y^*)} - b(y^*) = -b(y^*)$  and  $\sqrt{r(c^*, y^*)} - b(y^*) > \sqrt{r(\underline{c}, y^*)} - b(y^*)$ , if  $r(\underline{c}, y^*) \geq 0$ .

For the second case in (7.20) ( $r(c, y) = 0$ ), we have

$$\min_{c \in [\underline{c}, \bar{c}], y \in [y, \bar{y}], r(c, y) = 0} -b(y) = \min\{-b(y) : a_y y^2 + b_y y + b(y)^2 \in [\underline{c}, \bar{c}], y \in [y, \bar{y}]\}$$

The set  $\{y \in [y, \bar{y}] : a_y y^2 + b_y y + b(y)^2 \in [\underline{c}, \bar{c}]\}$  is given by (7.14) as one interval or union of two intervals. The minimum of the linear function  $-b(y)$  over this set is easily found.

For the first case in (7.20) ( $r(\underline{c}, y) > 0$ ), finding a minimum of  $\sqrt{r(\underline{c}, y)} - b(y)$ ,  $y \in [y, \bar{y}]$ , with nonzero value for  $r(\underline{c}, y)$  requires evaluation of the function at the boundary of  $[y, \bar{y}]$  (provided  $r(\underline{c}, y) > 0$ ) and finding its unrestricted minimum. For that latter, we compute

#### 7.4. Bound Tightening for Quadratic Functions

the zeros of the first derivative of  $\sqrt{r(\underline{c}, y)} - b(y)$ ,

$$\frac{r'(\underline{c}, y)}{2\sqrt{r(\underline{c}, y)}} - b'(y) = \frac{-2a_y y - b_y + 2b(y)b'(y)}{2\sqrt{\underline{c} - a_y y^2 - b_y y + b(y)^2}} - b'(y) \stackrel{!}{=} 0. \quad (7.21)$$

This equation can be solved by solving the quadratic ( $b'(y)$  is constant) equation

$$(-2a_y y - b_y + 2b(y)b'(y))^2 = 4b'(y)^2(\underline{c} - a_y y^2 - b_y y + b(y)^2) \quad (7.22)$$

and checking whether its solutions satisfy  $r(\underline{c}, y) > 0$  and (7.21). (7.22) is equivalent to

$$(a_x b_y^2 - a_{xy}^2 \underline{c} - a_{xy} b_x b_y) + (4a_x b_y - 2a_{xy} b_x) a_y y + (4a_x a_y - a_{xy}^2) a_y y^2 = 0.$$

For  $a_{xy}^2 \neq 4a_x a_y$  and  $a_y \neq 0$ , the solutions are

$$y_{+,-} := \frac{a_{xy} a_y b_x - 2a_x a_y b_y \pm \sqrt{a_{xy}^2 a_y (a_y b_x^2 - a_{xy} b_x b_y + a_x b_y^2 - a_{xy}^2 \underline{c} + 4a_x a_y \underline{c})}}{a_y (-a_{xy}^2 + 4a_x a_y)}.$$

For  $a_{xy}^2 = 4a_x a_y$  and  $2a_x b_y \neq a_{xy} b_x$  and  $a_y \neq 0$ , the solution is

$$y_{+,-} = -\frac{4a_y b_x b_y - a_{xy} b_y^2 + 4a_{xy} a_y \underline{c}}{4a_y (2a_x b_y - a_{xy} b_x)}.$$

In the case  $a_y = 0$  and in the case  $a_{xy}^2 = 4a_x a_y$  and  $2a_x b_y = a_{xy} b_x$ , the quadratic equation has either no solution or is always satisfied, i.e.,  $\sqrt{r(\underline{c}, y)} - b(y)$  is either constant or strictly monotone, thus, considering the extreme points of  $[y, \bar{y}]$  is sufficient.

**Maximum of  $\sqrt{r(\underline{c}, y)} - b(y)$ .** Similar considerations as for the minimum can be done to determine

$$\begin{aligned} & \max_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}], r(c, y) \geq 0} \left( \sqrt{r(c, y)} - b(y) \right) \\ &= \max \left\{ \max_{y \in [\underline{y}, \bar{y}], r(\bar{c}, y) > 0} \left( \sqrt{r(\bar{c}, y)} - b(y) \right), \max_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}], r(c, y) = 0} -b(y) \right\}. \end{aligned}$$

**Minimum/Maximum of  $-\sqrt{r(\underline{c}, y)} - b(y)$ .** Here, we have

$$\begin{aligned} & \min_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}], r(c, y) \geq 0} \left( -\sqrt{r(c, y)} - b(y) \right) \\ &= \min \left\{ \min_{y \in [\underline{y}, \bar{y}], r(\bar{c}, y) > 0} \left( -\sqrt{r(\bar{c}, y)} - b(y) \right), \min_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}], r(c, y) = 0} -b(y) \right\} \end{aligned}$$

## 7. A Constraint Integer Programming Approach to MINLP

$$\begin{aligned} & \max_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}], r(c, y) \geq 0} \left( -\sqrt{r(c, y)} - b(y) \right) \\ & = \max \left\{ \max_{y \in [\underline{y}, \bar{y}], r(\underline{c}, y) > 0} \left( -\sqrt{r(\underline{c}, y)} - b(y) \right), \max_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}], r(c, y) = 0} -b(y) \right\} \end{aligned}$$

These minima and maxima can be found via (7.21) as above.

**Unbounded**  $[\underline{c}, \bar{c}]$ . The following limits can be derived.

For  $\bar{c} = +\infty$ , we have

$$\begin{aligned} \inf_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}]} \left( -\sqrt{r(c, y)} - b(y) \right) &= -\infty, \\ \sup_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}]} \left( +\sqrt{r(c, y)} - b(y) \right) &= +\infty. \end{aligned}$$

Thus, tightened bounds for  $\sqrt{a_x x}$  are only available if  $\sqrt{a_x \bar{x}} < \min_y (\sqrt{r(\underline{c}, y)} - b(y))$  or  $\sqrt{a_x \underline{x}} > \max_y (-\sqrt{r(\underline{c}, y)} - b(y))$ .

For  $\underline{c} = -\infty$ , we have

$$\begin{aligned} \inf_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}]} \left( +\sqrt{r(c, y)} - b(y) \right) &= \inf_{y \in [\underline{y}, \bar{y}]} -b(y), \\ \sup_{c \in [\underline{c}, \bar{c}], y \in [\underline{y}, \bar{y}]} \left( -\sqrt{r(c, y)} - b(y) \right) &= \sup_{y \in [\underline{y}, \bar{y}]} -b(y). \end{aligned}$$

Thus, the two intervals  $\pm \sqrt{r([\underline{c}, \bar{c}], [\underline{y}, \bar{y}])} - b([\underline{y}, \bar{y}])$  overlap and only  $\min_y (-\sqrt{r(\bar{c}, y)} - b(y))$  and  $\max_y (\sqrt{r(\bar{c}, y)} - b(y))$  need to be computed.

**Unbounded**  $[\underline{y}, \bar{y}]$ . Let  $[\underline{c}, \bar{c}]$  be bounded. If  $a_y \neq 0$  and  $a_{xy}^2 - 4a_x a_y \geq 0$ , then

$$\begin{aligned} \lim_{y \rightarrow -\infty} \left( -\sqrt{r(c, y)} - b(y) \right) &= \text{sign} \left( a_{xy} - \sqrt{a_{xy}^2 - 4a_x a_y} \right) \infty, \\ \lim_{y \rightarrow +\infty} \left( -\sqrt{r(c, y)} - b(y) \right) &= \text{sign} \left( -a_{xy} - \sqrt{a_{xy}^2 - 4a_x a_y} \right) \infty, \\ \lim_{y \rightarrow -\infty} \left( +\sqrt{r(c, y)} - b(y) \right) &= \text{sign} \left( a_{xy} + \sqrt{a_{xy}^2 - 4a_x a_y} \right) \infty, \\ \lim_{y \rightarrow +\infty} \left( +\sqrt{r(c, y)} - b(y) \right) &= \text{sign} \left( -a_{xy} + \sqrt{a_{xy}^2 - 4a_x a_y} \right) \infty. \end{aligned}$$

Since  $a_x > 0$ , this implies

$$\lim_{y \rightarrow -\infty} \left( -\sqrt{r(c, y)} - b(y) \right) = \begin{cases} \text{sign}(a_y) \infty, & a_{xy} > 0, \\ -\infty, & a_{xy} < 0, \end{cases}$$

#### 7.4. Bound Tightening for Quadratic Functions

$$\begin{aligned}\lim_{y \rightarrow +\infty} \left( -\sqrt{r(c, y)} - b(y) \right) &= \begin{cases} -\infty, & a_{xy} > 0, \\ \text{sign}(a_y)\infty, & a_{xy} < 0, \end{cases} \\ \lim_{y \rightarrow -\infty} \left( +\sqrt{r(c, y)} - b(y) \right) &= \begin{cases} -\text{sign}(a_y)\infty, & a_{xy} < 0, \\ \infty, & a_{xy} > 0, \end{cases} \\ \lim_{y \rightarrow +\infty} \left( +\sqrt{r(c, y)} - b(y) \right) &= \begin{cases} \infty, & a_{xy} < 0, \\ -\text{sign}(a_y)\infty, & a_{xy} > 0. \end{cases}\end{aligned}$$

If  $a_y \neq 0$  and  $a_{xy}^2 - 4a_x a_y < 0$ , the limits for  $y \rightarrow \pm\infty$  do not exist, since  $r(c, y) \rightarrow -\infty$ . However, in these cases the minimum/maximum of  $\pm\sqrt{r(c, y)} - b(y)$  is defined by the one for  $-b(y)$  with respect to  $r(c, y) = 0$ . If  $a_y = 0$ , then

$$\begin{aligned}\lim_{y \rightarrow -\infty} \left( -\sqrt{r(c, y)} - b(y) \right) &= -\frac{b_y}{2}, \\ \lim_{y \rightarrow +\infty} \left( -\sqrt{r(c, y)} - b(y) \right) &= -\infty, \\ \lim_{y \rightarrow -\infty} \left( +\sqrt{r(c, y)} - b(y) \right) &= +\infty, \\ \lim_{y \rightarrow +\infty} \left( +\sqrt{r(c, y)} - b(y) \right) &= -\frac{b_y}{2}.\end{aligned}$$

#### Solving Bivariate Quadratic Expressions ( $a_x = 0$ )

With  $a_x = 0$ , the problem of finding all  $x \in [\underline{x}, \bar{x}]$  for which a  $y \in [\underline{y}, \bar{y}]$  with (7.17) exists, reduces to finding all solutions of

$$\exists y \in [\underline{y}, \bar{y}] : (b_x + a_{xy}y)x \in [\underline{c}, \bar{c}] - a_y y^2 - b_y y. \quad (7.23)$$

Consider first the case  $-\frac{b_x}{a_{xy}} \in [\underline{y}, \bar{y}]$ . If  $0 \in [\underline{c}, \bar{c}] - a_y \left(\frac{b_x}{a_{xy}}\right)^2 + b_y \frac{b_x}{a_{xy}}$  or  $-\frac{b_x}{a_{xy}} \in (\underline{y}, \bar{y})$ , then every  $x \in \mathbb{R}$  is a solution of (7.23).

If  $-\frac{b_x}{a_{xy}} \notin (\underline{y}, \bar{y})$ , then all solutions of (7.23) are given by

$$\left\{ \frac{[\underline{c}, \bar{c}] - a_y y^2 - b_y y}{b_x + a_{xy}y} : y \in [\underline{y}, \bar{y}] \right\}. \quad (7.24)$$

To find an interval that encloses this set, we can argue again that only the interval ends of  $[\underline{c}, \bar{c}]$  need to be considered. The minima and maxima of  $\frac{c - a_y y^2 - b_y y}{b_x + a_{xy}y}$ ,  $c \in \{\underline{c}, \bar{c}\}$ , can be computed by evaluating at  $y = \underline{y}$  and  $y = \bar{y}$ <sup>17</sup> and by finding the zeros of the first derivative. The latter means to solve the quadratic equation

$$(-2a_y y - b_y)(b_x + a_{xy}y) - (c - a_y y^2 - b_y y)a_{xy} = 0,$$

<sup>17</sup>For  $-\frac{b_x}{a_{xy}} \in \{\underline{y}, \bar{y}\}$ , the case  $0 \in [\underline{c}, \bar{c}] - a_y \left(\frac{b_x}{a_{xy}}\right)^2 + b_y \frac{b_x}{a_{xy}}$  has been discussed before. For  $0 \notin [\underline{c}, \bar{c}] - a_y \left(\frac{b_x}{a_{xy}}\right)^2 + \frac{b_x b_y}{a_{xy}}$ , evaluating (7.24) with a 0 nominator gives a definite value of  $+\infty$  or  $-\infty$ .

## 7. A Constraint Integer Programming Approach to MINLP

which has the solutions

$$y_{1,2} := -\frac{a_y b_x \pm \sqrt{a_y (a_y b_x^2 - a_{xy} (b_x b_y + a_{xy} c))}}{a_{xy} a_y},$$

if  $a_y \neq 0$ . Otherwise,  $\frac{\partial}{\partial y} \left( \frac{c - a_y y^2 - b_y y}{b_x + a_{xy} y} \right) = -\frac{b_x b_y + a_{xy} c}{(b_x + a_{xy} y)^2}$ . Thus  $\frac{c - a_y y^2 - b_y y}{b_x + a_{xy} y}$  is either constant (so that evaluating at  $\underline{y}$  or  $\bar{y}$  is sufficient) or has no unrestricted minima and maxima.

Further, if  $a_y \neq 0$ , then

$$\lim_{y \rightarrow -\infty} \frac{c - a_y y^2 - b_y y}{b_x + a_{xy} y} = +\text{sign} \left( \frac{a_y}{a_{xy}} \right) \infty,$$

$$\lim_{y \rightarrow +\infty} \frac{c - a_y y^2 - b_y y}{b_x + a_{xy} y} = -\text{sign} \left( \frac{a_y}{a_{xy}} \right) \infty,$$

and if  $a_y = 0$ , then

$$\lim_{y \rightarrow \pm\infty} \frac{c - a_y y^2 - b_y y}{b_x + a_{xy} y} = -\frac{b_y}{a_{xy}}.$$

**Example 7.12.** Consider the quadratic inequality

$$2x^2 - y^2 + xy - y \leq 0 \tag{7.25}$$

and bounds  $x \in [0, 1]$  and  $y \in [-1, 1]$ , see also Figure 7.7. We are interested in deriving a tighter upper bound for  $x$ .

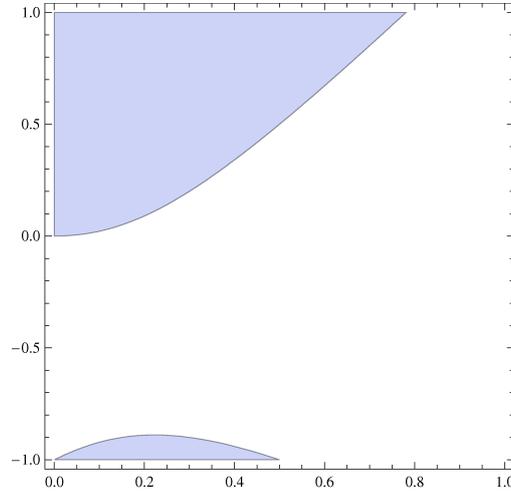


Figure 7.7.: Feasible region of inequality (7.25).

First, let us relax (7.25) into a univariate form by rewriting as

$$2x^2 + [-1, 1]x \leq \max\{y^2 + y : y \in [-1, 1]\}.$$

By (7.15), we have to compute

$$(\{x : 2x^2 - x \leq 2\} \cap \mathbb{R}_+) \cup (\{-x : 2x^2 - x \leq 2\} \cap \mathbb{R}_-).$$

This gives the new bounds  $[1/4(1 - \sqrt{17}), 1/4(1 + \sqrt{17})] \approx [-0.780776, 1.28078]$  for  $x$ , which do not improve on the existing ones.

Next, we write (7.25) in the form of (7.18), which gives

$$(\sqrt{2}x + b(y))^2 \in r(\mathbb{R}_-, y),$$

with

$$r(c, y) = c + y^2 + y + b(y)^2 \quad \text{and} \quad b(y) = \frac{y}{2\sqrt{2}}.$$

Due to (7.19), an upper bound on  $\sqrt{2}x$  is given by

$$\begin{aligned} \max \left\{ \sqrt{r(0, y)} - b(y) : y \in [-1, 1] \right\} \\ = \max \left\{ \sqrt{y + \frac{9y^2}{8}} - \frac{y}{2\sqrt{2}} : y \in [-1, 1] \right\} = \frac{\sqrt{17} - 1}{2\sqrt{2}}, \end{aligned}$$

which yields  $(\sqrt{17} - 1)/4 \approx 0.780776$  as new upper bound for  $x$ .

## 7.5. Outer-Approximation

The MINLP constraint handlers in SCIP strengthen the LP relaxation by adding valid cuts. Cuts can be added either during the cut-and-price loop or in the constraint enforcement step, see also Figure 7.1. During the cut-and-price loop, all separators and the separation routines of all constraint handlers can be called. Here, the MINLP constraint handlers only add cuts which are sufficiently much violated by the relaxations solution<sup>18</sup>. However, the enforcement routine of a constraint handler is only called if no other constraint handler with a higher priority was able to resolve a possible infeasibility of the current solution. If a constraint handler finds that the current relaxation solution violates one of its constraints, the constraint handler needs to enforce its constraints (cf. Section 7.2), e.g., by adding a cut that separates the solution from the relaxation. Thus, if a MINLP constraint handler enforces a violated nonlinear constraint and finds a cut that is violated by the current solution of the LP relaxation by more than the feasibility tolerance, then this cut is usually added to the relaxation (exceptions are for numerically dubious cuts). Note, that SCIP is configured such that MINLP constraints are enforced after the integrality constraints. Thus, as long as the relaxation solution is fractional (w.r.t.  $x_I$ ), cuts from nonlinear constraints are only added during the cut-and-price loop. By default, the latter does at most five rounds per branch-and-bound node.

<sup>18</sup>By default, a cut  $\langle a, x \rangle \leq \bar{a}$  is added during the cut-and-price loop if  $\langle a, \tilde{x} \rangle - \bar{a} \geq 10^{-4} \|a\|_\infty$ , where  $\tilde{x}$  is the solution of the relaxation that we aim to cut off.

## 7. A Constraint Integer Programming Approach to MINLP

In the following, we discuss the cut generation routines for the various types of nonlinear functions that are implemented in SCIP. Let  $\tilde{x}$  be a solution of the LP relaxation that we aim to separate.

### 7.5.1. General Nonlinear Functions

Assume that the right-hand-side of a constraint

$$\ell \leq \langle a, x \rangle + \sum_{j=1}^k h_j(x_{J_j}) \leq u, \quad (7.26)$$

where  $\ell, u \in \bar{\mathbb{R}}$ ,  $a \in \mathbb{R}^n$ ,  $J_j \subseteq [n]$ ,  $j \in [k]$ , is violated by  $\tilde{x}$ . Due to the convexity check from Section 7.3.3, SCIP may know that some  $h_j : \mathbb{R}^{|J_j|} \rightarrow \mathbb{R}$  are convex or concave on  $[\underline{x}, \bar{x}]$ . SCIP computes a cut by adding up linear underestimators for each  $h_j(\cdot)$ .

**Convex**  $h_j(\cdot)$ . If  $h_j(\cdot)$  is convex, a linear underestimator is obtained by linearization of  $h_j(\cdot)$  at  $\tilde{x}_{J_j}$ , see also Section 6.1.1.

**Concave**  $h_j(\cdot)$ . If  $h_j(\cdot)$  is concave and univariate ( $|J_j| = 1$ ), then the secant underestimator (6.10) is used.

If  $h_j(\cdot)$  is concave and multivariate, the characterization (6.9) is employed. Note, that by duality, (6.9) is equivalent to

$$h_j^e(x) = \sup \left\{ \langle \mu, x \rangle + \sigma : \langle \mu, x^i \rangle + \sigma \leq f(x^i), i \in [2^{|J_j|}] \right\} \quad (7.27)$$

where  $\{x^i : i \in [2^{|J_j|}]\}$  denotes the vertices of the box  $[\underline{x}_{J_j}, \bar{x}_{J_j}]$  (if bounded). Thus, solving the linear program (7.27) for  $x = \tilde{x}_{J_j}$  yields a linear function  $\langle \mu, x \rangle + \sigma$  that underestimates  $h_j(\cdot)$  on  $[\underline{x}_{J_j}, \bar{x}_{J_j}]$  and that takes the value of the convex envelope at  $x = \tilde{x}_{J_j}$ . No cut is generated if  $[\underline{x}, \bar{x}]$  is unbounded or  $h_j(\cdot)$  cannot be evaluated for some  $x^i$ ,  $i \in [2^{|J_j|}]$ . In such cases, the constraint need to be enforced by other means (bound tightening, branching).

**Indefinite**  $h_j(\cdot)$ . If  $h_j(\cdot)$  is neither convex nor concave, but continuously differentiable, and  $[\underline{x}, \bar{x}]$  is bounded, then SCIP can compute a linear underestimator of  $h_j(\cdot)$  by using interval arithmetic on the gradient of  $h_j(\cdot)$ .

Note, that by Taylor's theorem,

$$h_j(x_{J_j}) \geq h_j(\tilde{x}_{J_j}) + \min_{y \in [\underline{x}_{J_j}, \bar{x}_{J_j}]} \langle \nabla h(y), x_{J_j} - \tilde{x}_{J_j} \rangle \quad (x_{J_j} \in [\underline{x}_{J_j}, \bar{x}_{J_j}]). \quad (7.28)$$

Let  $[\underline{d}, \bar{d}]$  be such that  $\nabla h_j(x_{J_j}) \in [\underline{d}, \bar{d}]$  for all  $x_{J_j} \in [\underline{x}_{J_j}, \bar{x}_{J_j}]$ . Then (7.28) yields

$$h_j(x_{J_j}) \geq h_j(\tilde{x}_{J_j}) + \sum_{i \in J_j: x_i \geq \tilde{x}_i} \underline{d}_i(x_i - \tilde{x}_i) + \sum_{i \in J_j: x_i < \tilde{x}_i} \bar{d}_i(x_i - \tilde{x}_i)$$

By moving the reference point  $\tilde{x}_{J_j}$  to a corner of the box  $[\underline{x}, \bar{x}]$ , we can derive a linear underestimator. Thus, let  $\hat{x}_{J_j}$  and  $d_{J_j}$  be defined by

$$\hat{x}_i = \begin{cases} \underline{x}_i, & \text{if } \tilde{x}_i \leq (\underline{x}_i + \bar{x}_i)/2, \\ \bar{x}_i, & \text{otherwise,} \end{cases} \quad d_i = \begin{cases} \underline{d}_i, & \text{if } \hat{x}_i = \underline{x}_i, \\ \bar{d}_i, & \text{if } \hat{x}_i = \bar{x}_i, \end{cases} \quad (i \in J_j). \quad (7.29)$$

Then

$$h_j(x_{J_j}) \geq h_j(\hat{x}_{J_j}) + \langle d_{J_j}, x_{J_j} - \hat{x}_{J_j} \rangle \quad (7.30)$$

is a valid underestimator, which is used to derive *interval gradient cuts* [Nowak, 2005, Section 7.1.3]. A generalization of (7.30) by using interval slopes instead of interval gradients is discussed by Schichl and Neumaier [2005] and Gay [2010].

If  $[\underline{x}, \bar{x}]$  is unbounded and this results in infinite values in  $\hat{x}_{J_j}$  or  $d_{J_j}$ , then no cut is generated. The box  $[\underline{d}, \bar{d}]$  is computed in SCIP by calling the automatic differentiation methods for the computation of gradients [Griewank and Walther, 2008] in CPPAD<sup>19</sup> with the base data type changed from usual floating-point numbers to intervals.

Note, that the underestimator (7.30) can be very weak, even though it usually improves when the box  $[\underline{x}, \bar{x}]$  shrinks (by bound tightening or branching). Thus, SCIP avoids general indefinite functions  $h_j(\cdot)$  by reformulating the expression graph during presolve. This will be discussed in Section 7.6.1.

**Summing up.** Hence, if  $[\underline{x}_{J_j}, \bar{x}_{J_j}]$  is bounded for all  $j \in [k]$  with concave or indefinite  $h_j(\cdot)$ , then a valid cut for (7.26) is given by

$$\langle a, x \rangle + \sum_{j \in [k]: h_j(\cdot) \text{ convex}} \left( h_j(\tilde{x}_{J_j}) + \langle \nabla h_j(\tilde{x}_{J_j}), x_{J_j} - \tilde{x}_{J_j} \rangle \right) + \sum_{j \in [k]: h_j(\cdot) \text{ concave}} \left( \sigma^j + \langle \mu^j, x_{J_j} \rangle \right) + \sum_{j \in [k]: h_j(\cdot) \text{ indefinite}} \left( h_j(\hat{x}_{J_j}) + \langle d_{J_j}, x_{J_j} - \hat{x}_{J_j} \rangle \right) \leq u,$$

where  $\langle \mu^j, x_{J_j} \rangle + \sigma^j$  is a linear underestimator for  $h_j(\cdot)$  computed by (7.27) and  $d_{J_j}$  and  $\hat{x}_{J_j}$  are given by (7.29) and correspond to the interval gradient underestimator (7.30) of  $h_j(\cdot)$ . If  $\tilde{x}$  violates the left-hand-side of (7.26), an analog method is used.

### 7.5.2. Odd and Signed Power Functions

Consider the constraint

$$\ell \leq \text{sign}(x+b)|x+b|^a + cz \leq u, \quad (7.31)$$

where  $a > 1$ ,  $b, c \in \mathbb{R}$ ,  $\ell, u \in \bar{\mathbb{R}}$ . For  $a \in 2\mathbb{Z} + 1$  (odd power), (7.31) is equivalent to  $\ell \leq (x+b)^a + cz \leq u$ . Assume  $\tilde{x}$  violates the right-hand-side of constraint (7.31).

The convex envelope of  $x \mapsto \text{sign}(x+b)|x+b|^a$  on  $[\underline{x}, \bar{x}]$  is given by generalization of (6.15) from the case  $a \in 2\mathbb{Z} + 1$  to arbitrary  $a > 1$ : As in (6.15), the convex envelope for

<sup>19</sup><http://www.coin-or.org/CppAD>

## 7. A Constraint Integer Programming Approach to MINLP

exponent $a$	root of (7.33)
1.852	0.398217
2	$\sqrt{2} - 1 \approx 0.41421356237309504880$
3	$1/2 = 0.50000000000000000000$
4	0.56042566045031785945
5	0.60582958618826802099
6	0.64146546982884663257
7	0.67033204760309682774
8	0.69428385661425826738
9	0.71453772716733489700
10	0.73192937842370733350

Table 7.3.: Approximate root of polynomial (7.33) for exponents  $a \in \{1.852, 2, 3, \dots, 10\}$ .

the interesting case  $\underline{x} + b < 0 < \bar{x} + b$  is given by the secant between  $(\underline{x}, -|\underline{x} + b|^a)$  and  $(x^*, (x^* + b)^a)$  for  $x \in [\underline{x}, x^*]$  and by the function itself for  $x \geq x^*$ , where  $x^* > -b$  is such that the slope of the secant coincides with the gradient of the function at  $x^*$ , i.e.,

$$\frac{(x^* + b)^a + |\underline{x} + b|^a}{x^* - \underline{x}} = a(x^* + b)^{a-1} \quad (7.32)$$

W.l.o.g., let  $b = 0$ . Then (7.32) can be rewritten as

$$(-\underline{x})^a = (a - 1)(x^*)^a - a(x^*)^{a-1}\underline{x}.$$

Division by  $(-\underline{x})^a$  yields

$$1 = (a - 1) \left( \frac{x^*}{-\underline{x}} \right)^a + a \left( \frac{x^*}{-\underline{x}} \right)^{a-1}.$$

As in Liberti and Pantelides [2003], it is easily seen that

$$(a - 1)y^a + ay^{a-1} - 1 \quad (7.33)$$

has exactly one root in  $p \in [0, 1]$  by noting that it is strictly increasing in  $[0, 1]$  and takes values in  $[-1, 2a - 2]$ . The root  $p$  can easily be found numerically via Newton's method, which then yields  $x^* = -\underline{x}p$ . The value of  $p$  for a few exponents  $a$  is given in Table 7.3.

Finally, if  $\underline{x} \neq -\infty$ , then a linear underestimator of  $x \mapsto \text{sign}(x + b)|x + b|^a$  is obtained by linearization of the convex envelope in  $\tilde{x}$ , which yields the cut

$$\begin{aligned} -|\underline{x} + b|^a + \frac{(x^* + b)^a + |\underline{x} + b|^a}{x^* - \underline{x}}(x - \underline{x}) + cz &\leq u & \text{for } \tilde{x} + b \leq p\underline{x}, \\ (\tilde{x} + b)^a + a(\tilde{x} + b)^{a-1}(x - \tilde{x}) + cz &\leq u & \text{for } \tilde{x} + b \geq p\underline{x}. \end{aligned}$$

If the left-hand-side of (7.31) is violated, then a similar method is used to derive a cut.

### 7.5.3. Quadratic Functions

Consider a quadratic constraint

$$\langle x, Qx \rangle + \langle q, x \rangle + \bar{q} \leq 0 \quad (7.34)$$

with  $Q \in \mathbb{R}^{n \times n}$ ,  $q \in \mathbb{R}^n$ , and  $\bar{q} \in \mathbb{R}$ .

#### Convex Quadratic Functions

If the quadratic function in (7.34) is convex ( $Q \succeq 0$ ), SCIP generates the cut

$$\langle \tilde{x}, Q\tilde{x} \rangle + \langle 2Q\tilde{x}, x - \tilde{x} \rangle + \langle q, x \rangle + \bar{q} \leq 0 \quad (7.35)$$

obtained by simple linearization. In the special case that  $\langle x, Qx \rangle \equiv ax_i^2$  for some  $a > 0$  and  $i \in I$  with  $\tilde{x}_i \notin \mathbb{Z}$ , SCIP generates the cut

$$\bar{q} + \langle q, x \rangle + a(2\lfloor \tilde{x}_i \rfloor + 1)x_i - a\lfloor \tilde{x}_i \rfloor \lceil \tilde{x}_i \rceil \leq 0, \quad (7.36)$$

which is obtained by underestimating  $x_i \in \mathbb{Z} \mapsto x_i^2$  by the secant defined by the points  $(\lfloor \tilde{x}_i \rfloor, \lfloor \tilde{x}_i \rfloor^2)$  and  $(\lceil \tilde{x}_i \rceil, \lceil \tilde{x}_i \rceil^2)$ . Note, that the violation of (7.36) by  $\tilde{x}$  is larger than that of (7.35).

#### Nonconvex Quadratic Functions

For a violated nonconvex constraint, SCIP underestimates each term of  $\langle x, Qx \rangle$  separately, if none of the special structures discussed below are recognized. A convex term  $ax_i^2$  with  $a > 0$ ,  $i \in [n]$ , is underestimated as discussed above. For the concave case  $a < 0$ , the secant underestimator  $a(\underline{x}_i + \bar{x}_i)x_i - a\underline{x}_i\bar{x}_i$  is used (cf. (6.10)), if both  $\underline{x}_i$  and  $\bar{x}_i$  are finite. Otherwise, if  $\underline{x}_i = -\infty$  or  $\bar{x}_i = \infty$ , SCIP does not generate a cut. For a bilinear term  $ax_ix_j$  with  $a > 0$ , we utilize the McCormick underestimators (6.11),

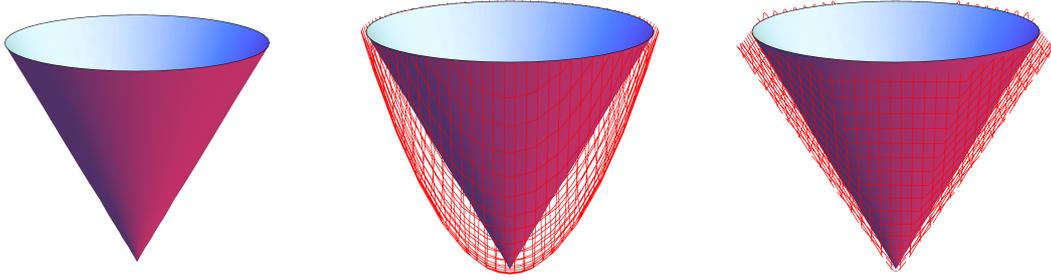
$$\begin{aligned} ax_ix_j &\geq a\underline{x}_i x_j + a\underline{x}_j x_i - a\underline{x}_i \underline{x}_j, \\ ax_ix_j &\geq a\bar{x}_i x_j + a\bar{x}_j x_i - a\bar{x}_i \bar{x}_j. \end{aligned}$$

If  $(\bar{x}_i - \underline{x}_i)\tilde{x}_j + (\bar{x}_j - \underline{x}_j)\tilde{x}_i \leq \bar{x}_i\bar{x}_j - \underline{x}_i\underline{x}_j$  and the bounds  $\underline{x}_i$  and  $\underline{x}_j$  are finite, the former is used for cut generation, otherwise the latter is used. If both  $\underline{x}_i$  or  $\underline{x}_j$  and  $\bar{x}_i$  or  $\bar{x}_j$  are infinite, SCIP does not generate a cut. Similar, for a bilinear term  $ax_ix_j$  with  $a < 0$ , the McCormick underestimators are

$$\begin{aligned} ax_ix_j &\geq a\bar{x}_i x_j + a\underline{x}_j x_i - a\bar{x}_i \underline{x}_j, \\ ax_ix_j &\geq a\underline{x}_i x_j + a\bar{x}_j x_i - a\underline{x}_i \bar{x}_j. \end{aligned}$$

If  $(\bar{x}_i - \underline{x}_i)\tilde{x}_j - (\bar{x}_j - \underline{x}_j)\tilde{x}_i \leq \bar{x}_i\underline{x}_j - \underline{x}_i\bar{x}_j$  and the bounds  $\bar{x}_i$  and  $\underline{x}_j$  are finite, the former is used for cut generation, otherwise the latter is used.

## 7. A Constraint Integer Programming Approach to MINLP



(a) Second-order cone      (b) Best possible outer-approximat- (c) Linear outer-approximation de-  
 tion for constraint (7.37) when      rived from reformulation (7.38)  
 structure is not exploited.      via eight cuts of the form (7.39).

Figure 7.8.: Second-order cone ( $\sqrt{x_1^2 + x_2^2} \leq x_3$ ) and its approximations depending on whether structure is exploited.

### Second-Order Cones

Quadratic constraints of the form

$$\gamma + \sum_{i \in [n-1]} (\alpha_i(x_i + \beta_i))^2 \leq (\alpha_n(x_n + \beta_n))^2, \quad (7.37)$$

where  $\alpha_i, \beta_i \in \mathbb{R}$ ,  $i \in [n]$ ,  $\gamma \in \mathbb{R}_+$ , and  $\underline{x}_n \geq -\beta_n$  or  $\bar{x}_n \leq -\beta_n$  are recognized as *second-order cone* constraints by SCIP. Figure 7.8a illustrates the set defined by (7.37). Assume w.l.o.g.  $\alpha_n \geq 0$  and  $\underline{x}_n \geq -\beta_n$ . Then (7.37) can be reformulated as

$$\sqrt{\gamma + \sum_{i \in [n-1]} (\alpha_i(x_i + \beta_i))^2} \leq \alpha_n(x_n + \beta_n). \quad (7.38)$$

Note, that (7.38) is a convex constraint. Thus, SCIP generates cuts by linearization of the function on the left-hand-side of (7.38) as it does for general convex nonlinear constraints. The resulting inequality to separate a point  $\tilde{x}$  that violates (7.37) is

$$\eta + \frac{1}{\eta} \sum_{i \in [n-1]} \alpha_i^2(\tilde{x}_i + \beta_i)(x_i - \tilde{x}_i) \leq \alpha_n(x_n + \beta_n), \quad (7.39)$$

where  $\eta := \gamma + \sqrt{\sum_{i \in [n-1]} (\alpha_i(\tilde{x}_i + \beta_i))^2}$ . Note, that since  $\tilde{x}$  violates (7.37), one has  $\eta > \alpha_n \tilde{x}_n + \beta_n \geq 0$ .

Figure 7.8c illustrates the outer-approximation obtained by the cuts (7.39). For comparison, Figure 7.8b shows the best-possible outer-approximation that SCIP would generate for (7.37) if it does not reformulate into the form (7.38). Note, that to further tighten the relaxation in Figure 7.8b, branching on the variable  $x_n$  would be necessary.

As suggested in Ben-Tal and Nemirovski [2001] and Glineur [2000], SCIP can also add an a priori linear outer-approximation that uses additional variables. However, since we did not observe computational benefits, this option is disabled by default. Further, it is known that certain quadratic constraints (7.34) where  $Q$  has only one negative eigenvalue can be written as second-order cone constraint (7.37) after a suitable variable transformation, see Mahajan and Munson [2010] for details. SCIP does not exploit this structure yet. Further, recognizing if nonlinear constraints describe a union of second-order cones (e.g., (7.37) defines a union of two second-order cones if  $\underline{x}_n < -\beta_n < \bar{x}_n$ ) could be exploited in a branch-and-bound algorithm, see Mahajan and Munson [2010].

### Factorable Quadratic Functions

SCIP checks, whether a quadratic constraint (7.34) can be written in a form

$$(\langle a^1, x \rangle + b^1)(\langle a^2, x \rangle + b^2) \leq u, \quad (7.40)$$

where  $a^1, a^2 \in \mathbb{R}^n$ ,  $b^1, b^2 \in \mathbb{R}$ , and  $u \in \mathbb{R} \setminus \{0\}$ . If a form (7.40) exists and  $\langle a^1, x \rangle + b^1$  or  $\langle a^2, x \rangle + b^2$  is bounded away from 0 on  $[\underline{x}, \bar{x}]$ , then (7.40) can be reformulated by dividing by  $\langle a^1, x \rangle + b^1$  or  $\langle a^2, x \rangle + b^2$ . Assume, w.l.o.g., that  $\langle a^1, x \rangle + b^1 > 0$  on  $[\underline{x}, \bar{x}]$  (if it is negative, then multiply  $a^1, a^2, b^1, b^2$  by  $-1$ ). Then (7.40) can equivalently be written as

$$\langle a^2, x \rangle + b^2 - \frac{u}{\langle a^1, x \rangle + b^1} \leq 0 \quad (7.41)$$

Since  $y \mapsto \frac{1}{y}$  is convex for  $y > 0$ ,  $-\frac{u}{\langle a^1, x \rangle + b^1}$  is convex for  $u < 0$  and concave for  $u > 0$ .

For  $u < 0$ , we have thus found a description of the (convex) set defined by (7.34) via a convex function. Thus, linearization yields the cut

$$\langle a^2, x \rangle + b^2 - \frac{u}{\langle a^1, \tilde{x} \rangle + b^1} + \frac{u}{(\langle a^1, \tilde{x} \rangle + b^1)^2} \langle a^1, x - \tilde{x} \rangle \leq 0. \quad (7.42)$$

As for second-order cone constraints, recognizing a factorable form (7.40) with  $u < 0$  has the advantage that SCIP can exploit the convexity of the set defined by the quadratic constraint for the construction of a linear outer-approximation, see also Figure 7.9.

To find the form (7.40), Sylvester's law of inertia can be utilized. SCIP computes eigenvalues and an orthonormal eigenvector basis of the matrix

$$\hat{Q} := \begin{pmatrix} Q & q/2 \\ q/2 & 0 \end{pmatrix}.$$

Note, that  $\langle x, Qx \rangle + \langle q, x \rangle = \langle \hat{x}, \hat{Q}\hat{x} \rangle$ , where  $\hat{x} := (x, 1)$  is the vector  $x$  extended by an additional entry 1. If  $\hat{Q}$  has exactly one positive eigenvalue  $\sigma_1^2$  with eigenvector  $v^1$  and one negative eigenvalue  $-\sigma_2^2$  with eigenvector  $v^2$ , then

$$\langle \hat{x}, \hat{Q}\hat{x} \rangle = \sigma_1^2 \langle v^1, \hat{x} \rangle^2 - \sigma_2^2 \langle v^2, \hat{x} \rangle^2 = (\sigma_1 \langle v^1, \hat{x} \rangle - \sigma_2 \langle v^2, \hat{x} \rangle)(\sigma_1 \langle v^1, \hat{x} \rangle + \sigma_2 \langle v^2, \hat{x} \rangle).$$

## 7. A Constraint Integer Programming Approach to MINLP

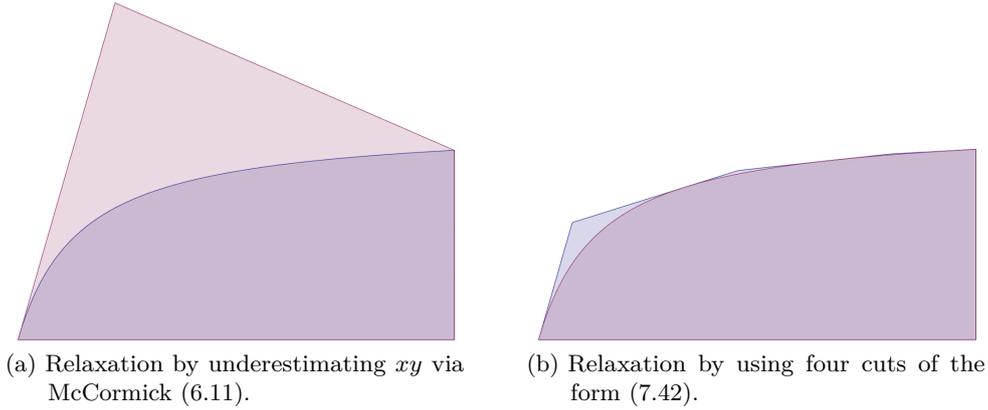


Figure 7.9.: Feasible region of quadratic constraint  $xy \leq -\frac{1}{4}$  for  $x \in [\frac{1}{8}, 1]$  and  $y \in [-2, 2]$  and overestimate by linear relaxation without or with exploiting structure.

Thus, as in Amato and Mensch [1971] and Cottle [1975], the desired form is given by

$$\langle x, Qx \rangle + \langle q, x \rangle = (\langle \sigma_1 v_{[n]}^1 - \sigma_2 v_{[n]}^2, x \rangle + \sigma_1 v_{n+1}^1 - \sigma_2 v_{n+1}^2) (\langle \sigma_1 v_{[n]}^1 + \sigma_2 v_{[n]}^2, x \rangle + \sigma_1 v_{n+1}^1 + \sigma_2 v_{n+1}^2)$$

If linear variables exist in constraint (7.34), i.e.,  $i \in [n]$  with  $q_i \neq 0$  but  $Q_{i,j} = 0$  for all  $j \in [n]$ , then a reformulation into the form (7.40) does not exist<sup>20</sup>. However, by eliminating the rows and columns from  $\hat{Q}$  that correspond to linear variables and treating them separately, one may find a reformulation of (7.34) into a form

$$(\langle a^1, x \rangle + b^1) (\langle a^2, x \rangle + b^2) \leq u + \langle c, x \rangle, \quad (7.43)$$

where  $c \in \mathbb{R}^n$ . For  $u + \langle c, x \rangle < 0$ , similar to (7.42), a linear underestimator of  $\langle a^1, x \rangle + b^1 - \frac{u + \langle c, \hat{x} \rangle}{\langle a^2, x \rangle + b^2}$  can be derived for fixed linear variables ( $\langle c, x \rangle$  fixed to  $\langle c, \hat{x} \rangle$ ). Next, Belotti et al. [2011] have shown how to lift such an underestimator to be valid for all  $\hat{x} \in [x, \bar{x}]$ . SCIP uses COUENNE's implementation of these so-called *lifted tangent inequalities*.

## 7.6. Reformulation

### 7.6.1. Reformulating the Expression Graph

Nonlinear constraints that are given neither by a sum of convex, concave, power, nor quadratic functions are reformulated by SCIP to obey one of these forms. The aim is to

<sup>20</sup>For a matrix  $\begin{pmatrix} A & 0 & 0 \\ 0 & 0 & a \\ 0 & a^\top & 0 \end{pmatrix}$  with nonzero matrix  $A$  and nonzero vector  $a$ , at least one nonzero eigenvalue

is contributed by  $A$  and at least one negative and one positive eigenvalue is contributed by  $\begin{pmatrix} 0 & a \\ a^\top & 0 \end{pmatrix}$ .

reformulate the MINLP into a form such that for all nonlinear constraints, methods to generate linear underestimators exist (other than the one that uses interval gradients). The reformulated MINLP has a form similar to the standard form (6.16), but also allows for (sums of) multivariate quadratic, convex, and concave functions.

The reformulation routine inspects the expression graph and turns vertices that correspond to certain subexpressions into sinks by adding a new variable and constraint. That is, for a vertex  $v$  of the expression graph with  $o(v) = (w_1, \dots, w_m; f)$  associated subexpression  $f(g_1(x), \dots, g_m(x))$ , the following happens (see Algorithm 7.3 for details):

- if  $f(g(x))$  is known to be convex or concave (cf. Section 7.3.3), do nothing,
- if the function  $y \mapsto f(y)$  is quadratic or convex or concave or a (signed) power function, ensure that all children correspond to linear expressions by adding new auxiliary variables  $z_j$  and new constraints  $z_j = g_j(x)$ ,  $j \in [m]$ ,
- if  $f(g(x)) = \alpha \prod_{j \in [m]} g_j(x)^{\beta_j}$ , add auxiliary variables  $z_1$  and  $z_2$ , new constraints

$$z_1 = \prod_{j \in [m/2]} g_j(x)^{\beta_j}, \quad z_2 = \prod_{j \in [m/2]+1:m} g_j(x)^{\beta_j},$$

and associate  $v$  with the expression  $\alpha z_1 z_2$ ,

- if  $f(g(x)) = \sum_{j \in [k]} \alpha_j g_1(x)^{\beta_{j,1}} \dots g_m(x)^{\beta_{j,m}}$ , add auxiliary variables  $z_j$ , new constraints  $z_j = g_1(x)^{\beta_{j,1}} \dots g_m(x)^{\beta_{j,m}}$ ,  $j \in [k]$ , and associate  $v$  with expr.  $\sum_{j \in [k]} \alpha_j z_j$ .

The reformulation algorithm has been implemented in a way that other constraint handler can interact with it by registering callback functions that are called for each vertex of the expression graph and can reformulate the associated expression, e.g., by substituting the vertex with a source for a new auxiliary variable and adding an own specialized nonlinear constraint. For example, SCIP's constraint handler for **and** constraints  $r = \ell_1 \wedge \dots \wedge \ell_n$ , where  $\ell_j$  is a literal as defined in Definition 7.2, replaces vertices that correspond to a product of at least three binary variables ( $\prod_{i \in B} x_i$ ,  $B \subseteq I$ ,  $\underline{x}_i = 0$ ,  $\bar{x}_i = 1$ ,  $i \in B$ ) by a new auxiliary variable  $r$  and a new **and**-constraint  $r = \bigwedge_{i \in B} x_i$ .

**Example 7.13.** Recall Example 7.7 with the constraints

$$\begin{aligned} 420.169\sqrt{900 + x_1^2} - x_3 x_1 x_2 &= 0 \\ \frac{2960.88 + 18505.5x_2^2}{7200 + x_1^2} - x_3 &\geq 0 \\ x_{\text{obj}} - 0.047x_2\sqrt{900 + x_1^2} &\geq 0 \end{aligned}$$

and the simplified expression graph in Figure 7.3. By reformulation, the following equivalent<sup>21</sup>

<sup>21</sup>Equivalence in the sense that the projection of the feasible set onto the original variables is the same.

**Algorithm 7.3:** Reformulation of expression graph

---

```

input : expression graph  $G = (V, E, o)$ , box  $[x, \bar{x}]$ 
for  $d = 1$  to  $d_{\max}$  do
  foreach  $v \in V$  with  $d(v) = d$  do
    let  $o(v) = (w_1, \dots, w_m; f)$ ;
    if  $\text{convex}(v) \vee \text{concave}(v)$  then continue;
    if  $y \mapsto f(y)$  is convex or concave or quadratic or (signed) power then
      /* ensure that all children correspond to linear expr. */
      foreach  $w \in c(v) : \neg(\text{convex}(w) \wedge \text{concave}(w))$  do
        add new auxiliary variable  $z$ ;
        add new constraints  $z = g(x)$  where  $g(x)$  is associated with vertex  $w$ ;
         $V \leftarrow V \cup \{u\}$ ;  $o(u) \leftarrow (\emptyset; z)$ ;
         $E \leftarrow (E \setminus \{(w, v)\}) \cup \{(u, v)\}$ ;
      end
    else if  $f = \alpha \prod_{j \in [m]} y_j^{\beta_j}$  then /* split products */
      add new auxiliary variables  $z_1$  and  $z_2$ ;
       $V \leftarrow V \cup \{u_1, u_2\}$ ;  $o(u_1) \leftarrow (\emptyset; z_1)$ ,  $o(u_2) \leftarrow (\emptyset; z_2)$ ;
       $E \leftarrow (E \setminus \{(w_j, v) : j \in [m]\}) \cup \{(u_1, v), (u_2, v)\}$ ;
       $o(v) \leftarrow (u_1, u_2; y \mapsto \alpha y_1 y_2)$ ;
      add new constraints  $z_1 = g_1(x)$  and  $z_2 = g_2(x)$  where  $g_1(x)$  and  $g_2(x)$  are associated with the new sinks  $p_1$  and  $p_2$ :  $V \leftarrow V \cup \{p_1, p_2\}$ ;
       $o(p_1) \leftarrow (w_1, \dots, w_{\lfloor m/2 \rfloor}; y \mapsto \prod_{j \in [\lfloor m/2 \rfloor]} y_j^{\beta_j})$ ;
       $o(p_2) \leftarrow (w_{\lfloor m/2 \rfloor + 1}, \dots, w_m; y \mapsto \prod_{j \in [\lfloor m/2 \rfloor + 1 : m]} y_j^{\beta_j})$ ;
       $E \leftarrow E \cup \{(w_j, p_1) : j \in [\lfloor m/2 \rfloor]\} \cup \{(w_j, p_2) : j \in [\lfloor m/2 \rfloor + 1 : m]\}$ ;
    else ( $f = \sum_{j \in [k]} \alpha_j y_1^{\beta_{j,1}} \dots y_m^{\beta_{j,m}}$ ) /* split sums */
      add new auxiliary variables  $z_j$ ,  $j \in [k]$ ;
       $V \leftarrow V \cup \{u_j : j \in [k]\}$ ;  $o(u_j) \leftarrow (\emptyset; z_j)$ ,  $j \in [k]$ ;
       $E \leftarrow (E \setminus \{(w_j, v) : j \in [m]\}) \cup \{(u_j, v) : j \in [k]\}$ ;
       $o(v) \leftarrow (u_1, \dots, u_k; y \mapsto \sum_{j \in [k]} \alpha_j y_j)$ ;
      add new constraints  $z_j = g_j(x)$ , where  $g_j(x)$  is associated with the new sink  $p_j$ ,  $j \in [k]$ :  $V \leftarrow V \cup \{p_j : j \in [k]\}$ ;
       $o(p_j) \leftarrow (w_1, \dots, w_m; y \mapsto y_1^{\beta_{j,1}} \dots y_m^{\beta_{j,m}})$ ,  $j \in [k]$ ;
       $E \leftarrow E \cup \{(w_{j'}, p_j) : j' \in [m], j \in [k]\}$ ;
    end
  end
end

```

---

set of constraints is constructed:

$$\begin{array}{lll}
 900 + x_1^2 = z_1 & 7200 + x_1^2 = z_4 & 420.169\sqrt{z_1} - x_3z_5 = 0 \\
 -z_3 + z_2z_4 = 0 & x_1x_2 = z_5 & z_2 - x_3 \geq 0 \\
 2960.88 + 18505.5x_2^2 = z_3 & \sqrt{z_1} = z_6 & 0.047x_2z_6 \leq x_{\text{obj}}
 \end{array}$$

Figure 7.10 shows the corresponding expression graph.

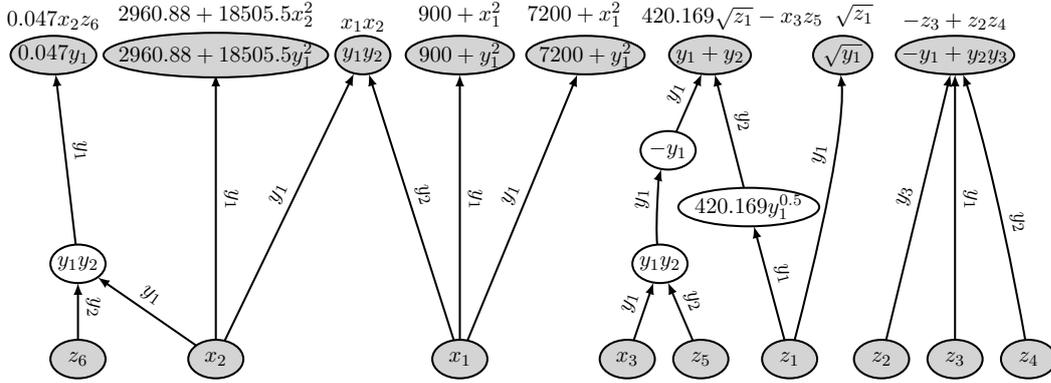


Figure 7.10.: Reformulation of expression graph from Figure 7.3.

### 7.6.2. Quadratic Constraints with Binary Variables

Binary variables in quadratic constraints (7.34) experience a special treatment by SCIP. Obviously, the square of a binary variable can be replaced by the binary variable itself. Further, if a quadratic term contains a product of a binary variable with a linear term,

$$x_i \sum_{j \in [n]} Q_{i,j} x_j,$$

where  $x_i$  is a binary variable, and all  $x_j$  with  $Q_{i,j} \neq 0$  have finite bounds, then this product is replaced by a new variable  $z \in \mathbb{R}$  and the linear constraints

$$\begin{aligned}
 \underline{M}_1 x_i &\leq z \leq \overline{M}_1 x_i \\
 \sum_{j \in [n]} Q_{i,j} x_j - \overline{M}_0 (1 - x_i) &\leq z \leq \sum_{j \in [n]} Q_{i,j} x_j - \underline{M}_0 (1 - x_i),
 \end{aligned}$$

where  $[\underline{M}_0, \overline{M}_0]$  and  $[\underline{M}_1, \overline{M}_1]$  are bounds on  $\sum_{j \in [n]} Q_{i,j} x_j$  for the cases  $x_i = 0$  and  $x_i = 1$ , respectively. A simple choice is

$$\begin{aligned}
 \underline{M}_0 = \underline{M}_1 &= \sum_{j \in [n]: Q_{i,j} > 0} Q_{i,j} \underline{x}_j + \sum_{j \in [n]: Q_{i,j} < 0} Q_{i,j} \overline{x}_j, \\
 \overline{M}_0 = \overline{M}_1 &= \sum_{j \in [n]: Q_{i,j} > 0} Q_{i,j} \overline{x}_j + \sum_{j \in [n]: Q_{i,j} < 0} Q_{i,j} \underline{x}_j.
 \end{aligned}$$

## 7. A Constraint Integer Programming Approach to MINLP

Possibly tighter values are computed by taking variable bounds of the form  $x_i = a \rightarrow (\underline{b} \leq x_j \leq \bar{b})$  into account ( $a \in \{0, 1\}$ ). SCIP stores these implications in a central data structure [Achterberg, 2007, Section 3.3], which is then checked during reformulation.

If all variables in the product  $x_i \sum_{j \in [n]} Q_{i,j} x_j$  are of integral type, i.e.,  $\{j : Q_{i,j} \neq 0\} \subseteq I$ , and also all coefficients  $Q_{i,j}$  are integral, then also the auxiliary variable  $z$  can take only integral values. Thus, SCIP marks  $z$  to be an *implicit integral* variable, which can be advantageous for bound tightening and cut generation.

A stronger linear relaxation is usually obtained by replacing each product  $x_i x_j$  with a new variable. However, this comes at the costs of a highly increased number of variables and constraints. So far, we have not recognized a computational benefit when considering each product separately. Thus, by default, as few new variables as possible are introduced.

The reformulation of products into linear inequalities sometimes leads to a complete linearization of a MIQCP. The problem is then completely represented by the LP relaxation plus the integrality requirements, which is useful for heuristics that work on the LP relaxation itself. Further, the added linear constraints can participate in the constructing of cutting planes that separate fractional solutions from the LP relaxation. Finally, linear constraints are already handled very efficiently in SCIP, while the support for quadratic constraints has been added just recently.

A disadvantage of the reformulation is the loss of structure due to the addition of new variables and constraints and the general problem of loose relaxations when using big- $M$  constraints, i.e., the values  $\underline{M}_{0,1}$  and  $\overline{M}_{0,1}$  are not updated during the solving process, even though the bounds that originally defined these values may be tightened. It thus may be promising to use SCIP's constraint handler for indicator constraints to model the constraints  $x_i = 0 \rightarrow z = 0$  and  $x_i = 1 \rightarrow z = \sum_{j \in [n]} Q_{i,j} x_j$ .

### 7.6.3. Quadratic Complementarity Constraints

A quadratic constraint

$$Q_{i,j} x_i x_j + q_i x + q_j y_j + \frac{q_i q_j}{Q_{i,j}^2} = 0 \quad (7.44)$$

is equivalent to

$$(x_i - a)(x_j - b) = 0 \quad \text{with} \quad a = -\frac{q_j}{Q_{i,j}} \quad \text{and} \quad b = -\frac{q_i}{Q_{i,j}}. \quad (7.45)$$

Thus, (7.44) can also be written as  $(x_i = a) \vee (x_j = b)$ , which yields the following conjunction of *bound disjunction constraints*,

$$(x_i \leq a \vee x_j \leq b) \wedge (x_i \leq a \vee x_j \geq b) \wedge (x_i \geq a \vee x_j \leq b) \wedge (x_i \geq a \vee x_j \geq b), \quad (7.46)$$

which may be shortened by considering the bounds on  $x_i$  and  $x_j$ .

SCIP reformulates a quadratic complementarity constraint (7.45) into the form (7.46) and replaces it by (at most 4) bound disjunction constraints. The idea is that the specialized bound disjunction constraint handler can handle these kinds of constraints more efficiently than the constraint handler for general quadratic constraints does, e.g., the

bound disjunction constraint handler contributes to SCIP's conflict analysis [Achterberg, 2007, Chapter 11], while this feature is not implemented for quadratic constraints yet.

Similar reformulations are done for quadratic constraints that can be written as

$$(x_i - a)(x_j - b) \geq 0 \quad \text{or} \quad (x_i - a)(x_j - b) \leq 0.$$

#### 7.6.4. Signed Square Functions

Recall constraints of the form (7.31), which are handled by a specialized constraint handler in SCIP. For pairs of (7.31) with  $\ell = u$  and  $a = 2$ , a special presolving is applied.

Assume, two constraints

$$\text{sign}(x + b_1)(x + b_1)^2 + c_1 z = u_1 \tag{7.47a}$$

$$\text{sign}(x + b_2)(x + b_2)^2 + c_2 z = u_2 \tag{7.47b}$$

with  $c_1 \neq 0$ ,  $c_2 \neq 0$ , and  $c_1 u_2 = c_2 u_1$  are given.

Subtracting  $c_1(7.47b)$  from  $c_2(7.47a)$  yields

$$c_2 \text{sign}(x + b_1)(x + b_1)^2 = c_1 \text{sign}(x + b_2)(x + b_2)^2 \tag{7.48}$$

If  $c_1 = c_2$ , then also  $u_1 = u_2$ . If, additionally,  $b_1 = b_2$ , then either (7.47a) or (7.47b) can be removed. If, however,  $b_1 \neq b_2$ , then (7.48) (and thus the system (7.47)) has no solution due to monotonicity of  $\text{sign}(x)|x|$ .

If  $c_1 \neq c_2$ , then (7.48) can be shown to have a unique solution, which is

$$x = \frac{b_2 - b_1}{\text{sign}(c_2/c_1)\sqrt{|c_2/c_1|} - 1} - b_1.$$

As a consequence,

$$z = \frac{b_1 - \text{sign}(x + b_1)(x + b_1)^2}{c_1}$$

and both constraints (7.47a) and (7.47b) can be removed from the problem.

## 7.7. Branching

We consider each unfixed variable  $x_i$  that appears in a nonconvex nonlinear constraint that is violated by the current relaxation solution  $\tilde{x}$  as a branching candidate. More specifically, for violated general nonlinear constraints (7.26), we consider variables  $x_i$  with  $i \in J_j$  for concave or indefinite  $h_j(\cdot)$ ,  $j \in [k]$ ; for violated odd or signed power constraints (7.31), we consider the variable  $x$ ; for violated nonconvex quadratic constraints (7.34), we consider variables in bilinear terms<sup>22</sup> and in concave terms  $ax_i^2$  with  $a < 0$ .

<sup>22</sup>For bilinear terms  $x_i x_j$  that involve binary variables, we first consider only the binary variable for branching, since it linearizes this term in both child nodes. If a bilinear term involves unbounded variables, then we first consider only the unbounded variables for branching.

## 7. A Constraint Integer Programming Approach to MINLP

Let  $\hat{x}_i \in (\underline{x}_i, \bar{x}_i)$  be the potential branching point for branching on  $x_i$ . Usually, we choose  $\hat{x}_i = \tilde{x}_i$ . If, however,  $\tilde{x}_i$  is very close to one of the bounds,  $\hat{x}_i$  is shifted inwards the interval. Thus, for  $\underline{x}_i, \bar{x}_i \in \mathbb{R}$ , we let  $\hat{x}_i := \min\{\max\{\tilde{x}_i, \lambda\underline{x}_i + (1-\lambda)\bar{x}_i\}, \lambda\bar{x}_i + (1-\lambda)\underline{x}_i\}$ , where the parameter  $\lambda$  is set to 0.2 in our experiments. Further, for power constraints (7.31) we usually choose  $\hat{x} = -b$  as branching point if  $\underline{x} < -b < \bar{x}$ .

As suggested in Belotti et al. [2009], we select the branching variable w.r.t. its pseudo cost, cf. Section 6.1.4. The pseudo costs are used to estimate the objective change in the LP relaxation when branching downwards and upwards on a particular variable. In classical pseudo cost branching for integer variables, the distances of  $\tilde{x}_i$  to the nearest integers are used as multipliers of the pseudo cost. For continuous variables, we use another measure that is similar to “rb-int-br” in Belotti et al. [2009] (last measure suggested in “Pseudo Cost Branching” part of Section 6.1.4): the distance of  $\hat{x}_i$  to the bounds  $\underline{x}_i$  and  $\bar{x}_i$  for a variable  $x_i$ . If the domain of  $x_i$  is unbounded, then the “infeasibility of the variable  $x_i$ ” will be used as factor, see also Section 6.1.4. The estimates for down- and upwards branching are combined by multiplication, see (6.21).

### 7.8. Primal Heuristics

When solving MINLPs, SCIP still makes use of all its default MIP primal heuristics [Berthold, 2006]. Most of these heuristics aim at finding good integer and LP feasible solutions starting from an optimum of the LP relaxation or the incumbent solution. Further, some heuristics that target especially on MINLP or on CIP in general are available and reviewed shortly in the following.

#### 7.8.1. NLP Local Search

There are several cases, where the MIP primal heuristics already find feasible solutions for the MINLP. However, the heuristics usually construct a point  $\hat{x}$  which is feasible for the MIP relaxation, i.e., the LP relaxation plus the integrality requirements, but violates some of the nonlinear constraints. Such a point may, nevertheless, provide useful information, since it can serve as starting point for a local search.

The NLP local search heuristic considers the space of continuous variables, i.e., it searches for a local optimum of the NLP obtained from the MINLP by fixing all integer variables to the values of  $\hat{x}$  and using  $\hat{x}$  as starting point for the NLP solver. Each feasible solution of this NLP is also a feasible solution of the MINLP. So far, the only available NLP solver in SCIP is IPOPT [Wächter and Biegler, 2006].

Before transferring the MINLP with fixed discrete variables to a NLP solver, the presolving is applied again, which may find additional reductions. For example, fixing a binary variable to zero often leads to fixations for some continuous variables, too.

To decide when to run the NLP local search heuristic, one has to compromise between frequently calling the heuristic in order to evaluate the fixations found by SCIP’s MIP heuristics or branch-and-bound search and rarely calling the heuristic in order to save computation time. By default, the heuristic uses a rule that is similar to the one applied in other large neighborhood search heuristics in SCIP. That is, the heuristic runs during

root node processing (provided a starting point is available) and every few hundred nodes during the branch-and-bound search. The exact decisions on whether to run the heuristic at a specific node and which limit on the number of iterations is given to the NLP solver are based on the number of iterations spend in previous calls, the number of solutions found by the heuristic so far, the number of nodes that SCIP has processed since the heuristic was called the last time, and, of course, the availability of a starting point.

### 7.8.2. Undercover Heuristic

Berthold and Gleixner [2009] have developed a MINLP heuristic in SCIP that is based on the observation that it often suffices to fix only a comparatively small number of variables such as to yield a MIP subproblem. Every solution of such a sub-MIP is then a feasible solution for the original MINLP. The variables to fix are chosen by solving a set covering problem, which aims at minimizing the number of variables to fix. The values for the fixed variables are initially taken from the solution of the LP or NLP relaxation or a known feasible solution of the MINLP. However, instead of fixing all variables at once, a diving alike procedure is applied that calls fast domain propagation routines after each fixing and allows for backtracking and choosing alternative fixing values in case that infeasibility is recognized for a particular partial fixing.

The sub-MIP is solved by a new SCIP instance, where limits on the number of nodes and the number of nodes without improvement in upper bound are used to restrict the effort spend for solving the subproblem. If a feasible solution for the sub-MIP (and thus also the MINLP) is found, SCIP tries to improve this point further by using it as a starting point in the NLP local search heuristic, cf. Section 7.8.1, i.e., fixing the integer variables to the values found by the undercover heuristic and optimizing w.r.t. the remaining continuous variables. We refer to Berthold and Gleixner [2009, 2012] for more details about this powerful heuristic. In default settings, the undercover heuristic is called only once during root node processing.

### 7.8.3. Sub-MINLP Heuristics

SCIP's large neighborhood search (LNS) heuristics for the MIP relaxation have recently been extended to work on the CIP itself [Berthold, Heinz, Pfetsch, and Vigerske, 2011]. The extension is generic in the sense that these heuristics do not treat specific constraint types in a special way, in contrast to the NLP local search heuristic that employs a NLP solver and the undercover heuristic that analysis the structure of the nonlinear constraints to select variables that have to be fixed. Since a MINLP is handled in SCIP as any other CIP, SCIP's LNS heuristics are now applicable to MINLPs, too.

The main idea of LNS heuristics is to restrict the search for "good" solutions to a neighborhood of specific points – usually close to optimal or feasible solutions. The hope is that such a restriction makes the subproblem much easier to solve, while still providing solutions of high quality. Since the restriction to a neighborhood of some solution again generates a CIP, SCIP is called recursively to handle the subproblems. Note that, during the solution process of the subproblem, the NLP local search heuristic may be used along

## 7. A Constraint Integer Programming Approach to MINLP

with the default SCIP heuristics (only the LNS heuristics discussed below are usually disabled for the solution of the subproblems).

Obviously, a good definition of the neighborhood is crucial for the success of a large neighborhood search heuristic. The neighborhood should contain high quality solutions, these solutions should be easy to find, and the neighborhood should be easy to process. Naturally, these three goals are conflicting in practice. The neighborhood is usually defined around a small set of starting points such as the a best known feasible solution or the optimal solution of an relaxation.

In the following, we list the LNS heuristics that are typically used in SCIP.

LOCAL BRANCHING [Fischetti and Lodi, 2003] measures the distance to the starting point in Manhattan norm on the integer variables and only considers solutions which are inside a  $k$ -neighborhood of the reference solution, where  $k$  is typically between 10 and 20.

*The relaxation induced neighborhood search* (RINS) [Danna et al., 2004] uses two starting points: The incumbent CIP solution (which is feasible, but may not have a small objective function value) and the optimum of the LP relaxation (which is not feasible, but has a small objective function value). RINS defines the neighborhood by fixing all integer variables that take the same value in both solutions.

In contrast to RINS, the *relaxation enforced neighborhood search* (RENS) [Berthold, 2007, 2012] does not require an incumbent solution. Thus, it can be used as a start heuristic. RENS fixes all integer variables that take an integral value in the optimal solution of the LP relaxation. For the remaining integer variables, the bounds get tightened to the two nearest integral values.

CROSSOVER is an improvement heuristic that is inspired by genetic algorithms [Berthold, 2006, Rothberg, 2007] and requires more than one feasible solution. For a set of feasible solutions, e.g., the three best found so far, it fixes variables that take identical values in all of them.

DINS [Ghosh, 2007] combines the ideas of RINS and LOCAL BRANCHING. It defines the neighborhood by introducing a distance function between the incumbent solution and the optimum of the LP relaxation. When applied during a branch-and-bound search, it further takes into account how variables change their values at different nodes of the tree.

## 8. Computational Study

This chapter presents a comprehensive study of the computational performance of the MINLP extensions to the framework SCIP, cf. Chapter 7. In Section 8.1, we present computation results for a mine production scheduling problem. This application constitutes the first use of SCIP's new MINLP features in a practical relevant application. Section 8.2 discusses a water network design application from the literature [Bragalli et al., 2012]. We have chosen this application because of its difficulty, the availability of test data, and the similarity (w.r.t. the type of nonlinearity) to applications in gas network design, the latter being one of the main motivations for the development of MINLP features in SCIP [Pfetsch et al., 2012]. Further, Section 8.3 evaluates the performance of SCIP on various libraries of MINLP instances that are commonly used to compare and test MINLP solvers. Finally, in Section 8.4 we study the computational impact of single components of the solver SCIP. That is, we compare how the performance of SCIP changes when specific features are used more or less aggressively.

All computational experiments were conducted within the GAMS environment [Brooke et al., 2012], because it supports general MINLPs, provides access to many state-of-the-art MINLP solvers (see also Table 6.2 at page 157), and due to the availability of libraries with test instances. The following solvers as available with GAMS 23.9.2 were used in our computational experiments:

- ALPHAECP 2.09.02 with CPLEX as MIP solver and CONOPT 3.15F [Drud, 1994] as NLP solver,
- BARON 11.3.0 with CPLEX as LP solver and MINOS 5.51 [Murtagh and Saunders, 2003] as NLP solver,
- BONMIN 1.6 with CLP as LP solver, IPOPT (using HSL MA27) as NLP solver, and CBC or CPLEX as MIP solver
- COUENNE 0.4 with CLP as LP solver and IPOPT (using MA27) as NLP solver
- CPLEX 12.4.0.1,
- DICOPT with CPLEX as MIP solver and CONOPT 3.15F as NLP solver,
- KNITRO 8.0.0,
- LINDOAPI (global solver) 7.0.1.497 with CONOPT 3.15F as NLP solver,
- MOSEK 6.0.0.137, and
- SBB with CONOPT 3.15F as NLP solver.

## 8. Computational Study

Further, we installed SCIP 2.1.2 linked with CPLEX 12.4.0.0, IPOPT 3.10 rev. 2107 (using HSL MA27), and CPPAD 20120101 rev. 2431 in GAMS (by using the interfaces available in the COIN-OR/GAMSLinks project<sup>1</sup>; compiled with gcc 4.6.2).

All computational results have been obtained under openSUSE Linux 12.1 64bit on a Dell PowerEdge M1000e blade with 48GB RAM and two Intel Xeon X5672 CPUs running at 3.20 GHz.

### 8.1. Open Pit Mine Production Scheduling with a Single Stockpile

The following presentation is taken from Bley, Gleixner, Koch, and Vigerske [2012b]. The computational results have been updated.

This section investigates the performance of SCIP and several state-of-the-art MINLP solvers on an open pit mine production scheduling problem with mixing constraints. We compare the solvers BARON, COUENNE, SBB, and SCIP to a problem-specific algorithm on two different MIQCP formulations. The computational results presented show that general-purpose solvers with no particular knowledge of problem structure are able to nearly match the performance of a hand-crafted algorithm.

#### 8.1.1. Open Pit Mine Production Scheduling with Stockpiles

In the following, we describe our model of the *open pit mine production scheduling problem* (OPMPSP) [Osanloo et al., 2008, Fricke, 2006, Boland et al., 2009]. Typically, the orebody of an open pit mine is discretized into small mining units called *blocks*, however, recently also continuous models have been suggested [Alvarez et al., 2011]. Block models of real-world open pit mines may consist of hundreds of thousands of blocks resulting in large-scale optimization problems. Groups of blocks are often aggregated to form larger mining units with possibly heterogeneous ore distribution, which we call *aggregates*. We assume such an aggregation of a block model is given a priori, with the set of aggregate indices  $\mathcal{N} = \{1, \dots, N\}$ .<sup>2</sup>

Moreover, we assume complete knowledge<sup>3</sup> about the contents of each aggregate  $i$ : First, its *rock* tonnage  $R_i$ , i.e. the amount of material which has to be extracted from the mine. Second, its *ore* tonnage  $O_i$ , i.e. the fraction of the rock tonnage sufficiently valuable to be processed further; in contrast, the non-ore fraction of each aggregate is discarded as waste immediately after its extraction from the mine. Finally, the tonnages  $A_i^1, \dots, A_i^K$  quantify a number of mineral *attributes* contained in the ore fraction. Attributes may be desirable, such as valuable mineral, or undesirable, such as chemical impurities.

The mining operations consist of several processes: First, rock is extracted from the pit, which we refer to as *mining*. Subsequently, the valuable part of the extracted material is refined further for sale, which is called *processing*; the remaining material not sufficiently

---

<sup>1</sup><https://projects.coin-or.org/GAMSLinks>

<sup>2</sup>Various techniques exist for aggregating blocks, e.g., the fundamental tree method [Ramazan, 2007].

<sup>3</sup>For an investigation of a stochastic model that deals with uncertain geology, see Boland et al. [2008].

## 8.1. Open Pit Mine Production Scheduling with a Single Stockpile

$N, \mathcal{N}$	number of aggregates and set of aggregate indices $\{1, \dots, N\}$ , respectively
$\mathcal{P}(i)$	set of immediate predecessors of aggregate $i$
$R_i, O_i$	rock and ore tonnage of aggregate $i$ , respectively [tonnes]
$A_i^k$	tonnage of attribute $k$ in aggregate $i$ ( $A_i$ for a single attribute) [tonnes]
$c^k$	sales price of attribute $k$ ( $c$ for a single attribute) [\$m/tonne]
$m, p$	mining and processing cost, respectively [\$m/tonne]
$T$	number of time periods
$\delta_t$	discount factor for time period $t$ (typically $1/(1+q)^t$ with interest rate $q \geq 0$ )
$M_t, P_t$	mining and processing capacity, respectively, for time period $t$ [tonnes]

Table 8.1.: List of notation

valuable is simply discarded as waste. In an intermediate stage between mining and processing, the valuable material may be stored on *stockpiles*. A stockpile can be imagined as “bucket” in which all material is immediately mixed and becomes homogeneous.

The lifespan of the mine is discretized into several, not necessarily homogeneous periods  $1, \dots, T$ . A feasible mine schedule determines, for each time period, the amount of rock which is to be mined from each aggregate, the fraction of the mined ore which is to be sent for processing or stockpiled, as well as the amount of ore sent from the stockpiles to the processing plant. *Resource constraints* restrict the amount of rock which may be mined and the amount of ore which may be processed during each time period  $t$  by limits  $M_t$  and  $P_t$ , respectively. *Precedence constraints* model the requirement that wall slopes are not too steep, ensuring the safety of the mine. Technically, these constraints demand that, before the mining of aggregate  $i$  may be started, a set of predecessor aggregates  $\mathcal{P}(i)$  must have been completely mined.

Long-term mining schedules have to be evaluated by their *net present value*: For each time period, we take the return from the processed and sold minerals minus the cost for mining and processing, multiplied by a decreasing discount factor to account for the time value of money. For homogeneous time periods and constant interest rate  $q \geq 0$  per time period, the profit made in time period  $t$  is multiplied by a factor of  $1/(1+q)^t$ . The objective is to find a feasible mine schedule with maximum net present value. Already without considering stockpiles, open pit mine production scheduling poses an NP-hard optimization problem, see, e.g., Gleixner [2008].

In the following, we focus on the special case of one attribute (some valuable mineral) and a single stockpile. A more general setting comprising multiple attributes, multiple stockpiles, or blending constraints in case of multiple attributes can easily be modeled by minor extensions and modifications, see Bley et al. [2012a]. Table 8.1 summarizes the notation introduced above.

### 8.1.2. MIQCP Formulations

In the following, we provide MIQCP formulations of the open pit mine production scheduling problem with one attribute (“metal”) and a single, infinite-capacity stockpile,

## 8. Computational Study

as presented in Bley et al. [2012a]: an aggregated “*basic*” formulation and an extended “*warehouse*” formulation. These formulations are theoretically equivalent. The results in Bley et al. [2012a], however, clearly speak in favor of the extended formulation, which is also confirmed by the computational study presented below.

### Basic Formulation

To track the various material flows, we define the following continuous decision variables for each aggregate  $i$  and time period  $t$ :

- $y_{i,t}^m \in [0, 1]$  as the fraction of aggregate  $i$  mined at time period  $t$ ,
- $y_{i,t}^p \in [0, 1]$  as the fraction of aggregate  $i$  mined at time period  $t$  and sent immediately for processing,
- $y_{i,t}^s \in [0, 1]$  as the fraction of aggregate  $i$  mined at time period  $t$  and sent to the stockpile,
- $o_t^s, a_t^s \geq 0$  as the absolute amount of ore respectively metal on the stockpile at time period  $t$ , and
- $o_t^p, a_t^p \geq 0$  as the absolute amount of ore respectively metal sent from the stockpile to the processing plant at time period  $t$ .

With this, the net present value of a mine schedule is calculated as

$$NPV(y^m, y^p, o^p, a^p) = \sum_{t=1}^T \delta_t \left[ c \left( a_t^p + \sum_{i=1}^N A_i y_{i,t}^p \right) - p \left( o_t^p + \sum_{i=1}^N O_i y_{i,t}^p \right) - m \sum_{i=1}^N R_i y_{i,t}^m \right].$$

In order to model the precedence constraints, we define the binary variables

- $x_{i,t} \in \{0, 1\}$  as equal to 1 if aggregate  $i$  is completely mined within time periods  $1, \dots, t$ .

A precedence-feasible extraction sequence is then ensured by the constraints

$$x_{i,t} \leq \sum_{\tau=1}^t y_{i,\tau}^m \quad (i \in \mathcal{N}, t \in [T]), \quad (8.1a)$$

$$\sum_{\tau=1}^t y_{i,\tau}^m \leq x_{j,t} \quad (i \in \mathcal{N}, j \in \mathcal{P}(i), t \in [T]). \quad (8.1b)$$

Additionally, we may, without altering the set of feasible solutions, require the sequence  $x_{i,1}, \dots, x_{i,T}$  to be nondecreasing for each aggregate  $i$ :

$$x_{i,t-1} \leq x_{i,t} \quad (i \in \mathcal{N}, t \in [2 : T]). \quad (8.1c)$$

Though redundant from a modeling point of view, these inequalities may help (or hinder) computationally, and have been used in the benchmark algorithm from Bley et al. [2012a].

### 8.1. Open Pit Mine Production Scheduling with a Single Stockpile

Conservation of the mined material is enforced by

$$\sum_{t=1}^T y_{i,t}^m \leq 1 \quad (i \in \mathcal{N}), \quad (8.1d)$$

$$y_{i,t}^p + y_{i,t}^s \leq y_{i,t}^m \quad (i \in \mathcal{N}, t \in [T]), \quad (8.1e)$$

i.e., for each aggregate, the amount sent for processing or to the stockpile in one time period must not exceed the total amount mined. (The difference  $y_{i,t}^m - y_{i,t}^p - y_{i,t}^s$  is discarded as waste.)

To model the state of the stockpile, we assume that material sent from the stockpile to processing is removed *at the beginning* of each time period, while material extracted from the pit (and not immediately processed) is stockpiled *at the end of each time period*.

Following this assumption, we must not send more material from the stockpile to processing than is available at the end of the previous period:

$$o_t^p \leq o_{t-1}^s \text{ and } a_t^p \leq a_{t-1}^s \quad (t \in [2 : T]). \quad (8.1f)$$

If we assume the stockpile to be empty at the start of the mining operations, we have  $o_1^p = a_1^p = 0$ . Now, book-keeping constraints for the amount of ore on the stockpile read

$$o_1^s = \sum_{i=1}^N O_i y_{i,1}^s, \quad (8.1g)$$

$$o_t^s = o_{t-1}^s - o_t^p + \sum_{i=1}^N O_i y_{i,t}^s, \quad (t \in [2 : T]), \quad (8.1h)$$

and analogously for the amount of metal on the stockpile

$$a_1^s = \sum_{i=1}^N A_i y_{i,1}^s, \quad (8.1i)$$

$$a_t^s = a_{t-1}^s - a_t^p + \sum_{i=1}^N A_i y_{i,t}^s m \quad (t \in [2 : T]). \quad (8.1j)$$

The resource constraints on mining and processing read

$$\sum_{i=1}^N R_i y_{i,t}^m \leq M_t \quad (t \in [T]), \quad (8.1k)$$

$$o_t^p + \sum_{i=1}^N O_i y_{i,t}^p \leq P_t \quad (t \in [T]). \quad (8.1l)$$

Last, we need to ensure that the ore-metal-ratio of the material sent from stockpile to processing equals the ore-metal-ratio in the stockpile itself. Otherwise, only the profitable metal could be sent to processing and for sale while the ore, only causing processing

## 8. Computational Study

costs, could remain in the stockpile. This involves the nonconvex quadratic mixing constraints  $a_t^p/o_t^p = a_{t-1}^s/o_{t-1}^s$  for  $t = 2, \dots, T$ . To avoid singularities, we reformulate these constraints as

$$a_t^p o_{t-1}^s = a_{t-1}^s o_t^p \quad (t \in [2 : T]). \quad (8.1m)$$

All in all, we obtain the *basic formulation*

$$\max \left\{ \begin{array}{l} \text{(8.1a) – (8.1m)} \\ x \in \{0, 1\}^{N \times T}, \\ y^m, y^p, y^s \in [0, 1]^{N \times T}, \\ o^s, a^s, o^p, a^p \geq 0. \end{array} \right. \quad (8.2)$$

Stockpiling capacities can be incorporated as upper bounds on  $o^s$  and  $a^s$ .

### Warehouse Formulation

In the basic formulation (8.2) the material of all aggregates sent from the pit to the stockpile is aggregated into variables  $o^s$  and  $a^s$ . Alternatively, we may track the material flows via the stockpile individually. Instead of variables  $o^s$ ,  $a^s$ ,  $o^p$ , and  $a^p$ , we then define for each aggregate  $i$  and time period  $t$ :

$z_{i,t}^p \in [0, 1]$  as the fraction of aggregate  $i$  sent from stockpile for processing at time period  $t$  and

$z_{i,t}^s \in [0, 1]$  as the fraction of aggregate  $i$  remaining in the stockpile at period  $t$ .

The net present values in terms of these variables is calculated as

$$NPV(y^m, y^p, z^p) = \sum_{t=1}^T \delta_t \left[ c \sum_{i=1}^N A_i (y_{i,t}^p + z_{i,t}^p) - p \sum_{i=1}^N O_i (y_{i,t}^p + z_{i,t}^p) - m \sum_{i=1}^N R_i y_{i,t}^m \right].$$

Constraints (8.1a) – (8.1e) remain unchanged. Starting with an empty stockpile gives  $z_{i,1}^s = z_{i,1}^p = 0$  for  $i \in \mathcal{N}$ . The stockpile balancing equations read

$$z_{i,t-1}^s + y_{i,t-1}^s = z_{i,t}^s + z_{i,t}^p \quad (i \in \mathcal{N}, t \in [2 : T]). \quad (8.3a)$$

The resource constraints on mining are the same as (8.1k), those for processing become

$$\sum_{i=1}^N O_i (y_{i,t}^p + z_{i,t}^p) \leq P_t \quad (t \in [T]). \quad (8.3b)$$

Instead of the mixing constraints (8.1m), now we demand that for each time period  $t$ , the fraction  $z_{i,t}^p/z_{i,t}^s$  is equal for each aggregate  $i$ . We obtain a better formulation by introducing, for each time period  $t$ , a new variable  $f_t \in [0, 1]$  called *out-fraction*, and requiring for all  $i \in \mathcal{N}$ , that  $z_{i,t}^p/(z_{i,t}^s + z_{i,t}^p) = f_t$ . To avoid zero denominators, we reformulate this as

$$z_{i,t}^p(1 - f_t) = z_{i,t}^s f_t \quad (i \in \mathcal{N}, t \in [2 : T]). \quad (8.3c)$$

This gives the *warehouse formulation*

$$\max \left\{ \begin{array}{l} \text{NPV}(y^m, y^p, z^p) : \\ y^m, y^p, y^s, z^p, z^s \in [0, 1]^{N \times T}, \\ f \in [0, 1]^T. \end{array} \right. \quad (8.4)$$

(8.1a) – (8.1e), (8.1k),  
(8.3a) – (8.3c),  
 $x \in \{0, 1\}^{N \times T}$ ,

Note that the basic formulation is an aggregated version of the warehouse formulation, and thus the LP relaxation (obtained by dropping integrality and mixing constraints) is tighter for the warehouse formulation.

### 8.1.3. Application-specific Benchmark Algorithm

As benchmark algorithm we used the application-specific approach developed by Bley et al. [2012a]. It features a branch-and-bound algorithm based on the linear MIP relaxation of the problem obtained by dropping the nonlinear mixing constraints (8.1m) for the basic and (8.3c) for the warehouse formulation, respectively. A specialized branching scheme is used to force the maximum violation of the nonlinear constraints arbitrarily close to zero.

As long as integer variables with fractional values are present, the algorithm branches on these to obtain an integer feasible solution. At nodes of the branch-and-bound tree with integer feasible relaxation solution, but violated nonlinear constraints, a specialized spatial branching is performed.

For the basic formulation, if constraint (8.1m) is violated for some time period  $t \in [2 : T]$ , then – since the current solution is LP optimal – the metal fraction taken out of the stockpile exceeds the ore fraction taken out. Hence, there is a ratio  $\phi$  with

$$\frac{o_t^p}{o_{t-1}^s} < \phi < \frac{a_t^p}{a_{t-1}^s}.$$

From this, two branches are created: one with

$$o_t^p \leq \phi o_{t-1}^s \text{ and } a_t^p \leq \phi a_{t-1}^s,$$

the other branch with

$$o_t^p \geq \phi o_{t-1}^s \text{ and } a_t^p \geq \phi a_{t-1}^s.$$

In both branches, the current solution is cut off and the possible maximum violation of the mixing constraint for time period  $t$  is reduced.

Similarly, for the warehouse formulation, suppose that constraint (8.3c) is violated for some time period  $t \in [T]$ . Then there exist at least two aggregates  $i_1, i_2 \in \mathcal{N}$  with different out-fractions, thus there is a ratio  $\phi$  with

$$\frac{z_{i_1,t}^p}{z_{i_1,t}^s + z_{i_1,t}^p} < \phi < \frac{z_{i_2,t}^p}{z_{i_2,t}^s + z_{i_2,t}^p}.$$

## 8. Computational Study

This gives rise to two branches, one with the additional constraints

$$(1 - \phi)z_{i,t}^p \leq \phi z_{i,t}^s \quad (i \in \mathcal{N}),$$

forcing the out-fractions of all aggregates in time period  $t$  below  $\phi$ , the other branch with

$$(1 - \phi)z_{i,t}^p \geq \phi z_{i,t}^s \quad (i \in \mathcal{N}),$$

forcing them above  $\phi$ . Again, in both branches, the current solution is cut off and the possible maximum violation of the mixing constraints for time period  $t$  is reduced.

Note, that the branches are created by adding multiple linear inequalities. Such an aggressive branching strategy is usually invalid for general MINLP solvers. In this special application it is feasible because of the additional knowledge, that the out-fractions of each aggregate must be equal for each time period.

The approach was implemented using the state-of-the-art MIP solver CPLEX with tuned parameter settings. Additionally, problem-specific heuristics as well as a variable fixation scheme and cutting planes derived from the underlying precedence constrained knapsack structure [Bley et al., 2010], which have been shown to improve the dual bound<sup>4</sup> for linear mine production scheduling models, are applied. To obtain good primal solutions, CPLEX’s rounding heuristics are extensively used and integer feasible solutions are post-processed by adapting mixing ratios heuristically. For further details, see Bley et al. [2012a]. We used the same implementation in our computational study.

### 8.1.4. Test Instances

Our industry partner BHP Billiton Pty. Ltd.<sup>5</sup> has provided us with realistic data from two open pit mines, see also Adams [1979]. Data set *Marvin* is based on a block model provided with the Whittle 4X mine planning software<sup>6</sup>, originally consisting of 8513 blocks which were aggregated to 85 so-called “panels”, i.e., single layers of blocks without block-to-block precedence relations. The lifespan of this mine, i.e., the time in which the profitable part of the orebody can be fully mined, is 15 years. Each panel has an average of 2.2 immediate predecessor aggregates. Data set *Dent* is based on the block model of a real-world open pit mine in Western Australia, originally consisting of 96821 blocks, which were aggregated to 125 panels. Each panel has an average of 2.0 immediate predecessor aggregates. The lifespan of this mine is 25 years.

The aggregations to panels, the cutoff grades (determining which blocks in each panel are immediately discarded as waste), and precedence relations between the panels were pre-computed by our industry partner. Scheduling periods are time periods of one year each with a discount rate of 10% per year. Realistic values for mining costs and processing profits as well as for mining and processing capacities per year were chosen by our industry partner.

---

<sup>4</sup>By the *dual bound* we denote the best lower bound on the optimal value of a minimization problem and the best upper bound on the optimal value of a maximization problem.

<sup>5</sup><http://www.bhpbilliton.com/>

<sup>6</sup>Gemcom Whittle, <http://www.gemcomsoftware.com/products/whittle/>

### 8.1. Open Pit Mine Production Scheduling with a Single Stockpile

	<i>Marvin</i>						<i>Dent</i>					
	# variables			# constraints			# variables			# constraints		
	total	bin	cont	total	linear	quad	total	bin	cont	total	linear	quad
(8.2)	5848	1445	4403	7598	7582	16	12600	3125	9475	15774	15750	24
(8.4)	8687	1445	7242	10404	9044	1360	18775	3125	15650	21900	18900	3000

Table 8.2.: Size of basic and warehouse formulations for instances *Marvin* and *Dent*.

We tested the performance of the general-purpose MINLP solvers on this data using the basic and the warehouse formulation – the same formulations on which the benchmark algorithm is based. Table 8.2 gives an overview over the size of these MIQCPs.

#### 8.1.5. Computational Results

For our computational experiments, we used the general purpose solvers BARON, COUENNE, SCIP, and SBB (cf. Section 6.2 and Chapter 7) and the application specific algorithm discussed in Section 8.1.3 with CPLEX 12.3.0.0. SCIP, BARON, and COUENNE are global solvers, i.e., they provide both a valid dual bound and can tentatively find a global optimal solution. All of them implement a spatial branch-and-bound algorithm that employs a LP relaxation. On the contrary, SBB implements a NLP-based branch-and-bound algorithm (cf. Algorithm 6.1). Since we run SBB with CONOPT as NLP solver, global optimality for our nonconvex MIQCPs is not guaranteed. However, we decided to include also SBB into our set of solvers, since NLP based branch-and-bound algorithms often obtain very good primal solutions also for nonconvex MINLPs.

For each run, we imposed a time limit of 10000 seconds, within which no solver was able to close the optimality gap. We report primal and dual bound and number of nodes processed after one hour and at the end of the time limit. We parenthesized dual bound and gap for solver SBB, since they may be invalid due to the problem’s nonconvexity.

#### Solver Settings

We run each solver with default and “hand-tuned” settings. The results for tuned settings are indicated by ‘\*’ in tables and figures.

**BARON.** In default settings, BARON spend most time for probing. Thus, for the tuned settings, we reduced the amount of probing to at most depth 10 of the branch-and-bound tree (option PEnd 10).

**Couenne.** Tuning COUENNE is difficult, since the solver does not print much statistic information about the time spend in single components. However, turning off expensive bound propagation techniques (options `aggressive_fbbt no` and `optimality_bt no`) increases the number of nodes that are processed by COUENNE within the time limit.

## 8. Computational Study

Without success, we also tried enabling single heuristics (feasibility pump, iterative rounding, local branching) or the creation of disjunctive cuts. Changing the LP solver to CPLEX increases the number of nodes processed by COUENNE, but also worsens the dual bound, maybe because the underlying MIP-solver CBC is tuned to work with CLP.

**SBB.** We generally switched off any node limit by setting the SBB option `memnodes` and the GAMS option `nodlim` to huge values. For the tuned settings, we enabled the option `acceptnonopt`, ensuring that SBB did not prune a node if the NLP subsolver did not conclude optimality or infeasibility of the node's relaxation. Additionally, we set the option `dfsstay 25`. Usually, SBB uses a mix of depth first search and best bound node selection rule. If a node is processed without creating subnodes, e.g., because an integer feasible solution is found, SBB jumps to the branch-and-bound node with best dual bound. With option `dfsstay n`, SBB is forced to continue searching the neighborhood of this node in a depth first search manner for `n` more nodes before applying the best bound node selection rule. This setting can help to improve previously found solutions.

**SCIP.** To find a good feasible solution early in the search, we changed the source for the fixing values in the undercover heuristic (cf. Section 7.8.2) to the NLP relaxation, which is therefor solved once in the root node. Additionally, we set the emphasis for the separators and heuristics in SCIP to aggressive, which leads to enabling the separation of MIP cuts also during branch-and-bound and the more excessive use of the large-neighborhood search heuristics that solve sub-MINLPs (cf. Section 7.8.3) – all of them found a solution every now and then. Finally, we changed the settings of the (time-consuming) propagator for variable bound constraints so that it runs only in depth 3%.

### Results for the Basic Formulation

Table 8.3 shows the performance of the application-specific benchmark algorithm from Section 8.1.3 and the general-purpose solvers when using the basic formulation. The application-specific algorithm yields the smallest primal-dual gaps among the LP relaxation based solvers, all of which, however, terminate with large dual bounds. Among the LP-based general-purpose solvers, BARON has best dual bounds, while it is outperformed by SBB and SCIP in terms of primal solutions. However, including the benchmark algorithm, all LP-based solvers perform rather unsatisfactory on this formulation.

In contrast, the tightest dual bounds clearly are obtained by SBB. Unfortunately, since it is not sure that the NLP subproblems have been solved to global optimality, these bounds cannot be trusted. Further, in default settings, CONOPT failed to find a local optimal solution for the root node relaxation, so that SBB stopped prematurely.

### Results for the Warehouse Formulation

Table 8.4 shows the results for the warehouse formulation. First, note that the LP-based approaches perform significantly better on this formulation. The application-specific algorithm shows excellent performance on the warehouse formulation. It produces the

### 8.1. Open Pit Mine Production Scheduling with a Single Stockpile

Instance Solver		after 3600 seconds			after 10000 seconds			
		primal	dual	nodes	primal	dual	nodes	gap[%]
<i>Marvin</i>	Benchmark	676.39	919.48	367000	676.39	919.43	1300592	35.93
	BARON	142.22	1216.85	438	142.22	1166.29	1066	720.09
	BARON*	622.18	1144.80	3430	641.40	1105.53	9577	72.36
	COUENNE	-	1625.54	100	-	1617.74	774	-
	COUENNE*	-	1631.24	2700	-	1626.27	9980	-
	SBB		failed			failed		
	SBB*	679.75	(706.11)	4780	689.86	(704.62)	13027	(2.14)
	SCIP	670.57	1581.66	180900	670.57	1576.93	514032	135.16
SCIP*	673.28	1580.58	193200	673.28	1574.52	591171	133.86	
<i>Dent</i>	Benchmark	47.32	54.07	45000	47.32	53.95	179735	13.71
	BARON	0.00	106.69	241	0.00	106.69	847	-
	BARON*	0.00	106.69	1297	0.00	106.69	4042	-
	COUENNE	-	113.94	0	-	113.94	0	-
	COUENNE*	-	113.54	400	-	112.56	2138	-
	SBB		failed			failed		
	SBB*	40.04	(50.27)	440	40.04	(50.20)	1117	(25.38)
	SCIP	44.21	110.43	37100	44.83	110.15	112798	145.70
SCIP*	48.77	110.77	30200	48.79	110.33	122551	126.16	

Table 8.3.: Results for the basic formulation (8.2)

best primal solutions and terminates with the smallest primal-dual gaps of 0.05% for instance *Marvin* and 0.51% for instance *Dent*. Nevertheless, the best primal solutions found by SCIP are only 0.39% (*Marvin*) and 2.23% (*Dent*) below the solution found by the benchmark algorithm.

The best dual bounds from SCIP are 1.26% and 0.14% away from the benchmark values for *Marvin* and *Dent*, respectively. Note that this difference is not only due to the handling of the nonlinear constraints. Also, the benchmark algorithm uses knowledge about the underlying precedence constrained knapsack structure of the linear constraints in order to fix binary variables and separate induced cover inequalities. This structure is not directly exploited by the general-purpose solvers.

In contrast to the LP relaxation based solvers, the NLP relaxation based approach of SBB appears to be less dependent on the change in formulation for instance *Marvin*. For the basic formulation, SBB computed a dual bound that is even slightly better than the one computed for the warehouse formulation. Notably, the *Dent* instance appears more challenging to SBB than *Marvin*, while for SCIP the situation is reversed. This

## 8. Computational Study

Instance Solver		after 3600 seconds			after 10000 seconds			
		primal	dual	nodes	primal	dual	nodes	gap[%]
<i>Marvin</i>	Benchmark	694.95	695.43	36580	694.97	695.33	140055	0.05
	BARON	450.73	716.83	485	450.73	716.24	1351	58.91
	BARON*	325.15	716.34	2549	458.65	715.99	7003	56.11
	COUENNE	-	719.47	0	-	719.47	2	-
	COUENNE*	-	716.63	1400	-	714.94	4539	-
	SBB	674.36	(706.53)	4260	674.36	(705.37)	11815	(4.60)
	SBB*	679.68	(706.50)	4400	680.28	(705.37)	11970	(3.69)
	SCIP	691.12	704.90	28400	691.92	704.14	91571	1.77
	SCIP*	692.26	705.26	18700	692.26	704.12	74708	1.71
<i>Dent</i>	Benchmark	48.80	49.15	3700	48.80	49.05	12826	0.51
	BARON	0.00	50.01	479	0.00	50.01	1855	-
	BARON*	0.00	50.01	650	0.00	50.01	2548	-
	COUENNE	-	50.34	0	-	50.34	0	-
	COUENNE*	-	50.08	100	-	50.00	814	-
	SBB	32.83	(50.19)	480	32.83	(50.10)	1276	(52.62)
	SBB*	32.83	(50.19)	500	32.83	(50.10)	1295	(52.62)
	SCIP	48.76	49.21	7800	48.76	49.12	33201	0.75
	SCIP*	48.71	49.34	600	48.71	49.16	20536	0.92

Table 8.4.: Results for the warehouse formulation (8.2)

is probably due to the increased problem size, which affects the solvability of the NLP relaxation in SBB more than the solvability of the LP relaxation in SCIP.

For both instances, SCIP computed better primal solutions than SBB on the warehouse formulation. For *Marvin*, SBB produced better solutions when using the option `dfsstay` 25. The forced depth first search after nodes with integer feasible solution appears to function as an improvement heuristic, compensating for SBB's lack of heuristics.

### Comparison of LP-based Solvers BARON, Couenne, and SCIP

For the basic formulation, the best dual bounds were found by BARON, while for the warehouse formulations SCIP computed tighter bounds. For all formulations, SCIP computed the best primal solutions and terminated with the smallest gaps.

Figure 8.1 compares the progress of the primal and dual bounds from the start to the time limit of 10000 seconds for all three solvers. It can be seen that even with tuned settings BARON and COUENNE spent a significant amount of time in presolving for

### 8.1. Open Pit Mine Production Scheduling with a Single Stockpile

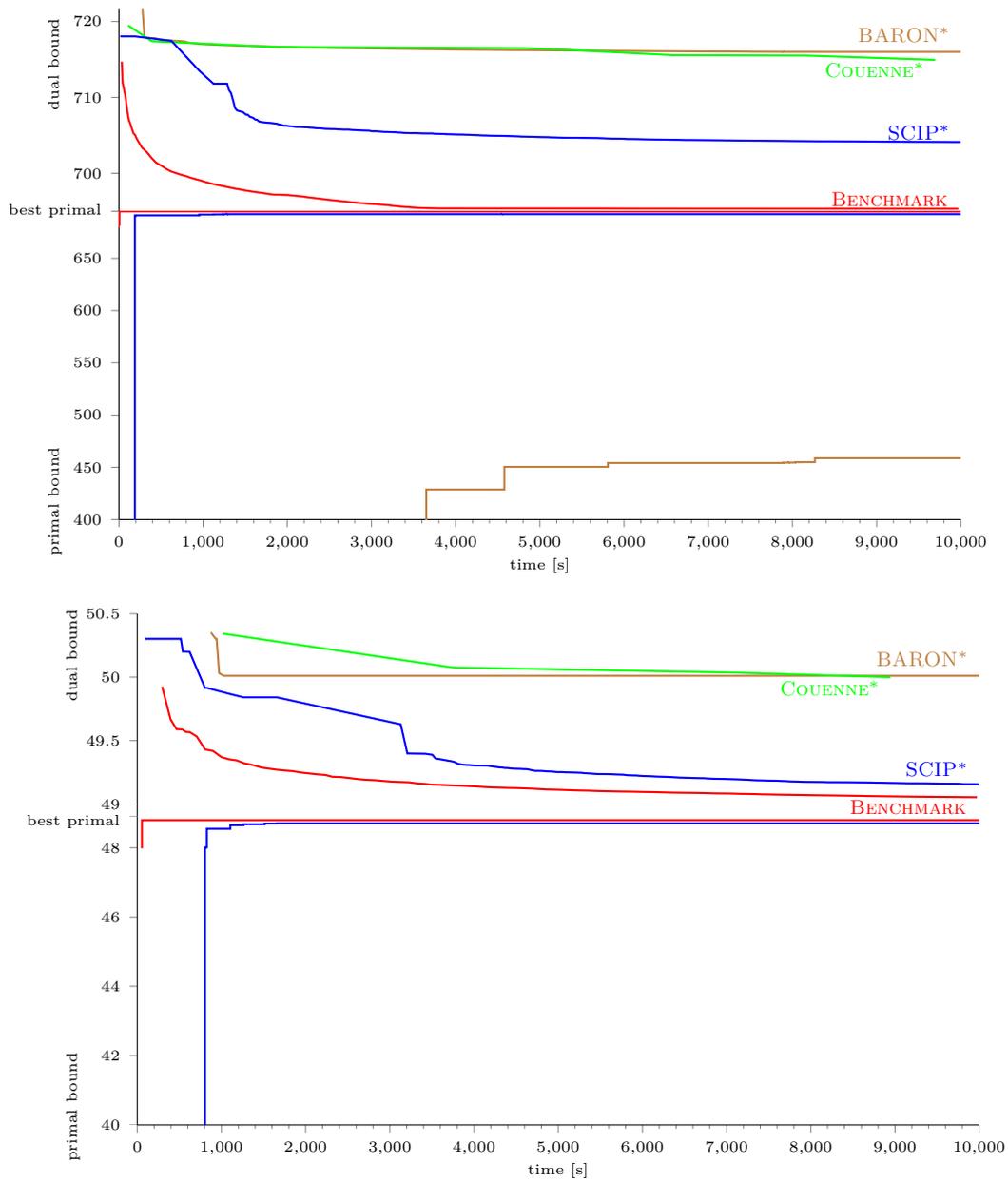


Figure 8.1.: Progress in primal and dual bounds for the application-specific algorithm (red) and the global solvers for the warehouse formulation (8.4) for instances *Marvin* (top) and *Dent* (bottom). The “best primal” axis is level with the best known primal solution value from the application-specific benchmark algorithm.

## 8. Computational Study

instance *Dent*. SCIP spend much time in the undercover heuristic during the root node processing, especially for instance *Dent*. However, this time is well spent, since SCIP's other heuristics are able find improved solutions short after, e.g., for both instances, the best solution was found by the improvement heuristic DINS. While COUENNE in version 0.2 was still able to find good primal solutions (see Bley et al. [2012b]), the version 0.4 that we used here had no success in finding any primal solutions<sup>7</sup>.

For the dual bound it can be seen that all three solvers start almost equal. SCIP, however, is able to decrease it more rapidly and comes closer to the best known primal solution values from the application-specific algorithm. For both instances, SCIP's dual bound after 3000 seconds is already less than 0.31% above its final value at 10000 seconds.

### 8.2. Water Distribution Network Design

In the following, we consider a MINLP formulation of a water network design optimization problem that has been discussed by Bragalli, D'Ambrosio, Lee, Lodi, and Toth [2012]. Given the topology of a water distribution network (WDN), the problem of finding an optimal design consists of choosing the diameter of pipes that have to be installed such that installation costs are minimized and the demand on the nodes can be satisfied while respecting hydraulic constraints. The diameter can be chosen from a discrete set of elements. The model combines a combinatorial part (choosing the diameter for each pipe) with nonconvex nonlinear constraints. Nonlinearity arises from modeling the pressure loss in the pipes, which includes a nonsmooth function, which can be problematic when employing NLP solvers on the continuous relaxation or a subproblem of the MINLP.

Bragalli et al. [2012] applied BONMIN's NLP-based branch-and-bound algorithm to find good feasible solutions for their WDN instances. They customized BONMIN by using a different formulation of the objective function in the bounding step and in being more conservative when deciding whether to prune a node due to inferiority (since the NLP solver employed by BONMIN guarantees only a local optimal solution, its objective function value may not yield a correct dual bound). Further, they used a smoothed formulation of the pressure loss constraint.

In the following, we investigate the performance of MINLP solvers that can also compute dual bounds. We will use the same instances as Bragalli et al. [2012].

#### 8.2.1. Model

We use a notation similar to Bragalli et al. [2012]. Given the topology of a WDN, denote by  $N$  the set of junctions (nodes) and by  $E$  the set of pipes (edges). For each pipe  $e \in E$ , the available diameters belong to a finite set  $\mathcal{D}_e \subset \mathbb{R}_+$ . The diameters for each pipe are

---

<sup>7</sup>COUENNE 0.4 also processed less nodes in the same time on a faster machine when compared to previously published results with version 0.2. Unfortunately, it is difficult to say where COUENNE spend most of the time. It may be the construction or solution of the LP relaxation, since, for instance *Marvin* with the warehouse formulation, in previous runs, COUENNE reported to have generated  $\approx 12$  million cuts to estimate the quadratic constraints while processing  $\approx 4300$  nodes, while SCIP generated for the same instance “only”  $\approx 1.2$  million cuts while processing  $\approx 93000$  nodes.

associated with a cost function  $C_e(\cdot) : \mathfrak{D}_e \rightarrow \mathbb{R}_+$ . Further pipe parameters are the length  $\ell_e$ , a physical constant  $k_e$  that depends on the roughness of the pipe, and a limit  $\bar{v}_e$  on the speed of water in a pipe.

Let  $S \subset N$  be the set of source junctions. For each junction  $n \in N \setminus S$ , we have parameters for the physical elevation  $h_n$ , the minimal and maximal pressure head  $\underline{p}_n$  and  $\bar{p}_n$ , respectively, and the demand  $q_n$ . For a source junction  $n \in S$ , the pressure head is fixed,  $\underline{p}_n = \bar{p}_n$ . Finally, by  $\delta_+(n) \subset E$  we denote the set of pipes with tail at junction  $n$  and by  $\delta_-(n) \subset E$  the set of pipes with head at junction  $n$ .

The task is to decide for a single diameter for each pipe, such that there exists for each junction a hydraulic pressure head within the given bounds that realizes a flow on the adjacent edges that satisfies the demand on all non-source junctions. Thereby, the flow on a pipe is uniquely determined by the pipe diameter and the pressure head on both ends of the pipe. We refer to Bragalli et al. [2012] and the references therein for a more detailed discussion of the model.

We consider the following variables:

- $x_{e,d}$  binary variable indicating the diameter chosen for a pipe ( $e \in E$ ,  $d \in \mathfrak{D}_e$ )
- $A_e$  cross-sectional area of a pipe ( $e \in E$ )
- $Q_e$  flow through a pipe ( $e \in E$ )
- $H_n$  hydraulic pressure head at a junction ( $n \in N$ )

The variables  $x_{e,d} \in \{0, 1\}$  model the decision whether diameter  $d \in \mathfrak{D}_e$  is selected for pipe  $e \in E$ . Since exactly one diameter has to be chosen for each pipe, we have

$$\sum_{d \in \mathfrak{D}_e} x_{e,d} = 1 \quad (e \in E). \quad (8.5a)$$

The cross-sectional area of a pipe is linked to the diameter by

$$A_e = \frac{\pi}{4} \sum_{d \in \mathfrak{D}_e} d^2 x_{e,d} \quad (e \in E). \quad (8.5b)$$

The flow on each pipe is limited by the cross-sectional area via the constraint

$$|Q_e| \leq A_e \bar{v}_e \quad (e \in E). \quad (8.5c)$$

Further, flow conservation constraints need to hold on all non-source nodes,

$$\sum_{e \in \delta_-(n)} Q_e - \sum_{e \in \delta_+(n)} Q_e = q_n \quad (n \in N \setminus S). \quad (8.5d)$$

A flow on a pipe is caused by differing head pressure in the adjacent junctions, where pressure loss occurs due to friction. An accepted model is the Hazen-Williams equation

$$\text{sign}(Q_e) |Q_e|^{1.852} = \frac{k_e^{1.852}}{10.7 \ell_e} (H_n - H_m) \left( \frac{4}{\pi} A_e \right)^{2.435} \quad (e = (n, m) \in E). \quad (8.5e)$$

Additionally, the head pressure is bounded by the minimal and maximal pressure head,



name	junctions $ N $	sources $ S $	pipes $ E $	diameters $ \mathcal{D}_e $	variables	binary	constr.
shamir	7	1	8	14	135	112	46
hanoi	32	1	34	6	304	204	201
blacksburg	31	1	35	14	591	490	205
New York	20	1	21	12	335	273	124
foss_poly_0	37	1	58	7	559	406	326
foss_iron	37	1	58	13	907	754	326
foss_poly_1	37	1	58	22	1429	1276	326
pescara	71	3	99	13	1556	1287	563
modena	272	4	317	13	5027	4121	1853

Table 8.5.: Statistics on water network instances from Bragalli et al. [2012]. The last three columns specify the number of variables, binary variables, and constraints when using the signpow-formulation.

and `modena` correspond to real-world instances of Italian water networks. `foss_*` are variations for a water network in a neighborhood of Bologna. For instance `blacksburg`, the diameter of twelve pipes is already fixed.

The instance `New York` is special in the sense that the network is to be renovated instead of build, that is, for an existing network with fixed pipe diameters, the task is here to select edges where an *additional pipe* should be built and to decide for the diameter of the additional pipes. Fortunately, this task can also be handled with model (8.6) by treating a possible pair of pipes on an edges as one single pipe, which diameter is computed from the diameter of the existing and the new pipe, cf. Theorem 2 in Bragalli et al. [2012]. Further, the case where no additional pipe is built is accounted by adding the existing diameter to the set  $\mathcal{D}_e$ , associating with zero cost.

### 8.2.3. Computational Results

We have run SCIP, BARON, COUENNE, and LINDOAPI on model (8.6) with all formulations for the term  $\text{sign}(Q_e)|Q_e|^{1.852}$  for all available test instances with a time limit of 10,000 seconds. For SCIP, we set the emphasis for heuristics and separators to aggressive. Table 8.6 shows the best primal and dual bounds computed by all solvers and state the best primal bound reported by Bragalli et al. [2012]. We further state the formulation of the  $\text{sign}(Q_e)|Q_e|^{1.852}$  term that yield the corresponding bound. A ‘\*’ indicates that all formulations gave the same result. If a solver reported a dual bound that contradicts with the best primal bound for some formulation, then the results for this formulation are ignored for this instance<sup>9</sup>. If no formulation yield a correct dual bound, a dash is printed. Best dual and primal bounds are printed in bold face.

<sup>9</sup>LINDOAPI reported wrong dual bounds on instance `foss_poly_0` with `ifthen-` and `abssign-`formulations. BARON reported a wrong dual bound for instances `hanoi`, `blacksburg`, and `pescara` with all formulations. COUENNE reported wrong dual bounds for instance `shamir` with all formulations.

## 8. Computational Study

instance	SCIP		BARON		COUENNE		LINDOAPI		BONMIN
	dual	primal	dual	primal	dual	primal	dual	primal	primal
shamir	<b>419,000</b>		<b>419,000</b>		–	–	<b>419,000</b>		<b>419,000</b>
	signpow		signvar		*	*	*		
hanoi	<b>6,109,621</b>		–	–	<b>6,109,621</b>		5,868,343	6,147,427	<b>6,109,621</b>
	signpow		*	*	abs		ifthen	abssign	
blacksburg	113,755	<b>117,248</b>	–	–	105,774	161,818	<b>116,842</b>	118,223	118,251
	signpow	signpow	*	*	abs	abs	ifthen	ifthen	
New York	18,614,769	40,737,281	<b>28,359,301</b>	45,112,035	27,804,136	40,938,286	22,067,766	39,309,310	<b>39,307,780</b>
	signpow	signpow	signvar	signvar	signvar	signvar	signvar	signvar	
foss_poly_0	<b>67,559,218</b>		64,753,820	∞	66,292,084	∞	67,231,420	70,680,508	70,680,508
	signpow		*	*	abs	*	abs	ifthen	
foss_iron	<b>175,922</b>		170,552	950,978	175,468	∞	174,311	278,146	178,494
	signpow		*	abs	abs	*	abs	ifthen	
foss_poly_1	<b>26,207.48</b>	<b>28,936.70</b>	25,308.12	∞	25,701.94	37,299.61	25,425.87	∞	29,177.04
	signpow	signpow	*	*	abs	abs	abs	*	
pescara	<b>1,635,507</b>	3,642,604	–	–	1,575,067	∞	1,588,214	∞	<b>1,820,264</b>
	signpow	signpow	*	*	abs	*	signvar	*	
modena	<b>2,113,768</b>	∞	2,073,054	∞	2,089,281	∞	2,077,055	∞	<b>2,576,589</b>
	signpow	*	*	*	abs	*	abs	*	

Table 8.6.: Computational results for water distribution network design problem. For each solver, we report the best dual and primal bound that was found and the formulation of the  $\text{sign}(Q_e)|Q_e|^{1.852}$  term in the pressure loss constraint (8.5e) that was used to compute that bound. The last column reports the best primal bound as reported by Bragalli et al. [2012].

For SCIP, best bounds are always obtained by using the `signpow`-formulation, likely because it then handles  $\text{sign}(Q_e)|Q_e|^{1.852}$  as a signed power function for which it can generate tight linear underestimators, see also Section 7.5.2. SCIP improves the best known solutions from Bragalli et al. [2012] for four instances and finds proven optimal solutions for four instances. However, for the two largest instances (`pescara` and `modena`), SCIP finds either no feasible solution or one which objective function value is much worse than the best known primal bound. SCIP reports the best dual bound for 7 instances.

Finally, we run SCIP 3.0.0 with a time limit of 24 hours. As a result, we could solve `blacksburg` to optimality (116,945), solve `foss_poly_1` to optimality (28,043.86), and improve the dual bounds for `pescara` and `modena` to 1,643,525 and 2,122,182, respectively.

### 8.3. Benchmarks

To get an impression of SCIP’s performance on MINLPs in general, we conducted numerical experiments on four different test sets, each of them allowing for comparisons with a different set of solvers: a set of general MINLPs from MINLPLib to compare with global solvers for convex and nonconvex MINLPs, a set of convex MINLPs from various sources to compare with solvers for convex MINLPs, a set of “CPLEX-compatible” MIQPPs, and a set of mixed-integer second-order cone programs (MISOCPs) to compare with CPLEX and MOSEK. The test sets are discussed in more detail below. For almost every instance, a feasible solution is known, and in many cases also an optimal one.

**Problem Statistics.** For each test set, we report problem statistics like the number of variables and linear and nonlinear constraints before and after SCIP’s presolve. Note, that next to SCIP’s MIP presolving routines, also the reformulation of the expression graph and quadratic functions as discussed in Section 7.6 are applied during presolve. Thus, the presolved problem may have more variables and constraints than the original one. Further, due to the reformulation of quadratic functions with binary variables (cf. Section 7.6.2), some MINLPs may be completely linearized during presolve, which is indicated by  $m' = 0$  in the problem statistics. If a best known or global optimal solution is known, its objective function value is also given in the problem statistics.

**Solve Outcome.** For each problem instance, we run SCIP and a set of other solvers with a time limit of 2 hours and a feasibility and gap tolerance of  $10^{-6}$ . For each run of a solver on an instance, we record the number of seconds for solving the problem, or, if a time limit was hit, the primal and dual bound<sup>10</sup> at termination. Additionally, for branch-and-bound solvers, we collect the number of nodes that have been processed. A solver is marked to have *failed* on an instance, if it reports a dual bound that contradicts with the objective value of a best known solution or reports a primal bound that is better than a known optimal value or if it stops before the time limit with an upper bound  $\bar{v}$  and lower bound  $\underline{v}$  on the optimal value such that  $\bar{v} - \underline{v} > 10^{-4} \max(1, |\bar{v}|)$ . A solver is marked to have *aborted* on an instance, if it stops without reporting a result (e.g., segmentation fault) or does not stop within twice the time limit.

In the detailed result tables, each entry shows the time in seconds a solver spent to solve a problem and the number of processed nodes. If the problem has not been solved within the given time limit and the solver did not fail, then we report either the *dual gap* and *primal gap* in parentheses, or the dual bound and primal bound in brackets. The gaps are reported if the optimal value of the instance is known, otherwise the bounds are given. Inspired by Achterberg, Berthold, and Hendel [2012], we define the *dual gap* and the *primal gap* for a problem with dual bound  $\underline{v}$ , primal bound  $\bar{v}$ , and optimal value  $v^*$  as

$$\text{dual gap} := \min \left( 1, \frac{|v^* - \underline{v}|}{\max(1, |v^*|)} \right), \quad \text{primal gap} := \min \left( 1, \frac{|\bar{v} - v^*|}{\max(1, |v^*|)} \right).$$

We use these values, to evaluate the quality of a dual and a primal bound reported by a solver. A justification for the truncation at 1 is the claim that all bounds that are far away from the optimal value are equally useless. Further, limiting the gaps to  $[0, 1]$  allows for meaningful arithmetic means.

For each instance, the fastest solution time or – in case all solvers hit the time limit – the best primal and dual gaps (or bounds), are depicted in bold face<sup>11</sup>.

<sup>10</sup>As in Section 8.1, the *dual bound* corresponds to the lower bound for a minimization problem and to the upper bound for a maximization problem. The *primal bound* is the objective function value of the incumbent solution, if any found.

<sup>11</sup>A solution time is marked as best time, if it is within one second of the minimal solution time for that instance. A dual or primal gap/bound for a solver that hit the time limit is marked as best gap/bound, if no solver solved the instance and if it is within 0.01% of the best one for that instance.

## 8. Computational Study

**Aggregated Solution Statistics.** Further, for each solver we calculated mean values of the solution time (in which unsolved instances are accounted for with the time limit), the number of processed nodes, and the primal and dual gap at termination. The mean values for solution time and nodes are computed w.r.t. all instances where no solver aborted or computed a wrong bound (fail). For the mean values for primal and dual gap, we restrict further to instances with known optimal value. The number of instances that satisfy these criteria are specified in the tables next to the mean values. Finally, statistics on how often a solver solved a problem, reported a wrong bound (fail) or aborted, hit the time limit, computed the best dual bound, found the best primal solution value, or was the fastest among all solvers are collected. For the last three, we count only instances that can be handled by all considered solver.

We consider three different mean values for a set of positive data points  $\{v_1, \dots, v_n\} \subset \mathbb{R}_+^n$ , see also Section A.3 in Achterberg [2007]. The *arithmetic mean* is defined by

$$\frac{1}{n} \sum_{i \in [n]} v_i$$

and is sensitive to variations of data points with relatively large range, while variations of data points with relatively small range are neglected. The *geometric mean* is defined by

$$\left( \prod_{i \in [n]} \max(\varepsilon, v_i) \right)^{1/n},$$

for some  $\varepsilon > 0$ . Contrary to the arithmetic mean, the geometric mean is more sensitive to variations close to zero. The *shifted geometric mean* is a compromise that reduces the effect of data points close to zero in the geometric mean by shifting. It is defined by

$$\left( \prod_{i \in [n]} \max(\varepsilon, v_i + s) \right)^{1/n} - s,$$

for some shift value  $s \in \mathbb{R}_+$ . In the summarizing tables, we report only the shifted geometric means for solution time and processed number of nodes and the arithmetic means for primal and dual gap. In the detailed tables, also other mean values are given. We used  $\varepsilon = 1$  and  $s = 10$  for solution times and  $\varepsilon = 1$  and  $s = 100$  for node counts.

**Virtually Best Solver.** Finally, we collect values for a *virtually best solver*, whose solution time, primal, and dual bound on an instance is defined as the best one among running all (non-virtual) solvers, that is, the minimal time that any solver spend on the instance, and the best primal and dual bounds that have been found by any solver. If all solvers failed on an instance, then also the virtually best solver failed. The virtually best solver allows to estimate the loss from having only one instead of all solvers available.

**Performance Profiles.** Additionally to the mean values, we plot for each solver the number of instances (i) solved within a certain time, (ii) with gap below a certain value,

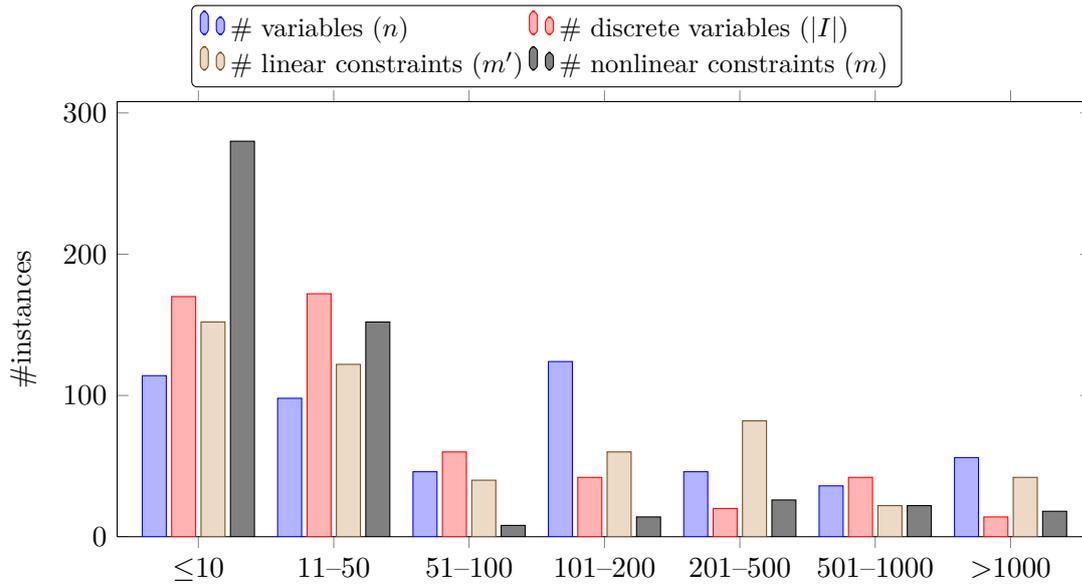


Figure 8.2.: Histogram on characteristic sizes of MINLPLib instances.

(iii) with primal gap below a certain value, or (iv) with dual gap below a certain value. The gap between a primal bound  $\bar{v}$  and a dual bound  $\underline{v}$  is defined as

$$\text{gap} := \begin{cases} \infty, & \text{if } \bar{v} = +\infty \text{ or } \underline{v} = -\infty, \\ 0, & \text{if } \bar{v} - \underline{v} \leq 10^{-6}, \\ \infty, & \text{if } \underline{v} = 0 \text{ or } \bar{v} = 0, \\ \frac{\bar{v} - \underline{v}}{\min(|\underline{v}|, |\bar{v}|)}, & \text{otherwise,} \end{cases}$$

for a minimization problem and analogously for a maximization problem.

### 8.3.1. MINLPLib

The MINLPLib [Bussieck et al., 2003] is a collection of (currently) 272 MINLP instances that contains both small-scale models from the literature and large industrial models. For most instances, also feasible solutions are included. The MINLPLib is frequently used to test but also to benchmark MINLP solvers. Since the collection of instances is not as well-balanced as, e.g., the benchmark sets of the MIPLIB library [Koch et al., 2011], benchmarks based on the MINLPLib have to be taken with a pinch of salt. See Figure 8.2 for histograms that visualize the number of instances with certain numbers of variables, discrete variables, linear constraints, and nonlinear constraints.

Nevertheless, since we are not aware of a generally accepted benchmark set for MINLP, we also used the MINLPLib (version from 8th of August 2012) to compare our implementation of MINLP extensions in SCIP with the state-of-the-art MINLP solvers BARON, LINDOAPI, and COUENNE, see also Section 6.2. All these solvers are global solvers, in

## 8. Computational Study

	# instances	SCIP	BARON	COUENNE	LINDOAPI	virt. best
#solved	260	<b>165</b>	140	137	138	190
#timeout	260	87	97	97	97	70
#failed or aborted	260	<b>8</b>	15	18	18	0
#fastest	252	<b>88</b>	64	20	12	
#best dual bound	252	<b>184</b>	155	151	159	
#best primal bound	252	<b>193</b>	164	156	168	
time (sh. geom. mean)	210	<b>91.2</b>	166.1	186.5	222.1	67.0
nodes (sh. geom. mean)	210	4981.8	2178.6	4308.4	<b>122.2</b>	726.4
dual gap (arith. mean)	172	<b>4.44%</b>	11.32%	11.23%	14.74%	1.50%
primal gap (arith. mean)	172	<b>0.45%</b>	5.70%	4.28%	11.75%	0.01%

Table 8.7.: Performance of different solvers on MINLPLib test set.

the sense that they compute valid dual bounds also for nonconvex problems (leaving numerical issues and programming bugs aside). For SCIP, we set a memory limit of 20GB<sup>12</sup>. When running BARON, we set the GAMS option `workfactor` to its maximal value of 1500, which increases the memory that is available to BARON<sup>13</sup>. COUENNE was run with CLP as LP solver<sup>14</sup>.

**Solver Capabilities.** For our tests, we excluded only those instances from MINLPLib that cannot be handled by SCIP because they use the functions `sin`, `cos`, or `erf`<sup>15</sup>. Problem statistics for the remaining 260 instances are given in Table A.1. The test set includes both convex and nonconvex models and both MIQQP and general MINLPs. Seven instances include special-ordered-set constraints, which can be handled by SCIP, but not by BARON, COUENNE, and LINDOAPI<sup>16</sup>. Further, instance `meanvarxsc` has semi-continuous variables, which can be handled by SCIP, but none of the other solvers. Finally, instance `fuzzy` uses the `min` function that cannot be handled by COUENNE. If a solver cannot handle a specific instance, this is marked by a dash in the detailed result tables. These instances are also not considered when counting how often a solver was fastest, produced the best dual bound, or produced the best primal bound.

Detailed results are given in Table A.4. The various performance measures are summarized in Table 8.7.

<sup>12</sup>The memory limit regards only memory allocated by SCIP itself, e.g., excludes memory allocated by LP or NLP solvers. When 18 GB of memory have been allocated by SCIP itself, the node selection strategy is changed to depth-first-search, since this strategy consumes the least amount of memory.

<sup>13</sup>When approaching the memory limit, also BARON changes the node selection to depth-first-search.

<sup>14</sup>We have also run an experiment with CPLEX as LP solver in COUENNE, but the number of failed and aborted instances increased to almost half of the test set.

<sup>15</sup>The excluded instances are `blendgap` (erf), `deb6,7,8,9,10` (sin, cos), `dosemin2d,3d` (erf), `prob10` (sin), `var_con5,10` (sin, cos), and `windfac` (sin).

<sup>16</sup>LINDOAPI allows models with special-ordered-set constraints, but returns solutions where these constraints are violated. We decided to treat this as inability to handle special-ordered-set constraints.

**Solves and Fails.** We note, that SCIP solved the largest number of instances, also when excluding the eight instances that cannot be handled by every solver. From the 190 instances that could be solved by any solver within the time limit, SCIP solved 87%, while BARON, COUENNE, and LINDOAPI solved 74%, 72%, and 73%, respectively. The remaining 70 instances could not be solved by any solver. Further, SCIP seems to work most robust on this test set, that is, it failed on the least number of instances. SCIP failed on `ex1252a`, `gasnet`, `gear4`, `parallel`, `pump`, `water4`, and `waterz` and aborted on `contvar`. These failures are usually caused by numerical issues<sup>17</sup>. BARON seems to have problems to read the `pb35*` instances within the time limit<sup>18</sup>. Instances `hda`, `product*`, `super1,2,3`<sup>19</sup>, `uselinear`, `water4`, and `waterz` are wrongly declared as infeasible by BARON and for instances `ex1252*`, `minlpnix`, `nvs22`, and `pump` wrong optimal values are reported. COUENNE wrongly declared the instances `hda`, `nvs22`, `procsol`, and `product*` as infeasible and reported a wrong optimal value or wrong bounds for the instances `du-opt*` and `oil*`. For the instances `csched2a` and `nuclearv*`, COUENNE did not stop after the time limit. The aborts for instances `csched2` and `st_e32` are due to segmentation faults. LINDOAPI wrongly declared the instances `ghg_1veh`, `ghg_3veh`, `oil` and `uselinear` as infeasible and reported wrong optimal values for instances `eg_all_s`, `eg_int_s`, `eniplac`, `hda`, `nuclearvb`, `nuclearvd`, `nuclearvf`, `space25`, and `st_e40`, and `tls2`. For instance `mbtd`, LINDOAPI aborted with a segmentation fault. For instances `nuclearva`, `super1`, and `super2`, LINDOAPI did not stop within twice the timelimit. There is no instance, where every solver failed.

**Fastest and Best Bounds.** From the 252 instances that all solvers could handle, there are 88 instances where SCIP is as fast as the fastest among all solvers on these instances, followed by BARON, which is the fastest for 64 instances. The best dual bounds are reported by SCIP for 73% of the 252 instances, compared to between 60% and 63% for the other solvers. The best feasible solutions are found by SCIP for 77% of the 252 instances, compared to between 62% and 67% for the other solvers.

**Mean Values.** On only 83% of all instances that all solvers could handle, no solver failed or aborted (210 out of 252). On these 210 instances, the mean solving time of BARON, COUENNE, and LINDOAPI are 1.8, 2.0, and 2.4 times the one of SCIP, respectively. Regarding the number of nodes, SCIP enumerates about twice as much nodes as BARON in average, while COUENNE's mean number of nodes is 14% below the one of SCIP. On the contrary, the mean on the reported number of nodes of LINDOAPI is only 2% of

<sup>17</sup>For example, when SCIP finished its branch-and-bound on instance `parallel`, it believed that the optimal value is 923.87. Unfortunately, SCIP then recognized that the solution that was computed for the presolved and reformulated problem violates a constraint of the original problem by  $\approx 9.9 \cdot 10^{-5}$ , which is above the feasibility tolerance. In an attempt to repair this infeasibility, SCIP then calls IPOPT to solve (locally) the NLP obtained by fixing all integer variables in the original MINLP to the values in the almost feasible solution. In this case, the (now feasible) solution reported by Ipopt had an objective function value of 924.296. As a result, SCIP finished before the time limit without having the gap closed, which is evaluated as a failure.

<sup>18</sup>The `pb*` instances have quadratic objective functions with many terms ( $\approx 15000$  products for `pb302035`).

<sup>19</sup>The `super1,2,3` instances contain constants like  $10^{-13}$ , which is numerically "quite challenging".

## 8. Computational Study

SCIP's mean. Out of the 210 instances which all solvers processed without fail or abort, the optimal value is known for 172 of them. While the average dual bound reported by SCIP differs only by 4.4% from the optimal value, the mean dual gap reported by BARON, COUENNE, and LINDOAPI is at least 11.2%. Similarly, the average primal gap for SCIP is 0.5%, while it is at least 4.3% for the other solvers. This indicates, that SCIP finds a good feasible or optimal solution in most cases. When comparing the best bounds reported by any solver, then the best dual and primal bound are in average close (1.5%) and very close (0.01%), respectively, to the optimal value.

**Performance Profiles.** Performance profiles are shown in Figure 8.3. The profile on the number of instances solved by a certain point in time shows that within one second, LINDOAPI solves a bit more than 70 instances, COUENNE solves almost 90 instances, BARON solves almost 100 instances, and SCIP solves  $\approx 115$  instances. The number of solved instances then gradually improves until they reach the numbers given in the first row of Table 8.7. One can observe, that LINDOAPI has solved the lowest number of instances in the beginning, but catches up with BARON and COUENNE after  $\approx 1000$  seconds. This behavior aligns with the high mean solution time of LINDOAPI compared to the relatively high number of solved instances, see Table 8.7.

Regarding the gap between dual and primal bound at termination, Figure 8.3 shows that SCIP terminates with a nonzero gap below 100% for only approximately 20 of the instances. For the other solvers, the number of instances with a gap below  $x\%$  increases similarly slow. The profiles of the primal and dual gap indicate, that the main difficulty in closing the gap is the dual bound. While the number of instances with a primal gap below  $x\%$  is increasing rapidly with increasing  $x$ , the profile for the dual gap resembles the same behavior as the one for the remaining gap. The high gradient of the primal gap curve for SCIP at the beginning shows that for most instances where SCIP found a feasible solution, the primal gap is below 5%. Recall, that primal and dual gaps are only computed for the 195 instances with known optimal value.

**Instances where SCIP performs best.** Instances that SCIP seems to handle much better than the other solvers include `du-opt*`, `fo*`, `netmod*`, `no*`, `o7,8,9*`, `pb*`, `tln12`, and `tls7`. `du-opt*` are convex MIQPs, where half of the variables are of general integer type. It may be a combination of recognizing the convexity of the objective function (by checking positive-semidefiniteness of the coefficient matrix) and the advanced MIP machinery that makes SCIP the winner on these instances. The `fo*`, `netmod*`, `no*`, `o7,8,9*` instances are also convex MINLPs with a challenging combinatorial part, which may be handled better by SCIP than by the other solvers. The `pb*` instances are MIQPs that contain only binary variables. SCIP reformulates them as MIP, see Section 7.6.2, which increases the number of variables by a factor of two and the number of constraints by a factor of three. Even though the final gap reported by SCIP is still huge, the reformulated problem seem to give better dual bounds in comparison to those computed by other solvers. `tln*` and `tls*` are challenging MINLP formulations of trim loss problems (`tls*` is a convex reformulation of `tln*` [Harjunkski et al., 1998]).

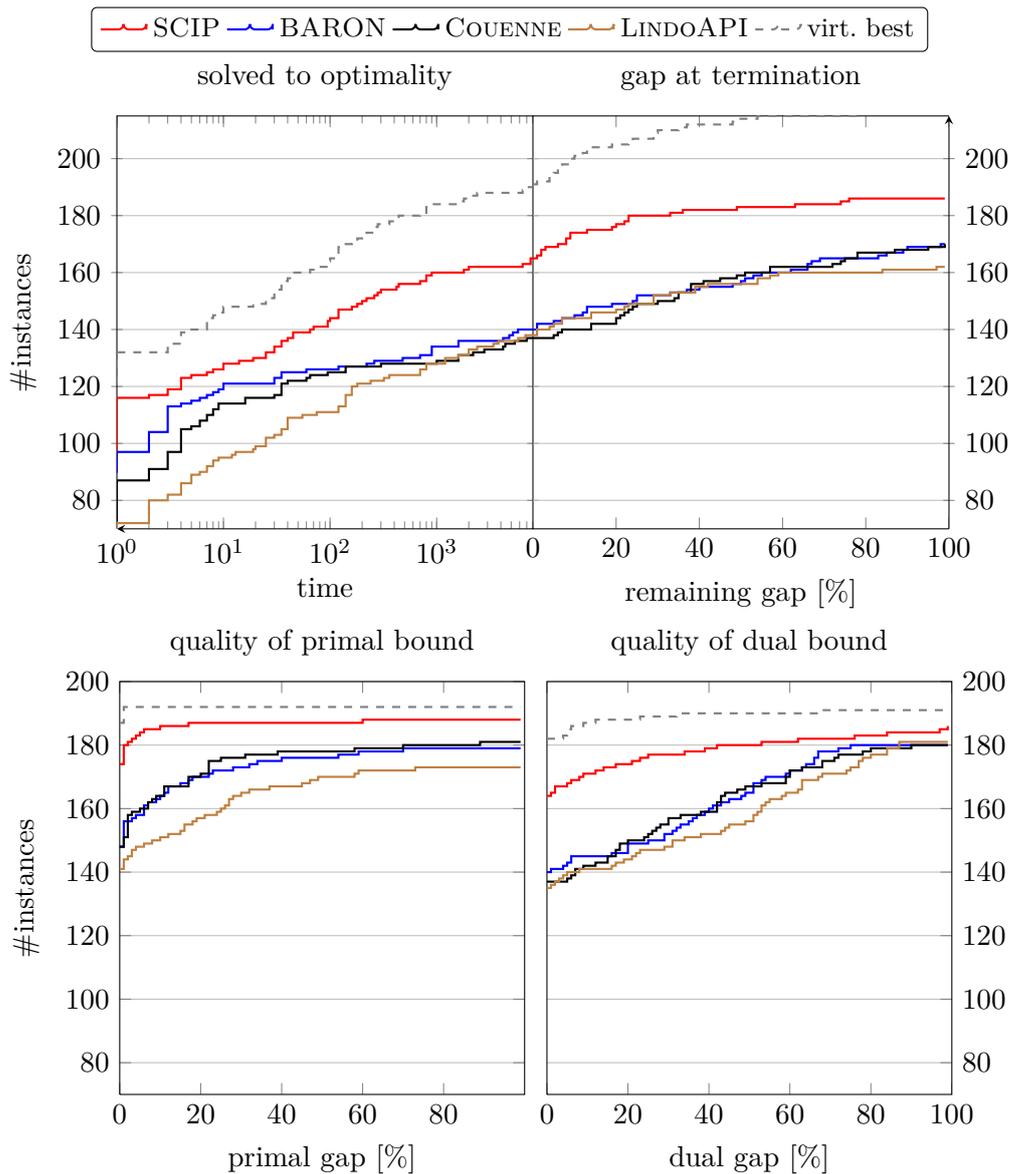


Figure 8.3.: Performance profiles for MINLPLib test set (260 instances in total, 195 with known optimal value).

**Instances where SCIP performs worst.** Finally, there are also several instances that SCIP handles much worse than other solvers. `csched1*` is solved very fast by all solvers except for SCIP, which finds very good primal solutions, but does not proof optimality before the time limit. Also `minlpnix` is solved very fast by COUENNE, while SCIP could not find a finite dual bound. Also the best solution found by SCIP is still far from the optimal one. A reason for these problems may be missing bound tightening techniques (cf. Section 6.1.5). For the nonlinear part of the problem, the used SCIP

## 8. Computational Study

version only implements a constraint-based bound tightening procedure, see also Section 7.3.2, but does not apply probing to non-binary variables or relaxation-based bound tightening<sup>20</sup>. However, advanced bound tightening techniques may be essential here. For `csched1*`, SCIP often refrains from adding linear underestimators to the LP relaxation due to numerical issues<sup>21</sup>, which then leads to more branching. Instance `minlpix` has unbounded variables in nonconvex nonlinear variables, due to which no bounded linear relaxation can be constructed. SCIP tries to bound these variables by branching, but at all time at least one node with infinite dual bound remains. For instance `ghg_1veh`, SCIP reduces the gap between dual and primal bound to below 0.01% within 90 seconds, but for the remaining two hours the dual bound improves only very slowly. It may be a too cautious bound tightening (too much widening of variable bounds before domain propagation) that prevents SCIP from stopping earlier here.

### 8.3.2. Convex MINLPs

To compare SCIP with solvers for convex MINLPs, we extended the test set from Bonami et al. [2010]. From a webpage of an CMU-IBM project on MINLP<sup>22</sup>, we took the multi-product batch plant design instances `Batch*`, the constrained layout problems `clay*`, the farmland layout models `FLay*`, the retrofit planning models `RSyn*`, the safety layout problems `SLay*`, and the synthesis design models `Syn*`, see also Bonami et al. [2010] and the references therein. From MINLPLib, we took the block layout design problems `fo*`, `m*`, `no*`, and `o*` [Castillo et al., 2005], the network community structure identification problems `net*` [Xu et al., 2007], and the convex reformulations of trimloss problems `tls*` [Harjunkoski et al., 1998]. From Günlük and Linderoth [2012] are the network design with congestion constraint problems `nd*`, the stochastic service system design problems `sssd*`, and the quadratic uncapacitated facility location problems `uflquad*`. The complete test set has 155 instances. Detailed problem statistics, including optimal values, for all instances not from MINLPLib are given in Table A.2. For those from MINLPLib, see Table A.1. For all instances in this test set, the optimal value is known.

**Solvers and Settings.** On this test set, we compared SCIP only with solvers for convex MINLPs. Even though also general MINLP solvers are applicable, they may not always recognize the convexity of some instances, while the solvers for convex MINLPs can assume convexity. For example, the instances which names ends with an `h` or `H` (except for the `SLay*` instances) use a special convex hull construction [Grossmann and Lee, 2003], which uses a function which convexity is not evidently seen, see also (7.10) on page 181 for an example taken from instance `clay0203h`.

We compared SCIP with the solvers ALPHAECP, BONMIN, DICOPT, KNITRO, and SBB, see also Section 6.2. We run SCIP in two variants. In the first one, we set the SCIP option `constraints/nonlinear/assumeconvex` to `TRUE`, which instructs SCIP

<sup>20</sup>A LP-relaxation-based bound tightening method was added in SCIP 3.0 [Gleixner and Weltge, 2013].

<sup>21</sup>By default, a cut  $\langle a, x \rangle \leq \bar{a}$  computed by the MINLP constraint handlers in SCIP is only added to the LP relaxation, if the maximal range of the cut coefficients  $\max_i |a_i| / \min_{i:a_i \neq 0} |a_i|$  is below  $10^7$ .

<sup>22</sup><http://egon.cheme.cmu.edu/ibm/page.htm>

to treat every general nonlinear function as convex, which is analog to the assumption a convex MINLP solver does. In the SCIP-nc setting, we did not set this option. As a result, SCIP does not always recognize convexity, may reformulate the expression graph and then applies convexification techniques and spatial branching. In both settings, we set again a memory limit of 20GB, see also Section 8.3.1. For ALPHA ECP we set the option `ECPstrategy` to 1, which disables extra steps for handling pseudo-convex functions, set the option `CUTdelcrit` to 0, which disables a cut selection heuristic that is only useful for nonconvex problems, set the feasibility tolerance in ALPHA ECP `TOLepsg` to  $10^{-6}$  and the one for the MIP relaxations in CPLEX to  $10^{-9}$ . Since BONMIN implements many algorithmic strategies and the default NLP-based branch-and-bound is not always the most efficient for a convex MINLP, we run BONMIN in 6 variants: BONMIN-BB is a NLP-based branch-and-bound (cf. Algorithm 6.1), BONMIN-ECP is a LP-based branch-and-bound similar to the one that SCIP implements (see also the discussion of the extended cutting plane variant of Algorithm 6.4 on page 122), BONMIN-Hyb is a hybrid of a LP-based and a NLP-based branch-and-bound (thus, in between BONMIN-BB and BONMIN-QG), BONMIN-OA is an outer-approximation algorithm (cf. Algorithm 6.2) that uses CBC to solve the MIP relaxation, BONMIN-OA-cpx is the same outer-approximation algorithm but uses CPLEX (in single-thread mode) to solve the MIP relaxation, and BONMIN-QG implements a LP/NLP-based branch-and-bound algorithm (cf. Algorithm 6.4). For DICOPT, we set the option `stop` to 1, which instructs DICOPT to solve the instances to optimality, and `maxcycles` to 10000, which avoids prematurely stopping after 20 outer-approximation iterations. For KNITRO, we set the options `mip_maxnodes` and `mip_maxsolves` to 0, which disables limits on the number of nodes to explore and number of subproblems to solve. Finally, for SBB we set the option `memnodes` to 9999999 to disable a limit on the number of open nodes in SBB.

**Statistics.** Tables A.5–A.7 present the detailed computational results of this comparison and Table 8.8 summarizes the results. For the latter, we excluded the number of nodes, because not all employed solvers run a branch-and-bound algorithm.

As expected, SCIP-nc performs much worse than SCIP, which is solely due to worse performance on those instances where it does not recognize convexity.

We observe, that BONMIN-OA-cpx solves most instances (142 out of 155) and has the smallest mean solution time. SCIP solves only six instances less and its mean solution time is only 60% higher. Accordingly, BONMIN-OA-cpx has also the highest number of instances where it is a fastest solver. More than 120 instances are also solved by ALPHA ECP (127), BONMIN-ECP (125), BONMIN-Hyb (129), and BONMIN-QG (124).

Regarding the number of instances where a best primal bound is found, SCIP, BONMIN-OA-cpx, and ALPHA ECP have roughly the same number, then BONMIN-Hyb, BONMIN-ECP, and BONMIN-QG. However, the average dual and primal gaps are lower for SCIP than for BONMIN-OA-cpx, indicating that SCIP often finds very good or optimal solutions, but does not always succeed in proving optimality before the time limit.

SCIP-nc and BONMIN-OA are the only solvers that did not fail or abort on any instance, followed by SCIP, ALPHA ECP, BONMIN-OA-cpx, and BONMIN-QG, who failed on only

## 8. Computational Study

	# instances	SCIP	SCIP-nc	ALPHAECP	BONMIN-OA-cpx	virt. best
#solved	155	136	103	127	<b>142</b>	146
#timeout	155	18	52	27	12	9
#failed or aborted	155	1	<b>0</b>	1	1	0
#fastest	155	13	10	1	<b>76</b>	
#best dual bound	155	140	104	129	<b>142</b>	
#best primal bound	155	<b>144</b>	110	141	142	
time (sh. geom. mean)	140	69.3	255.4	146.2	<b>43.3</b>	26.9
dual gap (arith. mean)	137	<b>1.71%</b>	12.66%	5.75%	3.40%	0.26%
primal gap (arith. mean)	137	<b>0.09%</b>	5.22%	0.47%	5.84%	0.00%

---

	# instances	BONMIN-BB	BONMIN-ECP	BONMIN-Hyb	BONMIN-OA	virt. best
#solved	155	90	125	129	82	146
#timeout	155	63	25	24	73	9
#failed or aborted	155	2	5	2	<b>0</b>	0
#fastest	155	6	2	0	1	
#best dual bound	155	91	125	129	82	
#best primal bound	155	102	132	136	82	
time (sh. geom. mean)	140	452.9	142.6	157.2	654.4	26.9
dual gap (arith. mean)	137	13.66%	2.09%	2.41%	35.98%	0.26%
primal gap (arith. mean)	137	8.35%	0.27%	0.28%	46.72%	0.00%

---

	# instances	BONMIN-QG	DICOPT	KNITRO	SBB	virt. best
#solved	155	124	84	72	78	146
#timeout	155	30	33	68	73	9
#failed or aborted	155	1	38	15	4	0
#fastest	155	7	25	3	2	
#best dual bound	155	125	85	73	78	
#best primal bound	155	128	90	76	86	
time (sh. geom. mean)	140	154.7	–	–	675.1	26.9
dual gap (arith. mean)	137	9.41%	–	–	25.79%	0.26%
primal gap (arith. mean)	137	0.67%	–	–	14.95%	0.00%

Table 8.8.: Performance of different solvers on convex MINLP test set.

one instance and several others that failed on two, four, or five instances. DICOPT and KNITRO have the highest number of fails or aborts (at least 10% of the test set). To still obtain meaningful mean values for solution time, dual gap, and primal gap (recall, that these means are w.r.t. all instances where no solver failed), we excluded DICOPT and KNITRO from the mean value computations. Fails are usually due to reporting a wrong dual bound. DICOPT reports to have aborted solving some instances (mainly `f0*` and `nd-*`) due to an decrease in the lower bound given by the MIP solver after having added integer cuts<sup>23</sup>. KNITRO aborted on the `nd-*` instances due to segmentation faults.

<sup>23</sup>The purpose of these “integer cuts” is to forbid a combination of values for the binary variables in the MIP approximation that has been proven to be infeasible for the MINLP by the NLP solver.

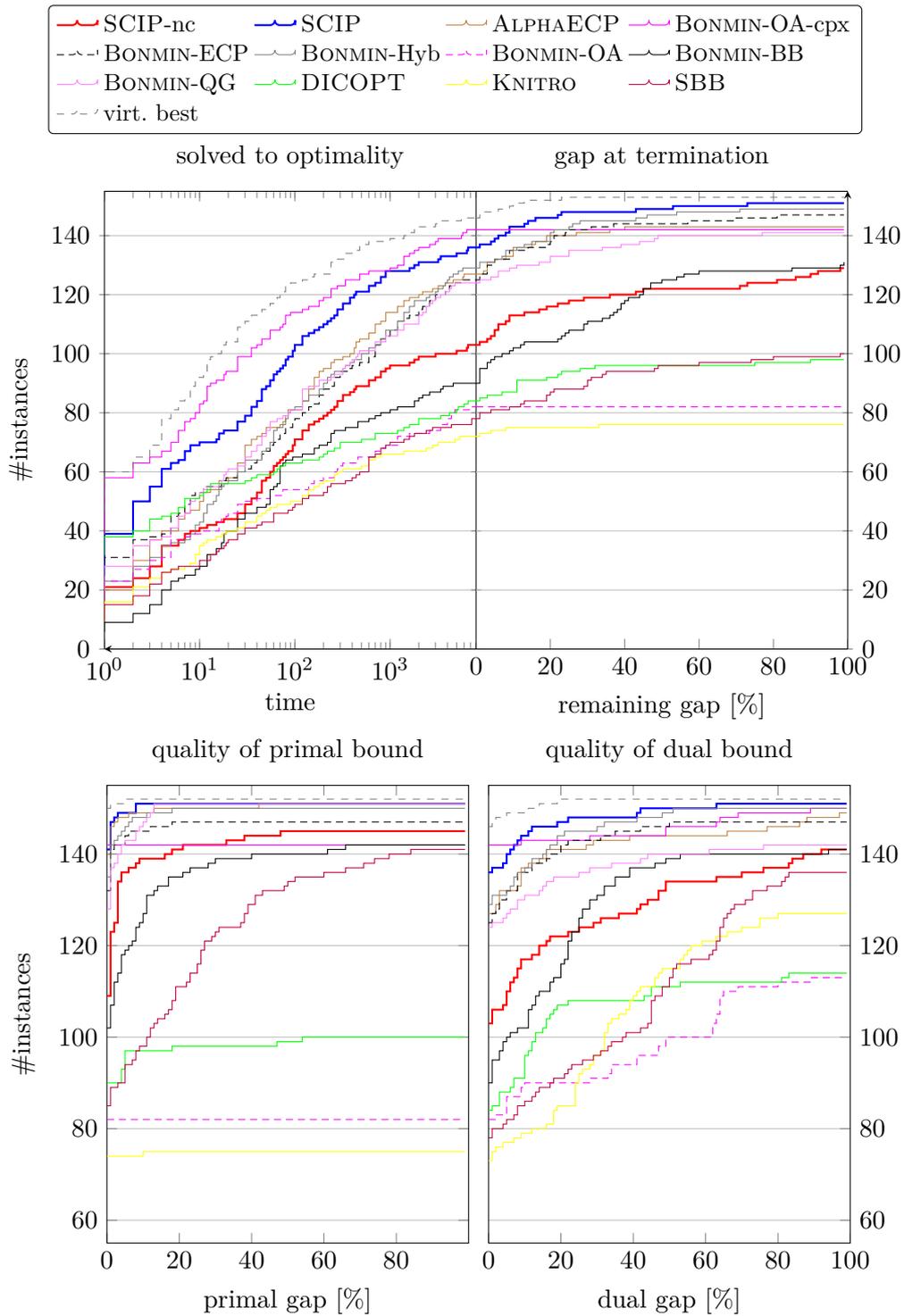


Figure 8.4.: Performance profiles for convex MINLP test set (155 instances in total, all with known optimal value).

## 8. Computational Study

**Performance Profiles.** Figure 8.4 shows the performance profiles. We observe, that even though BONMIN-OA-cpx solves six more instances than SCIP, the number of instances with a gap below 10% is the same.

Further, we observe that the number of instances with a small primal gap is relatively high for all solvers except for DICOPT, BONMIN-OA, KNITRO, and SBB. For DICOPT and SBB, we know that these solvers do not apply special heuristics to find feasible solutions early. The plot for the dual gap resembles again the plot for the remaining gap.

### 8.3.3. MIQPs

To allow for a comparison with CPLEX (see also Section 6.2), we consider a third test set which comprises only MIQPs where every quadratic objective and quadratic constraint is either convex or of second-order cone type (see also page 194) or is nonconvex but has only binary variables involved in nonconvex quadratic terms. A nonconvex quadratic term  $\langle x, Qx \rangle$  where all variables are of binary type ( $x_i \in \{0, 1\}$ ,  $i \in [n]$ ) is reformulated by CPLEX as  $\langle x, Qx \rangle + \sum_{i=1}^n u_i(x_i^2 - x_i)$ , where the vector  $u \in \mathbb{R}^n$  is chosen such that  $Q - \text{Diag}(u) \succeq 0$ . A simple choice is  $u_i = -\lambda_1(Q)$ ,  $i \in [n]$ , where  $\lambda_1(Q)$  denotes the minimal eigenvalue of the matrix  $Q$ , see also Hammer and Rubin [1970]. Since  $x_i^2 - x_i < 0$  for  $x_i \in (0, 1)$ , the reformulation yields a less tight continuous relaxation than the original (nonconvex) one<sup>24</sup>, but it allows CPLEX to handle the problem as a convex MIQP.

The MIQP test set has been assembled from all MINLPLib instances that CPLEX could handle, the convex quadratic constrained layout problems `clay*m`, the safety layout problems `SLay*`, and the quadratic uncapacitated facility location problems `uflquad*` from the convex MINLP test set (cf. Section 8.3.2), and all instances from the MIQP test set of H. Mittelmann<sup>25</sup> except for instance `ivalues`, which CPLEX cannot handle. For the instances from the Mittelmann test set, Table A.3 gives detailed problem statistics. Note, that due to SCIP's reformulation of products with binary variable, cf. Section 7.6.2, 15 instances are reformulated as MIP. The complete test set has 91 instances, whereof for 80 instances the optimal value is known.

Next to SCIP and CPLEX, we run COUENNE and LINDOAPI again. We also run BARON, but do not include the results here, since it aborted or failed on 42% of the Mittelmann instances. We also did not include convex MINLP solvers, since not all instances in the test set are convex.

**Statistics.** Table A.8 presents detailed computational results of this comparison and Table 8.9 summarizes the results. We observe, that CPLEX solves 81% of all instances, which is only one instance less than those solved by any solver. The only instances not

<sup>24</sup>In fact, the reformulation by adding the null-term  $\sum_{i=1}^n u_i(x_i^2 - x_i)$  can also be applied if  $Q$  is already positive-semidefinite, which can help to tighten the continuous relaxation, see also Billionnet et al. [2008] and Ahlatçioğlu et al. [2012] for this “de-convexification” approach.

<sup>25</sup><http://plato.asu.edu/ftp/miqp>. In order to run all solvers on the same input, we converted them into GAMS format by using SCIP's reader for MPS files and writer for GAMS files. Since SCIP can only handle linear objective functions, the conversion has the side effect that a quadratic term  $\langle x, Qx \rangle$  in the objective is replaced by an auxiliary variable  $z$  and a constraint  $\langle x, Qx \rangle \leq z$  is added.

	# instances	SCIP	CPLEX	COUENNE	LINDOAPI	virt. best
#solved	91	71	<b>74</b>	55	48	75
#timeout	91	19	16	31	40	16
#failed or aborted	91	<b>1</b>	<b>1</b>	5	3	0
#fastest	91	16	<b>53</b>	5	1	
#best dual bound	91	79	<b>80</b>	58	48	
#best primal bound	91	72	<b>90</b>	58	52	
time (sh. geom. mean)	82	68.7	<b>41.4</b>	195.9	427.7	32.9
nodes (sh. geom. mean)	82	1961.1	1266.4	1660.5	<b>87.4</b>	922.3
dual gap (arith. mean)	72	3.04%	<b>1.39%</b>	6.10%	16.53%	1.11%
primal gap (arith. mean)	72	0.30%	<b>0.01%</b>	5.40%	15.63%	0.01%

Table 8.9.: Performance of different solvers on MIQQP test set.

solved by CPLEX within the time limit are 8 instances from the Mittelmann test set and the MIQPs `pb*` and `qap` from MINLPLib. For the unsolved instances for the Mittelmann test set, CPLEX reports for most of the instances the best primal and dual bounds, and for most of the `pb*` instances, CPLEX reports the best primal bound. However, the dual bound reported by CPLEX on the `pb*` instances is negative, even though 0 would be a trivial and better dual bound here. This is likely due to the convexifying reformulation that adds negative  $x_i^2 - x_i$  terms to the objective function.

SCIP solves all instances that also CPLEX solves, except for `netmod_dol1`, `SLay10H`, and two of the `uflquad*` instances. COUENNE and LINDOAPI solve 19 and 26 instances less than CPLEX, respectively. Further, CPLEX, SCIP, and LINDOAPI are most robust on this test set. CPLEX failed on instance `ibienst1`, SCIP failed on instance `ilaser0`, LINDOAPI failed on instances `iportfolio` and `itointqor` and aborted on instance `clay0303m` with an out-of-memory message, and COUENNE failed on instances `iqap10` and `du-opt*` and aborted on instances `ilaser0` and `imod011` with a segmentation fault. CPLEX is the fastest solver for 58% of all instances, but SCIP is also the fastest solver on 18% of the instances. The detailed results show, that there are several Mittelmann instances, where SCIP is considerably faster than CPLEX (and vice-versa). The best dual bounds are computed most often by CPLEX (88%) and SCIP (87%). Also the best feasible solutions are found most often by CPLEX (99%) and SCIP (79%).

The mean values on time, primal gap, and dual gap for those 82 instances where no solver failed show a similar picture. SCIP's mean time is 65% above the one of CPLEX, while COUENNE's and LINDOAPI's mean time are 4.7 and 10.3 times the one of CPLEX, respectively. The average dual gap for SCIP is twice the one of CPLEX, while the factor is much higher when comparing COUENNE or LINDOAPI with CPLEX. The average primal gap for SCIP is 0.3%, while it is almost zero for CPLEX.

**Performance Profiles.** Figure 8.5 shows the performance profiles. We observe, that within 1000 seconds, CPLEX and SCIP have solved about the same number of instances,

## 8. Computational Study

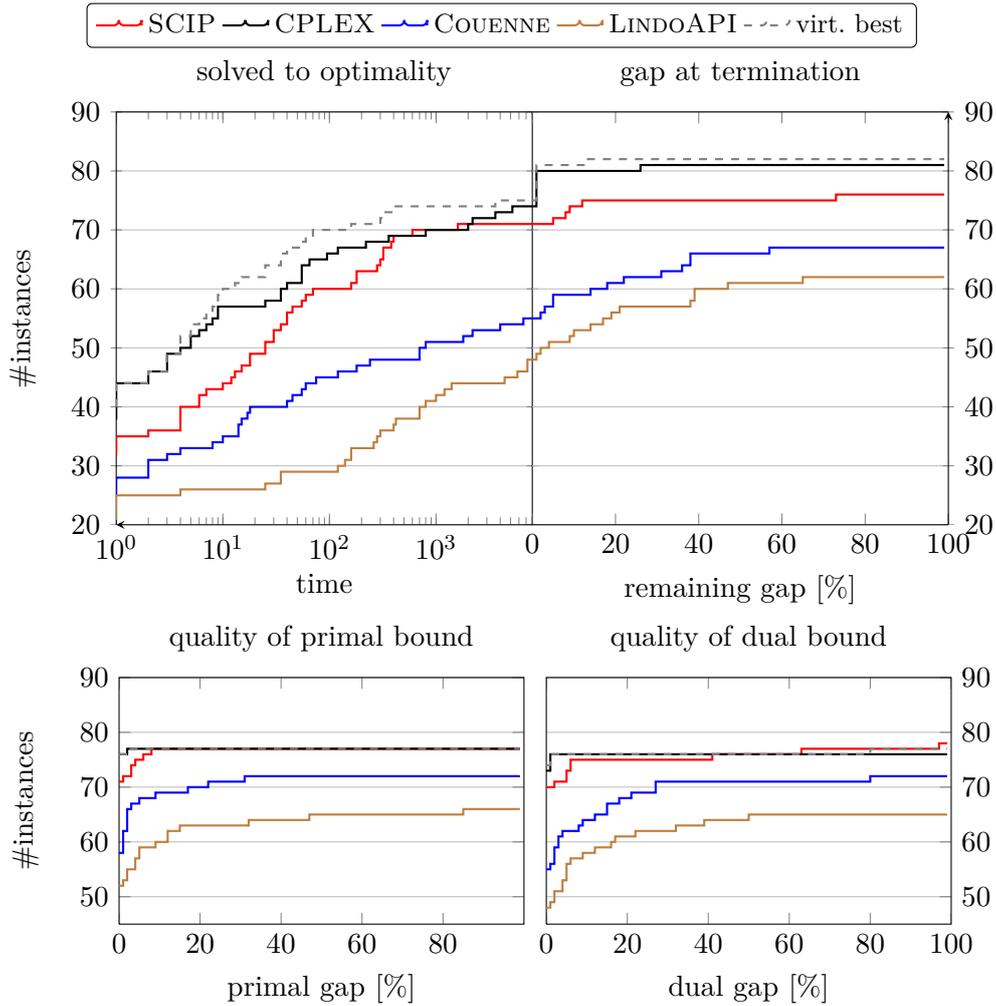


Figure 8.5.: Performance profiles for MIQP test set (91 instances in total, 80 with known optimal value).

but when the time limit is reached, CPLEX has closed the gap to the virtually best solver. From the remaining unsolved instances, there are about five instances with a very small gap at termination for CPLEX, while most of the remaining ones have a very large gap at termination (obviously, these are the  $\text{pb}^*$  instances).

### 8.3.4. MISOCPS

Finally, we consider instances from the portfolio optimization test set of Vielma et al. [2008]. All instances are MISOCs, the problem statistics are summarized in Table 8.10. In comparison to the previous test sets which comprised instances from various sources, here all instances have their origin in the same application. The `classical_k_*` instances contain one convex quadratic constraint of the form  $\sum_{i \in [k]} x_i^2 \leq u$  for some  $u \in \mathbb{Q}$ . The

instance	original problem				presolved problem				
	$n$	bin	int	$m' + m$	$n$	bin	int	$m'$	$m$
classical_k_*	$3k$	$k$	0	$2k + 3$	$3k$	$k$	0	$2k + 2$	1
robust_k_*	$4k + 3$	$k + 1$	0	$3k + 6$	$4k + 3$	$k + 1$	0	$3k + 4$	2
shortfall_k_*	$4k + 4$	$k + 1$	0	$3k + 7$	$4k + 4$	$k + 1$	0	$3k + 5$	2

Table 8.10.: MISOCP test set problem statistics. In some cases, SCIP can fix a few variables in the presolved problem, which reduces  $n$  and  $m'$ .

	# instances	SCIP	CPLEX	MOSEK	virt. best
#solved	170	<b>135</b>	134	<b>135</b>	142
#timeout	170	35	36	34	28
#failed or aborted	170	<b>0</b>	<b>0</b>	1	0
#fastest	170	32	<b>72</b>	38	
#best dual bound	170	140	<b>147</b>	146	
#best primal bound	170	153	<b>157</b>	149	
time (sh. geom. mean)	169	<b>95.1</b>	<b>87.7</b>	<b>88.6</b>	58.6
dual gap (arith. mean)	144	0.02%	0.04%	<b>0.02%</b>	0.00%
primal gap (arith. mean)	144	<b>0.00%</b>	0.00%	0.00%	0.00%

Table 8.11.: Performance of different solvers on MISOCP test set.

`robust_k_*` instances contain one convex quadratic and one SOC constraint of dimension  $k$ . The `shortfall_k_*` instances contain two SOC constraints of dimension  $k$ . For `classical_k_*`, instances for dimension  $k \in \{20, 30, 40, 50, 200\}$  were available. For `robust_k_*` and `shortfall_k_*`, instances for dimension  $k \in \{20, 30, 40, 50, 100, 200\}$  were available. For each dimension, we took the first 10 available instances. Thus, the whole test set consists of 170 instances, the optimal value is known for 145 instances.

On this test set, we compared SCIP with two other solvers for MISOCPs, which are CPLEX and MOSEK. We did not run any of the solvers for general or convex MINLPs, since they do not recognize the SOC structure (see Berthold et al. [2009b] for a comparison that includes BARON, COUENNE, and LINDOAPI).

**Statistics.** Table A.9 presents detailed computational results of this comparison and Table 8.11 summarizes the results. We observe, that SCIP, CPLEX, and MOSEK solve 79% of all instances, whereas the percentage of instances solved by any solver is 84%, which is due to a few instances that are solved only by one of the solvers. MOSEK aborted on the instance `shortfall_100_5` due to numerical problems.

CPLEX is the fastest solver approximately twice as often as SCIP and MOSEK. The mean solution times of MOSEK and SCIP are only 1% and 8%, respectively, higher than the one of CPLEX. However, CPLEX's mean solution time is 50% above the one of the virtually best solver.

## 8. Computational Study

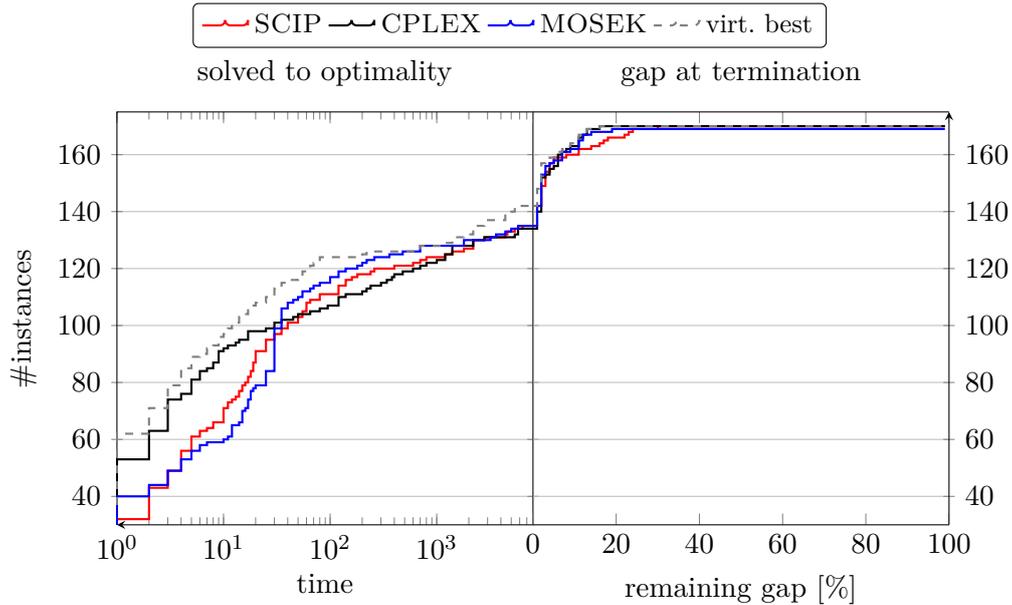


Figure 8.6.: Performance profiles for MISOCP test set (170 instances in total).

**Performance Profiles.** Figure 8.6 shows the performance profiles. We did not include profiles for primal and dual gap here, because these gaps were already close to zero when a solver hit the time limit. It is seen, that SCIP, CPLEX, and MOSEK behave quite similar, but for time points before 20 seconds, CPLEX has solved most instances.

### 8.4. Impact of Solver Components

In the following, we investigate the impact of certain individual SCIP components onto the computational performance for MINLP solving. The analysis is inspired by the publication Berthold, Gleixner, Heinz, and Vigerske [2012], where the impact of individual components of an earlier version of SCIP was investigated for a set of MIQCPs.

For our experiments, we selected a set of 230 publicly available MINLP instances from the test sets considered in the previous section (MINLPLib, convex MINLPs, Mittelmann’s MIQPs, MISOCPs) in the following way. From the union of all test sets, we excluded all seemingly trivial instances by removing instances where SCIP in default settings requires less than 10 seconds and less than 100 nodes. Then we removed instances where SCIP in default settings failed or aborted or terminated with an infinite dual bound. Finally, when there were many instances generated apparently from the same model, we removed some of them. The instances taken from MINLPLib and the sets of convex MINLPs and Mittelmann MIQPs are marked by a “\*” in front of the instance name in the Tables A.1–A.3. Additionally, from the MISOCP test set, the instances `classical_{20,50,200}_{0,1}`, `shortfall_{20,50,100,200}_{0,1}`, and `robust_{20,50,100,200}_{0,1}` are used.

To measure the impact of individual components, we compare the default run (see Section 8.3) to the performance with a feature disabled or switched to a different strategy. Since many MINLP instances contain a considerable linear and discrete part, we also investigate the effect of the classical MIP components. All in all, we compared 18 alternative settings against the SCIP default:

quad. reform off	no linear reformulation of products $x_i \sum_j Q_{i,j} x_j$ with $x_i$ a binary variable (cf. Section 7.6.2)
quad. reform aggr.	linear reformulation of each product $x_i x_j$ with $x_i$ a binary variable separately ( $n = 1$ in Section 7.6.2), which leads to more variables and constraints, but also a tighter relaxation
expr. reform off	no reformulation of expression graph (cf. Section 7.6.1), i.e., indefinite terms $h_j(\cdot)$ in general nonlinear constraints are underestimated by interval-gradient cuts (cf. Section 7.5.1)
SOC recogn. off	no recognition of second-order cone constraints (cf. Section 7.5.3), i.e., quadratic constraints are handled like nonconvex constraints, including spatial branching
linear presolve off	no presolving for linear constraints, e.g., no upgrades to specialized constraint handlers, no coefficient tightening [Achterberg, 2007, Section 10.1]
heuristics off	no heuristics at all, i.e., feasible solutions can only be found by tree search
NLP heur. off	no NLP local search to repair violated nonlinear constraints in solutions of MIP relaxation (cf. Section 7.8.1)
LNS heur. aggr.	call large-neighborhood search heuristics more often (cf. Section 7.8.3); in default settings, only CROSSOVER is called during tree search and RENS is called at the root node
heuristics aggr.	call all heuristics more often
nonlin. sepa. off	disable separation routines in MINLP constraint handlers, i.e., outer-approximation is improved only when nonlinear constraints are enforced
MIP cuts off	disable MIP cutting plane separators
MIP cuts aggr.	run MIP cutting plane separators more often (default is to separate only at root node)
inference branch.	change branching rule to inference branching, i.e., prefer branching on variables which lead to many bound tightenings before (default is reliability branching for fractional variables and pseudo-cost branching for nonlinear variables, cf. Section 6.1.4)
most inf. branch.	change branching rule to most infeasible branching, i.e., score integer variables by fractionality and nonlinear variables by “variable infeasibility” (cf. Section 6.1.4)
random branch.	change branching rule to random branching, i.e., choose branching variable uniformly at random from all branching candidates

## 8. Computational Study

depth-first search	explore branch-and-bound tree by depth first search; default is to select nodes by a best estimate rule [Achterberg, 2007, Chapter 6]
no domain prop.	disable domain propagation (cf. Sections 6.1.5, 7.3.2, and 7.4), i.e., bound tightening may only be done during constraint enforcement
no conflicts	disable conflict analysis, i.e., do not learn conflict clauses from infeasible subproblems [Achterberg, 2007, Chapter 11]

With default settings, SCIP could solve 131 of the 230 instances within the time limit of two hours. For 11 of the unsolved instances, no feasible solution was found. Table 8.13 shows the difference when a particular setting is used. Obviously, some of the features may only have an effect on a certain subset of the test set, e.g., disabling recognition of SOC constraints only has an effect if such constraints are present in the model. For such settings, we split the test set in a “relevant” and a “control” group, expecting no change in the performance for the control group. Column “size” gives the number of instances in the respective test group. For some settings, SCIP fails on a few instances (e.g., compute a wrong dual bound). These fails lead to a reduction in the size of the test set that is considered for this setting.

All performance measures are with respect to the default settings of SCIP. As a rough indicator of the usefulness of a component, the third column reports how many instances more or less were solved. The remaining columns provide a more detailed comparison to the SCIP default. For instances that could not be solved within the time limit, we count on how many instances the primal and dual bounds were “better” or “worse” by at least 2%, respectively. Additionally, we count the absolute number of instances for which a particular setting was more than 10% faster or slower. Further, we compare the shifted geometric mean of the overall running time, the time until the first and a best solution were found<sup>26</sup>, and the number of branch-and-bound nodes.

For each of the eight performance measures we indicate whether it shows an improvement or a degradation: when the performance measure gets worse after disabling a certain component, the corresponding numbers are set in a bold font; when it improves they are set in italics. Loosely speaking, having a row with more bold than italic indicates that a certain setting degrades the performance on the test set.

Table 8.13 shows that each setting leads to at most a small increase in the total number of instances solved and that for each setting the overall computation time does not decrease by more than 1%. This indicates that the default settings are reasonable.

We further see that the MINLP specific features expression graph reformulation, second-order cone recognition, and LP relaxation improvement also in nodes where integrality requirements are still violated have a large impact on the performance. Disabling the reformulation of products with binary variables actually seems to be beneficial<sup>27</sup>, as we can see an improvement in six out of eight performance measures and a degradation

---

<sup>26</sup>The time to a best solution in a particular run refers here to the time where the primal bound was improved the last time. If an instance was solved to optimality, the time to a best solution is the time until an optimal solution is found.

<sup>27</sup>The change in the primal bound for two instances of the control group is due to the use of this reformulation technique in a sub-MINLP in one of SCIP’s primal heuristics.

setting	size solved	better : worse			running time			nodes	
		primal	dual	time	mean to first sol.	to best sol.			
quad. reform off	230	+2	11 : 4	14 : 10	10 : 7	0%	-4%	+3%	-13%
relevant	54	+2	10 : 3	14 : 10	10 : 7	-2%	-13%	+18%	-44%
control	176	0	1 : 1	0 : 0	0 : 0	0%	0%	0%	0%
quad. reform aggr.	228	+1	<b>4 : 6</b>	<b>6 : 11</b>	<b>1 : 5</b>	-1%	+13%	+4%	-12%
relevant	31	+1	<b>4 : 6</b>	<b>6 : 11</b>	<b>1 : 5</b>	-6%	+152%	+74%	-61%
control	197	0	0 : 0	0 : 0	0 : 0	0%	0%	0%	0%
expr. reform off	222	-8	<b>10 : 24</b>	<b>12 : 37</b>	<b>3 : 9</b>	+15%	+18%	+3%	+57%
relevant	68	-8	<b>10 : 24</b>	<b>12 : 37</b>	<b>3 : 9</b>	+62%	+55%	+118%	+337%
control	154	0	0 : 0	0 : 0	0 : 0	0%	0%	0%	0%
SOC recogn. off	230	-7	<b>0 : 2</b>	<b>0 : 10</b>	<b>0 : 11</b>	+20%	+8%	+10%	+26%
relevant	16	-7	<b>0 : 2</b>	<b>0 : 10</b>	<b>0 : 11</b>	+1490%	+156%	+44003%	+2588%
control	214	0	0 : 0	0 : 0	0 : 0	0%	0%	0%	0%
linear presolve off	228	+2	18 : 18	26 : 18	<b>35 : 43</b>	+1%	-2%	+2%	+1%
heuristics off	229	-2	<b>4 : 46</b>	35 : 21	34 : 34	+9%	+273%	+21%	+15%
NLP heur. off	230	-4	<b>0 : 33</b>	13 : 13	<b>11 : 29</b>	+10%	+133%	+21%	+13%
LNS heur. aggr.	230	+2	19 : 3	<b>14 : 20</b>	<b>10 : 96</b>	+16%	-4%	+39%	-25%
heuristics aggr.	229	+2	28 : 4	22 : 16	<b>16 : 80</b>	+10%	-27%	+19%	-28%
nonlin. sepa. off	230	-14	<b>12 : 23</b>	<b>17 : 65</b>	<b>21 : 65</b>	+57%	+7%	+41%	+231%
MIP cuts off	227	0	16 : 7	24 : 23	37 : 35	+10%	0%	+7%	+27%
MIP cuts aggr.	227	-1	<b>9 : 19</b>	31 : 20	<b>25 : 68</b>	+18%	+30%	+44%	-9%
inference branch.	229	-5	<b>7 : 10</b>	<b>18 : 40</b>	<b>10 : 69</b>	+23%	-6%	+44%	+43%
most inf. branch.	230	-20	14 : 13	<b>7 : 68</b>	<b>12 : 68</b>	+79%	-8%	+84%	+104%
random branch.	229	-27	<b>10 : 21</b>	4 : 81	<b>7 : 80</b>	+110%	-2%	+117%	+168%
depth-first search	229	-18	<b>8 : 44</b>	<b>2 : 85</b>	<b>12 : 64</b>	+71%	+7%	+136%	+150%
no domain prop.	225	-5	<b>6 : 30</b>	<b>11 : 53</b>	<b>22 : 59</b>	+20%	+75%	+51%	+68%
no conflicts	230	+1	6 : 6	<b>7 : 18</b>	<b>13 : 30</b>	0%	+11%	+11%	+2%

Table 8.13.: Impact of SCIP components. Column “size” gives the number of instances in the test group. Performance measures are absolute/relative differences compared to SCIP with default settings.

in only one measure. Replacing each single product that involves at least one binary variable by a new auxiliary variable leads to a high increase in the time to find a first or best solution, but also reduces the number of nodes to be explored, probably because the relaxation is now more difficult to solve (also the dual bound worsens in eleven cases, while it improves in only six cases).

Domain propagation works on the linear and the nonlinear part, additionally exploiting

## 8. Computational Study

global information. It gives a clear benefit w.r.t. the computation time and even more w.r.t. the number of branch-and-bound nodes and the dual bound for unsolved instances.

Primal heuristics slightly improve the overall computation time, but very much help to find a first feasible solution and to obtain a good primal bound if a run has to be terminated due to a time limit. Disabling all primal heuristics or only the NLP local search heuristic increase the overall computation time only moderately, but very much increase the time to find a first feasible solution or to obtain a good primal bound if a run has to be terminated due to a time limit. Using sub-MINLP heuristics more aggressively improves the primal bound on 19 instances and helps to solve two more instances, but also increases running time. A similar effect is seen when running all heuristics more aggressively, even though the increase on the running time is lower and the times until a first or a best solution is found are much more reduced here.

MIP specific features like linear presolving, MIP cuts, and conflict analysis are less successful than in pure MIP. Disabling MIP cuts still increases running time and number of branch-and-bound nodes, while disabling the linear presolving routines can even be slightly advantageous. We note, that most of the MINLP constraint handler do not support conflict analysis yet, which may explain why disabling it has only little effect.

Using sophisticated branching and node selection rules is extremely important. The deterioration due to switching to inference branching is less than when switching to most infeasible or random branching.

Altogether, most of the components turned out to be beneficial for the performance, only the reformulation of products with binary variables is on the borderline. These examples show that often using more than one criterion for measuring performance gives a better picture of the overall behavior.

# A. Tables

## A.1. Problem Statistics

The following tables show statistics on the problem instances considered in the test sets in Section 8.3. We give the total number of variables ( $n$ ), the number of binary variables, the number of non-binary integer variables, the number of linear constraints ( $m'$ ) and the number of nonlinear constraints ( $m$ ). We show the statistics for both the original problem<sup>1</sup> and the presolved problem in SCIP. Further, we give the optimal value, if known, otherwise, a best known upper bound (if known). If an instance is included into the test set for the study in Section 8.4, it is marked with a \* in front of its name.

### A.1.1. MINLPLib

Table A.1.: MINLPLib problem statistics

instance	opt. value	original problem					presolved problem				
		$n$	bin	int	$m'$	$m$	$n$	bin	int	$m'$	$m$
* 4stufen	$\leq 116329.671$	150	48	0	64	34	125	48	0	21	55
alan	2.925	9	4	0	7	1	9	4	0	7	1
batchdes	167427.657	20	9	0	18	2	14	5	0	11	2
batch	285506.508	47	24	0	72	2	43	20	0	67	2
* beuster	$\leq 116347.950$	158	52	0	67	47	164	51	0	108	62
* cecil_13	-115656.528	840	180	0	718	180	659	96	0	736	174
* chp_partload	$\leq 23.554$	2248	45	0	2026	490	1244	42	0	528	981
contvar	$\leq 809149.827$	297	88	0	165	120	735	87	0	204	531
* csched1a	-30430.177	29	15	0	22	1	34	12	0	13	12
* csched1	-30639.258	77	63	0	22	1	80	60	0	13	10
* csched2a	$\leq -165398.701$	233	140	0	137	1	291	140	0	109	87
* csched2	$\leq -166101.996$	401	308	0	137	1	457	308	0	109	85
du-opt5	8.074	21	0	13	9	1	19	0	11	4	1
* du-opt	3.556	21	0	13	9	1	21	0	13	5	1
* eg_all_s	$\leq 7.658$	8	0	7	0	28	46546	2	5	0	46570
* eg_disc2_s	$\leq 5.642$	8	0	3	0	28	46558	0	3	0	46578
* eg_disc_s	$\leq 5.761$	8	0	4	0	28	46552	0	4	0	46572
* eg_int_s	$\leq 6.453$	8	0	3	0	28	46546	2	1	0	46570
* elf	0.192	54	24	0	11	27	54	24	0	11	27

continue next page...

<sup>1</sup>Note, that when a MINLP with nonlinear objective function is passed to SCIP, then an additional variable and constraint is added.

A. Tables

... continued statistics for MINLPLib instances											
instance	opt. value	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>
* eniplac	-132117.083	141	24	0	165	24	122	23	0	131	24
* enpro48	187277.259	154	92	0	213	2	154	92	0	213	2
* enpro48pb	187277.259	154	92	0	213	2	154	92	0	213	2
* enpro56	263428.301	128	73	0	190	2	125	70	0	190	2
* enpro56pb	263428.301	128	73	0	190	2	125	70	0	190	2
ex1221	7.667	5	3	0	3	2	5	3	0	3	2
ex1222	1.077	4	1	0	2	2	2	0	0	0	2
ex1223a	4.580	8	4	0	5	5	9	4	0	7	4
ex1223b	4.580	8	4	0	5	5	9	4	0	7	4
ex1223	4.580	12	4	0	9	5	9	4	0	7	4
ex1224	-0.944	12	8	0	4	4	13	8	0	4	5
ex1225	31.000	8	6	0	9	1	8	4	0	4	3
ex1226	-17.000	5	3	0	4	1	8	3	0	4	4
* ex1233	155010.671	53	12	0	64	1	83	10	0	46	41
ex1243	83402.506	69	16	0	96	1	64	11	0	63	13
* ex1244	82042.905	96	23	0	129	1	90	15	0	86	18
ex1252a	128893.741	25	3	6	22	13	41	3	6	28	25
ex1252	128893.741	40	15	0	31	13	52	14	0	34	25
* ex1263a	19.600	24	4	20	31	4	23	3	20	27	4
* ex1263	19.600	92	72	0	51	4	91	71	0	47	4
ex1264a	8.600	24	4	20	31	4	23	3	20	27	4
* ex1264	8.600	88	68	0	51	4	82	62	0	47	4
* ex1265a	10.300	35	5	30	39	5	34	4	30	35	5
* ex1265	10.300	130	100	0	69	5	122	92	0	65	5
ex1266a	16.300	48	6	42	47	6	46	4	42	39	6
ex1266	16.300	180	138	0	89	6	168	126	0	81	6
ex3	68.010	33	8	0	26	5	23	7	0	20	5
ex3pb	68.010	33	8	0	26	5	23	7	0	20	5
ex4	-8.064	37	25	0	5	26	37	25	0	4	26
fac1	160912610.000	23	6	0	18	1	21	4	0	18	1
fac2	331837500.000	67	12	0	33	1	67	12	0	39	1
fac3	31982310.000	67	12	0	33	1	67	12	0	39	1
feedtray2	0.000	88	36	0	137	147	300	12	0	1001	147
* feedtray	-13.406	98	7	0	30	62	251	7	0	19	224
fo7_2	17.749	114	42	0	197	14	82	42	0	137	14
* fo7_ar2_1	24.840	112	0	42	255	14	81	1	40	164	14
* fo7_ar25_1	23.094	112	0	42	255	14	81	1	40	164	14
fo7_ar3_1	22.517	112	0	42	255	14	81	1	40	164	14
fo7_ar4_1	20.730	112	0	42	255	14	81	1	40	164	14
* fo7_ar5_1	17.749	112	0	42	255	14	81	1	40	164	14
* fo7	20.730	114	42	0	197	14	82	42	0	137	14
* fo8_ar2_1	30.341	144	0	56	331	16	101	1	54	204	16
* fo8_ar25_1	28.045	144	0	56	331	16	101	1	54	204	16

continue next page...

... continued statistics for MINLPLib instances

instance	opt. value	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>
fo8_ar3_1	23.910	144	0	56	331	16	101	1	54	204	16
fo8_ar4_1	22.382	144	0	56	331	16	101	1	54	204	16
* fo8_ar5_1	22.382	144	0	56	331	16	101	1	54	204	16
* fo8	22.382	146	56	0	257	16	102	56	0	173	16
* fo9_ar2_1	32.625	180	0	72	417	18	123	1	70	248	18
* fo9_ar25_1	32.186	180	0	72	417	18	123	1	70	248	18
fo9_ar3_1	24.816	180	0	72	417	18	123	1	70	248	18
fo9_ar4_1	23.464	180	0	72	417	18	123	1	70	248	18
* fo9_ar5_1	23.464	180	0	72	417	18	123	1	70	248	18
* fo9	23.464	182	72	0	325	18	124	72	0	213	18
fuel	8566.119	16	3	0	12	4	13	2	0	9	4
* fuzzy	$\leq -0.538$	897	120	0	1038	19	819	110	0	694	55
gasnet	$\leq 6999381.562$	90	10	0	25	44	148	10	0	61	88
gastrans	89.086	106	21	0	125	24	88	12	0	126	22
gbd	2.200	5	3	0	4	1	4	2	0	3	1
* ghg_1veh	7.782	30	12	0	10	28	82	6	0	51	57
* ghg_2veh	$\leq 7.771$	57	18	0	14	48	200	16	0	160	128
* ghg_3veh	$\leq 7.754$	96	36	0	31	88	366	35	0	351	227
* gear2	0.000	29	24	0	4	1	34	24	0	4	6
* gear3	0.000	9	0	4	4	1	10	0	4	0	6
gear4	1.643	6	0	4	0	1	10	0	4	0	5
* gear	0.000	5	0	4	0	1	10	0	4	0	6
gkocis	-1.923	11	3	0	6	2	7	3	0	3	2
* hda	$\leq -5964.534$	723	13	0	568	151	466	7	0	232	233
hmittelman	13.000	17	16	0	0	8	18	18	0	7	8
* johnall	-224.730	195	190	0	2	191	2094	190	0	2	2091
* lop97ic	$\leq 4041.832$	1754	831	831	52	40	5220	708	704	11509	0
* lop97icx	4099.060	987	68	831	48	40	486	68	67	1135	0
m3	37.800	26	6	0	37	6	22	6	0	33	6
* m6	82.257	86	30	0	145	12	62	30	0	101	12
* m7_ar2_1	190.235	112	0	42	255	14	68	12	18	158	14
* m7_ar25_1	143.585	112	0	42	255	14	70	10	22	158	14
m7_ar3_1	143.585	112	0	42	255	14	76	4	34	158	14
m7_ar4_1	106.757	112	0	42	255	14	79	1	40	158	14
* m7_ar5_1	106.460	112	0	42	255	14	79	1	40	158	14
* m7	106.757	114	42	0	197	14	80	42	0	133	14
* mbtd	$\leq 4.833$	210	200	0	50	20	81310	81200	0	360	81000
meanvarx	14.369	36	14	0	44	1	30	12	0	36	1
meanvarxsc	14.369	36	14	0	30	15	23	5	0	17	13
minlphix	316.693	85	20	0	88	5	28	6	0	19	13
* netmod_dol1	-0.560	1999	462	0	3137	1	1993	462	0	3131	1
* netmod_dol2	-0.560	1999	462	0	3080	1	1592	454	0	2637	1
* netmod_kar1	-0.420	457	136	0	666	1	453	136	0	662	1

continue next page...

A. Tables

---

... continued statistics for MINLPLib instances

---

instance	opt. value	$n$	bin	int	$m'$	$m$	$n$	bin	int	$m'$	$m$
* netmod_kar2	-0.420	457	136	0	666	1	453	136	0	662	1
* no7_ar2_1	107.815	112	0	42	255	14	87	1	40	182	14
* no7_ar25_1	107.815	112	0	42	255	14	87	1	40	182	14
* no7_ar3_1	107.815	112	0	42	255	14	87	1	40	182	14
* no7_ar4_1	98.518	112	0	42	255	14	87	1	40	182	14
* no7_ar5_1	90.623	112	0	42	255	14	87	1	40	182	14
* nous1	1.567	51	2	0	15	29	47	2	0	11	29
* nous2	0.626	51	2	0	15	29	46	1	0	10	29
nuclear104		23813	10816	0	11024	3221	12997	10816	0	208	3221
* nuclear10a		13010	10920	0	209	3130	23826	10920	0	32761	3026
* nuclear10b		23826	10920	0	21945	3026	23826	10920	0	21945	3026
nuclear14a	$\leq -1.128$	992	600	0	49	584	1568	600	0	1801	560
nuclear14b	$\leq -1.117$	1568	600	0	1225	560	1568	600	0	1225	560
nuclear14	$\leq -1.128$	1562	576	0	624	602	986	576	0	48	602
* nuclear24a	$\leq -1.128$	992	600	0	49	584	1568	600	0	1801	560
* nuclear24b	$\leq -1.117$	1568	600	0	1225	560	1568	600	0	1225	560
nuclear24	$\leq -1.128$	1562	576	0	624	602	986	576	0	48	602
nuclear25a	$\leq -1.120$	1058	650	0	51	608	1683	650	0	1951	583
nuclear25b	$\leq -1.101$	1683	650	0	1326	583	1683	650	0	1326	583
nuclear25	$\leq -1.119$	1678	625	0	675	628	1053	625	0	50	628
* nuclear49a	$\leq -1.151$	3341	2450	0	99	1332	5742	2450	0	7351	1283
* nuclear49b	$\leq -1.117$	5742	2450	0	4950	1283	5742	2450	0	4950	1283
nuclear49	$\leq -1.151$	5735	2401	0	2499	1374	3334	2401	0	98	1374
nuclearva	$\leq -1.012$	351	168	0	50	267	327	144	0	24	267
nuclearvb	$\leq -1.030$	351	168	0	50	267	327	144	0	24	267
nuclearvc	$\leq -0.998$	351	168	0	50	267	327	144	0	24	267
nuclearvd	$\leq -1.034$	351	168	0	50	267	327	144	0	24	267
nuclearve	$\leq -1.035$	351	168	0	50	267	327	144	0	24	267
nuclearvf	$\leq -1.019$	351	168	0	50	267	327	144	0	24	267
nvs01	12.470	4	0	2	1	3	10	0	2	0	9
nvs02	5.964	9	0	5	0	4	9	0	5	0	4
nvs03	16.000	3	0	2	1	2	3	0	2	1	2
nvs04	0.720	3	0	2	0	1	4	0	2	0	2
nvs05	5.471	9	0	2	1	9	27	0	2	1	27
nvs06	1.770	3	0	2	0	1	10	0	2	0	8
nvs07	4.000	4	0	3	1	2	5	0	3	1	3
nvs08	23.450	4	0	2	0	4	5	0	2	1	4
* nvs09	-43.134	11	0	10	0	1	40	0	10	0	30
nvs10	-310.800	3	0	2	0	3	3	0	2	0	3
nvs11	-431.000	4	0	3	0	4	4	0	3	0	4
nvs12	-481.200	5	0	4	0	5	5	0	4	0	5
nvs13	-585.200	6	0	5	0	6	6	0	5	0	6
nvs14	-40358.155	9	0	5	0	4	9	0	5	0	4

---

continue next page...

---

... continued statistics for MINLPLib instances

---

instance	opt. value	n	bin	int	m'	m	n	bin	int	m'	m
nvs15	1.000	4	0	3	1	1	5	1	2	4	1
nvs16	0.703	3	0	2	0	1	8	0	2	0	6
nvs17	-1100.400	8	0	7	0	8	8	0	7	0	8
nvs18	-778.400	7	0	6	0	7	7	0	6	0	7
nvs19	-1098.400	9	0	8	0	9	9	0	8	0	9
* nvs20	230.922	17	0	5	8	1	33	0	5	8	17
nvs21	-5.685	4	0	2	0	3	9	0	2	0	8
nvs22	6.058	9	0	4	1	9	27	0	4	1	27
* nvs23	-1125.200	10	0	9	0	10	10	0	9	0	10
* nvs24	-1033.200	11	0	10	0	11	11	0	10	0	11
o7_2	116.946	114	42	0	197	14	90	42	0	153	14
* o7_ar2_1	140.412	112	0	42	255	14	89	1	40	188	14
* o7_ar25_1	140.412	112	0	42	255	14	89	1	40	188	14
o7_ar3_1	137.932	112	0	42	255	14	89	1	40	188	14
o7_ar4_1	131.653	112	0	42	255	14	89	1	40	188	14
* o7_ar5_1	116.946	112	0	42	255	14	89	1	40	188	14
* o7	131.653	114	42	0	197	14	90	42	0	153	14
* o8_ar4_1	243.071	144	0	56	331	16	117	1	54	252	16
* o9_ar4_1	236.138	180	0	72	417	18	137	1	70	290	18
oaer	-1.923	9	3	0	5	2	8	3	0	4	2
* oil2	-0.733	937	2	0	643	284	1065	2	0	119	952
* oil	-0.932	1535	19	0	1128	418	1406	19	0	407	1143
ortez	-9532.039	87	18	0	47	27	88	15	0	46	30
parallel	924.296	206	25	0	111	5	259	20	0	103	114
* pb302035	≤ 3379359.000	601	600	0	50	1	1180	600	0	1790	0
pb302055	≤ 3599602.000	601	600	0	50	1	1152	585	0	1751	0
pb302075	≤ 4050938.000	601	600	0	50	1	1074	544	0	1640	0
* pb302095	≤ 5726530.000	601	600	0	50	1	931	469	0	1436	0
* pb351535	≤ 4456670.000	526	525	0	50	1	1035	525	0	1580	0
pb351555	≤ 4639128.000	526	525	0	50	1	1035	525	0	1580	0
pb351575	≤ 6301723.000	526	525	0	50	1	1035	525	0	1580	0
* pb351595	≤ 6670264.000	526	525	0	50	1	1013	514	0	1547	0
prob02	112235.000	6	0	6	3	5	0	0	0	0	0
prob03	10.000	2	0	2	0	1	2	0	2	0	1
protsel	-1.923	10	3	0	5	2	7	3	0	3	2
* product2	≤ -2102.389	2842	128	0	2597	528	668	128	0	338	316
* product	-2142.948	1553	107	0	1793	132	450	92	0	450	86
pump	128893.741	25	3	6	22	13	41	3	6	28	25
* qap	388214.000	226	225	0	30	1	435	225	0	660	0
* qapw	388214.000	451	225	0	255	1	675	225	0	930	0
ravem	269590.219	113	54	0	185	2	109	50	0	185	2
ravempb	269590.219	113	54	0	185	2	109	50	0	185	2
* risk2b	-55.876	464	14	0	580	1	143	12	0	177	1

---

continue next page...

A. Tables

... continued statistics for MINLPLib instances											
instance	opt. value	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>
risk2bpb	-55.876	464	14	0	580	1	143	12	0	177	1
* saa_2	≤ 12.692	4407	400	0	2405	3800	24115	366	0	2170	23743
sep1	-510.081	29	2	0	25	6	19	2	0	15	6
* space25a	≤ 484.329	383	240	0	176	25	308	240	0	101	25
* space25	≤ 484.329	893	750	0	210	25	766	715	0	118	25
* space960	≤ 7610305.246	5537	0	960	5537	960	2657	0	960	2657	960
spectra2	13.978	70	30	0	65	8	68	30	0	30	8
* spring	0.846	18	11	1	3	6	28	11	1	3	16
st_e13	2.000	2	1	0	1	1	2	1	0	1	1
st_e14	4.580	12	4	0	9	5	9	4	0	7	4
st_e15	7.667	5	3	0	3	2	5	3	0	3	2
st_e27	2.000	5	2	0	6	1	0	0	0	0	0
st_e29	-0.944	12	8	0	4	4	13	8	0	4	5
* st_e31	-2.000	112	24	0	130	5	59	12	0	54	5
* st_e32	-1.430	36	1	18	6	13	64	0	18	4	43
* st_e35	64868.077	33	7	0	39	1	61	7	0	30	33
* st_e36	-246.000	3	0	1	0	3	13	0	1	0	14
st_e38	7197.727	5	0	2	2	2	9	0	2	2	6
st_e40	30.414	4	0	3	4	4	19	0	3	4	22
st_miqp1	281.000	6	5	0	1	1	4	4	0	2	0
st_miqp2	2.000	5	2	2	3	1	5	2	2	3	1
st_miqp3	-6.000	3	0	2	1	1	2	0	1	0	1
st_miqp4	-4574.000	7	3	0	4	1	6	2	0	3	1
st_miqp5	-333.889	8	2	0	13	1	8	2	0	9	1
* stockcycle	119948.688	481	432	0	97	1	481	432	0	97	1
st_test1	0.000	6	5	0	1	1	0	0	0	0	0
st_test2	-9.250	7	5	1	2	1	0	0	0	0	0
st_test3	-7.000	14	10	3	10	1	0	0	0	0	0
st_test4	-7.000	7	2	4	5	1	7	2	4	5	1
st_test5	-110.000	11	10	0	11	1	0	0	0	0	0
st_test6	471.000	11	10	0	5	1	0	0	0	0	0
st_test8	-29605.000	25	0	24	20	1	25	0	24	10	1
st_testgr1	-12.812	11	0	10	5	1	11	0	10	5	1
st_testgr3	-20.590	21	0	20	20	1	21	0	20	20	1
st_testph4	-80.500	4	0	3	10	1	4	0	3	4	1
super1	≤ 9.508	1308	44	0	1285	374	931	31	0	595	667
super2	≤ 4.934	1308	44	0	1285	374	932	31	0	595	667
super3	≤ 12.628	1308	44	0	1285	374	946	37	0	606	672
super3t	≤ -0.669	1056	44	0	1103	240	677	37	0	488	449
* synheat	154997.335	57	12	0	64	1	75	10	0	42	33
* synthes1	6.010	7	3	0	4	3	6	2	0	4	3
* synthes2	73.035	12	5	0	11	4	11	4	0	9	4
* synthes3	68.010	18	8	0	19	5	17	7	0	17	5

continue next page...

... continued statistics for MINLPLib instances

instance	opt. value	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>
* tln12	90.500	168	12	156	60	12	180	24	144	85	11
tln2	5.300	8	2	6	10	2	6	0	6	4	2
* tln4	8.300	24	4	20	20	4	24	4	20	16	4
* tln5	10.300	35	5	30	25	5	35	5	30	20	5
* tln6	15.300	48	6	42	30	6	48	6	42	24	6
* tln7	15.000	63	7	56	35	7	63	7	56	28	7
tloss	16.300	48	6	42	47	6	46	4	42	39	6
* tlns12	≤ 112.800	812	656	12	372	12	743	429	0	936	35
tls2	5.300	37	31	2	22	2	26	19	0	24	2
* tlns4	8.300	105	85	4	60	4	124	85	0	101	13
* tlns5	10.300	161	131	5	85	5	214	131	0	217	14
* tlns6	≤ 15.300	215	173	6	114	6	278	165	0	316	16
* tlns7	≤ 15.500	345	289	7	147	7	362	257	0	159	56
tltr	48.067	48	12	36	51	3	56	20	27	73	2
useliner	≤ -1050.336	6793	58	0	1265	5765	6285	58	0	2028	5025
* util	999.579	146	28	0	164	4	32	12	0	11	4
* waste	598.919	2484	400	0	623	1368	1238	400	0	516	2140
* water3	≤ 907.016	195	28	0	122	29	207	28	0	120	43
water4	≤ 907.017	195	126	0	122	15	207	126	0	120	29
* waterful2	≤ 1012.609	629	56	0	326	113	683	56	0	324	169
* watersbp	≤ 913.776	195	28	0	122	29	207	28	0	120	43
* waters	≤ 913.776	195	14	0	122	29	207	14	0	120	43
* watersym1	≤ 913.776	321	28	0	172	57	347	28	0	170	85
* watersym2	≤ 940.894	321	28	0	172	57	320	24	0	154	79
* waterx	≤ 909.040	70	14	0	38	16	96	14	0	36	44
waterz	≤ 907.017	195	126	0	122	15	207	126	0	120	29

### A.1.2. Convex MINLPs

Table A.2.: convex MINLP test set problem statistics

instance	opt. value	original problem					presolved problem				
		<i>n</i>	bin	int	<i>m'</i>	<i>m</i>	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>
* BatchS101006M	769440.420	279	129	0	1018	2	270	120	0	689	2
* BatchS121208M	1241125.510	407	203	0	1510	2	395	191	0	1042	2
* BatchS151208M	1543472.400	446	203	0	1780	2	433	190	0	1212	2
* BatchS201210M	2295348.850	559	251	0	2326	2	533	225	0	1576	2
clay0203h	41573.302	90	18	0	108	24	303	15	0	141	228
clay0203m	41572.982	30	18	0	30	24	27	15	0	27	24
* clay0204h	6545.000	164	32	0	202	32	448	28	0	246	304
* clay0204m	6545.000	52	32	0	58	32	48	28	0	54	32
clay0205h	8092.500	260	50	0	325	40	615	45	0	380	380

continue next page...

A. Tables

---

... continued statistics for convex MINLP test set instances

---

instance	opt. value	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>
* clay0205m	8092.500	80	50	0	95	40	75	45	0	90	40
clay0303h	26669.134	99	21	0	114	36	405	21	0	114	342
clay0303m	26668.420	33	21	0	30	36	31	19	0	29	36
* clay0304h	40262.424	176	36	0	210	48	584	36	0	210	456
* clay0304m	40262.100	56	36	0	58	48	54	34	0	57	48
clay0305h	8092.500	275	55	0	335	60	785	55	0	335	570
* clay0305m	8092.500	85	55	0	95	60	81	51	0	93	60
* FLay03H	48.990	122	12	0	141	3	122	12	0	141	3
* FLay03M	48.990	26	12	0	21	3	26	12	0	21	3
FLay04H	54.406	234	24	0	278	4	234	24	0	278	4
* FLay04M	54.406	42	24	0	38	4	42	24	0	38	4
* FLay05H	64.498	382	40	0	460	5	382	40	0	460	5
* FLay05M	64.498	62	40	0	60	5	62	40	0	60	5
FLay06H	66.933	566	60	0	687	6	566	60	0	687	6
* FLay06M	66.933	86	60	0	87	6	86	60	0	87	6
* nd-10	52.461	336	28	0	149	28	363	27	0	147	56
* nd-12	56.670	560	40	0	209	40	582	34	0	186	80
* nd-13	63.005	660	44	0	240	44	704	44	0	240	88
* nd-15	74.192	850	50	0	306	50	898	48	0	302	100
* nd-16	97.979	936	52	0	341	52	967	47	0	317	104
* RSyn0810H	1721.448	343	42	0	477	6	190	40	0	187	32
RSyn0810M02H	1741.388	790	168	0	1176	12	431	119	0	506	68
RSyn0810M02M	1741.387	410	168	0	854	12	223	119	0	583	12
* RSyn0810M03H	2722.449	1185	252	0	1917	18	656	198	0	922	104
* RSyn0810M03M	2722.448	615	252	0	1434	18	342	198	0	999	18
RSyn0810M04H	6581.937	1580	336	0	2760	24	889	277	0	1421	140
RSyn0810M04M	6581.937	820	336	0	2116	24	469	277	0	1518	24
* RSyn0810M	1721.448	185	74	0	306	6	80	40	0	166	6
* RSyn0820H	1150.301	417	52	0	590	14	263	48	0	260	74
RSyn0820M02H	1092.092	978	208	0	1472	28	587	145	0	680	152
RSyn0820M02M	1092.093	510	208	0	1046	28	283	145	0	697	28
* RSyn0820M03H	2028.813	1467	312	0	2406	42	895	242	0	1215	230
* RSyn0820M03M	2028.812	765	312	0	1767	42	437	242	0	1202	42
RSyn0820M04H	2450.773	1956	416	0	3472	56	1211	339	0	1853	308
RSyn0820M04M	2450.773	1020	416	0	2620	56	599	339	0	1828	56
* RSyn0820M	1150.301	215	84	0	357	14	105	48	0	209	14
* RSyn0840H	325.555	568	72	0	809	28	397	65	0	388	148
RSyn0840M02H	734.984	1360	288	0	2050	56	875	199	0	993	300
RSyn0840M02M	734.987	720	288	0	1424	56	407	199	0	918	56
* RSyn0840M03H	2742.646	2040	432	0	3363	84	1337	333	0	1750	452
* RSyn0840M03M	2742.648	1080	432	0	2424	84	633	333	0	1599	84
RSyn0840M04H	2564.500	2720	576	0	4868	112	1823	467	0	2679	604
RSyn0840M04M	2564.499	1440	576	0	3616	112	883	467	0	2480	112

---

continue next page...

---

... continued statistics for convex MINLP test set instances

---

instance	opt. value	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>	<i>n</i>	bin	int	<i>m'</i>	<i>m</i>
* RSyn0840M	325.555	280	104	0	456	28	157	65	0	291	28
SLay05H	22664.679	231	40	0	290	1	231	40	0	290	1
SLay05M	22664.679	71	40	0	90	1	71	40	0	90	1
* SLay06H	32757.020	343	60	0	435	1	343	60	0	435	1
* SLay06M	32757.020	103	60	0	135	1	103	60	0	135	1
SLay07H	64748.825	477	84	0	609	1	477	84	0	609	1
SLay07M	64748.825	141	84	0	189	1	141	84	0	189	1
* SLay08H	84960.212	633	112	0	812	1	633	112	0	812	1
* SLay08M	84960.212	185	112	0	252	1	185	112	0	252	1
SLay09H	107805.753	811	144	0	1044	1	811	144	0	1044	1
SLay09M	107805.753	235	144	0	324	1	235	144	0	324	1
* SLay10H	129579.884	1011	180	0	1305	1	1011	180	0	1305	1
* SLay10M	129579.884	291	180	0	405	1	291	180	0	405	1
* sssd-10-4-3	152709.944	68	52	0	30	12	72	52	0	30	16
* sssd-12-5-3	261142.543	95	75	0	37	15	100	75	0	37	20
* sssd-15-6-3	440748.198	132	108	0	45	18	138	108	0	45	24
sssd-16-8-3	615763.366	184	152	0	56	24	192	152	0	56	32
sssd-18-8-3	524062.788	200	168	0	58	24	208	168	0	58	32
* sssd-20-9-3	478093.435	243	207	0	65	27	252	207	0	65	36
sssd-22-8-3	595970.689	232	200	0	62	24	240	200	0	62	32
* sssd-8-4-3	197181.071	60	44	0	28	12	64	44	0	28	16
Syn10H	1267.354	77	10	0	106	6	61	8	0	54	32
Syn10M02H	2310.301	194	40	0	282	12	133	23	0	130	68
Syn10M02M	2310.302	110	40	0	186	12	61	23	0	99	12
* Syn10M03H	3354.683	291	60	0	468	18	205	38	0	221	104
* Syn10M03M	3354.683	165	60	0	324	18	95	38	0	175	18
Syn10M04H	4557.063	388	80	0	684	24	277	53	0	327	140
Syn10M04M	4557.062	220	80	0	492	24	129	53	0	266	24
Syn10M	1267.354	35	10	0	48	6	27	8	0	38	6
* Syn20H	924.263	151	20	0	219	14	134	16	0	126	74
Syn20M02H	1752.133	382	80	0	578	28	289	49	0	302	152
Syn20M02M	1752.133	210	80	0	378	28	121	49	0	211	28
* Syn20M03H	2646.951	573	120	0	957	42	444	82	0	511	230
* Syn20M03M	2646.951	315	120	0	657	42	190	82	0	375	42
Syn20M04H	3532.744	764	160	0	1396	56	599	115	0	753	308
Syn20M04M	3532.744	420	160	0	996	56	259	115	0	572	56
* Syn20M	924.263	65	20	0	99	14	52	16	0	80	14
Syn40H	67.713	302	40	0	438	28	268	33	0	254	148
Syn40M02H	388.773	764	160	0	1156	56	577	103	0	615	300
Syn40M02M	388.772	420	160	0	756	56	245	103	0	432	56
* Syn40M03H	395.149	1146	240	0	1914	84	886	173	0	1046	452
* Syn40M03M	395.151	630	240	0	1314	84	386	173	0	772	84
Syn40M04H	901.752	1528	320	0	2792	112	1195	243	0	1547	604

---

continue next page...

## A. Tables

... continued statistics for convex MINLP test set instances

instance	opt. value	$n$	bin	int	$m'$	$m$	$n$	bin	int	$m'$	$m$
Syn40M04M	901.756	840	320	0	1992	112	527	243	0	1182	112
Syn40M	67.713	130	40	0	198	28	104	33	0	162	28
* uflquad-15-60	1063.193	916	15	0	960	1	916	15	0	960	1
* uflquad-15-80	1217.989	1216	15	0	1280	1	1216	15	0	1280	1
* uflquad-20-40	860.961	821	20	0	840	1	821	20	0	840	1
* uflquad-20-50	375.919	1021	20	0	1050	1	1021	20	0	1050	1
uflquad-25-25	673.697	651	25	0	650	1	651	25	0	650	1
* uflquad-25-30	669.840	776	25	0	780	1	776	25	0	780	1
* uflquad-25-40	828.165	1026	25	0	1040	1	1026	25	0	1040	1

### A.1.3. Mittelmann's MIQPs

Table A.3.: Mittelmann test set problem statistics

instance	opt. value	original problem					presolved problem				
		$n$	bin	int	$m'$	$m$	$n$	bin	int	$m'$	$m$
* iair04	56162.206	8905	8904	0	823	1	12997	7362	0	17546	0
* iair05	26391.327	7196	7195	0	426	1	10702	6116	0	14142	0
* ibc1	3.544	1752	252	0	1913	1	981	252	0	1783	0
* ibell13a	878784.998	123	31	29	104	1	128	31	29	160	1
* ibienst1	48.738	506	28	0	576	1	476	28	0	601	0
* icap6000	-2448337.000	6001	6000	0	2171	1	7057	5867	0	5598	0
* icvxqp1	$\leq$ 395592.000	10001	0	10000	5000	1	10003	2	9998	5006	1
* ieild76	888.691	1899	1898	0	75	1	2664	1898	0	2374	0
ilaser0	$\leq$ 2412628.290	1003	0	151	2000	1	1003	0	151	1000	1
* imas284	91407.993	152	150	0	68	1	300	150	0	515	0
* imisc07	2814.191	261	259	0	212	1	429	238	0	812	0
imod011	$\leq$ 0.000	10958	96	1	4480	1	8804	96	1	2633	1
* inug06-3rd	1318.000	2887	2886	0	3972	1	3364	2886	0	5988	0
* inug08	7213.000	1633	1632	0	912	1	2236	1632	0	3143	0
* iportfolio	$\leq$ -0.494	1201	775	192	201	1	1201	775	192	201	1
* iqap10	353.828	4151	4150	0	1820	1	5934	4150	0	9512	0
* iqu	-127.081	841	48	0	1192	1	887	48	0	1333	0
* iran13x13	3258.557	339	169	0	195	1	506	169	0	699	0
* iran8x32	5255.448	513	256	0	296	1	767	256	0	1061	0
* isqp0	-20319.514	1001	0	50	249	1	1001	0	50	249	1
* isqp1	-18992.680	1001	100	0	249	1	1065	100	0	471	1
* isqp	$\leq$ -21001.451	1001	0	50	249	1	1001	0	50	249	1
* iswath2	387.163	6405	2213	0	483	1	8291	2213	0	6470	0
* itointqor	-1146.700	51	0	50	0	1	51	0	50	0	1

## A.2. Detailed Benchmark Results

The following tables give the detailed results for the benchmarks from Section 8.3. At the end of each table, aggregated statistics are given. In the first column, the number in brackets is the number of instances considered when calculating entries for the corresponding row, see also Section 8.3.

### A.2.1. MINLPLib

Table A.4.: Detailed results on MINLPLib test set.

instance	SCIP		BARON		COUENNE		LINDOAPI	
	(dual gap) [dual bnd]	(pr. gap) [pr. bnd]						
4stufen	[88433]	[119944]	<b>[104903]</b>	<b>[117070]</b>	[102092]	[121573]	[17631]	[117099]
alan	<b>1.13</b>	5	2.27	7	<b>0.21</b>	18	<b>0.32</b>	1
batchdes	<b>1.28</b>	3	2.00	5	<b>0.29</b>	2	2.04	1
batch	<b>1.56</b>	1	2.76	31	<b>1.01</b>	26	4.97	1
beuster	[14127]	[125308]	<b>[103869]</b>	[116495]	[17785]	<b>[107716]</b>	[45905]	[117147]
cecil_13	(1.88%)	(0.04%)	(4.26%)	(0.08%)	(4.58%)	(0.08%)	<b>6793.20</b>	2494
chp_partload	<b>[20.16]</b>	[∞]	<b>[20.16]</b>	[25.13]	<b>[20.16]</b>	<b>[24.86]</b>	<b>[20.16]</b>	[∞]
contvar	<i>abort</i>		[372504]	<b>[809150]</b>	[412569]	[∞]	<b>[412681]</b>	<b>[809150]</b>
csched1a	(7.97%)	(0.00%)	3.35	1	<b>0.90</b>	44	2.44	20
csched1	(22.25%)	(0.01%)	<b>3.82</b>	87	<b>4.60</b>	3655	12.12	134
csched2a	[-352760000]	[-149135]	<b>[-165692]</b>	<b>[-165240]</b>	<i>abort</i>		[-3264940]	[-164274]
csched2	[-2806410]	[-134227]	<b>[-181023]</b>	[-160492]	<i>abort</i>		[-724313]	<b>[-165364]</b>
du-opt5	<b>0.71</b>	80	10.04	458	<i>fail</i>		5742.53	740
du-opt	<b>0.82</b>	240	5.44	113	<i>fail</i>		(1.52%)	(1.90%)
eg_all_s	<b>[-3.305]</b>	[8.666]	[-4.89]	<b>[7.822]</b>	[-6.424]	[11.81]	<i>fail</i>	
eg_disc2_s	[-7.735]	[12.79]	[-6.162]	[5.679]	[-8.087]	[5.679]	<b>2316.01</b>	25
eg_disc_s	[-7.613]	[9.129]	[-6.075]	[5.974]	[-8.087]	[6.061]	<b>[-5.146]</b>	<b>[5.761]</b>
eg_int_s	[-6.063]	[100000]	<b>[-1.895]</b>	<b>[6.453]</b>	[-6.394]	[8.127]	<i>fail</i>	
elf	<b>0.49</b>	293	2.67	410	16.57	21.4k	6.20	24
eniplac	<b>0.90</b>	154	4069.50	454.6k	139.17	72.6k	<i>fail</i>	
enpro48	<b>1.10</b>	37	3.17	307	6.49	325	24.41	1
enpro48pb	<b>1.09</b>	31	3.25	405	6.35	325	28.84	1
enpro56	<b>1.51</b>	1211	35.10	9905	9.20	1459	128.05	3
enpro56pb	<b>1.49</b>	591	59.91	15.9k	9.18	1459	142.08	5
ex1221	<b>0.09</b>	1	<b>0.12</b>	1	<b>0.06</b>	0	<b>0.06</b>	1
ex1222	<b>0.09</b>	1	<b>0.50</b>	0	<b>0.05</b>	0	<b>0.10</b>	0
ex1223a	<b>0.17</b>	1	<b>0.62</b>	1	<b>0.13</b>	0	<b>0.12</b>	1
ex1223b	<b>0.20</b>	1	<b>0.12</b>	7	<b>0.18</b>	2	<b>0.18</b>	1
ex1223	<b>0.20</b>	1	<b>0.17</b>	7	<b>0.18</b>	2	<b>0.20</b>	1
ex1224	<b>0.18</b>	6	<b>0.58</b>	12	<b>0.40</b>	6	<b>0.67</b>	3
ex1225	<b>0.13</b>	1	<b>0.12</b>	1	<b>0.11</b>	0	<b>0.11</b>	1
ex1226	<b>0.14</b>	5	<b>0.17</b>	1	<b>0.12</b>	0	<b>0.12</b>	2
ex1233	(1.43%)	(0.00%)	<b>114.20</b>	4705	(5.59%)	(0.00%)	(0.31%)	(0.00%)
ex1243	<b>0.90</b>	71	2.66	47	<b>1.66</b>	19	4.99	21
ex1244	<b>1.96</b>	629	6.71	394	4.61	107	34.48	95
ex1252a	<i>fail</i>		<i>fail</i>		<b>7.94</b>	3244	122.55	1155

continue next page...

A. Tables

... continued detailed results on MINLPLib test set									
instance	SCIP		BARON		COUENNE		LINDOAPI		
ex1252	<b>5.04</b>	1390	<i>fail</i>		8.30	3136	1033.40	83.4k	
ex1263a	<b>0.35</b>	229	<b>0.82</b>	225	1.65	1258	3.62	2	
ex1263	<b>1.04</b>	596	3.06	836	3.81	1003	9.05	37	
ex1264a	<b>0.16</b>	176	3.14	305	1.77	1662	1.64	1	
ex1264	<b>0.28</b>	86	3.19	686	4.49	1068	38.20	149	
ex1265a	<b>0.22</b>	72	2.63	130	2.06	1045	2.40	1	
ex1265	<b>0.40</b>	69	3.05	516	2.70	616	7.20	7	
ex1266a	<b>0.14</b>	1	<b>0.87</b>	72	1.71	321	<b>0.41</b>	1	
ex1266	<b>0.17</b>	1	1.31	200	3.71	624	2.41	1	
ex3	<b>0.29</b>	1	<b>0.24</b>	18	<b>0.21</b>	6	<b>0.31</b>	1	
ex3pb	<b>0.29</b>	1	<b>0.43</b>	18	<b>0.42</b>	6	<b>0.33</b>	1	
ex4	<b>0.76</b>	9	<b>1.32</b>	39	<b>1.45</b>	84	5.93	1	
fac1	<b>0.10</b>	1	<b>0.31</b>	7	<b>0.32</b>	4	<b>1.00</b>	1	
fac2	<b>0.58</b>	12	<b>0.29</b>	43	<b>1.26</b>	84	8.10	1	
fac3	<b>0.48</b>	7	<b>0.62</b>	387	(17.44%)	(0.00%)	31.73	1	
feedtray2	<b>0.25</b>	1	<b>0.85</b>	0	4.89	1	24.39	3	
feedtray	(100.00%)	<b>(0.00%)</b>	<b>(100.00%)</b>	<b>(0.00%)</b>	(100.00%)	<b>(0.00%)</b>	(100.00%)	<b>(0.00%)</b>	
fo7_2	<b>35.20</b>	53.7k	(47.09%)	(0.00%)	5272.97	3515.3k	(30.32%)	(58.94%)	
fo7_ar2_1	<b>33.42</b>	71.5k	1583.31	923.9k	908.10	543.2k	(33.05%)	(72.30%)	
fo7_ar25_1	<b>25.03</b>	45.9k	(32.82%)	(11.19%)	2800.81	1458.0k	(18.34%)	(0.12%)	
fo7_ar3_1	<b>30.81</b>	51.4k	(28.51%)	(0.00%)	4279.95	2203.1k	(50.99%)	(27.53%)	
fo7_ar4_1	<b>44.01</b>	70.9k	(19.16%)	(0.00%)	4028.90	2080.1k	(43.45%)	(57.92%)	
fo7_ar5_1	<b>10.25</b>	16.0k	(28.98%)	(17.23%)	1558.43	840.1k	(30.52%)	(48.43%)	
fo7	<b>104.16</b>	168.6k	(44.88%)	(8.62%)	(16.92%)	(0.00%)	(42.73%)	(44.60%)	
fo8_ar2_1	<b>249.03</b>	473.2k	(36.63%)	(2.13%)	(41.37%)	(16.40%)	(54.94%)	(∞)	
fo8_ar25_1	<b>170.58</b>	350.9k	(74.73%)	(69.71%)	(46.07%)	(10.44%)	(74.46%)	(23.19%)	
fo8_ar3_1	<b>44.13</b>	70.5k	(52.79%)	(33.66%)	(43.18%)	(21.34%)	(62.57%)	(100.00%)	
fo8_ar4_1	<b>103.33</b>	146.6k	(35.92%)	(21.27%)	(37.33%)	(21.66%)	(62.02%)	(∞)	
fo8_ar5_1	<b>96.99</b>	151.2k	(65.12%)	(39.71%)	(42.78%)	(6.83%)	(66.47%)	(∞)	
fo8	<b>203.80</b>	298.9k	(65.37%)	(31.28%)	(52.26%)	(10.14%)	(76.30%)	(∞)	
fo9_ar2_1	<b>(5.29%)</b>	<b>(0.00%)</b>	(59.90%)	(27.14%)	(71.86%)	(69.87%)	(87.00%)	(26.05%)	
fo9_ar25_1	<b>6269.99</b>	11.5M	(62.62%)	(53.01%)	(58.09%)	(19.22%)	(62.69%)	(∞)	
fo9_ar3_1	<b>189.36</b>	261.1k	(100.00%)	(∞)	(67.43%)	(57.70%)	(73.41%)	(∞)	
fo9_ar4_1	<b>405.48</b>	513.1k	(100.00%)	(∞)	(59.58%)	(88.51%)	(76.56%)	(100.00%)	
fo9_ar5_1	<b>792.74</b>	1047.4k	(71.01%)	(100.00%)	(59.53%)	(21.37%)	(77.26%)	(∞)	
fo9	<b>724.77</b>	1018.8k	(66.07%)	(58.32%)	(70.88%)	(38.52%)	(79.76%)	(∞)	
fuel	<b>0.23</b>	1	<b>0.07</b>	1	<b>0.14</b>	0	<b>0.25</b>	1	
fuzzy	<b>[-0.5398]</b>	<b>[-0.5261]</b>	-		-		<b>[-0.5398]</b>	<b>[-0.5064]</b>	
gasnet	<i>fail</i>		[2492350]	[6999380]	[3079330]	[7076690]	<b>351.61</b>	172	
gastrans	<b>0.32</b>	4	<b>0.44</b>	10	1.74	190	2.22	1	
gbd	<b>0.09</b>	1	<b>0.05</b>	0	<b>0.05</b>	0	<b>0.10</b>	1	
ghg_1veh	7409.37	19.1M	<b>9.34</b>	429	1471.75	229.0k	<i>fail</i>		
ghg_2veh	[3.604]	<b>[7.771]</b>	[3.779]	<b>[7.771]</b>	[5.99]	<b>[7.771]</b>	<b>[6.851]</b>	<b>[7.771]</b>	
ghg_3veh	[0.5457]	[7.761]	[0]	<b>[7.754]</b>	<b>[2.789]</b>	[7.77]	<i>fail</i>		
gear2	3.36	5960	<b>0.15</b>	87	<b>0.19</b>	15	<b>0.27</b>	0	
gear3	1.28	518	<b>0.25</b>	110	<b>0.14</b>	9	<b>0.23</b>	0	
gear4	<i>fail</i>		<b>0.36</b>	935	<b>1.18</b>	2386	74.49	79	
gear	1.99	518	<b>0.22</b>	116	<b>0.07</b>	9	<b>0.14</b>	0	
gkocis	<b>0.19</b>	3	<b>0.10</b>	1	<b>0.10</b>	2	<b>0.13</b>	1	
hda	<b>[-13010]</b>	<b>[-5965]</b>	<i>fail</i>		<i>fail</i>		<i>fail</i>		
hmittelman	<b>0.10</b>	1	<b>0.11</b>	35	<b>0.18</b>	2	<b>0.11</b>	0	
johnall	10.05	1	<b>4.27</b>	1	<b>3.63</b>	0	52.89	4	
lop97ic	[2541]	[5295]	[3383]	[5040]	[3846]	[6035]	<b>[3886]</b>	<b>[4119]</b>	
lop97icx	(33.07%)	(16.14%)	<i>abort</i>		(6.10%)	(6.71%)	<b>(2.40%)</b>	<b>(2.08%)</b>	

continue next page...

A.2. Detailed Benchmark Results

... continued detailed results on MINLPLib test set

instance	SCIP		BARON		COUENNE		LINDOAPI	
m3	<b>0.30</b>	17	<b>0.07</b>	29	<b>0.94</b>	24	1.58	4
m6	<b>1.24</b>	791	665.96	537.0k	31.79	21.1k	142.34	17
m7_ar2_1	<b>4.49</b>	8431	476.44	403.4k	39.10	19.7k	683.37	136
m7_ar25_1	<b>1.42</b>	1714	203.94	156.0k	16.73	8435	314.99	29
m7_ar3_1	<b>4.46</b>	7445	(15.32%)	(0.00%)	56.15	27.6k	1468.86	292
m7_ar4_1	<b>1.61</b>	822	5051.64	3198.7k	62.11	26.8k	709.41	185
m7_ar5_1	<b>8.53</b>	13.1k	5603.04	3309.3k	94.10	44.3k	1976.42	435
m7	<b>7.87</b>	14.2k	(24.32%)	(0.00%)	300.39	218.8k	1110.10	195
mbtd	[ <b>2.5</b> ]	[ <b>6.667</b> ]	[ <b>2.5</b> ]	[∞]	[ <b>2.5</b> ]	[7.917]	<i>abort</i>	
meanvarx	<b>0.24</b>	3	<b>0.12</b>	3	<b>0.89</b>	96	1.20	2
meanvarxsc	<b>0.31</b>	7	-	-	-	-	-	-
minlphix	(∞)	(9.92%)	<i>fail</i>		<b>1.53</b>	26	(100.00%)	(0.00%)
netmod_dol1	( <b>4.65%</b> )	( <b>0.00%</b> )	(19.61%)	(14.64%)	(26.77%)	(21.59%)	(22.00%)	(46.12%)
netmod_dol2	<b>69.14</b>	794	4612.84	7192	(8.40%)	(30.13%)	(3.91%)	(3.04%)
netmod_kar1	<b>4.55</b>	315	881.48	53.1k	(14.21%)	(1.04%)	(15.83%)	(0.24%)
netmod_kar2	<b>4.39</b>	315	880.99	53.1k	(14.21%)	(1.04%)	(16.24%)	(0.00%)
no7_ar2_1	<b>22.56</b>	33.0k	(5.84%)	(0.05%)	2157.06	1145.7k	(48.46%)	(31.14%)
no7_ar25_1	<b>40.22</b>	57.2k	(50.04%)	(9.40%)	(19.61%)	(0.00%)	(37.50%)	(36.02%)
no7_ar3_1	<b>141.75</b>	221.2k	(50.92%)	(0.00%)	(24.46%)	(1.47%)	(51.69%)	(7.93%)
no7_ar4_1	<b>92.84</b>	128.6k	(34.29%)	(0.37%)	(23.90%)	(1.93%)	(52.08%)	(18.61%)
no7_ar5_1	<b>63.87</b>	94.2k	(30.20%)	(5.17%)	(16.63%)	(0.00%)	(53.64%)	(29.19%)
nous1	(24.71%)	(0.00%)	<b>28.89</b>	2979	(11.97%)	(0.00%)	36.09	317
nous2	3.78	3589	<b>0.55</b>	13	2.81	70	<b>0.91</b>	8
nuclear104	[−∞]	[∞]	[−∞]	[∞]	[−∞]	[∞]	[−∞]	[∞]
nuclear10a	[−244.5]	[∞]	[−24]	[∞]	[− <b>12.33</b> ]	[∞]	[−∞]	[∞]
nuclear10b	[−216.8]	[∞]	[−24]	[∞]	[− <b>12.33</b> ]	[∞]	[−24]	[∞]
nuclear14a	[−238]	[−1.119]	[−24]	[∞]	[− <b>12.26</b> ]	[∞]	[− <b>12.26</b> ]	[− <b>1.129</b> ]
nuclear14b	[−144.1]	[− <b>1.11</b> ]	[−3.253]	[−1.097]	[−1.449]	[∞]	[− <b>1.426</b> ]	[− <b>1.11</b> ]
nuclear14	[−∞]	[−1.122]	[−∞]	[− <b>1.127</b> ]	[−∞]	[∞]	[−∞]	[−1.126]
nuclear24a	[−238]	[−1.119]	[−24]	[∞]	[− <b>12.26</b> ]	[∞]	[− <b>12.26</b> ]	[− <b>1.129</b> ]
nuclear24b	[−144.1]	[− <b>1.11</b> ]	[−3.252]	[−1.097]	[−1.449]	[∞]	[− <b>1.426</b> ]	[− <b>1.11</b> ]
nuclear24	[−∞]	[−1.122]	[−∞]	[− <b>1.127</b> ]	[−∞]	[∞]	[−∞]	[−1.126]
nuclear25a	[−225.3]	[−1.098]	[−24]	[∞]	[− <b>12.32</b> ]	[∞]	[− <b>12.32</b> ]	[− <b>1.12</b> ]
nuclear25b	[−111.2]	[− <b>1.078</b> ]	[−24]	[∞]	[− <b>1.453</b> ]	[∞]	[−1.524]	[−1.077]
nuclear25	[−∞]	[−1.1]	[−∞]	[∞]	[−∞]	[∞]	[−∞]	[− <b>1.115</b> ]
nuclear49a	[−238.1]	[∞]	[−24]	[∞]	[− <b>12.36</b> ]	[∞]	[− <b>12.36</b> ]	[− <b>1.151</b> ]
nuclear49b	[−197.8]	[− <b>1.129</b> ]	[−24]	[∞]	[− <b>3.222</b> ]	[∞]	[−24]	[∞]
nuclear49	[−∞]	[− <b>1.136</b> ]	[−∞]	[∞]	[−∞]	[∞]	[−∞]	[∞]
nuclearva	[−∞]	[∞]	[−∞]	[∞]	<i>abort</i>		<i>abort</i>	
nuclearvb	[−∞]	[∞]	[−∞]	[∞]	<i>abort</i>		<i>fail</i>	
nuclearvc	[−∞]	[−0.9932]	[−∞]	[−0.988]	<i>abort</i>		[−∞]	[− <b>1.001</b> ]
nuclearvd	[−∞]	[∞]	[−∞]	[∞]	<i>abort</i>		<i>fail</i>	
nuclearve	[−∞]	[∞]	[−∞]	[∞]	<i>abort</i>		[−∞]	[− <b>1.036</b> ]
nuclearvf	[−∞]	[∞]	[−∞]	[∞]	<i>abort</i>		<i>fail</i>	
nvs01	<b>0.28</b>	29	<b>0.16</b>	1	<b>0.14</b>	1	<b>0.26</b>	2
nvs02	<b>0.14</b>	1	<b>0.13</b>	4	<b>0.11</b>	0	<b>0.45</b>	1
nvs03	<b>0.12</b>	1	<b>0.09</b>	3	<b>0.09</b>	0	<b>0.16</b>	1
nvs04	<b>0.15</b>	3	<b>0.09</b>	3	<b>0.13</b>	2	<b>0.22</b>	3
nvs05	2.01	853	1.49	117	<b>0.67</b>	0	<b>0.31</b>	2
nvs06	<b>0.17</b>	11	<b>0.04</b>	0	<b>0.09</b>	0	<b>0.17</b>	2
nvs07	<b>0.12</b>	1	<b>0.07</b>	0	<b>0.10</b>	0	<b>0.12</b>	0
nvs08	<b>0.17</b>	1	<b>0.12</b>	7	<b>0.12</b>	2	<b>0.26</b>	3
nvs09	(52.28%)	(59.39%)	<b>0.14</b>	1	<b>0.30</b>	2	8.10	12
nvs10	<b>0.12</b>	1	<b>0.08</b>	0	<b>0.10</b>	1	<b>0.14</b>	1

continue next page...

A. Tables

... continued detailed results on MINLPLib test set								
instance	SCIP		BARON		COUENNE		LINDOAPI	
nvs11	<b>0.15</b>	3	<b>0.15</b>	3	<b>0.13</b>	4	<b>1.08</b>	1
nvs12	<b>0.16</b>	6	<b>0.10</b>	3	<b>0.23</b>	10	4.54	1
nvs13	<b>0.17</b>	12	<b>0.27</b>	9	<b>0.57</b>	83	19.91	1
nvs14	<b>0.14</b>	1	<b>0.10</b>	4	<b>0.11</b>	0	<b>0.34</b>	1
nvs15	<b>0.15</b>	3	<b>0.08</b>	1	<b>0.09</b>	0	<b>0.16</b>	1
nvs16	<b>0.15</b>	7	<b>0.09</b>	3	<b>0.09</b>	0	<b>0.15</b>	3
nvs17	<b>0.26</b>	51	<b>1.17</b>	78	1.68	1400	3255.31	69
nvs18	<b>0.21</b>	23	<b>0.61</b>	37	<b>0.97</b>	276	610.24	11
nvs19	<b>0.36</b>	96	2.12	113	3.24	3234	(28.64%)	(0.00%)
nvs20	<b>0.66</b>	125	<b>0.92</b>	13	<b>1.07</b>	74	5.89	9
nvs21	<b>0.28</b>	28	<b>0.12</b>	3	<b>0.20</b>	4	1.26	11
nvs22	<b>0.21</b>	11	<i>fail</i>		<i>fail</i>		<b>0.60</b>	2
nvs23	<b>0.46</b>	106	8.02	376	8.55	7320	(83.67%)	(0.00%)
nvs24	<b>0.47</b>	104	26.90	797	26.03	20.4k	(100.00%)	(0.00%)
o7_2	<b>860.74</b>	1350.0k	(53.68%)	(0.00%)	(29.05%)	(0.00%)	(52.82%)	(26.96%)
o7_ar2_1	<b>94.40</b>	150.7k	(19.47%)	(0.00%)	6843.70	3318.7k	(58.85%)	(∞)
o7_ar25_1	<b>268.37</b>	419.4k	(50.69%)	(6.39%)	(27.20%)	(0.00%)	(44.88%)	(20.47%)
o7_ar3_1	<b>683.06</b>	1069.9k	(38.39%)	(3.82%)	(32.80%)	(0.00%)	(67.19%)	(14.08%)
o7_ar4_1	<b>1676.39</b>	2287.5k	(48.27%)	(16.14%)	(29.63%)	(1.91%)	(50.57%)	(17.18%)
o7_ar5_1	<b>430.53</b>	667.2k	(40.48%)	(10.03%)	(17.17%)	(1.82%)	(61.08%)	(25.40%)
o7	<b>1810.76</b>	2787.1k	(63.17%)	(11.82%)	(41.45%)	(4.22%)	(57.82%)	(15.65%)
o8_ar4_1	( <b>8.03%</b> )	( <b>0.00%</b> )	(58.62%)	(22.08%)	(58.17%)	(10.25%)	(83.31%)	(∞)
o9_ar4_1	( <b>11.15%</b> )	( <b>0.00%</b> )	(100.00%)	(∞)	(67.92%)	(16.39%)	(86.56%)	(∞)
oacr	<b>0.18</b>	1	<b>0.09</b>	1	<b>0.15</b>	2	<b>0.13</b>	1
oil2	(0.01%)	(0.00%)	(0.00%)	(0.01%)	<i>fail</i>		<b>105.54</b>	5
oil	( <b>20.64%</b> )	( <b>0.00%</b> )	(42.64%)	(5.15%)	<i>fail</i>		<i>fail</i>	
ortez	<b>0.29</b>	5	<b>0.23</b>	11	1.52	22	2.43	3
parallel	<i>fail</i>		<b>10.71</b>	461	32.51	5001	223.78	767
pb302035	[ <b>941406</b> ]	[4350760]	[403709]	[4432220]	[591654]	[∞]	[709694]	[ <b>3923680</b> ]
pb302055	[ <b>916942</b> ]	[4293990]	[435301]	[∞]	[638020]	[∞]	[758891]	[ <b>4158700</b> ]
pb302075	[ <b>1088470</b> ]	[4501660]	[424724]	[∞]	[653918]	[∞]	[838373]	[ <b>4449650</b> ]
pb302095	[ <b>2227670</b> ]	[ <b>5928560</b> ]	[956625]	[∞]	[1216070]	[∞]	[1898830]	[6006690]
pb351535	[ <b>1484400</b> ]	[5722240]	[0]	[∞]	[1061080]	[∞]	[1223760]	[ <b>5116370</b> ]
pb351555	[ <b>1594210</b> ]	[5249910]	[0]	[∞]	[1151040]	[∞]	[1273050]	[ <b>5206860</b> ]
pb351575	[ <b>1692200</b> ]	[6776200]	[0]	[∞]	[1179110]	[∞]	[1417750]	[ <b>6527830</b> ]
pb351595	[ <b>1675830</b> ]	[7237100]	[0]	[∞]	[1146550]	[∞]	[1359910]	[ <b>7206560</b> ]
prob02	<b>0.10</b>	0	<b>0.07</b>	0	<b>0.10</b>	0	<b>0.12</b>	1
prob03	<b>0.11</b>	1	<b>0.07</b>	0	<b>0.09</b>	0	<b>0.12</b>	1
procse1	<b>0.21</b>	1	<b>0.09</b>	1	<i>fail</i>		<b>0.12</b>	1
product2	7426.28	5921.7k	<i>fail</i>		<i>fail</i>		[-2108]	[-2099]
product	<b>19.11</b>	6151	<i>fail</i>		<i>fail</i>		3728.07	2396
pump	<i>fail</i>		<i>fail</i>		<b>7.79</b>	3000	118.70	1338
qap	(96.02%)	(3.51%)	( <b>66.97%</b> )	( <b>0.01%</b> )	(79.90%)	(16.33%)	(100.00%)	(11.04%)
qapw	(98.45%)	(4.89%)	( <b>31.78%</b> )	( <b>0.77%</b> )	(100.00%)	(∞)	(100.00%)	(5.77%)
ravem	<b>1.17</b>	151	<b>0.97</b>	71	3.25	74	20.93	3
ravempb	<b>1.03</b>	184	<b>0.86</b>	101	3.26	74	13.06	1
risk2b	<b>0.18</b>	25	<b>0.70</b>	13	5.98	4	5.43	1
risk2bpb	<b>0.74</b>	7	<b>0.78</b>	11	4.47	4	2.78	1
saa_2	[-3.903]	[12.88]	[ <b>9.285</b> ]	[ <b>12.7</b> ]	[-57.94]	[12.78]	[-121.4]	[12.71]
sep1	<b>0.49</b>	37	<b>0.13</b>	1	<b>0.39</b>	6	<b>0.30</b>	5
space25a	[73.48]	[588.4]	[123.1]	[488.6]	[73.01]	[∞]	<b>129.29</b>	3
space25	[72.46]	[∞]	[ <b>114.8</b> ]	[ <b>536.5</b> ]	[73.01]	[∞]	<i>fail</i>	
space960	[ <b>6516160</b> ]	[ <b>17355000</b> ]	[6515330]	[42155000]	[6491010]	[∞]	[6495000]	[∞]
spectra2	<b>0.87</b>	25	2.60	185	(100.00%)	(24.69%)	39.67	51

continue next page...

A.2. Detailed Benchmark Results

... continued detailed results on MINLPLib test set								
instance	SCIP		BARON		COUENNE		LINDOAPI	
spring	<b>0.51</b>	102	<b>0.42</b>	6	<b>0.51</b>	6	1.83	14
st_e13	<b>0.09</b>	1	<b>0.07</b>	0	<b>0.09</b>	0	<b>0.09</b>	2
st_e14	<b>0.21</b>	1	<b>0.12</b>	7	<b>0.18</b>	2	<b>0.20</b>	1
st_e15	<b>0.09</b>	1	<b>0.08</b>	1	<b>0.09</b>	0	<b>0.11</b>	1
st_e27	<b>0.08</b>	0	<b>0.08</b>	1	<b>0.10</b>	0	<b>0.10</b>	1
st_e29	<b>0.19</b>	6	<b>0.14</b>	12	<b>0.40</b>	6	<b>0.68</b>	3
st_e31	<b>1.15</b>	1735	<b>0.74</b>	146	4.76	1906	3.97	205
st_e32	14.25	9773	<b>0.85</b>	55	<i>abort</i>		23.67	216
st_e35	(13.25%)	(5.47%)	<b>1.51</b>	166	33.30	731	37.75	620
st_e36	1.62	401	<b>0.18</b>	5	<b>0.14</b>	0	2.14	1
st_e38	<b>0.21</b>	7	<b>0.14</b>	1	<b>0.13</b>	2	<b>0.41</b>	4
st_e40	<b>0.17</b>	27	<b>0.14</b>	7	<b>0.41</b>	12	<i>fail</i>	
st_miqp1	<b>0.09</b>	1	<b>0.09</b>	3	<b>0.09</b>	0	<b>0.12</b>	1
st_miqp2	<b>0.12</b>	1	<b>0.09</b>	5	<b>0.10</b>	2	<b>0.15</b>	1
st_miqp3	<b>0.11</b>	1	<b>0.07</b>	0	<b>0.09</b>	0	<b>0.11</b>	1
st_miqp4	<b>0.15</b>	1	<b>0.08</b>	1	<b>0.10</b>	0	<b>0.10</b>	1
st_miqp5	<b>0.16</b>	1	<b>0.09</b>	1	<b>0.12</b>	0	<b>0.11</b>	1
stockcycle	282.45	47.5k	35.27	11.4k	<b>32.64</b>	19.0k	122.09	1
st_test1	<b>0.05</b>	0	<b>0.09</b>	7	<b>0.10</b>	0	<b>0.16</b>	1
st_test2	<b>0.05</b>	1	<b>0.09</b>	3	<b>0.10</b>	0	<b>0.12</b>	1
st_test3	<b>0.09</b>	1	<b>0.07</b>	0	<b>0.10</b>	0	<b>0.50</b>	1
st_test4	<b>0.12</b>	1	<b>0.08</b>	1	<b>0.09</b>	0	<b>0.16</b>	1
st_test5	<b>0.09</b>	1	<b>0.09</b>	13	<b>0.12</b>	2	<b>0.11</b>	1
st_test6	<b>0.09</b>	1	<b>0.10</b>	13	<b>0.11</b>	2	<b>0.07</b>	1
st_test8	<b>0.12</b>	1	<b>0.09</b>	1	<b>0.20</b>	0	<b>0.57</b>	1
st_testgr1	<b>0.19</b>	47	<b>0.15</b>	31	<b>0.29</b>	94	1.48	3
st_testgr3	<b>0.18</b>	27	<b>0.55</b>	166	<b>1.15</b>	1128	1.80	1
st_testph4	<b>0.12</b>	1	<b>0.07</b>	0	<b>0.06</b>	0	<b>0.05</b>	1
super1	[4.198]	[∞]	<i>fail</i>		[4.208]	[9.665]	<i>abort</i>	
super2	[2.938]	[∞]	<i>fail</i>		[2.951]	[5.246]	<i>abort</i>	
super3	[6.612]	[∞]	<i>fail</i>		[6.458]	[∞]	[6.68]	[∞]
super3t	[-1]	[-0.6742]	[-1]	[∞]	[-1]	[-0.6645]	[-1]	[-0.6535]
synheat	(16.40%)	(0.00%)	<b>252.29</b>	8957	(6.33%)	(0.00%)	1981.72	5986
synthes1	<b>0.12</b>	3	<b>0.10</b>	5	<b>0.14</b>	2	<b>0.15</b>	1
synthes2	<b>0.27</b>	1	<b>0.12</b>	10	<b>0.18</b>	2	<b>0.32</b>	1
synthes3	<b>0.30</b>	5	<b>0.15</b>	12	<b>0.25</b>	4	1.35	1
tln12	(83.38%)	<b>(0.77%)</b>	(5.69%)	(∞)	(77.23%)	(∞)	<b>(4.93%)</b>	(15.47%)
tln2	<b>0.14</b>	1	<b>0.11</b>	13	<b>0.13</b>	2	<b>0.22</b>	0
tln4	<b>1.73</b>	2658	<b>1.25</b>	893	133.81	240.0k	7.45	1
tln5	114.22	171.0k	<b>3.17</b>	1967	(25.71%)	(0.00%)	163.39	9
tln6	(38.41%)	(0.00%)	<b>7.97</b>	3371	(41.64%)	(1.96%)	158.94	11
tln7	(61.87%)	<b>(0.00%)</b>	<b>(3.60%)</b>	(6.00%)	(62.39%)	(8.00%)	(7.60%)	(10.00%)
tloss	<b>0.15</b>	1	<b>0.68</b>	55	2.26	604	<b>0.90</b>	1
tls12	[0]	[∞]	[0]	[∞]	[2.029]	[∞]	[3.337]	[∞]
tls2	<b>0.22</b>	6	<b>0.63</b>	305	<b>0.43</b>	40	<i>fail</i>	
tls4	<b>26.52</b>	26.3k	825.38	217.2k	(42.31%)	(0.00%)	713.92	214
tls5	(75.40%)	(1.94%)	(39.39%)	<b>(0.97%)</b>	(89.12%)	(2.91%)	<b>(22.33%)</b>	(2.91%)
tls6	[2.708]	[15.4]	[5.282]	[16.6]	[1.081]	[16.6]	[5.726]	[27.5]
tls7	[2.514]	[15.8]	[0]	[∞]	[0.3691]	[∞]	[3.925]	[27.3]
tltr	<b>0.30</b>	38	<b>0.27</b>	61	4.50	6242	<b>1.20</b>	1
uselinear	[-∞]	[∞]	<i>fail</i>		[-∞]	[∞]	<i>fail</i>	
util	<b>0.71</b>	475	<b>0.77</b>	643	4.52	274	4.36	2
waste	(41.69%)	(2.09%)	<b>1437.84</b>	7035	(48.64%)	(8.16%)	(20.63%)	(24.54%)
water3	[225.6]	[983.1]	-	-	-	-	-	-

continue next page...

## A. Tables

... continued detailed results on MINLPLib test set

instance	SCIP		BARON		COUENNE		LINDOAPI	
water4		<i>fail</i>		<i>fail</i>	[577.7]	[989.7]	<b>[600.1]</b>	<b>[923.8]</b>
waterful2	[363]	<b>[1802]</b>		–		–		–
watersbp	<b>[229.5]</b>	<b>[1064]</b>		–		–		–
waters	<b>[209]</b>	<b>[961.6]</b>		–		–		–
watersym1	[476.2]	<b>[987.9]</b>		–		–		–
watersym2	<b>[877]</b>	<b>[941.2]</b>		–		–		–
waterx	<b>[770]</b>	<b>[909]</b>	[254.3]	[912.1]	[622.9]	[974.1]	[599]	[948.9]
waterz		<i>fail</i>		<i>fail</i>	[198.1]	<b>[948.4]</b>	<b>[286.8]</b>	[958.7]
geom. mean	[210]	28.9 1115.0	55.7 457.7		68.3 895.5		92.0 15.3	
sh. geom. mean	[210]	91.2 4981.8	166.1 2178.6		186.5 4308.4		222.1 122.2	
arith. mean	[210]	2066.2 1369.9k	2851.8 526.7k		2983.0 590.5k		2991.8 1434	
arith. mean	[172]	(4.44%) (0.45%)	(11.32%) (5.70%)		(11.23%) (4.28%)		(14.74%) (11.75%)	
#solved	[260]	165	140		137		138	
#timeout	[260]	87	97		97		97	
#failed/aborted	[260]	8	15		18		18	
#fastest	[252]	88	64		20		12	
#best dual bound	[252]	184	155		151		159	
#best primal bnd.	[252]	193	164		156		168	

### A.2.2. Convex MINLPs

Table A.5.: Detailed results on convex MINLP test set (part I).

instance	SCIP		SCIP-nc		ALPHAECP		BONMIN-OA-cpx	
	time	nodes	time	nodes	time	nodes	time	nodes
	(dual gap) [dual bnd]	(pr. gap) [pr. bnd]						
BatchS101006M	15.50	16.2k	15.05	16.2k	46.34	0	<b>5.80</b>	0
BatchS121208M	112.35	91.7k	112.05	91.7k	887.72	0	<b>5.62</b>	0
BatchS151208M	75.38	58.9k	75.14	58.9k	1088.10	0	<b>9.51</b>	0
BatchS201210M	99.39	66.7k	98.89	66.7k	127.09	0	18.87	0
clay0203h	<b>0.59</b>	74	(91.44%)	(32.52%)	3.01	0	<b>1.19</b>	0
clay0203m	<b>0.39</b>	48	<b>0.34</b>	48	2.01	0	<b>0.84</b>	0
clay0204h	2.29	1254	12.47	7535	11.32	0	<b>1.12</b>	0
clay0204m	<b>0.93</b>	671	<b>1.16</b>	671	6.15	0	<b>0.84</b>	0
clay0205h	16.81	11.4k	(0.09%)	(2.30%)	38.37	0	<b>13.70</b>	0
clay0205m	<b>5.18</b>	10.7k	<b>4.92</b>	10.7k	22.70	0	6.85	0
clay0303h	<b>0.72</b>	166	(86.65%)	( $\infty$ )	3.97	0	<b>1.66</b>	0
clay0303m	<b>0.39</b>	87	<b>0.36</b>	87	3.00	0	<b>1.27</b>	0
clay0304h	<b>2.39</b>	930	(83.74%)	(47.19%)	16.58	0	7.58	0
clay0304m	<b>0.83</b>	316	<b>0.58</b>	316	7.28	0	2.97	0
clay0305h	21.38	12.5k	(0.09%)	(100.00%)	43.16	0	<b>19.01</b>	0
clay0305m	<b>4.55</b>	9205	<b>4.66</b>	9205	24.54	0	10.16	0
FLay03H	<b>1.06</b>	117	<b>0.90</b>	117	8.62	0	<b>0.93</b>	0
FLay03M	<b>0.81</b>	141	<b>0.76</b>	141	3.33	0	<b>0.39</b>	0
FLay04H	<b>3.92</b>	2370	<b>4.12</b>	2370	147.26	0	24.48	0
FLay04M	<b>2.51</b>	2465	<b>2.19</b>	2465	118.65	0	6.19	0
FLay05H	236.99	114.5k	<b>235.87</b>	114.5k	(7.04%)	(0.00%)	5798.88	0
FLay05M	<b>57.32</b>	89.6k	<b>57.05</b>	89.6k	(15.65%)	(0.00%)	3181.84	0
FLay06H	(3.64%)	<b>(0.00%)</b>	(3.64%)	<b>(0.00%)</b>	<b>(1.83%)</b>	<b>(0.00%)</b>	(48.25%)	( $\infty$ )

continue next page...

## A.2. Detailed Benchmark Results

... continued detailed results on convex MINLP test set								
instance	SCIP		SCIP-nc		ALPHAECP		BONMIN-OA-cpx	
FLay06M	3003.76	4696.5k	<b>2998.05</b>	4696.5k	(2.47%)	(0.00%)	(48.25%)	(∞)
fo7_2	35.21	53.7k	35.20	53.7k	16.25	0	<b>12.24</b>	0
fo7_ar2_1	33.98	71.5k	33.42	71.5k	71.69	0	<b>9.68</b>	0
fo7_ar25_1	25.01	45.9k	25.03	45.9k	68.72	0	<b>12.17</b>	0
fo7_ar3_1	30.60	51.4k	30.81	51.4k	143.01	0	<b>15.41</b>	0
fo7_ar4_1	44.15	70.9k	44.01	70.9k	129.60	0	<b>11.59</b>	0
fo7_ar5_1	10.04	16.0k	10.25	16.0k	26.73	0	<b>4.25</b>	0
fo7	104.14	168.6k	104.16	168.6k	158.32	0	<b>37.07</b>	0
fo8_ar2_1	246.99	473.2k	249.03	473.2k	416.70	0	<b>52.48</b>	0
fo8_ar25_1	170.71	350.9k	170.58	350.9k	432.29	0	<b>87.11</b>	0
fo8_ar3_1	43.67	70.5k	44.13	70.5k	175.83	0	<b>12.05</b>	0
fo8_ar4_1	104.03	146.6k	103.33	146.6k	130.30	0	<b>7.00</b>	0
fo8_ar5_1	96.13	151.2k	96.99	151.2k	469.25	0	<b>22.00</b>	0
fo8	202.50	298.9k	203.80	298.9k	532.20	0	<b>130.27</b>	0
fo9_ar2_1	(5.28%)	(0.00%)	(5.29%)	(0.00%)	1938.45	0	<b>277.87</b>	0
fo9_ar25_1	6265.34	11.5M	6269.99	11.5M	(2.73%)	(0.00%)	<b>1307.23</b>	0
fo9_ar3_1	190.14	261.1k	189.36	261.1k	1033.52	0	<b>41.52</b>	0
fo9_ar4_1	404.77	513.1k	405.48	513.1k	1274.74	0	<b>31.77</b>	0
fo9_ar5_1	791.35	1047.4k	792.74	1047.4k	2653.34	0	<b>104.13</b>	0
fo9	722.72	1018.8k	724.77	1018.8k	3049.93	0	<b>475.58</b>	0
m3	<b>0.21</b>	17	<b>0.30</b>	17	<b>0.43</b>	0	<b>0.12</b>	0
m6	<b>1.17</b>	791	<b>1.24</b>	791	1.48	0	<b>0.38</b>	0
m7_ar2_1	4.42	8431	4.49	8431	6.84	0	<b>2.44</b>	0
m7_ar25_1	<b>1.38</b>	1714	<b>1.42</b>	1714	<b>1.06</b>	0	<b>0.74</b>	0
m7_ar3_1	4.19	7445	4.46	7445	10.86	0	<b>2.25</b>	0
m7_ar4_1	1.42	822	1.61	822	<b>0.98</b>	0	<b>0.29</b>	0
m7_ar5_1	7.71	13.1k	8.53	13.1k	3.74	0	<b>1.51</b>	0
m7	7.82	14.2k	7.87	14.2k	2.82	0	<b>0.92</b>	0
nd-10	34.13	4680	(7.86%)	(∞)	5943.30	0	34.49	0
nd-12	76.70	5694	(6.75%)	(∞)	684.68	0	<b>12.45</b>	0
nd-13	<i>fail</i>		(13.77%)	(∞)	(15.60%)	(0.00%)	<b>969.37</b>	0
nd-15	(6.87%)	<b>(7.58%)</b>	(8.01%)	(∞)	<i>fail</i>		<i>fail</i>	
nd-16	(10.94%)	(1.19%)	(13.74%)	(∞)	(8.57%)	(0.22%)	<b>1511.53</b>	0
netmod_dol1	<b>(4.66%)</b>	<b>(0.00%)</b>	<b>(4.65%)</b>	<b>(0.00%)</b>	(8.22%)	(0.02%)	(27.33%)	(∞)
netmod_dol2	<b>69.28</b>	794	<b>69.14</b>	794	408.51	0	235.51	0
netmod_kar1	<b>4.59</b>	315	<b>4.55</b>	315	81.99	0	80.35	0
netmod_kar2	<b>4.39</b>	315	<b>4.39</b>	315	82.56	0	80.19	0
no7_ar2_1	22.30	33.0k	22.56	33.0k	131.14	0	<b>7.27</b>	0
no7_ar25_1	40.11	57.2k	40.22	57.2k	221.21	0	<b>23.08</b>	0
no7_ar3_1	141.01	221.2k	141.75	221.2k	361.94	0	<b>77.24</b>	0
no7_ar4_1	92.01	128.6k	92.84	128.6k	308.83	0	<b>74.26</b>	0
no7_ar5_1	<b>63.50</b>	94.2k	<b>63.87</b>	94.2k	197.27	0	70.20	0
o7_2	855.86	1350.0k	860.74	1350.0k	1225.83	0	<b>518.85</b>	0
o7_ar2_1	93.97	150.7k	94.40	150.7k	164.54	0	<b>52.59</b>	0
o7_ar25_1	268.21	419.4k	268.37	419.4k	876.71	0	<b>239.03</b>	0
o7_ar3_1	684.72	1069.9k	683.06	1069.9k	689.20	0	<b>477.74</b>	0
o7_ar4_1	1684.69	2287.5k	1676.39	2287.5k	4416.51	0	<b>1632.07</b>	0
o7_ar5_1	429.79	667.2k	430.53	667.2k	645.03	0	<b>333.81</b>	0
o7	1818.63	2787.1k	<b>1810.76</b>	2787.1k	(26.24%)	(0.00%)	2462.84	0
o8_ar4_1	(8.08%)	(0.00%)	(8.03%)	(0.00%)	(28.29%)	(0.00%)	<b>5426.37</b>	0
o9_ar4_1	(11.17%)	<b>(0.00%)</b>	(11.15%)	<b>(0.00%)</b>	<b>(10.73%)</b>	(12.66%)	(100.00%)	(∞)
RSyn0810H	<b>0.29</b>	1	<b>0.95</b>	228	<b>0.66</b>	0	<b>0.32</b>	0
RSyn0810M02H	<b>2.04</b>	536	(16.33%)	(0.00%)	6.40	0	<b>1.05</b>	0
RSyn0810M02M	48.54	76.7k	48.80	76.7k	29.59	0	5.92	0

continue next page...

A. Tables

... continued detailed results on convex MINLP test set								
instance	SCIP		SCIP-nc		ALPHA ECP		BONMIN-OA-cpx	
RSyn0810M03H	<b>1.87</b>	7	(100.00%)	(0.25%)	11.52	0	<b>1.73</b>	0
RSyn0810M03M	42.49	28.9k	41.83	28.9k	35.62	0	12.49	0
RSyn0810M04H	<b>2.22</b>	3	(35.89%)	(0.05%)	16.92	0	<b>1.62</b>	0
RSyn0810M04M	83.30	33.4k	82.11	33.4k	32.50	0	<b>8.04</b>	0
RSyn0810M	<b>0.76</b>	411	<b>0.58</b>	411	<b>1.15</b>	0	<b>0.59</b>	0
RSyn0820H	<b>0.87</b>	5	(28.64%)	(0.00%)	<b>1.31</b>	0	<b>0.68</b>	0
RSyn0820M02H	<b>2.07</b>	678	(82.82%)	(1.35%)	4.43	0	<b>1.50</b>	0
RSyn0820M02M	317.38	864.8k	316.75	864.8k	28.00	0	<b>6.61</b>	0
RSyn0820M03H	<b>2.12</b>	15	(100.00%)	(2.41%)	7.72	0	<b>1.40</b>	0
RSyn0820M03M	6466.24	6674.4k	6426.25	6674.4k	23.07	0	<b>18.70</b>	0
RSyn0820M04H	<b>2.95</b>	198	(100.00%)	(2.23%)	11.63	0	<b>2.84</b>	0
RSyn0820M04M	(21.07%)	(0.07%)	(21.06%)	(0.07%)	<b>26.02</b>	0	53.83	0
RSyn0820M	1.73	1155	1.78	1155	2.30	0	<b>0.72</b>	0
RSyn0840H	<b>1.24</b>	3	(100.00%)	(3.81%)	2.43	0	<b>0.64</b>	0
RSyn0840M02H	<b>1.78</b>	30	(100.00%)	(8.98%)	13.86	0	<b>1.52</b>	0
RSyn0840M02M	126.89	73.0k	126.72	73.0k	29.11	0	<b>3.41</b>	0
RSyn0840M03H	3.50	1401	(100.00%)	(2.29%)	30.37	0	<b>1.84</b>	0
RSyn0840M03M	4851.84	2039.1k	4840.16	2039.1k	50.18	0	7.10	0
RSyn0840M04H	16.35	8149	(100.00%)	(3.08%)	77.17	0	<b>2.26</b>	0
RSyn0840M04M	(41.96%)	(0.15%)	(41.94%)	(0.15%)	543.76	0	<b>25.25</b>	0
RSyn0840M	3.03	4227	3.06	4227	2.58	0	<b>0.95</b>	0
SLay05H	2.10	197	2.00	197	118.83	0	<b>0.86</b>	0
SLay05M	<b>0.75</b>	24	<b>0.63</b>	24	77.81	0	<b>0.26</b>	0
SLay06H	6.51	1265	6.30	1265	244.10	0	<b>3.37</b>	0
SLay06M	<b>1.56</b>	266	<b>1.29</b>	266	157.31	0	<b>0.74</b>	0
SLay07H	52.62	5370	52.55	5370	746.23	0	10.14	0
SLay07M	5.98	1453	6.12	1453	431.63	0	<b>1.25</b>	0
SLay08H	45.79	2671	45.82	2671	1579.80	0	<b>35.01</b>	0
SLay08M	7.43	1520	7.18	1520	821.53	0	<b>4.02</b>	0
SLay09H	510.18	44.0k	511.59	44.0k	(8.50%)	(0.16%)	164.51	0
SLay09M	37.29	5277	37.42	5277	2096.70	0	<b>23.33</b>	0
SLay10H	(4.21%)	(0.15%)	(4.21%)	(0.15%)	(8.19%)	(0.23%)	(8.10%)	(∞)
SLay10M	377.42	52.6k	379.69	52.6k	(13.29%)	(0.17%)	1865.18	0
sssd-10-4-3	<b>1.58</b>	1353	972.20	2717.6k	6.81	0	<b>1.73</b>	0
sssd-12-5-3	<b>4.08</b>	5453	(30.86%)	(0.07%)	24.23	0	10.41	0
sssd-15-6-3	<b>8.95</b>	15.6k	(48.89%)	(0.80%)	118.77	0	41.01	0
sssd-16-8-3	372.18	1059.9k	(46.47%)	(2.62%)	3710.81	0	338.53	0
sssd-18-8-3	836.62	2436.3k	(46.02%)	(2.27%)	5649.01	0	1244.52	0
sssd-20-9-3	(0.12%)	( <b>0.01%</b> )	(43.51%)	(2.02%)	(2.26%)	(2.38%)	(62.32%)	(∞)
sssd-22-8-3	<b>3358.83</b>	9110.8k	(48.18%)	(17.25%)	(1.72%)	(1.08%)	(63.69%)	(∞)
sssd-8-4-3	<b>0.94</b>	650	128.80	324.7k	3.15	0	<b>1.26</b>	0
Syn10H	<b>0.16</b>	1	<b>0.60</b>	53	<b>0.09</b>	0	<b>0.08</b>	0
Syn10M02H	<b>0.44</b>	1	3.27	2906	<b>1.04</b>	0	<b>0.10</b>	0
Syn10M02M	<b>0.43</b>	21	<b>0.21</b>	21	<b>0.76</b>	0	<b>0.09</b>	0
Syn10M03H	<b>0.50</b>	1	(4.53%)	(0.12%)	<b>1.18</b>	0	<b>0.24</b>	0
Syn10M03M	2.45	4352	2.44	4352	1.10	0	<b>0.09</b>	0
Syn10M04H	<b>0.53</b>	1	(5.91%)	(0.06%)	2.12	0	<b>0.28</b>	0
Syn10M04M	3.19	7521	3.16	7521	1.93	0	<b>0.23</b>	0
Syn10M	<b>0.28</b>	1	<b>0.29</b>	1	<b>0.16</b>	0	<b>0.07</b>	0
Syn20H	<b>0.48</b>	1	(15.15%)	(2.88%)	<b>0.87</b>	0	<b>0.11</b>	0
Syn20M02H	<b>0.45</b>	3	(0.50%)	(0.11%)	<b>1.21</b>	0	<b>0.23</b>	0
Syn20M02M	3.13	6531	3.35	6531	<b>0.99</b>	0	<b>0.27</b>	0
Syn20M03H	<b>0.54</b>	3	(24.24%)	(0.56%)	2.21	0	<b>0.44</b>	0
Syn20M03M	57.92	221.7k	58.12	221.7k	1.52	0	<b>0.36</b>	0

continue next page...

A.2. Detailed Benchmark Results

... continued detailed results on convex MINLP test set

instance	SCIP		SCIP-nc		ALPHA-ECP		BONMIN-OA-cpx		
Syn20M04H	<b>0.73</b>	1	(69.15%)	(0.81%)	3.06	0	<b>0.59</b>	0	
Syn20M04M	534.06	1768.0k	532.38	1768.0k	2.53	0	<b>0.42</b>	0	
Syn20M	<b>1.01</b>	664	<b>1.02</b>	664	1.15	0	<b>0.16</b>	0	
Syn40H	<b>0.62</b>	3	(100.00%)	(37.20%)	2.26	0	<b>0.71</b>	0	
Syn40M02H	<b>1.09</b>	7	(100.00%)	(15.72%)	4.91	0	<b>0.65</b>	0	
Syn40M02M	78.45	230.6k	78.60	230.6k	4.39	0	<b>0.58</b>	0	
Syn40M03H	2.56	901	(100.00%)	(20.25%)	10.90	0	<b>1.28</b>	0	
Syn40M03M	55.57	47.6k	55.70	47.6k	10.42	0	<b>1.08</b>	0	
Syn40M04H	<b>1.87</b>	15	(100.00%)	(5.18%)	21.23	0	<b>1.86</b>	0	
Syn40M04M	(7.47%)	(0.01%)	(7.48%)	(0.01%)	20.94	0	<b>1.28</b>	0	
Syn40M	<b>1.12</b>	13	<b>1.19</b>	13	2.23	0	<b>0.29</b>	0	
tls12	[15.6]	[∞]	[0]	[∞]	[8.125]	[∞]	[2.312]	[∞]	
tls2	<b>0.11</b>	9	<b>0.22</b>	6	<b>0.30</b>	0	<b>0.38</b>	0	
tls4	<b>8.62</b>	4817	26.52	26.3k	224.49	0	236.70	0	
tls5	<b>(18.55%)</b>	<b>(0.00%)</b>	(75.40%)	(1.94%)	(38.83%)	(41.75%)	(88.55%)	(∞)	
tls6	[ <b>10.19</b> ]	[15.5]	[2.708]	[15.4]	[3.869]	[39.1]	[1.306]	[∞]	
tls7	[ <b>6.585</b> ]	[16.3]	[2.514]	[15.8]	[1.347]	[53.8]	[0.5935]	[∞]	
uflquad-15-60	1495.82	842	1495.51	842	(85.89%)	(0.00%)	1331.24	0	
uflquad-15-80	(40.35%)	(2.62%)	(40.35%)	(2.62%)	(100.00%)	(0.00%)	6321.99	0	
uflquad-20-40	276.61	1291	275.81	1291	(87.06%)	(0.00%)	1139.13	0	
uflquad-20-50	(62.56%)	(7.49%)	(62.56%)	(7.49%)	(100.00%)	(5.07%)	(68.14%)	(∞)	
uflquad-25-25	18.84	433	18.61	433	(65.93%)	(0.00%)	187.67	0	
uflquad-25-30	26.87	356	26.63	356	(96.65%)	(0.00%)	177.66	0	
uflquad-25-40	303.48	1757	303.92	1757	(76.58%)	(0.00%)	2968.56	0	
geom. mean	[140]	32.8	7163.5	155.6	68945.3	86.5	1.0	18.3	1.0
sh. geom. mean	[140]	69.3	12339.2	255.4	78614.6	146.2	0.0	43.3	0.0
arith. mean	[140]	1081.2	861.5k	2494.9	2352.1k	1566.3	0	816.4	0
arith. mean	[137]	(1.71%)	(0.09%)	(12.66%)	(5.22%)	(5.75%)	(0.47%)	(3.40%)	(5.84%)
#solved	[155]		136		103		127		142
#timeout	[155]		18		52		27		12
#failed/aborted	[155]		1		0		1		1
#fastest	[155]		13		10		1		76
#best dual bound	[155]		140		104		129		142
#best primal bnd.	[155]		144		110		141		142

Table A.6.: Detailed results on convex MINLP test set (part II).

instance	BONMIN-BB		BONMIN-ECP		BONMIN-Hyb		BONMIN-OA	
	time	nodes	time	nodes	time	nodes	time	nodes
	(dual gap)	(pr. gap)	(dual gap)	(pr. gap)	(dual gap)	(pr. gap)	(dual gap)	(pr. gap)
	[dual bnd]	[pr. bnd]	[dual bnd]	[pr. bnd]	[dual bnd]	[pr. bnd]	[dual bnd]	[pr. bnd]
BatchS101006M	18.35	436	18.88	490	25.87	513	443.55	0
BatchS121208M	37.37	570	40.01	272	46.46	435	655.49	0
BatchS151208M	110.92	1790	77.08	466	60.05	432	4421.22	0
BatchS201210M	144.97	1560	159.94	692	79.84	532	(1.74%)	(∞)
clay0203h	6.87	203	2.58	372	4.19	344	3.16	0
clay0203m	4.29	251	<b>0.90</b>	224	2.70	214	2.20	0
clay0204h	52.78	1622	5.16	193	11.43	292	7.69	0
clay0204m	9.15	1208	<b>1.29</b>	226	4.41	209	3.34	0
clay0205h	1775.54	23.9k	110.19	1139	110.89	1059	1044.70	0
clay0205m	708.55	21.4k	7.47	870	16.70	810	122.47	0

continue next page...

A. Tables

... continued detailed results on convex MINLP test set								
instance	BONMIN-BB		BONMIN-ECP		BONMIN-Hyb		BONMIN-OA	
clay0303h	18.61	335	7.63	959	9.70	1036	5.96	0
clay0303m	5.56	349	2.42	614	3.89	508	5.64	0
clay0304h	208.03	2684	361.61	38.7k	493.16	35.5k	263.10	0
clay0304m	61.53	2867	48.50	14.8k	56.53	14.7k	304.52	0
clay0305h	1662.50	15.4k	111.38	1792	171.67	1756	3838.39	0
clay0305m	250.63	16.9k	9.26	1344	11.68	1272	258.45	0
FLay03H	2.25	106	<b>0.96</b>	70	1.95	52	1.98	0
FLay03M	<b>0.86</b>	102	<b>0.19</b>	60	<b>1.05</b>	58	<b>0.70</b>	0
FLay04H	38.97	2536	8.03	860	18.53	852	188.16	0
FLay04M	12.84	2644	<b>2.25</b>	842	12.11	822	21.44	0
FLay05H	3506.48	89.6k	300.29	24.1k	315.75	24.3k	(46.29%)	(∞)
FLay05M	617.43	85.1k	69.65	21.9k	75.77	22.1k	(46.29%)	(∞)
FLay06H	(10.36%)	<b>(0.00%)</b>	(7.93%)	<b>(0.00%)</b>	(9.10%)	<b>(0.00%)</b>	(48.25%)	(∞)
FLay06M	(3.64%)	(0.00%)	(4.37%)	(0.00%)	(4.51%)	(0.00%)	(48.25%)	(∞)
fo7_2	2031.27	120.3k	669.63	52.1k	99.35	20.7k	2450.54	0
fo7_ar2_1	(2.41%)	(0.00%)	289.68	27.7k	401.35	34.1k	7014.39	0
fo7_ar25_1	(10.54%)	(10.81%)	98.50	9997	220.13	17.0k	(100.00%)	(∞)
fo7_ar3_1	(13.47%)	(7.45%)	253.58	23.6k	192.77	18.8k	(100.00%)	(∞)
fo7_ar4_1	(2.55%)	(0.00%)	255.64	21.8k	259.75	19.3k	(100.00%)	(∞)
fo7_ar5_1	4544.92	207.0k	52.88	3897	55.15	2682	3347.92	0
fo7	(0.34%)	(0.00%)	585.45	101.0k	1100.12	149.1k	(100.00%)	(∞)
fo8_ar2_1	(21.70%)	(60.45%)	2852.81	160.6k	2301.88	113.5k	(100.00%)	(∞)
fo8_ar25_1	(25.61%)	(∞)	1939.58	84.8k	3448.72	129.3k	(100.00%)	(∞)
fo8_ar3_1	(10.26%)	(65.84%)	741.78	26.4k	1526.50	47.4k	(100.00%)	(∞)
fo8_ar4_1	(11.76%)	(27.19%)	745.52	24.9k	802.06	23.6k	(100.00%)	(∞)
fo8_ar5_1	(21.77%)	(9.12%)	901.84	32.6k	748.53	25.9k	(100.00%)	(∞)
fo8	(12.16%)	(20.02%)	3786.54	340.1k	3903.72	154.8k	(100.00%)	(∞)
fo9_ar2_1	(24.49%)	(∞)	(11.66%)	(0.00%)	(15.31%)	(5.54%)	(100.00%)	(∞)
fo9_ar25_1	(31.80%)	(∞)	(15.51%)	(0.20%)	(20.04%)	(0.59%)	(100.00%)	(∞)
fo9_ar3_1	(16.04%)	(∞)	5716.61	128.0k	4415.42	82.8k	(100.00%)	(∞)
fo9_ar4_1	(22.24%)	(∞)	4651.03	98.9k	2857.26	55.3k	(100.00%)	(∞)
fo9_ar5_1	(19.97%)	(∞)	2552.45	52.2k	5114.29	120.0k	(100.00%)	(∞)
fo9	(34.59%)	(29.83%)	(16.69%)	(0.00%)	(29.90%)	(0.00%)	(100.00%)	(∞)
m3	1.35	69	<b>0.24</b>	39	<b>0.41</b>	26	<b>0.20</b>	0
m6	46.27	3035	7.32	2149	17.35	2086	19.77	0
m7_ar2_1	2569.77	96.3k	163.72	6322	79.18	7747	150.45	0
m7_ar25_1	166.50	6560	11.17	1134	20.53	902	51.63	0
m7_ar3_1	(7.66%)	(0.00%)	91.97	9775	84.14	7106	726.33	0
m7_ar4_1	(12.77%)	(39.21%)	34.59	3311	134.73	1699	864.49	0
m7_ar5_1	(10.74%)	(0.00%)	68.30	6902	173.03	10.3k	1070.70	0
m7	507.31	27.0k	184.24	24.0k	57.02	6244	140.01	0
nd-10	1189.28	9955	33.67	2014	42.59	1904	448.20	0
nd-12	(3.39%)	(2.88%)	170.71	3433	183.50	3735	660.68	0
nd-13	(20.09%)	(3.14%)	(7.65%)	(0.00%)	(6.56%)	(0.00%)	(40.25%)	(∞)
nd-15	(19.03%)	(12.32%)	(7.18%)	(11.33%)	(7.54%)	(10.56%)	(32.13%)	(∞)
nd-16	(17.18%)	(1.09%)	(4.15%)	(0.13%)	(5.27%)	(0.80%)	(40.40%)	(∞)
netmod_dol1	(5.13%)	<b>(0.00%)</b>	(11.31%)	<b>(0.00%)</b>	(11.24%)	(0.22%)	(27.33%)	(∞)
netmod_dol2	1936.81	3274	228.55	398	337.41	568	(9.10%)	(∞)
netmod_kar1	15.02	473	42.58	1148	29.94	870	(33.02%)	(∞)
netmod_kar2	15.04	473	42.85	1148	29.92	870	(33.02%)	(∞)
no7_ar2_1	(21.85%)	(16.49%)	242.31	37.7k	204.20	32.2k	(100.00%)	(∞)
no7_ar25_1	(20.00%)	(6.76%)	602.93	96.1k	613.46	91.3k	(100.00%)	(∞)
no7_ar3_1	(24.73%)	(8.73%)	1243.38	217.3k	1017.76	166.8k	(100.00%)	(∞)
no7_ar4_1	(16.25%)	(1.93%)	823.25	132.5k	879.87	135.1k	(100.00%)	(∞)

continue next page...

A.2. Detailed Benchmark Results

... continued detailed results on convex MINLP test set

instance	BONMIN-BB		BONMIN-ECP		BONMIN-Hyb		BONMIN-OA	
no7_ar5_1	(4.07%)	(0.00%)	477.52	72.4k	373.77	58.0k	(100.00%)	(∞)
o7_2	(22.28%)	(0.00%)	1422.88	417.7k	1688.17	560.7k	(100.00%)	(∞)
o7_ar2_1	(24.13%)	(9.14%)	663.40	110.9k	596.84	102.0k	(100.00%)	(∞)
o7_ar25_1	(29.92%)	(7.55%)	1119.57	228.9k	1431.40	254.7k	(100.00%)	(∞)
o7_ar3_1	(27.54%)	(5.92%)	1992.60	380.1k	1752.83	347.6k	(100.00%)	(∞)
o7_ar4_1	(25.95%)	(3.86%)	2899.66	531.1k	2517.33	466.9k	(100.00%)	(∞)
o7_ar5_1	(21.19%)	(1.56%)	1073.75	184.6k	1246.51	180.8k	(100.00%)	(∞)
o7	(27.01%)	(0.00%)	4180.13	1223.4k	4029.94	1462.0k	(100.00%)	(∞)
o8_ar4_1	(45.03%)	(22.81%)	(19.23%)	(5.37%)	(18.08%)	(4.18%)	(100.00%)	(∞)
o9_ar4_1	(38.72%)	(∞)	(30.56%)	(17.41%)	(31.04%)	(17.60%)	(100.00%)	(∞)
RSyn0810H	3.08	34	<b>0.51</b>	6	1.11	12	<b>0.38</b>	0
RSyn0810M02H	18.25	44	<b>1.89</b>	20	4.09	56	3.17	0
RSyn0810M02M	(52.80%)	(1.65%)	192.15	7661	83.90	2716	1576.37	0
RSyn0810M03H	53.81	150	5.89	44	10.85	142	16.15	0
RSyn0810M03M	(38.07%)	(0.37%)	516.21	9411	1045.39	16.2k	(100.00%)	(∞)
RSyn0810M04H	30.98	8	5.64	34	7.69	52	5.92	0
RSyn0810M04M	(34.50%)	(1.07%)	628.94	7144	1295.91	16.2k	(100.00%)	(∞)
RSyn0810M	450.35	25.9k	1.32	172	1.49	147	5.45	0
RSyn0820H	5.51	73	<b>1.23</b>	30	2.52	63	2.83	0
RSyn0820M02H	22.94	40	<i>fail</i>		4.12	56	4.78	0
RSyn0820M02M	(100.00%)	(10.88%)	2544.98	107.0k	544.87	16.0k	(100.00%)	(∞)
RSyn0820M03H	75.25	194	<i>fail</i>		12.27	126	40.62	0
RSyn0820M03M	(100.00%)	(3.77%)	(21.92%)	(0.97%)	(14.02%)	(0.00%)	(100.00%)	(∞)
RSyn0820M04H	91.39	89	<i>fail</i>		12.40	108	23.84	0
RSyn0820M04M	(100.00%)	(4.20%)	(34.08%)	(1.83%)	(40.85%)	(3.42%)	(100.00%)	(∞)
RSyn0820M	3363.64	154.6k	3.08	297	8.05	999	26.10	0
RSyn0840H	3.73	14	<i>fail</i>		1.43	8	<b>0.94</b>	0
RSyn0840M02H	23.32	36	<b>1.86</b>	16	3.35	12	2.63	0
RSyn0840M02M	(100.00%)	(13.39%)	(2.61%)	(0.38%)	(11.87%)	(0.65%)	4692.20	0
RSyn0840M03H	66.91	109	6.84	49	<i>fail</i>		16.82	0
RSyn0840M03M	(100.00%)	(3.60%)	(2.19%)	(0.25%)	(18.20%)	(1.05%)	(100.00%)	(∞)
RSyn0840M04H	240.46	423	17.80	120	21.55	113	20.53	0
RSyn0840M04M	(100.00%)	(10.56%)	(49.23%)	(4.52%)	(50.11%)	(2.42%)	(100.00%)	(∞)
RSyn0840M	(93.08%)	(2.15%)	2.93	146	16.78	514	71.22	0
SLay05H	4.45	35	2.49	152	8.45	201	17.26	0
SLay05M	2.58	47	<b>0.46</b>	62	<b>1.23</b>	72	2.31	0
SLay06H	54.54	96	11.50	401	25.80	462	1071.96	0
SLay06M	27.41	100	<b>0.89</b>	166	3.04	188	5.90	0
SLay07H	67.97	233	60.67	1101	89.77	1801	(4.62%)	(∞)
SLay07M	66.46	235	<b>1.72</b>	314	5.13	702	75.29	0
SLay08H	69.26	270	174.78	2801	231.41	3401	(4.95%)	(∞)
SLay08M	67.53	265	5.66	761	14.11	602	4354.49	0
SLay09H	<b>79.28</b>	416	(1.95%)	(0.16%)	3633.10	35.3k	(4.34%)	(∞)
SLay09M	69.88	393	28.11	2246	42.79	3251	(4.34%)	(∞)
SLay10H	<b>336.65</b>	7314	(7.42%)	(0.39%)	(6.74%)	(0.18%)	(8.10%)	(∞)
SLay10M	<b>144.97</b>	6692	396.93	21.1k	483.97	26.0k	(8.10%)	(∞)
sssd-10-4-3	12.56	1778	4.88	4443	<b>1.66</b>	750	295.47	0
sssd-12-5-3	52.54	6779	24.98	18.6k	42.04	33.0k	(61.73%)	(∞)
sssd-15-6-3	231.95	25.5k	93.92	52.6k	61.32	31.9k	(64.96%)	(∞)
sssd-16-8-3	(0.09%)	(0.05%)	966.87	407.9k	761.98	326.0k	(63.85%)	(∞)
sssd-18-8-3	(0.06%)	(0.01%)	2007.89	635.6k	841.67	289.3k	(63.48%)	(∞)
sssd-20-9-3	(0.18%)	(0.17%)	(0.34%)	(0.07%)	(0.20%)	(0.06%)	(62.32%)	(∞)
sssd-22-8-3	(0.04%)	(0.03%)	(0.06%)	(0.03%)	(0.04%)	(0.01%)	(63.69%)	(∞)
sssd-8-4-3	9.69	1393	<b>1.37</b>	1011	<b>1.32</b>	557	24.53	0

continue next page...

A. Tables

... continued detailed results on convex MINLP test set									
instance	BONMIN-BB		BONMIN-ECP		BONMIN-Hyb		BONMIN-OA		
Syn10H	<b>0.35</b>	2	<b>0.17</b>	0	<b>0.64</b>	0	<b>0.15</b>	0	
Syn10M02H	<b>0.32</b>	0	<b>0.26</b>	0	<b>0.85</b>	0	<b>0.22</b>	0	
Syn10M02M	4.55	246	<b>0.42</b>	20	<b>0.58</b>	12	<b>0.38</b>	0	
Syn10M03H	<b>0.72</b>	0	<b>0.30</b>	0	<b>1.04</b>	0	<b>0.26</b>	0	
Syn10M03M	19.67	878	<b>0.98</b>	24	<b>0.61</b>	16	1.21	0	
Syn10M04H	<b>1.06</b>	0	<b>0.18</b>	0	<b>1.08</b>	0	<b>0.24</b>	0	
Syn10M04M	56.47	1932	<b>1.02</b>	26	<b>1.07</b>	20	<b>1.11</b>	0	
Syn10M	<b>0.63</b>	30	<i>fail</i>		<i>fail</i>		<b>0.12</b>	0	
Syn20H	<b>0.41</b>	0	<b>0.32</b>	4	<b>0.94</b>	0	<b>0.23</b>	0	
Syn20M02H	2.25	6	<b>0.56</b>	0	<b>1.16</b>	0	<b>0.53</b>	0	
Syn20M02M	458.70	16.9k	1.35	70	1.30	62	1.79	0	
Syn20M03H	4.22	12	<b>0.58</b>	0	2.10	0	<b>0.40</b>	0	
Syn20M03M	<i>fail</i>		2.10	60	2.95	86	11.12	0	
Syn20M04H	7.61	16	<b>0.85</b>	0	2.99	0	<b>0.67</b>	0	
Syn20M04M	<i>fail</i>		8.15	222	9.05	320	8.90	0	
Syn20M	4.81	596	<b>0.22</b>	22	<b>0.46</b>	16	<b>0.34</b>	0	
Syn40H	<b>1.53</b>	10	<b>0.69</b>	10	<b>1.23</b>	18	<b>0.92</b>	0	
Syn40M02H	3.52	10	<b>1.42</b>	14	<b>1.56</b>	14	<b>1.44</b>	0	
Syn40M02M	(100.00%)	(7.29%)	8.19	314	9.76	342	255.97	0	
Syn40M03H	17.11	42	5.85	95	15.47	113	5.62	0	
Syn40M03M	(100.00%)	(16.20%)	76.37	1612	3068.07	74.8k	(100.00%)	( $\infty$ )	
Syn40M04H	53.44	110	5.75	62	15.33	70	16.98	0	
Syn40M04M	(100.00%)	(10.30%)	4042.35	65.5k	5722.48	94.3k	(100.00%)	( $\infty$ )	
Syn40M	973.03	57.1k	<b>0.76</b>	66	1.26	102	1.72	0	
tls12	<b>[21.19]</b>	[ $\infty$ ]	<b>[2.54]</b>	[ $\infty$ ]	<b>[2.593]</b>	[ $\infty$ ]	<b>[2.312]</b>	[ $\infty$ ]	
tls2	5.97	622	<b>0.50</b>	130	11.31	144	<b>0.95</b>	0	
tls4	(31.83%)	(0.00%)	1082.38	248.5k	1252.15	254.6k	(79.41%)	( $\infty$ )	
tls5	(48.34%)	( $\infty$ )	(41.82%)	(4.85%)	(47.57%)	(6.80%)	(88.55%)	( $\infty$ )	
tls6	[6.955]	[ $\infty$ ]	[3.018]	[ $\infty$ ]	[4.749]	[17]	[1.306]	[ $\infty$ ]	
tls7	[4.4]	[ $\infty$ ]	[0.846]	[ $\infty$ ]	[1.125]	[ $\infty$ ]	[0.5935]	[ $\infty$ ]	
uflquad-15-60	12.46	604	55.55	932	55.19	858	(61.76%)	( $\infty$ )	
uflquad-15-80	22.61	853	135.80	1412	124.76	1308	(63.17%)	( $\infty$ )	
uflquad-20-40	<b>17.25</b>	946	63.03	1510	58.32	1338	(62.15%)	( $\infty$ )	
uflquad-20-50	<b>225.14</b>	10.4k	(19.46%)	(0.00%)	(16.29%)	(0.00%)	(68.14%)	( $\infty$ )	
uflquad-25-25	10.52	602	19.60	614	28.77	630	3815.85	0	
uflquad-25-30	12.52	538	25.41	660	26.89	544	3352.38	0	
uflquad-25-40	<b>23.35</b>	1032	129.85	2384	107.65	1932	(61.92%)	( $\infty$ )	
geom. mean	[140]	326.6	6591.3	75.9	3049.9	94.5	3056.7	407.8	1.0
sh. geom. mean	[140]	452.9	9746.3	142.6	5062.6	157.2	5033.9	654.4	0.0
arith. mean	[140]	3207.5	123.5k	1495.0	107.7k	1497.7	102.7k	3790.4	0
arith. mean	[137]	(13.66%)	(8.35%)	(2.09%)	(0.27%)	(2.41%)	(0.28%)	(35.98%)	(46.72%)
#solved	[155]		90		125		129		82
#timeout	[155]		63		25		24		73
#failed/aborted	[155]		2		5		2		0
#fastest	[155]		6		2		0		1
#best dual bound	[155]		91		125		129		82
#best primal bnd.	[155]		102		132		136		82

Table A.7.: Detailed results on convex MINLP test set (part III).

instance	BONMIN-QG		DICOPT		KNITRO		SBB	
	time	nodes	time	nodes	time	nodes	time	nodes
	(dual gap) [dual bnd]	(pr. gap) [pr. bnd]						
BatchS101006M	8.34	308	7.67	0	143.27	3067	81.61	5566
BatchS121208M	38.64	498	7.21	0	5227.53	4283	99.40	4948
BatchS151208M	64.75	714	13.13	0	(0.36%)	( $\infty$ )	123.50	5255
BatchS201210M	86.60	502	<b>12.83</b>	0	(0.50%)	( $\infty$ )	517.82	17.6k
clay0203h	2.30	354	3.29	0	1.91	131	3.94	390
clay0203m	<b>0.79</b>	188	3.44	0	1.35	205	1.84	297
clay0204h	2.00	165	<b>0.38</b>	0	9.81	1507	43.69	3687
clay0204m	1.58	258	<b>0.41</b>	0	10.37	2629	10.93	2649
clay0205h	26.18	1201	38.33	0	268.14	19.7k	688.04	61.3k
clay0205m	6.93	1156	102.25	0	114.61	18.8k	231.49	49.8k
clay0303h	5.40	894	57.28	0	5.79	291	6.96	620
clay0303m	2.50	468	35.60	0	2.18	337	<i>fail</i>	
clay0304h	246.08	42.1k	(82.05%)	(3.84%)	37.25	1883	510.21	30.5k
clay0304m	49.96	15.1k	(80.99%)	(46.34%)	21.73	2289	94.46	17.1k
clay0305h	45.93	1692	243.57	0	200.50	12.6k	1521.99	103.1k
clay0305m	10.29	1577	1218.02	0	90.13	16.5k	186.26	34.4k
FLay03H	<b>0.54</b>	72	1.84	0	<b>0.25</b>	109	<b>1.02</b>	106
FLay03M	<b>0.31</b>	66	<b>0.73</b>	0	<b>0.21</b>	111	<b>0.64</b>	106
FLay04H	6.07	988	291.08	0	12.14	2655	25.21	2721
FLay04M	<b>1.99</b>	908	78.14	0	4.29	2875	10.09	2472
FLay05H	331.25	23.3k	(15.80%)	(0.00%)	647.44	98.1k	<i>fail</i>	
FLay05M	71.76	23.9k	(2.61%)	(0.00%)	169.65	93.5k	504.31	102.5k
FLay06H	(9.40%)	<b>(0.00%)</b>	(18.06%)	<b>(0.00%)</b>	(3.64%)	<b>(0.00%)</b>	(22.75%)	<b>(0.00%)</b>
FLay06M	(3.64%)	(0.00%)	(17.83%)	(0.00%)	(6.60%)	(0.00%)	<i>fail</i>	
fo7_2	136.06	52.2k	<i>abort</i>		(39.29%)	( $\infty$ )	(37.49%)	(0.00%)
fo7_ar2_1	119.24	27.2k	<i>abort</i>		(18.98%)	( $\infty$ )	(21.81%)	(40.38%)
fo7_ar25_1	112.58	22.6k	<i>abort</i>		(17.90%)	( $\infty$ )	(25.26%)	(11.19%)
fo7_ar3_1	161.87	31.7k	6582.81	0	(23.51%)	( $\infty$ )	(30.28%)	(2.68%)
fo7_ar4_1	83.85	17.0k	491.45	0	(24.94%)	( $\infty$ )	(28.96%)	(42.19%)
fo7_ar5_1	21.53	4443	<i>abort</i>		(23.24%)	( $\infty$ )	(17.53%)	(7.11%)
fo7	458.81	176.4k	<i>abort</i>		(37.35%)	( $\infty$ )	(50.83%)	(37.25%)
fo8_ar2_1	2570.56	377.2k	<i>abort</i>		(23.75%)	( $\infty$ )	(44.90%)	(59.59%)
fo8_ar25_1	1848.86	252.3k	<i>abort</i>		(28.60%)	( $\infty$ )	(49.11%)	(48.44%)
fo8_ar3_1	377.75	46.5k	<i>abort</i>		(28.77%)	( $\infty$ )	(51.16%)	(70.60%)
fo8_ar4_1	306.49	35.0k	(9.53%)	(0.00%)	(25.02%)	( $\infty$ )	(50.09%)	(49.52%)
fo8_ar5_1	433.68	49.2k	(12.12%)	(53.71%)	(31.30%)	( $\infty$ )	(45.97%)	(100.00%)
fo8	4112.32	1121.3k	<i>abort</i>		(52.56%)	( $\infty$ )	(65.40%)	(83.49%)
fo9_ar2_1	<i>abort</i>		(0.15%)	(0.00%)	(23.28%)	( $\infty$ )	(62.29%)	( $\infty$ )
fo9_ar25_1	(15.95%)	(0.59%)	<i>abort</i>		(32.71%)	( $\infty$ )	(81.32%)	( $\infty$ )
fo9_ar3_1	2350.42	202.2k	<i>abort</i>		(24.39%)	( $\infty$ )	(70.58%)	( $\infty$ )
fo9_ar4_1	2925.28	218.6k	<i>abort</i>		(30.75%)	( $\infty$ )	(67.42%)	( $\infty$ )
fo9_ar5_1	2093.26	143.6k	<i>abort</i>		(35.36%)	( $\infty$ )	(66.65%)	(77.55%)
fo9	(27.84%)	(0.00%)	<i>abort</i>		(62.42%)	( $\infty$ )	(72.26%)	(100.00%)
m3	<b>0.16</b>	37	<b>0.11</b>	0	<b>0.47</b>	51	<b>0.47</b>	52
m6	12.35	7344	<b>0.56</b>	0	504.44	11.5k	1321.29	112.7k
m7_ar2_1	38.61	9459	10.42	0	(17.55%)	( $\infty$ )	2283.07	170.9k
m7_ar25_1	6.91	1158	<b>0.83</b>	0	4701.29	11.8k	4391.67	237.9k

continue next page...

A. Tables

... continued detailed results on convex MINLP test set									
instance	BONMIN-QG		DICOPT		KNITRO		SBB		
m7_ar3_1	54.59	11.6k	1878.78	0	(15.78%)	(∞)	(16.47%)	(0.00%)	
m7_ar4_1	30.79	6768	<b>0.69</b>	0	(33.56%)	(∞)	7134.81	395.9k	
m7_ar5_1	34.19	8425	<i>abort</i>		(31.05%)	(∞)	(11.43%)	(0.00%)	
m7	113.52	50.4k	<b>0.50</b>	0	(11.66%)	(∞)	(7.53%)	(0.00%)	
nd-10	<b>28.30</b>	2080	<i>abort</i>		<i>abort</i>		472.24	20.5k	
nd-12	179.47	5538	<i>abort</i>		<i>abort</i>		2721.82	49.4k	
nd-13	(5.13%)	(0.02%)	<i>abort</i>		(17.71%)	(9.40%)	(13.24%)	(4.61%)	
nd-15	(7.75%)	(10.41%)	<b>(2.03%)</b>	(∞)	<i>abort</i>		(12.23%)	(17.16%)	
nd-16	(4.87%)	(0.52%)	(9.74%)	(∞)	<i>abort</i>		(20.16%)	(4.20%)	
netmod_doll1	(13.43%)	(0.22%)	(9.22%)	(4.79%)	(8.54%)	(∞)	(9.23%)	(0.86%)	
netmod_doll2	1022.55	1677	<i>fail</i>		244.43	1055	<i>abort</i>		
netmod_kar1	19.03	808	635.03	0	128.07	8639	204.35	6893	
netmod_kar2	18.89	808	634.37	0	128.00	8639	202.67	6893	
no7_ar2_1	304.92	80.9k	<i>fail</i>		(23.19%)	(∞)	(33.27%)	(40.85%)	
no7_ar25_1	647.52	185.1k	(9.20%)	(∞)	<i>abort</i>		(44.05%)	(38.00%)	
no7_ar3_1	1219.94	360.0k	(12.64%)	(∞)	(38.15%)	(∞)	(47.64%)	(24.68%)	
no7_ar4_1	840.60	224.0k	(11.15%)	(∞)	(38.61%)	(∞)	(44.03%)	(22.52%)	
no7_ar5_1	465.55	120.4k	<i>abort</i>		(46.57%)	(∞)	(44.35%)	(30.63%)	
o7_2	1918.85	829.8k	(42.24%)	(∞)	(74.29%)	(∞)	(49.28%)	(4.62%)	
o7_ar2_1	658.97	192.7k	4091.99	0	(32.22%)	(∞)	(41.66%)	(26.73%)	
o7_ar25_1	1394.31	429.4k	(9.77%)	(∞)	<i>abort</i>		(43.74%)	(28.85%)	
o7_ar3_1	2288.73	714.1k	(11.66%)	(∞)	(41.53%)	(∞)	(44.05%)	(18.88%)	
o7_ar4_1	2707.87	802.7k	(16.60%)	(∞)	(54.79%)	(∞)	(47.40%)	(39.00%)	
o7_ar5_1	1098.24	296.8k	(10.80%)	(17.41%)	(53.26%)	(∞)	(36.37%)	(10.51%)	
o7	4770.57	2094.7k	(44.38%)	(∞)	(52.58%)	(∞)	(60.53%)	(25.63%)	
o8_ar4_1	(17.53%)	(2.93%)	(14.39%)	(∞)	(79.46%)	(∞)	(55.25%)	(25.40%)	
o9_ar4_1	(24.19%)	(12.67%)	(15.94%)	(∞)	(69.55%)	(∞)	(76.07%)	(∞)	
RSyn0810H	<b>0.76</b>	39	<b>0.11</b>	0	3.30	111	2.95	304	
RSyn0810M02H	2.75	82	<b>1.37</b>	0	1227.89	183	20.71	498	
RSyn0810M02M	(33.84%)	(0.06%)	<b>4.28</b>	0	(74.50%)	(∞)	(80.18%)	(8.57%)	
RSyn0810M03H	9.08	200	<b>2.44</b>	0	<i>fail</i>		106.61	1295	
RSyn0810M03M	(75.45%)	(0.71%)	<b>6.35</b>	0	(100.00%)	(∞)	(100.00%)	(12.99%)	
RSyn0810M04H	4.61	68	<i>fail</i>		<i>fail</i>		30.64	256	
RSyn0810M04M	(43.91%)	(0.03%)	<i>fail</i>		(65.22%)	(∞)	(64.41%)	(17.45%)	
RSyn0810M	8.68	1939	<b>0.20</b>	0	493.87	42.7k	788.84	165.6k	
RSyn0820H	<b>1.16</b>	81	<b>0.46</b>	0	15.73	125	4.69	347	
RSyn0820M02H	3.44	92	<b>1.69</b>	0	<i>fail</i>		58.57	886	
RSyn0820M02M	(100.00%)	(3.73%)	7.80	0	(100.00%)	(∞)	(100.00%)	(38.69%)	
RSyn0820M03H	16.61	272	<i>fail</i>		<i>fail</i>		514.57	3987	
RSyn0820M03M	(100.00%)	(0.90%)	28.72	0	(100.00%)	(∞)	(100.00%)	(29.03%)	
RSyn0820M04H	10.78	158	5.41	0	<b>(0.00%)</b>	(∞)	427.05	2115	
RSyn0820M04M	(100.00%)	(8.66%)	215.33	0	(100.00%)	(∞)	(100.00%)	(36.19%)	
RSyn0820M	107.14	20.4k	<b>0.40</b>	0	2660.94	150.1k	7190.59	965.7k	
RSyn0840H	<b>1.02</b>	22	<b>0.33</b>	0	2.18	27	2.19	102	
RSyn0840M02H	<b>2.39</b>	36	<i>fail</i>		<i>abort</i>		18.93	279	
RSyn0840M02M	(100.00%)	(12.83%)	<i>fail</i>		(100.00%)	(∞)	(100.00%)	(74.00%)	
RSyn0840M03H	11.20	128	<i>fail</i>		<i>fail</i>		113.87	623	
RSyn0840M03M	(100.00%)	(2.59%)	<b>3.43</b>	0	(100.00%)	(∞)	(100.00%)	(18.28%)	
RSyn0840M04H	20.61	206	6.83	0	(1.74%)	(∞)	740.91	3023	
RSyn0840M04M	(100.00%)	(7.44%)	<i>fail</i>		(100.00%)	(∞)	(100.00%)	(10.26%)	
RSyn0840M	(41.16%)	(3.99%)	<b>0.44</b>	0	<i>fail</i>		(100.00%)	(24.01%)	
SLay05H	<b>0.91</b>	98	3.03	0	3.52	555	22.22	2786	
SLay05M	<b>0.52</b>	116	<b>0.72</b>	0	1.54	521	2.30	502	
SLay06H	<b>3.11</b>	310	13.60	0	10.04	1185	387.85	37.0k	

continue next page...

## A.2. Detailed Benchmark Results

... continued detailed results on convex MINLP test set									
instance	BONMIN-QG		DICOPT		KNITRO		SBB		
SLay06M	<b>0.98</b>	248	<b>1.63</b>	0	3.71	955	<b>1.12</b>	168	
SLay07H	<b>8.70</b>	655	11.94	0	27.87	2282	(0.08%)	(0.00%)	
SLay07M	2.88	610	67.75	0	11.65	2151	13.67	2658	
SLay08H	<b>35.50</b>	1501	3235.39	0	43.24	2690	(3.46%)	(0.18%)	
SLay08M	8.19	1488	62.28	0	26.81	3687	<b>4.82</b>	803	
SLay09H	1821.43	39.7k	(2.88%)	(3.10%)	112.61	4749	(4.21%)	(13.64%)	
SLay09M	44.27	4232	287.21	0	55.26	7253	330.16	48.2k	
SLay10H	(7.20%)	(1.11%)	(6.59%)	(3.55%)	741.34	25.8k	(7.91%)	(18.27%)	
SLay10M	852.12	32.6k	(6.26%)	(4.03%)	309.82	26.8k	538.42	57.9k	
sssd-10-4-3	<b>1.58</b>	1208	<b>2.20</b>	0	4.29	2763	(6.20%)	(4.33%)	
sssd-12-5-3	5.67	5030	<i>fail</i>		(47.27%)	(100.00%)	(61.12%)	(26.49%)	
sssd-15-6-3	47.48	27.6k	<i>fail</i>		(45.23%)	(100.00%)	(64.75%)	(16.59%)	
sssd-16-8-3	<b>231.39</b>	96.8k	1150.80	0	(58.43%)	(100.00%)	(63.70%)	(15.25%)	
sssd-18-8-3	<b>392.23</b>	158.7k	<i>fail</i>		(31.91%)	(100.00%)	(63.33%)	(21.61%)	
sssd-20-9-3	( <b>0.08%</b> )	(0.03%)	(5.55%)	(4.39%)	(55.24%)	(100.00%)	(62.26%)	(6.73%)	
sssd-22-8-3	3360.52	1126.4k	5898.28	0	(45.95%)	(100.00%)	(63.65%)	(22.52%)	
sssd-8-4-3	<b>1.30</b>	1096	<b>1.64</b>	0	9.32	2015	931.08	242.8k	
Syn10H	<b>0.17</b>	2	<b>0.07</b>	0	<b>0.08</b>	3	<b>0.11</b>	2	
Syn10M02H	<b>0.24</b>	4	<b>0.17</b>	0	<b>0.31</b>	7	<b>0.28</b>	7	
Syn10M02M	<b>0.85</b>	144	<b>0.10</b>	0	7.15	1325	3.34	551	
Syn10M03H	<b>0.29</b>	4	<b>0.20</b>	0	<b>0.23</b>	7	<b>0.45</b>	9	
Syn10M03M	2.90	366	<b>0.13</b>	0	49.71	2779	13.28	1828	
Syn10M04H	<b>0.19</b>	6	<b>0.20</b>	0	<b>0.68</b>	7	<b>0.59</b>	11	
Syn10M04M	7.03	830	<b>0.17</b>	0	524.74	8661	58.31	5847	
Syn10M	<b>0.08</b>	22	<b>0.06</b>	0	<b>0.26</b>	63	<b>0.22</b>	32	
Syn20H	<b>0.24</b>	2	<b>0.13</b>	0	<b>0.15</b>	5	<b>0.20</b>	4	
Syn20M02H	<b>0.65</b>	10	<i>fail</i>		<b>1.15</b>	13	<b>1.00</b>	14	
Syn20M02M	51.22	5716	<b>0.16</b>	0	3179.73	131.2k	819.53	81.3k	
Syn20M03H	<b>1.14</b>	18	<i>fail</i>		1.66	21	<b>1.17</b>	18	
Syn20M03M	633.09	52.2k	<b>0.25</b>	0	(31.47%)	( $\infty$ )	(0.45%)	(0.00%)	
Syn20M04H	1.60	22	<i>fail</i>		2.90	25	1.74	20	
Syn20M04M	4371.90	255.6k	<b>0.37</b>	0	(27.92%)	( $\infty$ )	(33.00%)	(0.91%)	
Syn20M	<b>0.96</b>	348	<b>0.09</b>	0	2.42	763	3.32	684	
Syn40H	<b>0.72</b>	24	<b>0.57</b>	0	<b>0.64</b>	13	<b>0.87</b>	11	
Syn40M02H	<b>0.88</b>	16	<b>0.91</b>	0	2.28	23	3.90	32	
Syn40M02M	(100.00%)	(4.70%)	<b>0.22</b>	0	(100.00%)	( $\infty$ )	(100.00%)	(11.39%)	
Syn40M03H	5.09	110	<b>1.42</b>	0	65.36	79	16.23	123	
Syn40M03M	(100.00%)	(9.39%)	<i>fail</i>		(100.00%)	( $\infty$ )	(100.00%)	(25.52%)	
Syn40M04H	6.65	94	<i>fail</i>		<i>fail</i>		60.92	262	
Syn40M04M	(100.00%)	(7.59%)	<i>fail</i>		(100.00%)	( $\infty$ )	(100.00%)	(7.75%)	
Syn40M	108.66	39.0k	<b>0.20</b>	0	2292.00	175.9k	(100.00%)	(0.00%)	
t1s12	[5.8]	[ $\infty$ ]	[7]	[ $\infty$ ]	[2.805]	[ $\infty$ ]	[2.703]	[ $\infty$ ]	
t1s2	<b>0.24</b>	256	<i>fail</i>		1.47	1042	4.23	1496	
t1s4	3786.17	1575.4k	(21.69%)	( $\infty$ )	<i>abort</i>		(72.59%)	(65.06%)	
t1s5	(60.19%)	(11.65%)	(52.43%)	( $\infty$ )	(40.09%)	( $\infty$ )	(82.42%)	(51.46%)	
t1s6	[4.871]	[ $\infty$ ]	[5.3]	[ $\infty$ ]	[3.573]	<b>[9.658]</b>	[1.929]	[21.4]	
t1s7	[2.4]	[ $\infty$ ]	[3.7]	[ $\infty$ ]	[1.03]	<b>[1.038]</b>	[1.117]	[ $\infty$ ]	
uflquad-15-60	45.70	876	1507.03	0	<b>9.72</b>	621	15.83	604	
uflquad-15-80	117.31	1322	5141.05	0	<b>17.87</b>	867	37.19	880	
uflquad-20-40	52.28	1410	1880.25	0	19.13	1351	19.00	1023	
uflquad-20-50	(14.20%)	(0.00%)	(44.98%)	(4.51%)	244.47	13.9k	1623.38	21.9k	
uflquad-25-25	17.32	618	187.46	0	8.92	617	<b>5.76</b>	474	
uflquad-25-30	18.70	558	163.90	0	10.01	523	<b>4.67</b>	306	
uflquad-25-40	105.81	2162	4831.32	0	40.02	2017	27.50	1202	

continue next page...

A. Tables

... continued detailed results on convex MINLP test set									
instance		BONMIN-QG		DICOPT		KNITRO		SBB	
geom. mean	[140]	85.4	6612.9	–	–	–	–	464.0	22421.0
sh. geom. mean	[140]	154.7	9043.4	–	–	–	–	675.1	28110.8
arith. mean	[140]	1738.2	175.8k	–	–	–	–	3763.3	236.6k
arith. mean	[137]	(9.41%)	(0.67%)	–	–	–	–	(25.79%)	(14.95%)
#solved	[155]		124		84		72		78
#timeout	[155]		30		33		68		73
#failed/aborted	[155]		1		38		15		4
#fastest	[155]		7		25		3		2
#best dual bound	[155]		125		85		73		78
#best primal bnd.	[155]		128		90		76		86

A.2.3. MIQPPs

Table A.8.: Detailed results on MIQQP test set.

instance	SCIP		CPLEX		COUENNE		LINDOAPI	
	time	nodes	time	nodes	time	nodes	time	nodes
	(dual gap) [dual bnd]	(pr. gap) [pr. bnd]						
clay0203m	<b>0.35</b>	48	<b>0.70</b>	97	2.17	280	24.30	22
clay0204m	<b>1.16</b>	671	<b>1.14</b>	850	4.09	988	109.41	11
clay0205m	4.85	10.7k	<b>2.17</b>	5009	15.85	12.5k	653.23	53
clay0303m	<b>0.40</b>	87	<b>0.84</b>	277	2.99	425	<i>fail</i>	
clay0304m	<b>0.72</b>	316	1.94	1707	10.60	4705	244.20	437
clay0305m	4.56	9205	<b>3.17</b>	5629	18.50	15.6k	697.23	67
SLay05H	2.10	197	<b>0.28</b>	32	8.91	2119	7200.59	274.5k
SLay05M	<b>0.76</b>	24	<b>0.30</b>	7	<b>1.12</b>	98	32.25	19
SLay06H	6.62	1265	<b>0.83</b>	83	75.82	25.4k	1036.30	508
SLay06M	<b>1.38</b>	266	<b>0.39</b>	11	1.69	417	7200.42	304.1k
SLay07H	52.78	5370	<b>1.20</b>	134	1758.77	516.6k	273.71	71
SLay07M	6.14	1453	<b>0.44</b>	35	2.82	494	142.69	11
SLay08H	45.83	2671	<b>2.56</b>	369	(1.84%)	(0.00%)	(4.95%)	(14.27%)
SLay08M	7.31	1520	<b>0.19</b>	77	3.15	726	151.23	12
SLay09H	512.12	44.0k	<b>9.40</b>	1408	(3.30%)	(1.22%)	(4.34%)	(11.67%)
SLay09M	37.07	5277	<b>0.45</b>	88	14.15	4349	287.31	21
SLay10H	(4.25%)	(0.15%)	<b>52.91</b>	25.5k	(7.55%)	(8.57%)	(8.10%)	(100.00%)
SLay10M	378.04	52.6k	<b>1.17</b>	755	167.03	60.9k	(0.26%)	(0.00%)
uflquad-15-60	1494.89	842	<b>4.20</b>	621	54.76	682	1394.00	75
uflquad-15-80	(40.35%)	(2.62%)	<b>7.30</b>	865	104.08	961	(1.26%)	(0.00%)
uflquad-20-40	275.97	1291	<b>3.55</b>	995	42.61	1069	800.88	57
uflquad-20-50	(62.56%)	(7.49%)	<b>32.47</b>	12.3k	766.83	20.3k	(38.34%)	(1.17%)
uflquad-25-25	18.64	433	<b>1.68</b>	544	14.94	529	407.13	13
uflquad-25-30	26.62	356	<b>1.89</b>	397	17.02	424	393.10	15
uflquad-25-40	303.59	1757	<b>5.54</b>	1311	59.65	1290	909.72	55
iair04	164.71	372	<b>39.78</b>	665	(1.07%)	( $\infty$ )	( $\infty$ )	( $\infty$ )
iair05	169.80	419	<b>22.97</b>	496	(1.63%)	( $\infty$ )	( $\infty$ )	( $\infty$ )
ibc1	<b>15.32</b>	167	774.38	59.6k	(11.58%)	(0.00%)	(4.33%)	(31.29%)
ibell3a	18.62	41.8k	<b>3.14</b>	25.9k	234.74	151.4k	126.73	1
ibienst1	<b>21.66</b>	9120	<i>abort</i>		643.43	35.5k	(31.69%)	(0.00%)

continue next page...

A.2. Detailed Benchmark Results

instance	SCIP		CPLEX		COUENNE		LINDOAPI	
icap6000	<b>10.96</b>	2129	106.95	15.0k	(0.00%)	(∞)	(∞)	(∞)
icvxp1	[13447]	[1182290]	[328153]	<b>[410303]</b>	<b>[363507]</b>	[568253]	[-∞]	[∞]
ieilD76	34.81	3	<b>8.85</b>	94	2193.96	20.3k	(∞)	(∞)
ilaser0	<i>fail</i>		<b>[2411910]</b>	<b>[2412500]</b>	<i>abort</i>		[-∞]	[∞]
imas284	12.63	17.0k	<b>5.82</b>	20.3k	680.20	156.4k	(5.70%)	(3.64%)
imisc07	39.71	36.7k	<b>31.93</b>	59.8k	(26.30%)	(0.02%)	(49.68%)	(8.95%)
imod011	<b>301.31</b>	1	5148.41	0	<i>abort</i>		[-∞]	[∞]
inug06-3rd	<b>399.69</b>	791	2010.51	1097	(20.56%)	(4.02%)	(∞)	(∞)
inug08	<b>13.69</b>	1	1841.00	8212	36.54	0	4326.97	1
iportfolio	[-0.5278]	[0]	<b>[-0.4944]</b>	<b>[-0.4943]</b>	[-0.4945]	[∞]	<i>fail</i>	
iqap10	<b>296.85</b>	32	345.07	45	<i>fail</i>		(∞)	(∞)
iqiu	<b>56.16</b>	12.2k	61.75	15.3k	3812.19	464.5k	(100.00%)	(84.48%)
iran13x13	28.53	39.7k	<b>9.66</b>	8527	(2.11%)	(0.60%)	(11.22%)	(4.78%)
iran8x32	23.57	20.8k	<b>6.01</b>	5121	(2.18%)	(2.53%)	(3.92%)	(4.62%)
isqp0	(5.54%)	(2.09%)	<b>(0.02%)</b>	<b>(0.00%)</b>	(∞)	(0.72%)	(∞)	(∞)
isqp1	(5.16%)	(5.85%)	<b>(0.13%)</b>	<b>(0.00%)</b>	(∞)	(0.97%)	(∞)	(∞)
isqp	[-21090]	[31396800000]	<b>[-21064]</b>	<b>[-21002]</b>	[-∞]	[-20740]	[-∞]	[∞]
iswath2	<b>148.52</b>	4189	206.78	108.9k	6487.36	4573	(∞)	(∞)
itointqor	(1.14%)	(100.00%)	<b>(0.16%)</b>	<b>(0.00%)</b>	(∞)	(2.00%)	<i>fail</i>	
alan	<b>1.13</b>	5	<b>0.23</b>	0	<b>0.21</b>	18	<b>0.32</b>	1
du-opt5	<b>0.71</b>	80	<b>0.05</b>	16	<i>fail</i>		5742.53	740
du-opt	<b>0.82</b>	240	<b>0.05</b>	41	<i>fail</i>		(1.52%)	(1.90%)
ex1223a	<b>0.17</b>	1	<b>0.14</b>	0	<b>0.13</b>	0	<b>0.12</b>	1
fac3	<b>0.48</b>	7	<b>0.06</b>	7	(17.44%)	(0.00%)	31.73	1
gbd	<b>0.09</b>	1	<b>0.03</b>	0	<b>0.05</b>	0	<b>0.10</b>	1
meanvarx	<b>0.24</b>	3	<b>0.04</b>	0	<b>0.89</b>	96	1.20	2
netmod_dol1	(4.65%)	(0.00%)	<b>3433.26</b>	68.6k	(26.77%)	(21.59%)	(22.00%)	(46.12%)
netmod_dol2	<b>69.14</b>	794	94.96	343	(8.40%)	(30.13%)	(3.91%)	(3.04%)
netmod_kar1	<b>4.55</b>	315	53.53	6299	(14.21%)	(1.04%)	(15.83%)	(0.24%)
netmod_kar2	<b>4.39</b>	315	53.56	6299	(14.21%)	(1.04%)	(16.24%)	(0.00%)
nvs03	<b>0.12</b>	1	<b>0.14</b>	0	<b>0.09</b>	0	<b>0.16</b>	1
nvs10	<b>0.12</b>	1	<b>0.04</b>	6	<b>0.10</b>	1	<b>0.14</b>	1
nvs11	<b>0.15</b>	3	<b>0.15</b>	21	<b>0.13</b>	4	<b>1.08</b>	1
nvs12	<b>0.16</b>	6	<b>0.06</b>	31	<b>0.23</b>	10	4.54	1
nvs15	<b>0.15</b>	3	<b>0.14</b>	0	<b>0.09</b>	0	<b>0.16</b>	1
pb302035	[ <b>941406</b> ]	[4350760]	[-14237700]	<b>[3732920]</b>	[591654]	[∞]	[709694]	[3923680]
pb302055	[ <b>916942</b> ]	[4293990]	[-12945500]	<b>[3657280]</b>	[638020]	[∞]	[758891]	[4158700]
pb302075	[ <b>1088470</b> ]	[4501660]	[-11421100]	<b>[4100330]</b>	[653918]	[∞]	[838373]	[4449650]
pb302095	[ <b>2227670</b> ]	[5928560]	[-5954450]	<b>[5726530]</b>	[1216070]	[∞]	[1898830]	[6006690]
pb351535	[ <b>1484400</b> ]	[5722240]	[-9858140]	<b>[4482780]</b>	[1061080]	[∞]	[1223760]	[5116370]
pb351555	[ <b>1594210</b> ]	[5249910]	[-12823400]	<b>[4639130]</b>	[1151040]	[∞]	[1273050]	[5206860]
pb351575	[ <b>1692200</b> ]	[6776200]	[-10438600]	<b>[6301720]</b>	[1179110]	[∞]	[1417750]	[6527830]
pb351595	[ <b>1675830</b> ]	[7237100]	[-15026100]	<b>[7192950]</b>	[1146550]	[∞]	[1359910]	[7206560]
prob02	<b>0.10</b>	0	<b>0.14</b>	0	<b>0.10</b>	0	<b>0.12</b>	1
prob03	<b>0.11</b>	1	<b>0.03</b>	3	<b>0.09</b>	0	<b>0.12</b>	1
qap	(96.02%)	(3.51%)	(100.00%)	<b>(1.06%)</b>	<b>(79.90%)</b>	(16.33%)	(100.00%)	(11.04%)
st_miqp1	<b>0.09</b>	1	<b>0.03</b>	0	<b>0.09</b>	0	<b>0.12</b>	1
st_miqp2	<b>0.12</b>	1	<b>0.04</b>	0	<b>0.10</b>	2	<b>0.15</b>	1
st_miqp3	<b>0.11</b>	1	<b>0.03</b>	0	<b>0.09</b>	0	<b>0.11</b>	1
st_miqp4	<b>0.15</b>	1	<b>0.04</b>	0	<b>0.10</b>	0	<b>0.10</b>	1
st_miqp5	<b>0.16</b>	1	<b>0.06</b>	0	<b>0.12</b>	0	<b>0.11</b>	1
st_test1	<b>0.05</b>	0	<b>0.06</b>	0	<b>0.10</b>	0	<b>0.16</b>	1
st_test2	<b>0.05</b>	1	<b>0.04</b>	0	<b>0.10</b>	0	<b>0.12</b>	1
st_test3	<b>0.09</b>	1	<b>0.03</b>	0	<b>0.10</b>	0	<b>0.50</b>	1
st_test4	<b>0.12</b>	1	<b>0.04</b>	0	<b>0.09</b>	0	<b>0.16</b>	1

continue next page...

## A. Tables

instance	SCIP		CPLEX		COUENNE		LINDOAPI	
st_test5	<b>0.09</b>	1	<b>0.03</b>	0	<b>0.12</b>	2	<b>0.11</b>	1
st_test6	<b>0.09</b>	1	<b>0.03</b>	0	<b>0.11</b>	2	<b>0.07</b>	1
st_test8	<b>0.12</b>	1	<b>0.03</b>	0	<b>0.20</b>	0	<b>0.57</b>	1
st_testgr1	<b>0.19</b>	47	<b>0.14</b>	38	<b>0.29</b>	94	1.48	3
st_testgr3	<b>0.18</b>	27	<b>0.06</b>	34	1.15	1128	1.80	1
st_testph4	<b>0.12</b>	1	<b>0.04</b>	0	<b>0.06</b>	0	<b>0.05</b>	1
geom. mean [82]	26.6	447.2	13.4	292.1	82.1	380.7	211.0	8.9
sh. geom. mean [82]	68.7	1961.1	41.4	1266.4	195.9	1660.5	427.7	87.4
arith. mean [82]	1546.4	299.6k	1250.0	266.1k	2748.7	169.5k	3395.8	7223
arith. mean [72]	(3.04%)	(0.30%)	(1.39%)	(0.01%)	(6.10%)	(5.40%)	(16.53%)	(15.63%)
#solved [91]		71		74		55		48
#timeout [91]		19		16		31		40
#failed/aborted [91]		1		1		5		3
#fastest [91]		16		53		5		1
#best dual bound [91]		79		80		58		48
#best primal bnd. [91]		72		90		58		52

### A.2.4. MISOCPs

Table A.9.: Detailed results on MISOCP test set.

instance	SCIP		CPLEX		MOSEK	
	time (dual gap) [dual bnd]	nodes (pr. gap) [pr. bnd]	time (dual gap) [dual bnd]	nodes (pr. gap) [pr. bnd]	time (dual gap) [dual bnd]	nodes (pr. gap) [pr. bnd]
classical_200_0	[-0.1287]	[-0.11]	<b>[-0.1223]</b>	<b>[-0.1108]</b>	[-0.1226]	[-0.1105]
classical_200_1	[-0.1286]	[-0.1162]	<b>[-0.1246]</b>	[-0.1162]	[-0.1268]	<b>[-0.1163]</b>
classical_200_2	[-0.126]	[-0.1087]	<b>[-0.1217]</b>	<b>[-0.1099]</b>	[-0.1222]	[-0.1096]
classical_200_3	[-0.1251]	[-0.1034]	<b>[-0.1191]</b>	<b>[-0.1061]</b>	[-0.1196]	[-0.1054]
classical_200_4	[-0.1222]	[-0.1077]	[-0.1168]	<b>[-0.1095]</b>	<b>[-0.1164]</b>	[-0.1095]
classical_200_5	[-0.1308]	[-0.1066]	[-0.1241]	<b>[-0.1112]</b>	<b>[-0.1237]</b>	[-0.1105]
classical_200_6	[-0.1174]	[-0.101]	<b>[-0.1121]</b>	<b>[-0.1029]</b>	[-0.113]	[-0.1021]
classical_200_7	[-0.1333]	[-0.1028]	[-0.1262]	<b>[-0.1124]</b>	<b>[-0.1243]</b>	[-0.1123]
classical_200_8	[-0.1282]	[-0.1156]	<b>[-0.1226]</b>	<b>[-0.1157]</b>	[-0.1233]	<b>[-0.1157]</b>
classical_200_9	[-0.1291]	[-0.1044]	<b>[-0.1234]</b>	<b>[-0.1069]</b>	[-0.1239]	[-0.1045]
classical_20_0	<b>0.68</b>	64	<b>1.01</b>	1095	<b>1.48</b>	555
classical_20_1	<b>1.22</b>	358	<b>1.85</b>	3001	<b>1.28</b>	522
classical_20_2	<b>1.59</b>	277	2.39	7072	<b>1.01</b>	283
classical_20_3	1.91	787	<b>0.68</b>	37	<b>1.05</b>	362
classical_20_4	<b>0.77</b>	160	<b>1.27</b>	2108	<b>0.74</b>	165
classical_20_5	<b>1.81</b>	616	3.41	4899	<b>1.30</b>	377
classical_20_6	<b>0.94</b>	166	<b>1.34</b>	2940	<b>1.16</b>	404
classical_20_7	<b>2.22</b>	478	6.44	10.5k	<b>1.78</b>	934
classical_20_8	<b>0.74</b>	83	<b>0.09</b>	0	<b>0.58</b>	34
classical_20_9	<b>0.73</b>	39	<b>0.08</b>	0	<b>0.16</b>	0
classical_30_0	<b>0.98</b>	51	<b>1.33</b>	2277	2.47	644
classical_30_1	27.30	15.4k	<b>4.38</b>	737	26.93	18.1k
classical_30_2	8.23	3823	<b>1.52</b>	163	27.92	18.4k
classical_30_3	15.55	8041	<b>1.31</b>	101	11.48	6327
classical_30_4	6.35	2885	<b>1.41</b>	149	21.98	13.3k
classical_30_5	<b>2.94</b>	330	<b>3.01</b>	4555	21.16	12.9k

continue next page...

A.2. Detailed Benchmark Results

... continued detailed results on MISOCP test set

instance	SCIP		CPLEX		MOSEK	
classical_30_6	4.88	1086	<b>1.35</b>	119	28.63	18.7k
classical_30_7	<b>1.85</b>	145	<b>0.89</b>	51	<b>1.86</b>	392
classical_30_8	10.58	5419	<b>2.37</b>	323	26.52	17.8k
classical_30_9	<b>1.22</b>	177	3.21	5183	26.91	16.9k
classical_40_0	11.75	4335	<b>1.98</b>	133	16.54	5018
classical_40_1	4.14	657	<b>1.41</b>	69	18.38	7358
classical_40_2	60.86	26.7k	<b>5.58</b>	618	12.46	2388
classical_40_3	20.52	8573	<b>3.91</b>	333	18.29	6708
classical_40_4	61.33	29.3k	<b>7.58</b>	861	34.69	14.1k
classical_40_5	5.35	1511	<b>1.60</b>	97	32.93	14.4k
classical_40_6	4.61	1191	<b>1.29</b>	59	6.67	1115
classical_40_7	23.22	9936	<b>1.95</b>	105	28.41	11.4k
classical_40_8	10.17	3584	<b>1.45</b>	73	5.87	1037
classical_40_9	46.39	19.8k	<b>5.40</b>	573	101.34	41.3k
classical_50_0	608.64	187.3k	<b>30.42</b>	2840	99.54	26.9k
classical_50_1	79.68	28.4k	<b>5.89</b>	431	15.78	2428
classical_50_2	53.40	18.8k	<b>5.93</b>	405	<b>5.97</b>	563
classical_50_3	227.92	83.2k	<b>26.30</b>	2263	29.93	5017
classical_50_4	113.52	39.6k	<b>17.83</b>	1543	48.97	9538
classical_50_5	(0.40%)	(0.00%)	<b>1373.12</b>	149.7k	3586.48	529.4k
classical_50_6	158.38	59.9k	<b>14.43</b>	1173	26.16	4201
classical_50_7	16.32	4653	<b>2.79</b>	125	15.54	5161
classical_50_8	132.23	45.9k	<b>17.05</b>	1396	27.56	4971
classical_50_9	<b>21.50</b>	7145	748.11	178.5k	33.48	11.9k
robust_100_0	790.03	53.2k	(0.54%)	(0.00%)	<b>691.59</b>	8165
robust_100_1	393.95	26.8k	363.60	2540	<b>200.22</b>	1750
robust_100_2	110.11	5351	1057.40	83.1k	<b>69.75</b>	511
robust_100_3	125.54	8408	388.65	41.6k	<b>51.99</b>	374
robust_100_4	180.55	10.3k	<b>76.95</b>	477	101.41	704
robust_100_5	(0.43%)	(0.00%)	<b>636.50</b>	5405	666.59	9083
robust_100_6	30.29	239	<b>9.69</b>	2808	39.27	621
robust_100_7	103.52	7265	110.91	721	<b>78.13</b>	467
robust_100_8	(0.08%)	(0.00%)	289.79	63.6k	<b>28.03</b>	167
robust_100_9	79.03	1940	225.36	35.5k	<b>53.24</b>	383
robust_200_0	(0.45%)	(0.04%)	<b>(0.21%)</b>	<b>(0.00%)</b>	(0.48%)	(0.03%)
robust_200_1	(0.47%)	<b>(0.00%)</b>	<b>(0.12%)</b>	<b>(0.00%)</b>	(0.33%)	(0.01%)
robust_200_2	<b>2076.92</b>	40.4k	2159.63	2040	3186.50	9408
robust_200_3	[-0.138]	[-0.1285]	<b>[-0.1358]</b>	<b>[-0.1286]</b>	[-0.1366]	[-0.1278]
robust_200_4	<b>5222.96</b>	104.5k	(0.11%)	(0.00%)	(0.29%)	(0.00%)
robust_200_5	<b>2012.13</b>	38.7k	(0.20%)	(0.00%)	(0.01%)	(0.00%)
robust_200_6	<b>[-0.1276]</b>	<b>[-0.1236]</b>	[-0.1368]	<b>[-0.1236]</b>	[-0.1289]	[-0.1234]
robust_200_7	<b>1265.35</b>	23.2k	(0.41%)	(0.00%)	(0.01%)	(0.00%)
robust_200_8	(0.33%)	(0.00%)	<b>2786.23</b>	2567	(0.20%)	(0.00%)
robust_200_9	<b>2685.25</b>	54.5k	5758.48	5321	(0.25%)	(0.00%)
robust_20_0	1.27	53	3.76	6496	<b>0.27</b>	0
robust_20_1	<b>0.84</b>	63	<b>0.67</b>	309	1.72	398
robust_20_2	<b>1.24</b>	143	<b>1.44</b>	1931	<b>1.68</b>	384
robust_20_3	<b>0.83</b>	83	1.51	3507	<b>0.12</b>	0
robust_20_4	1.69	547	<b>0.34</b>	306	<b>1.20</b>	52
robust_20_5	<b>0.77</b>	38	<b>0.76</b>	354	<b>0.85</b>	75
robust_20_6	<b>0.80</b>	35	<b>1.23</b>	3188	<b>0.35</b>	0
robust_20_7	<b>0.78</b>	33	<b>1.38</b>	2357	<b>0.62</b>	17
robust_20_8	2.88	943	<b>0.73</b>	555	<b>1.44</b>	296
robust_20_9	<b>0.64</b>	26	<b>0.55</b>	681	<b>1.17</b>	52

continue next page...

A. Tables

... continued detailed results on MISOCP test set						
instance	SCIP		CPLEX		MOSEK	
robust_30_0	2.09	139	<b>1.18</b>	914	<b>0.32</b>	0
robust_30_1	19.79	3776	<b>2.36</b>	2320	17.05	4474
robust_30_2	5.25	341	<b>0.90</b>	324	<b>0.90</b>	20
robust_30_3	1.98	273	<b>1.10</b>	393	<b>0.16</b>	0
robust_30_4	2.95	213	<b>1.02</b>	704	<b>0.18</b>	0
robust_30_5	3.69	87	<b>0.89</b>	977	15.10	4397
robust_30_6	5.70	803	<b>0.49</b>	183	27.68	8914
robust_30_7	5.66	559	<b>1.24</b>	1988	<b>1.67</b>	49
robust_30_8	3.54	157	<b>0.69</b>	536	2.08	127
robust_30_9	15.21	3912	<b>1.36</b>	813	26.53	8530
robust_40_0	10.54	709	<b>0.88</b>	299	<b>1.30</b>	7
robust_40_1	4.97	1067	<b>0.96</b>	737	3.18	33
robust_40_2	5.23	90	<b>2.55</b>	3187	30.84	5797
robust_40_3	4.43	667	<b>3.17</b>	4394	5.23	136
robust_40_4	14.37	252	8.15	5005	<b>2.38</b>	65
robust_40_5	2.54	125	<b>0.63</b>	294	1.87	119
robust_40_6	<b>11.82</b>	1733	183.68	71.3k	<b>12.65</b>	1077
robust_40_7	(0.12%)	(0.00%)	17.02	7065	<b>15.53</b>	2805
robust_40_8	6.09	403	<b>1.10</b>	1312	<b>1.49</b>	31
robust_40_9	20.07	255	9.19	5902	<b>4.38</b>	288
robust_50_0	<b>1.93</b>	23	5.98	13.2k	<b>1.03</b>	0
robust_50_1	3.72	31	<b>1.66</b>	486	<b>2.18</b>	14
robust_50_2	18.88	567	<b>3.83</b>	3466	<b>3.10</b>	26
robust_50_3	19.51	141	<b>3.13</b>	2093	17.25	1537
robust_50_4	17.48	1657	<b>3.81</b>	73	18.32	1306
robust_50_5	56.64	10.9k	<b>2.51</b>	1929	3.67	13
robust_50_6	3.68	377	<b>1.19</b>	571	<b>0.33</b>	0
robust_50_7	19.79	763	41.59	17.0k	<b>6.22</b>	165
robust_50_8	23.27	1690	<b>11.57</b>	363	17.28	731
robust_50_9	(0.09%)	(0.00%)	61.17	19.7k	<b>4.07</b>	38
shortfall_100_0	(0.18%)	(0.00%)	(1.61%)	(0.02%)	<b>4975.38</b>	78.1k
shortfall_100_1	<b>1193.32</b>	126.7k	2137.84	20.5k	1650.82	22.7k
shortfall_100_2	[-1.107]	[-1.1]	[-1.106]	[-1.101]	[-1.104]	[-1.101]
shortfall_100_3	<b>1740.53</b>	172.8k	(1.03%)	(0.08%)	1745.32	20.8k
shortfall_100_4	[-1.121]	[-1.118]	[-1.126]	[-1.118]	[-1.122]	[-1.118]
shortfall_100_5	(0.30%)	(0.00%)	(1.87%)	(0.00%)	<i>abort</i>	
shortfall_100_6	6320.72	647.8k	<b>5271.09</b>	49.6k	(0.37%)	(0.02%)
shortfall_100_7	<b>4328.38</b>	437.5k	5769.55	57.6k	5721.10	83.3k
shortfall_100_8	[-1.117]	[-1.111]	[-1.128]	[-1.111]	[-1.114]	[-1.111]
shortfall_100_9	<b>2128.83</b>	227.7k	(0.72%)	(0.00%)	(0.26%)	(0.00%)
shortfall_200_0	[-1.149]	[-1.127]	[-1.145]	[-1.126]	[-1.146]	[-1.126]
shortfall_200_1	[-1.146]	[-1.135]	[-1.148]	[-1.133]	[-1.15]	[-1.134]
shortfall_200_2	[-1.147]	[-1.121]	[-1.165]	[-1.123]	[-1.145]	[-1.126]
shortfall_200_3	[-1.143]	[-1.12]	[-1.14]	[-1.118]	[-1.139]	[-1.118]
shortfall_200_4	[-1.138]	[-1.123]	[-1.135]	[-1.122]	[-1.137]	[-1.121]
shortfall_200_5	[-1.149]	[-1.124]	[-1.148]	[-1.127]	[-1.148]	[-1.127]
shortfall_200_6	[-1.13]	[-1.113]	[-1.144]	[-1.111]	[-1.129]	[-1.114]
shortfall_200_7	[-1.152]	[-1.13]	[-1.183]	[-1.123]	[-1.151]	[-1.13]
shortfall_200_8	[-1.145]	[-1.135]	[-1.146]	[-1.135]	[-1.147]	[-1.135]
shortfall_200_9	[-1.147]	[-1.118]	[-1.165]	[-1.115]	[-1.143]	[-1.117]
shortfall_20_0	<b>1.05</b>	63	<b>0.31</b>	108	<b>0.18</b>	0
shortfall_20_1	<b>2.15</b>	759	<b>2.35</b>	2081	<b>1.92</b>	425
shortfall_20_2	<b>1.32</b>	190	2.34	3880	<b>1.78</b>	435
shortfall_20_3	<b>2.15</b>	694	3.68	5277	<b>1.59</b>	315

continue next page...

## A.2. Detailed Benchmark Results

... continued detailed results on MISOCP test set							
instance	SCIP		CPLEX		MOSEK		
shortfall_20_4	<b>1.01</b>	190	<b>1.22</b>	1144	<b>1.57</b>	389	
shortfall_20_5	<b>0.93</b>	33	<b>0.50</b>	499	4.24	2146	
shortfall_20_6	<b>0.84</b>	167	<b>0.81</b>	632	4.87	1912	
shortfall_20_7	<b>1.02</b>	17	<b>0.33</b>	167	<b>0.98</b>	127	
shortfall_20_8	<b>0.58</b>	23	<b>0.31</b>	90	<b>1.05</b>	85	
shortfall_20_9	<b>0.75</b>	33	<b>0.36</b>	118	<b>0.19</b>	0	
shortfall_30_0	<b>1.00</b>	98	<b>0.67</b>	249	3.08	680	
shortfall_30_1	<b>8.79</b>	3622	137.37	44.1k	19.55	5352	
shortfall_30_2	<b>7.66</b>	2785	8.85	700	37.09	10.1k	
shortfall_30_3	17.09	8159	<b>3.85</b>	247	30.23	8106	
shortfall_30_4	<b>2.86</b>	511	<b>2.64</b>	2555	14.92	4047	
shortfall_30_5	<b>2.03</b>	125	<b>2.01</b>	1628	10.28	1852	
shortfall_30_6	<b>4.52</b>	790	6.11	5668	35.09	9156	
shortfall_30_7	2.72	149	<b>1.32</b>	936	<b>1.53</b>	83	
shortfall_30_8	<b>22.00</b>	10.3k	<b>21.86</b>	2119	34.98	9623	
shortfall_30_9	3.84	245	<b>1.69</b>	809	28.73	8197	
shortfall_40_0	<b>14.51</b>	4046	110.78	38.9k	24.65	2156	
shortfall_40_1	<b>4.63</b>	399	6.74	7296	7.68	357	
shortfall_40_2	<b>34.00</b>	11.7k	917.16	146.7k	41.62	4378	
shortfall_40_3	20.99	7180	<b>9.29</b>	427	360.20	61.4k	
shortfall_40_4	<b>38.21</b>	14.0k	515.14	69.4k	476.26	79.3k	
shortfall_40_5	<b>3.07</b>	394	13.66	10.4k	35.06	5529	
shortfall_40_6	10.61	2547	<b>4.09</b>	171	123.22	19.7k	
shortfall_40_7	10.89	2353	<b>9.07</b>	290	21.18	3283	
shortfall_40_8	<b>12.74</b>	3680	49.44	26.6k	97.08	14.5k	
shortfall_40_9	<b>50.96</b>	19.0k	475.63	92.5k	244.45	38.0k	
shortfall_50_0	520.61	138.5k	1339.17	54.9k	<b>202.16</b>	17.1k	
shortfall_50_1	35.44	8873	<b>10.70</b>	293	12.22	853	
shortfall_50_2	20.86	5387	331.70	82.2k	<b>12.19</b>	436	
shortfall_50_3	260.51	76.1k	204.79	7157	<b>61.96</b>	3573	
shortfall_50_4	<b>59.19</b>	15.3k	1265.40	117.2k	177.51	16.7k	
shortfall_50_5	4447.67	1136.6k	(1.44%)	(0.04%)	<b>4353.09</b>	220.5k	
shortfall_50_6	51.41	14.4k	107.14	3541	<b>20.21</b>	818	
shortfall_50_7	13.26	2512	34.43	15.1k	<b>3.93</b>	39	
shortfall_50_8	37.12	9651	1097.59	143.0k	<b>34.25</b>	1651	
shortfall_50_9	<b>18.94</b>	4763	95.63	36.9k	23.00	2106	
geom. mean	[169]	47.8	3676.0	35.5	3143.5	45.0	1717.8
sh. geom. mean	[169]	95.1	4571.6	87.7	3720.9	88.6	2785.6
arith. mean	[169]	1677.0	72.3k	1707.3	27.5k	1628.7	17.8k
arith. mean	[144]	(0.02%)	(0.00%)	(0.04%)	(0.00%)	(0.02%)	(0.00%)
#solved	[170]		135		134		135
#timeout	[170]		35		36		34
#failed/aborted	[170]		0		0		1
#fastest	[170]		32		72		38
#best dual bound	[170]		140		147		146
#best primal bnd.	[170]		153		157		149



# Bibliography

- K. Abhishek, S. Leyffer, and J. T. Linderoth. FilmINT: An outer-approximation-based solver for nonlinear mixed integer programs. *INFORMS Journal On Computing*, 22(4): 555–567, 2010. doi:10.1287/ijoc.1090.0373. (cited at pp. 117, 122, and 152)
- T. Achterberg. *Constraint Integer Programming*. PhD thesis, TU Berlin, 2007. urn:nbn:de:0297-zib-11129. (cited at pp. iv, 136, 137, 139, 141, 155, 159, 160, 161, 162, 163, 164, 166, 177, 200, 201, 224, 239, and 240)
- T. Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. doi:10.1007/s12532-008-0001-1. (cited at pp. 159 and 162)
- T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, Special Issue 4(1):77–86, 2007. doi:j.disopt.2006.10.004. (cited at p. 164)
- T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005. doi:10.1016/j.orl.2004.04.002. (cited at pp. 135 and 137)
- T. Achterberg, R. Brinkmann, and M. Wedler. Property checking with constraint integer programming. ZIB-Report 07-37, Zuse Institute Berlin, 2007. urn:nbn:de:0297-zib-10376. (cited at p. 164)
- T. Achterberg, T. Berthold, T. Koch, and K. Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In Perron and Trick [2008], pages 6–20. doi:10.1007/978-3-540-68155-7\_4. (cited at p. 159)
- T. Achterberg, S. Heinz, and T. Koch. Counting solutions of integer programs using unrestricted subtree detection. In Perron and Trick [2008], pages 278–282. doi:10.1007/978-3-540-68155-7\_22. (cited at p. 164)
- T. Achterberg, T. Berthold, and G. Hendel. Rounding and propagation heuristics for mixed integer programming. In D. Klatte, H.-J. Lüthi, and K. Schmedders, editors, *Operations Research Proceedings 2011*, pages 71–76. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29210-1. doi:10.1007/978-3-642-29210-1\_12. (cited at pp. 164 and 223)
- D. Adams. *The Hitchhiker’s Guide to the Galaxy*. Pan Books, 1979. ISSN 0-330-25864-8. (cited at p. 212)
- C. S. Adjiman and C. A. Floudas. Rigorous convex underestimators for general twice-differentiable problems. *Journal of Global Optimization*, 9(1):23–40, 1996. doi:10.1007/BF00121749. (cited at p. 132)

## Bibliography

- C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. Global optimization of mixed-integer nonlinear problems. *Journal of the American Institute of Chemical Engineers*, 46: 1769–1797, 2000. doi:10.1002/aic.690460908. (cited at p. 149)
- T. Ahadi-Oskui. Optimierung des Entwurfs komplexer Energieumwandlungsanlagen. In *Fortschritt-Berichte*, number 543 in Series 6. VDI-Verlag, Düsseldorf, Germany, 2006. ISSN 0178-9414. (cited at p. iii)
- T. Ahadi-Oskui, S. Vigerske, I. Nowak, and G. Tsatsaronis. Optimizing the design of complex energy conversion systems by Branch and Cut. *Computers & Chemical Engineering*, 34(8):1226–1236, 2010. doi:10.1016/j.compchemeng.2010.03.007. (cited at pp. iii and 115)
- A. Ahlatçioğlu, M. Bussieck, M. Esen, M. Guignard, J.-H. Jagla, and A. Meeraus. Combining QCR and CHR for convex quadratic pure 0-1 programming problems with linear constraints. *Annals of Operations Research*, 199:33–49, 2012. doi:10.1007/s10479-011-0969-1. (cited at p. 234)
- S. Ahmed, M. Tawarmalani, and N. V. Sahinidis. A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377, 2004. doi:10.1007/s10107-003-0475-6. (cited at p. 30)
- I. G. Akrotirianakis and C. A. Floudas. A new class of improved convex underestimators for twice differentiable constrained NLPs. *Journal of Global Optimization*, 30(4): 367–390, 2004. doi:10.1007/s10898-004-6455-4. (cited at p. 132)
- F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983. doi:10.1287/moor.8.2.273. (cited at pp. 126 and 127)
- F. A. Al-Khayyal and H. D. Sherali. On finitely terminating branch-and-bound algorithms for some global optimization problems. *SIAM Journal on Optimization*, 10(4):1049–1057, 2000. doi:10.1137/S105262349935178X. (cited at p. 134)
- A. Alonso-Ayuso, L. F. Escudero, and M. T. Ortuño. BFC, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0-1 programs. *European Journal of Operational Research*, 151(3):503–519, 2003. doi:10.1016/S0377-2217(02)00628-8. (cited at p. 40)
- F. Alvarez, J. Amaya, A. Griewank, and N. Strogies. A continuous framework for open pit mine planning. *Mathematical Methods of Operations Research*, 73(1):29–54, 2011. doi:10.1007/s00186-010-0332-3. (cited at p. 206)
- H. Amato and G. Mensch. Rank restrictions on the quadratic form in indefinite quadratic programming. *Unternehmensforschung*, 15(1):214–216, 1971. doi:10.1007/BF01939829. (cited at p. 196)

- I. P. Androulakis, C. D. Maranas, and C. A. Floudas.  $\alpha$ BB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995. doi:10.1007/BF01099647. (cited at p. 149)
- K. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43(2):471–484, 2009. doi:10.1007/s10898-008-9372-0. (cited at p. 143)
- D. Applegate, R. E. Bixby, V. Chvátal, and W. Cook. On the solution of travel salesman problems. In *Documenta Mathematica, Extra Volume ICM III*, pages 645–656, 1998. (cited at p. 137)
- A. Atamtürk and V. Narayanan. Conic mixed-integer rounding cuts. *Mathematical Programming*, 122(1):1–20, 2010. doi:10.1007/s10107-008-0239-4. (cited at p. 142)
- E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3–44, 1998. originally MSRR#348, Carnegie Mellon University, Juli 1974. doi:10.1016/S0166-218X(98)00136-X. (cited at pp. 34, 142, and 144)
- E. Balas. Projection and lifting in combinatorial optimization. In M. Juenger and D. Naddef, editors, *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, volume 2241 of *Lecture Notes in Computer Science*, pages 26–56. Springer, 2001. doi:10.1007/3-540-45586-8\_2. (cited at p. 144)
- M. Ballerstein, D. Michaels, and S. Vigerske. Linear underestimators for bivariate functions with a fixed convexity behavior. ZIB-Report 13-02, Zuse Institute Berlin, 2013. urn:nbn:de:0297-zib-17641. (cited at pp. v and 166)
- V. Bally, G. Pagès, and J. Printems. A quantization tree method for pricing and hedging multidimensional american options. *Mathematical Finance*, 15(1):119–168, 2005. doi:10.1111/j.0960-1627.2005.00213.x. (cited at p. 67)
- B. Bank and R. Mandel. Quantitative stability of (mixed-) integer linear optimization problems. In *Optimization, parallel processing and applications, Proceedings of the Oberwolfach conference on operations research*, volume 304 of *Lecture Notes in Economics and Mathematical Systems*, pages 3–15, Oberwolfach, 1988. Springer. doi:10.1007/978-3-642-46631-1\_1. (cited at pp. 9 and 12)
- B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer. *Non-Linear Parametric Optimization*. Akademie-Verlag, Berlin, 1982. ISBN 3-7643-1375-7. (cited at pp. 9, 10, and 13)
- X. Bao. Automatic convexity detection for global optimization. Master’s thesis, University of Illinois at Urbana-Champaign, 2007. (cited at p. 178)

## Bibliography

- X. Bao, N. V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs. *Optimization Methods and Software*, 24(4-5):485–504, 2009. doi:10.1080/10556780902883184. (cited at pp. 144 and 150)
- X. Bao, N. V. Sahinidis, and M. Tawarmalani. Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. *Mathematical Programming*, 129(1):129–157, 2011. doi:10.1007/s10107-011-0462-2. (cited at p. 117)
- E. M. L. Beale. Branch and bound methods for numerical optimization of non-convex functions. In M. M. Barritt and D. Wishart, editors, *COMPSTAT 80 Proceedings in Computational Statistics*, pages 11–20, Vienna, 1980. Physica-Verlag. (cited at pp. 116 and 146)
- E. M. L. Beale and J. A. Tomlin. Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. In J. Lawrence, editor, *Proceedings of the Fifth International Conference on Operational Research*, volume 69 of *Operational Research*, pages 447–454, London, 1970. Tavistock Publishing. (cited at p. 146)
- P. Belotti. Disjunctive cuts for non-convex MINLP. In Lee and Leyffer [2012], pages 117–144. doi:10.1007/978-1-4614-1927-3\_5. (cited at pp. 142 and 151)
- P. Belotti. Bound reduction using pairs of linear inequalities. *Journal of Global Optimization*, to appear, 2012b. doi:10.1007/s10898-012-9848-9. (cited at p. 151)
- P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009. doi:10.1080/10556780903087124. (cited at pp. 135, 136, 137, 138, 141, 151, and 202)
- P. Belotti, S. Cafieri, J. Lee, and L. Liberti. Feasibility-based bounds tightening via fixed points. In W. Wu and O. Daescu, editors, *Combinatorial Optimization and Applications*, volume 6508 of *Lecture Notes in Computer Science*, pages 65–76. Springer, Berlin/Heidelberg, 2010. doi:10.1007/978-3-642-17458-2\_7. (cited at p. 151)
- P. Belotti, A. Miller, and M. Namazifar. Linear inequalities for bounded products of variables. *SIAG/OPT Views-and-News*, 12(1):1–8, March 2011. (cited at pp. 144 and 196)
- P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. Preprint ANL/MCS-P3060-1121, Argonne National Laboratory, 2012. URL [http://www.optimization-online.org/DB\\_HTML/2012/12/3698.html](http://www.optimization-online.org/DB_HTML/2012/12/3698.html). (cited at p. 117)
- A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26(2):193–205, 2001. doi:10.1287/moor.26.2.193.10561. (cited at p. 195)

- J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962. doi:10.1007/BF01386316. (cited at pp. 22 and 120)
- M. Bénichou, J. M. Gauthier, P. Girodet, G. Hentges, R. Ribi re, and O. Vincent. Experiments in mixed-integer linear programming. *Mathematical Programming*, 1(1):76–94, 1971. doi:10.1007/BF01584074. (cited at p. 135)
- H. Y. Benson. Mixed integer nonlinear programming using interior-point methods. *Optimization Methods and Software*, 26(6):911–931, 2011. doi:10.1080/10556781003799303. (cited at pp. 119 and 154)
- T. Berthold. Primal heuristics for mixed integer programs. Master’s thesis, Technische Universit t Berlin, 2006. urn:nbn:de:0297-zib-10293. (cited at pp. 145, 164, 202, and 204)
- T. Berthold. RENS – relaxation enforced neighborhood search. ZIB-Report 07-28, Zuse Institute Berlin, 2007. urn:nbn:de:0297-zib-4264. (cited at pp. 164 and 204)
- T. Berthold. RENS – the optimal rounding. ZIB-Report 12-17, Zuse Institute Berlin, 2012. urn:nbn:de:0297-zib-15203. (cited at p. 204)
- T. Berthold and A. M. Gleixner. Undercover – a primal heuristic for MINLP based on sub-MIPs generated by set covering. ZIB-Report 09-40, Zuse Institute Berlin, 2009. urn:nbn:de:0297-zib-11632. (cited at pp. v, 145, 164, and 203)
- T. Berthold and A. M. Gleixner. Undercover – a primal MINLP heuristic exploring a largest sub-MIP. ZIB-Report 12-07, Zuse Institute Berlin, 2012. urn:nbn:de:0297-zib-14631. (cited at pp. 145 and 203)
- T. Berthold and M. E. Pfetsch. Detecting orbitopal symmetries. In B. Fleischmann, K. H. Borgwardt, R. Klein, and A. Tuma, editors, *Operations Research Proceedings 2008*, pages 433–438. Springer-Verlag, 2009. doi:10.1007/978-3-642-00142-0\_70. (cited at p. 164)
- T. Berthold, S. Heinz, and M. E. Pfetsch. Nonlinear pseudo-boolean optimization: relaxation or propagation? In O. Kullmann, editor, *Theory and Applications of Satisfiability Testing – SAT 2009*, number 5584 in Lecture Notes in Computer Science, pages 441–446. Springer, 2009a. doi:10.1007/978-3-642-02777-2\_40. (cited at p. 164)
- T. Berthold, S. Heinz, and S. Vigerske. Extending a CIP framework to solve MIQCPs. In Lee and Leyffer [2012], pages 427–444. doi:10.1007/978-1-4614-1927-3\_15. (cited at pp. v, 155, 159, 164, and 237)
- T. Berthold, A. M. Gleixner, S. Heinz, and S. Vigerske. Extending SCIP for solving MIQCPs. In P. Bonami, L. Liberti, A. J. Miller, and A. Sartenaer, editors, *Proceedings of the European Workshop on Mixed Integer Nonlinear Programming*, pages 181–196, 2010a. URL <http://www.lix.polytechnique.fr/~liberti/ewminlp/ewminlp-proceedings.pdf>. (cited at p. v)

## Bibliography

- T. Berthold, S. Heinz, M. Lübbecke, R. H. Möhring, and J. Schulz. A constraint integer programming approach for resource-constrained project scheduling. In A. Lodi, M. Milano, and P. Toth, editors, *Proceedings of CPAIOR 2010*, volume 6140 of *Lecture Notes in Computer Science*, pages 313–317. Springer, June 2010b. doi:10.1007/978-3-642-13520-0\_34. (cited at p. 164)
- T. Berthold, S. Heinz, M. E. Pfetsch, and S. Vigerske. Large neighborhood search beyond MIP. In L. D. Gaspero, A. Schaerf, and T. Stützle, editors, *Proceedings of the 9th Metaheuristics International Conference (MIC 2011)*, pages 51–60, 2011. urn:nbn:de:0297-zib-12989. (cited at pp. v, 145, and 203)
- T. Berthold, A. M. Gleixner, S. Heinz, and S. Vigerske. Analyzing the computational impact of MIQCP solver components. *Numerical Algebra, Control and Optimization*, 2(4):739–748, 2012. doi:10.3934/naco.2012.2.739. (cited at pp. v and 238)
- P. Billingsley. *Convergence of Probability Measures*. Wiley, New York, 1968. doi:10.1002/9780470316962. (cited at p. 19)
- A. Billionnet, S. Elloumi, and M.-C. Plateau. Quadratic 0–1 programming: Tightening linear or quadratic convex reformulation by use of relaxations. *RAIRO - Operations Research*, 42:103–121, 2008. doi:10.1051/ro:2008011. (cited at p. 234)
- J. R. Birge. Decomposition and partitioning methods for multistage stochastic programming. *Operations Research*, 33(5):989–1007, 1985. doi:10.1287/opre.33.5.989. (cited at pp. 22, 26, and 71)
- J. R. Birge and F. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392, 1988. doi:10.1016/0377-2217(88)90159-2. (cited at p. 23)
- J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, 1997. ISBN 978-1-4614-0237-4. (cited at p. 3)
- J. R. Birge, C. J. Donohue, D. F. Holmes, and O. G. Svintsitski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. *Mathematical Programming*, 75(2):327–352, 1996. doi:10.1007/BF02592158. (cited at p. 28)
- R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1):3–15, 2002. doi:10.1287/opre.50.1.3.17780. (cited at p. 116)
- R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: theory and practice – closing the gap. In M. J. D. Powell and S. Scholtes, editors, *System Modelling and Optimization: Methods, Theory and Applications*, pages 19–49. Kluwer Dordrecht, 2000. ISBN 0792378814. (cited at p. 116)

- C. E. Blair and R. G. Jeroslow. The value function of a mixed integer program: I. *Discrete Mathematics*, 19(2):121–138, 1977. doi:10.1016/0012-365X(77)90028-0. (cited at pp. 9 and 12)
- C. E. Blair and R. G. Jeroslow. The value function of a mixed integer program: II. *Discrete Mathematics*, 25(1):7–19, 1979. doi:10.1016/0012-365X(79)90147-X. (cited at pp. 9 and 11)
- C. E. Blair and R. G. Jeroslow. The value function of an integer program. *Mathematical Programming*, 23(1):237–273, 1982. doi:10.1007/BF01583794. (cited at p. 33)
- A. Bley, N. Boland, C. Fricke, and G. Froyland. A strengthened formulation and cutting planes for the open pit mine production scheduling problem. *Computers and Operations Research*, 37(9):1641–1647, 2010. doi:10.1016/j.cor.2009.12.008. (cited at p. 212)
- A. Bley, N. Boland, G. Froyland, and M. Zuckerberg. Solving mixed integer nonlinear programming problems for mine production planning with stockpiling. Technical Report 31, Institute of Mathematics, TU Berlin, 2012a. (cited at pp. 207, 208, 211, and 212)
- A. Bley, A. M. Gleixner, T. Koch, and S. Vigerske. Comparing MIQCP solvers to a specialised algorithm for mine production scheduling. In H. G. Bock, X. P. Hoang, R. Rannacher, and J. P. Schlöder, editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 25–39. Springer Berlin Heidelberg, 2012b. doi:10.1007/978-3-642-25707-0\_3. (cited at pp. v, 206, and 218)
- C. Bliet, P. Spellucci, L. Vicente, A. Neumaier, L. Granvilliers, E. Monfroy, F. Benhamou, E. Huens, P. V. Hentenryck, D. Sam-Haroud, and B. Faltings. Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of The Art. Technical report, Universität Wien, Fakultät für Mathematik, 2001. URL <http://www.mat.univie.ac.at/~neum/glopt/coconut/StArt.html>. COCONUT Deliverable D1. (cited at p. 138)
- N. Boland, I. Dumitrescu, and G. Froyland. A multistage stochastic programming approach to open pit mining production scheduling with uncertain geology. 2008. URL [http://www.optimization-online.org/DB\\_HTML/2008/10/2123.html](http://www.optimization-online.org/DB_HTML/2008/10/2123.html). (cited at p. 206)
- N. Boland, I. Dumitrescu, G. Froyland, and A. M. Gleixner. LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. *Computers & Operations Research*, 36:1064–1089, 2009. doi:10.1016/j.cor.2007.12.006. (cited at p. 206)
- P. Bonami and J. P. M. Gonçalves. Heuristics for convex mixed integer nonlinear programs. *Computational Optimization and Applications*, 51(2), 2012. doi:10.1007/s10589-010-9350-6. (cited at pp. 145, 146, and 151)

## Bibliography

- P. Bonami and J. Lee. *BONMIN Users' Manual*, 1.5 edition, July 2011. <https://projects.coin-or.org/Bonmin>. (cited at p. 124)
- 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. doi:10.1016/j.disopt.2006.10.011. (cited at pp. 117, 122, and 151)
- P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, 119(2):331–352, 2009. doi:10.1007/s10107-008-0212-2. (cited at pp. 145 and 151)
- P. Bonami, M. Kiliç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In Lee and Leyffer [2012], pages 1–40. doi:10.1007/978-1-4614-1927-3\_1. (cited at pp. 118, 119, 121, and 230)
- P. Bonami, J. Lee, S. Leyffer, and A. Wächter. More branch-and-bound experiments in convex nonlinear integer programming. 2011. URL [http://www.optimization-online.org/DB\\_HTML/2011/09/3191.html](http://www.optimization-online.org/DB_HTML/2011/09/3191.html). (cited at pp. 119 and 135)
- B. Borchers and J. E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programming. *Computers & Operations Research*, 21(4):359–367, 1994. doi:10.1016/0305-0548(94)90024-8. (cited at p. 119)
- C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth. On the optimal design of water distribution networks: a practical MINLP approach. *Optimization and Engineering*, 13:219–246, 2012. <http://www.minlp.org/library/problem/index.php?i=134>. doi:10.1007/s11081-011-9141-7. (cited at pp. 147, 205, 218, 219, 220, 221, and 222)
- A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. *GAMS – A User's Guide*. GAMS Development Corp., 2012. URL <http://www.gams.com>. (cited at pp. 147 and 205)
- S. Burer and A. Saxena. The MILP road to MIQCP. In Lee and Leyffer [2012], pages 373–405. doi:10.1007/978-1-4614-1927-3\_13. (cited at p. 143)
- S. Burer and A. N. Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17(2):97–106, 2012. doi:10.1016/j.sorms.2012.08.001. (cited at p. 117)
- M. R. Bussieck. Introduction to GAMS Branch-and-Cut Facility. Technical report, GAMS Development Corp., 2003. URL <http://www.gams.com/docs/bch.htm>. (cited at pp. 147 and 155)
- M. R. Bussieck and S. Vigerske. MINLP solver software. In Cochran et al. [2010]. doi:10.1002/9780470400531.eorms0527. (cited at pp. v, 117, and 146)
- M. R. Bussieck, A. S. Drud, and A. Meeraus. MINLPLib - a collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1):

- 114–119, 2003. <http://www.gamsworld.org/minlp/minlplib.htm>. doi:10.1287/ijoc.15.1.114.15159. (cited at pp. 156, 170, and 225)
- R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In G. di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, pages 35–59. Springer, 2006. doi:10.1007/0-387-30065-1\_4. (cited at p. 153)
- S. Cafieri, J. Lee, and L. Liberti. On convex relaxations of quadrilinear terms. *Journal of Global Optimization*, 47(4):661–685, 2010. doi:10.1007/s10898-009-9484-1. (cited at p. 127)
- C. C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45, 1999. doi:10.1016/S0167-6377(98)00050-9. (cited at pp. 21, 39, 40, and 41)
- C. C. Carøe and J. Tind. A cutting-plane approach to mixed 0-1 stochastic integer programs. *European Journal of Operational Research*, 101(2):306–316, 1997. doi:10.1016/S0377-2217(96)00399-2. (cited at p. 30)
- C. C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1-3):451–464, 1998. doi:10.1007/BF02680570. (cited at p. 33)
- M. S. Casey and S. Sen. The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research*, 30(3):615–631, 2005. doi:10.1287/moor.1050.0146. (cited at p. 94)
- I. Castillo, J. Westerlund, S. Emet, and T. Westerlund. Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. *Computers & Chemical Engineering*, 30(1):54–69, 2005. doi:10.1016/j.compchemeng.2005.07.012. (cited at p. 230)
- M. T. Çezik and G. Iyengar. Cuts for mixed 0-1 conic programming. *Mathematical Programming*, 104(1):179–202, 2005. doi:10.1007/s10107-005-0578-3. (cited at p. 142)
- J.-S. Chen and C.-H. Huang. A note on convexity of two signomial functions. *Journal of Nonlinear and Convex Analysis*, 10(3):429–435, 2009. (cited at p. 179)
- M. Chen and S. Mehrotra. Epi-convergent scenario generation method for stochastic problems via sparse grid. Stochastic Programming E-Print Series 2007-08, 2008. urn:nbn:de:kobv:11-10087768. (cited at p. 7)
- A. Chiralaksanakul and D. P. Morton. Assessing policy quality in multi-stage stochastic programming. Stochastic Programming E-Print Series 2004-12, 2004. urn:nbn:de:kobv:11-10059513. (cited at pp. 89 and 93)

## Bibliography

- J. J. Cochran, L. A. Cox, Jr., P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors. *Wiley Encyclopedia of Operations Research and Management Science*. Wiley & Sons, Inc., 2010. doi:10.1002/9780470400531. (cited at pp. 282, 297, and 303)
- S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971. doi:10.1145/800157.805047. (cited at p. 160)
- W. Cook, A. M. H. Gerards, A. Schrijver, and É. Tardos. Sensitivity theorems in integer linear programming. *Mathematical Programming*, 34(3):251–264, 1986. doi:10.1007/BF01582230. (cited at pp. 9 and 11)
- W. Cook, T. Koch, D. E. Steffy, and K. Wolter. An exact rational mixed-integer programming solver. In O. Günlük and G. J. Woeginger, editors, *Integer Programming and Combinatorial Optimization - 15th International Conference*, volume 6655 of *Lecture Notes in Computer Science*, pages 104–116. Springer, 2011. doi:10.1007/978-3-642-20807-2\_9. (cited at pp. 164 and 172)
- R. W. Cottle. Three remarks about two papers on quadratic forms. *Mathematical Methods of Operations Research*, 19(3):123–124, 1975. doi:10.1007/BF01957172. (cited at p. 196)
- J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: a simplified approach. *Journal on Financial Economics*, 7(3):229–263, 1979. doi:10.1016/0304-405X(79)90015-1. (cited at p. 43)
- J. Czyzyk, M. P. Mesnier, and J. J. Moré. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5(3):68–75, 1998. URL <http://neos.mcs.anl.gov>. doi:10.1109/99.714603. (cited at p. 149)
- C. D’Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. Experiments with a feasibility pump approach for non-convex MINLPs. In Festa [2010], pages 350–360. doi:10.1007/978-3-642-13193-6\_30. (cited at p. 146)
- 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. doi:10.1007/s10107-012-0608-x. (cited at p. 146)
- E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2004. doi:10.1007/s10107-004-0518-7. (cited at p. 204)
- G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3-4):197–206, 1955. doi:10.1287/mnsc.1.3-4.197. (cited at p. 3)
- G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi:10.1287/opre.8.1.101. (cited at p. 53)

- M. A. H. Dempster. Sequential importance sampling algorithms for dynamic stochastic programming. *Journal of Mathematical Sciences*, 133(4):1422–1444, 2006. doi:10.1007/s10958-006-0058-1. (cited at p. 8)
- M. A. H. Dempster and R. T. Thompson. Parallelization and aggregation of nested Benders decomposition. *Annals of Operations Research*, 81:163–188, 1998. doi:10.1023/A:1018996821817. (cited at p. 28)
- D. Dentcheva and W. Römisch. Duality gaps in nonconvex stochastic optimization. *Mathematical Programming*, 101(3):515–535, 2004. doi:10.1007/s10107-003-0496-1. (cited at p. 21)
- A. Dolzmann and T. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997. URL <http://www.redlog.eu>. doi:10.1145/261320.261324. (cited at p. 178)
- F. Domes and A. Neumaier. Constraint propagation on quadratic constraints. *Constraints*, 15(3):404–429, 2010. doi:10.1007/s10601-009-9076-1. (cited at pp. 140 and 181)
- F. Domes and A. Neumaier. Rigorous enclosures of ellipsoids and directed Cholesky factorizations. *SIAM Journal on Matrix Analysis and Applications*, 32(1):262–285, 2011. doi:10.1137/090778110. (cited at p. 181)
- T. Dong. Efficient modeling with the IBM ILOG OPL-CPLEX Development Bundles, 2009. URL <http://www-01.ibm.com/software/integration/optimization/cplex-dev-bundles/library>. (cited at p. 147)
- S. Drewes. *Mixed Integer Second Order Cone Programming*. PhD thesis, Technische Universität Darmstadt, 2009. (cited at pp. 117 and 142)
- A. S. Drud. CONOPT – a large-scale GRG code. *INFORMS Journal on Computing*, 6(2):207–216, 1994. doi:10.1287/ijoc.6.2.207. (cited at pp. 154 and 205)
- O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999. doi:10.1016/S0012-365X(98)00213-1. (cited at p. 71)
- J. Dupačová, G. Consigli, and S. W. Wallace. Scenarios for multistage stochastic programs. *Annals of Operations Research*, 100(1-4):25–53, 2000. doi:10.1023/A:1019206915174. (cited at p. 8)
- J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenario reduction in stochastic programming: An approach using probability metrics. *Mathematical Programming*, 95(3):493–511, 2003. doi:10.1007/s10107-002-0331-0. (cited at p. 7)
- 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. doi:10.1007/BF02592064. (cited at pp. 119, 120, and 121)

## Bibliography

- A. Eichhorn and W. Römis. Stochastic integer programming: Limit theorems and confidence intervals. *Mathematics of Operations Research*, 32(1):118–135, 2007. doi:10.1287/moor.1060.0222. (cited at p. 89)
- A. Epe. *Stochastische Modellierung des Elektrizitätsspeicherzubaues in Deutschland*. PhD thesis, Ruhr-Universität Bochum, 2011. (cited at pp. iv and 112)
- A. Epe, C. Küchler, W. Römis, S. Vigerske, H.-J. Wagner, C. Weber, and O. Woll. Stochastische Optimierung mit rekombinierenden Szenariobäumen - Analyse dezentraler Energieversorgung mit Windenergie und Speichern. In *Optimierung in der Energiewirtschaft*, number 2018 in VDI-Berichte, pages 3–13. VDI-Verlag, Düsseldorf, 2007. ISBN 978-3-18-092018-4. (cited at pp. iii, 44, and 101)
- A. Epe, C. Küchler, W. Römis, S. Vigerske, H.-J. Wagner, C. Weber, and O. Woll. Optimization of dispersed energy supply – stochastic programming with recombining scenario trees. In J. Kallrath, P. Pardalos, S. Rebennack, and M. Scheidt, editors, *Optimization in the Energy Industry*, chapter 15, pages 347–364. Springer, 2009a. doi:10.1007/978-3-540-88965-6\_15. (cited at pp. iii and 101)
- A. Epe, C. Küchler, W. Römis, S. Vigerske, H.-J. Wagner, C. Weber, and O. Woll. Ökonomische Bewertung von elektrischen Energiespeichern - Ausbau und Betrieb im Kontext wachsender Windenergieerzeugung. In Schultz and Wagner [2009], chapter 7, pages 135–152. ISBN 978-3-8258-1359-8. (cited at pp. iii, 44, and 101)
- L. F. Escudero, A. Garín, M. Merino, and G. Pérez. A two-stage stochastic integer programming approach as a mixture of Branch-and-Fix Coordination and Benders Decomposition schemes. *Annals of Operations Research*, 152:395–420, 2007. doi:10.1007/s10479-006-0138-0. (cited at p. 40)
- European Wind Energy Association. Wind in power – 2010 European Statistics, February 2011. URL <http://www.ewea.org/statistics/european>. (cited at p. 101)
- I. V. Evstigneev. Measurable selection and dynamic programming. *Mathematics of Operations Research*, 1(3):267–272, 1976. doi:10.1287/moor.1.3.267. (cited at p. 5)
- O. Exler and K. Schittkowski. A trust region SQP algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1(3):269–280, 2007. doi:10.1007/s11590-006-0026-1. (cited at p. 155)
- O. Exler, T. Lehmann, and K. Schittkowski. A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization. *Mathematical Programming Computation*, 4(4):383–412, 2012. doi:10.1007/s12532-012-0045-0. (cited at p. 155)
- J. E. Falk and K. R. Hoffman. A successive underestimation method for concave minimization problems. *Mathematics of Operations Research*, 1(3):251–259, 1976. doi:10.1287/moor.1.3.251. (cited at p. 126)

- T. Farkas, B. Czuczai, E. Rev, and Z. Lelkes. New MINLP model and modified outer approximation algorithm for distillation column synthesis. *Industrial & Engineering Chemistry Research*, 47(9):3088–3103, 2008. doi:10.1021/ie0711426. (cited at p. 147)
- M. C. Ferris, S. P. Dirkse, J.-H. Jagla, and A. Meeraus. An extended mathematical programming framework. *Computers & Chemical Engineering*, 33(12):1973–1982, 2009. doi:10.1016/j.compchemeng.2009.06.013. (cited at p. 156)
- P. Festa, editor. *Proceedings of 9th International Symposium on Experimental Algorithms, SEA 2010*, volume 6049 of *Lecture Notes in Computer Science*, 2010. Springer. doi:10.1007/978-3-642-13193-6. (cited at pp. 284 and 288)
- FICO. *Xpress-SLP Program Reference Manual*, 1.41 edition, 2008. URL <http://www.fico.com/xpress>. (cited at pp. 117 and 152)
- FICO. *Xpress-Optimizer Reference manual*, 20.0 edition, 2009a. URL <http://www.fico.com/xpress>. (cited at p. 152)
- FICO. Xpress-Mosel 7.0, 2009b. URL <http://www.fico.com/xpress>. (cited at pp. 147 and 152)
- M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1-3):23–47, 2003. doi:10.1007/s10107-003-0395-5. (cited at p. 204)
- M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005. doi:10.1007/s10107-004-0570-3. (cited at p. 145)
- R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66(1-3):327–349, 1994. doi:10.1007/BF01581153. (cited at p. 119)
- R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002. doi:10.1007/s101070100244. (cited at pp. 151, 152, 154, and 155)
- C. A. Floudas. *Nonlinear and Mixed Integer Optimization: Fundamentals and Applications*. Oxford University Press, New York, 1995. ISBN 0195100565. (cited at p. 115)
- C. A. Floudas. *Deterministic Global Optimization: Theory, Algorithms and Applications*, volume 37 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers, 2000. ISBN 978-0-7923-6014-8. (cited at p. 116)
- C. A. Floudas, I. G. Akrotirianakis, C. Caratzoulas, C. A. Meyer, and J. Kallrath. Global optimization in the 21st century: Advances and challenges. *Computers & Chemical Engineering*, 29(6):1185–1202, 2005. doi:10.1016/j.compchemeng.2005.02.006. (cited at p. 116)

## Bibliography

- H. Föllmer and A. Schied. *Stochastic Finance – An Introduction in Discrete Time*, volume 27 of *De Gruyter Studies in Mathematics*. Walter de Gruyter, Berlin, 2nd edition, 2004. ISBN 3110183463. (cited at p. 3)
- L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multicommodity network flows. *Management Science*, 5:97–101, 1958. doi:10.1287/mnsc.1040.0269. (cited at p. 53)
- J. J. H. Forrest and J. A. Tomlin. Branch and bound, integer, and non-integer programming. *Annals of Operations Research*, 149(1):81–87, 2007. doi:10.1007/s10479-006-0112-x. (cited at pp. 116 and 146)
- R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Brooks/Cole Publishing Company, 1993. ISBN 0534388094. (cited at p. 147)
- R. Fourer, C. Maheshwari, A. Neumaier, D. Orban, and H. Schichl. Convexity and concavity detection in computational graphs: Tree walks for convexity assessment. *INFORMS Journal on Computing*, 22(1):26–43, 2009. doi:10.1287/ijoc.1090.0321. (cited at p. 178)
- K. Frauendorfer. Barycentric scenario trees in convex multistage stochastic programming. *Mathematical Programming*, 75(2):277–293, 1996. doi:10.1007/BF02592156. (cited at p. 8)
- C. Fricke. *Applications of Integer Programming in Open Pit Mining*. PhD thesis, University of Melbourne, August 2006. URL <http://repository.unimelb.edu.au/10187/14412>. (cited at p. 206)
- A. Fügenschuh, H. Homfeld, H. Schülldorf, and S. Vigerske. Mixed-integer nonlinear problems in transportation applications. In H. Rodrigues, editor, *Proceedings of the 2nd International Conference on Engineering Optimization*, 2010. (cited at p. 166)
- G. Gamrath. Generic branch-cut-and-price. Master’s thesis, Technische Universität Berlin, 2010. (cited at p. 164)
- G. Gamrath and M. Lübbecke. Experiments with a generic dantzig-wolfe decomposition for integer programs. In Festa [2010], pages 239–252. doi:10.1007/978-3-642-13193-6\_21. (cited at p. 164)
- GAMS Development Corp. *GAMS – The Solver Manuals*, 2012. URL <http://www.gams.com>. (cited at pp. 124, 151, 154, and 155)
- M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979. ISBN 0716710455. (cited at p. 50)
- H. I. Gassmann. MSLiP: a computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47(1-3):407–423, 1990. doi:10.1007/BF01580872. (cited at pp. 28 and 71)

- H. I. Gassmann and S. W. Wallace. Solving linear programs with multiple right-hand sides: Pricing and ordering schemes. *Annals of Operations Research*, 64(1):237–259, 1996. doi:10.1007/BF02187648. (cited at pp. 28 and 71)
- C. Gau and L. Schrage. Implementation and testing of a branch-and-bound based method for deterministic global optimization: Operations research applications. In C. A. Floudas and P. M. Pardalos, editors, *Frontiers in Global Optimization*, volume 74 of *Nonconvex Optimization and Its Applications*, pages 145–164. Springer, 2003. ISBN 978-1-4020-7699-2. (cited at p. 153)
- D. M. Gay. Bounds from slopes. Technical report, Sandia National Laboratories, 2010. URL <https://cfwebprod.sandia.gov/cfdocs/CCIM/docs/bounds10.pdf>. (cited at p. 191)
- B. Geißler, A. Martin, A. Morsi, and L. Schewe. Using piecewise linear functions for solving MINLPs. In Lee and Leyffer [2012], pages 287–314. doi:10.1007/978-1-4614-1927-3\_10. (cited at p. 117)
- A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972. doi:10.1007/BF00934810. (cited at pp. 120 and 121)
- V. Ghildyal and N. V. Sahinidis. Solving global optimization problems with BARON. In A. Migdalas, P. M. Pardalos, and P. Värbrand, editors, *From Local to Global Optimization*, volume 53 of *Nonconvex Optimization and Its Applications*, chapter 10, pages 205–230. Springer, 2001. ISBN 978-0-7923-6883-0. (cited at p. 150)
- S. Ghosh. DINS, a MIP improvement heuristic. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization - 12th International Conference*, volume 4513 of *Lecture Notes in Computer Science*, pages 310–323, 2007. doi:10.1007/978-3-540-72792-7\_24. (cited at p. 204)
- C. R. Glassey. Nested decomposition and multi-stage linear programs. *Management Science*, 20(3):282–292, 1973. doi:10.1287/mnsc.20.3.282. (cited at pp. 53 and 57)
- A. M. Gleixner. Solving large-scale open pit mining production scheduling problems by integer programming. Master’s thesis, Technische Universität Berlin, 2008. urn:nbn:de:0297-zib-11389. (cited at p. 207)
- A. M. Gleixner and S. Weltge. Learning and propagating Lagrangian variable bounds for mixed-integer nonlinear programming. ZIB-Report 13-04, Zuse Institute Berlin, 2013. urn:nbn:de:0297-zib-17631. (cited at pp. 141 and 230)
- A. M. Gleixner, H. Held, W. Huang, and S. Vigerske. Towards globally optimal operation of water supply networks. *Numerical Algebra, Control and Optimization*, 2(4):695–711, 2012. doi:10.3934/naco.2012.2.695. (cited at pp. v and 166)

## Bibliography

- F. Glineur. Computational experiments with a linear approximation of second order cone optimization. Technical Report 0001, Faculté Polytechnique de Mons, Belgium, 2000. (cited at p. 195)
- C. Gounaris and C. A. Floudas. Convexity of products of univariate functions and convexification transformations for geometric programming. *Journal of Optimization Theory and Applications*, 138(3):407–427, 2008a. doi:10.1007/s10957-008-9402-6. (cited at p. 179)
- C. Gounaris and C. A. Floudas. Tight convex underestimators for  $C^2$ -continuous problems: I. univariate functions. *Journal of Global Optimization*, 42(1):51–67, 2008b. doi:10.1007/s10898-008-9287-9. (cited at p. 132)
- C. Gounaris and C. A. Floudas. Tight convex underestimators for  $C^2$ -continuous problems: II. multivariate functions. *Journal of Global Optimization*, 42(1):69–89, 2008c. doi:10.1007/s10898-008-9288-8. (cited at p. 132)
- C. Gounaris and C. A. Floudas. A review of recent advances in global optimization. *Journal of Global Optimization*, 45:3–38, 2009. doi:10.1007/s10898-008-9332-8. (cited at p. 116)
- S. Graf and H. Luschgy. *Foundations of Quantization for Probability Distributions*, volume 1730 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 2000. doi:10.1007/BFb0103945. (cited at p. 7)
- A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 105 in Other Titles in Applied Mathematics. SIAM, Philadelphia, PA, 2nd edition, 2008. doi:10.1137/1.9780898717761. (cited at pp. 169 and 191)
- I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, 2002. doi:10.1023/A:1021039126272. (cited at p. 118)
- I. E. Grossmann and Z. Kravanja. Mixed-integer nonlinear programming: A survey of algorithms and applications. In A. R. Conn, L. T. Biegler, T. F. Coleman, and F. N. Santosa, editors, *Large-Scale Optimization with Applications, Part II: Optimal Design and Control*, volume 93 of *The IMA Volumes in Mathematics and its Applications*, pages 73–100. Springer, New York, 1997. doi:10.1007/978-1-4612-1960-6\_5. (cited at p. 115)
- I. E. Grossmann and J. Lee. Cyberinfrastructure for mixed-integer nonlinear programming. *SIAG/OPT Views-and-News*, 22(1):8–12, 2011. URL <http://www.minlp.org>. (cited at p. 156)
- I. E. Grossmann and S. Lee. Generalized convex disjunctive programming: Nonlinear convex hull relaxation. *Computational Optimization and Applications*, 26:83–100, 2003. doi:10.1023/A:1025154322278. (cited at pp. 156, 181, and 230)

- N. Gröwe-Kuska, K. C. Kiwiel, M. P. Nowak, W. Römisch, and I. Wegner. Power management in a hydro-thermal system under uncertainty by Lagrangian relaxation. In C. Greengard and A. Ruszczynski, editors, *Decision Making Under Uncertainty – Energy and Power*, volume 128 of *The IMA Volumes in Mathematics and its Applications*, pages 39–70. Springer-Verlag, New York, 2002. doi:10.1007/978-1-4684-9256-9\_3. (cited at p. 22)
- O. Günlük and J. T. Linderoth. Perspective reformulation and applications. In Lee and Leyffer [2012], pages 61–89. doi:10.1007/978-1-4614-1927-3\_3. (cited at p. 230)
- O. K. Gupta and V. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, 1985. doi:10.1287/mnsc.31.12.1533. (cited at p. 119)
- P. L. Hammer and A. A. Rubin. Some remarks on quadratic programming with 0-1 variables. *RAIRO - Operations Research - Recherche Opérationnelle*, 4(V3):67–79, 1970. (cited at p. 234)
- E. Handschin, F. Neise, H. Neumann, and R. Schultz. Optimal operation of dispersed generation under uncertainty using mathematical programming. *International Journal of Electrical Power & Energy Systems*, 28(9):618–626, 2006. doi:10.1016/j.ijepes.2006.03.003. (cited at p. 102)
- E. R. Hansen. *Global Optimization using Interval Analysis*. Marcel Dekker, Inc., 1992. (cited at p. 172)
- I. Harjunkoski, T. Westerlund, R. Pörn, and H. Skrifvars. Different transformations for solving non-convex trim-loss problems by MINLP. *European Journal of Operational Research*, 105(3):594–603, 1998. doi:10.1016/S0377-2217(97)00066-0. (cited at pp. 228 and 230)
- D. Haugland and S. W. Wallace. Solving many linear programs that differ only in the righthand side. *European Journal of Operational Research*, 37(3):318–324, 1988. doi:10.1016/0377-2217(88)90193-2. (cited at pp. 28 and 71)
- A. C. Hearn. REDUCE: a user-oriented interactive system for algebraic simplification. In M. Klerer and J. Reinfelds, editors, *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc.*, pages 79–90, New York, 1967. Academic Press. doi:10.1145/2402536.2402544. (cited at p. 178)
- A. C. Hearn. REDUCE: the first forty years. In A. Dolzmann, A. Seidl, and T. Sturm, editors, *Algorithmic Algebra and Logic*, Proceedings of the A3L, pages 19–24, Norderstedt, 2005. Books on Demand GmbH. ISBN 3833426691. (cited at p. 178)
- S. Heinz and J. C. Beck. Solving resource allocation/scheduling problems with constraint integer programming. In M. A. Salido, R. Bartak, and N. Policella, editors, *Proceedings*

## Bibliography

- of the *Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems, COPLAS 2011*, pages 23–30, 2011. urn:nbn:de:0297-zib-12691. (cited at p. 164)
- S. Heinz and J. Schulz. Explanations for the cumulative constraint: An experimental study. In P. M. Pardalos and S. Rebennack, editors, *Experimental Algorithms – 10th International Symposium, SEA 2011*, volume 6630 of *Lecture Notes in Computer Science*, pages 400–409. Springer, 2011. doi:10.1007/978-3-642-20662-7\_34. (cited at p. 164)
- T. Heinze. *Ein Verfahren zur Dekomposition mehrstufiger stochastischer Optimierungsprobleme mit Ganzzahligkeit und Risikoaversion*. PhD thesis, Universität Duisburg-Essen, Campus Duisburg, 2008. urn:nbn:de:hbz:464-20080723-112816-3. (cited at p. 41)
- T. Heinze and R. Schultz. A branch-and-bound method for multistage stochastic integer programs with risk objectives. *Optimization*, 57(2):277–293, 2008. doi:10.1080/02331930701779906. (cited at p. 41)
- H. Heitsch. *Stabilität und Approximation stochastischer Optimierungsprobleme*. PhD thesis, Humboldt-Universität zu Berlin, 2007. (cited at p. 7)
- H. Heitsch and W. Römis. A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters*, 35(6):731–738, 2007. doi:10.1016/j.orl.2006.12.008. (cited at pp. 7 and 9)
- H. Heitsch and W. Römis. Scenario tree reduction for multistage stochastic programs. *Computational Management Science*, 6(2):117–133, 2009a. doi:10.1007/s10287-008-0087-y. (cited at p. 8)
- H. Heitsch and W. Römis. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming*, 118(2):371–406, 2009b. doi:10.1007/s10107-007-0197-2. (cited at pp. 8, 65, 67, 72, and 94)
- H. Heitsch, W. Römis, and C. Strugarek. Stability of multistage stochastic programs. *SIAM Journal on Optimization*, 17(2):511–525, 2006. doi:10.1137/050632865. (cited at pp. 7 and 90)
- R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel. Nonlinear integer programming. In M. Jünger, T. M. Lieblich, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, volume 3: Current Topics, chapter 15, pages 561–618. Springer, 2010. doi:10.1007/978-3-540-68279-0\_15. (cited at p. 117)
- R. Henrion, C. Küchler, and W. Römis. Discrepancy distances and scenario reduction in two-stage stochastic integer programming. *Journal of Industrial and Management Optimization*, 4(2):363–384, 2008. doi:10.3934/jimo.2008.4.363. (cited at p. 9)

- R. Henrion, C. Küchler, and W. Römisch. Scenario reduction in stochastic programming with respect to discrepancy distances. *Computational Optimization and Applications*, 43(1):67–93, 2009. doi:10.1007/s10589-007-9123-z. (cited at p. 19)
- J. L. Higle and S. Sen. Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research*, 16(3):650–669, 1991. doi:10.1287/moor.16.3.650. (cited at p. 25)
- J. L. Higle, B. Rayco, and S. Sen. Stochastic scenario decomposition for multistage stochastic programs. *IMA Journal of Management Mathematics*, 21(1):39–66, 2010. doi:10.1093/imaman/dpp001. (cited at p. 39)
- P. Hilli and T. Pennanen. Numerical study of discretizations of multistage stochastic programs. *Kybernetika*, 44(2):185–204, 2008. (cited at p. 89)
- R. Hochreiter and G. C. Pflug. Financial scenario generation for stochastic multi-stage decision processes as facility location problem. *Annals of Operations Research*, 152(1):257–272, 2007. doi:10.1007/s10479-006-0140-6. (cited at p. 8)
- K. Holmström. The TOMLAB optimization environment in MATLAB. *Advanced Modeling and Optimization*, 1(1):47–69, 1999. URL <http://tomopt.com/tomlab>. (cited at pp. 154 and 155)
- J. N. Hooker. *Integrated Methods for Optimization*, volume 170 of *International Series in Operations Research & Management Science*. Springer, New York, 2007. doi:10.1007/978-1-4614-1900-6. (cited at p. 159)
- R. Horst and P. Pardalos. *Handbook of Global Optimization*, volume 2 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers, 1995. ISBN 978-0-7923-3120-9. (cited at pp. 116 and 307)
- R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer, Berlin, 1990. ISBN 978-3-540-61038-0. (cited at pp. 116, 133, and 134)
- K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001. doi:10.1287/mnsc.47.2.295.9834. (cited at p. 8)
- W. Huang. Operative planning of water supply networks by mixed integer nonlinear programming. Master’s thesis, Freie Universität Berlin, 2011. (cited at pp. v, 141, and 166)
- IBM. CPLEX, 2012. URL <http://www.cplex.com>. (cited at pp. 117 and 151)
- G. Infanger and D. P. Morton. Cut sharing for multistage stochastic linear programs with interstage dependency. *Mathematical Programming*, 75(2):241–256, 1996. doi:10.1007/BF02592154. (cited at pp. 29 and 43)

## Bibliography

- International Energy Agency. *World Energy Outlook 2006*. Stedi, Paris, 2006. ISBN 92-64-10989-7. URL <http://www.worldenergyoutlook.org>. (cited at p. 110)
- M. Jach, D. Michaels, and R. Weismantel. The convex envelope of  $(n-1)$ -convex functions. *SIAM Journal on Optimization*, 19(3):1451–1466, 2008. doi:10.1137/07069359X. (cited at pp. 127, 131, and 144)
- L. Jaulin, M. Kieffer, O. Didrit, and É. Walter. *Applied Interval Analysis*. Springer, 2001. doi:10.1007/978-1-4471-0249-6. (cited at p. 172)
- M. Jüdes. MINLP-Optimierung des Entwurfs und des stationären Betriebs von Kraftwerken mit mehreren Arbeitspunkten. In *Fortschritt-Berichte*, number 579 in Series 6. VDI-Verlag, Düsseldorf, Germany, 2009. ISSN 0178-9414. (cited at p. iv)
- M. Jüdes, G. Tsatsaronis, and S. Vigerske. Entwurfsoptimierung von Energieumwandlungsanlagen mit mehreren Betriebspunkten. In *Optimierung in der Energiewirtschaft*, number 2018 in VDI-Berichte, pages 199–210. VDI-Verlag, Düsseldorf, 2007. ISSN 0178-9414. (cited at p. iv)
- M. Jüdes, G. Tsatsaronis, and S. Vigerske. Optimization of the design and partial-load operation of power plants using mixed-integer nonlinear programming. In J. Kallrath, P. Pardalos, S. Rebennack, and M. Scheidt, editors, *Optimization in the Energy Industry*, chapter 9, pages 193–220. Springer, 2009. doi:10.1007/978-3-540-88965-6\_9. (cited at p. iv)
- P. Kall and S. W. Wallace. *Stochastic Programming*. Wiley, Chichester, 1994. ISBN 0471951080. (cited at p. 3)
- M. Kaut and S. W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271, 2007. (cited at p. 89)
- R. B. Kearfott. Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, 2(1):95–112, 1996. (cited at p. 170)
- J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960. doi:10.1137/0108053. (cited at p. 121)
- L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244(5):1093–1096, 1979. english translation in *Soviet Math. Dokl.* 20(1):191–194, 1979. (cited at p. 116)
- M. Kiliç, J. Linderoth, and J. Luedtke. Effective separation of disjunctive cuts for convex mixed integer nonlinear programs. Technical Report 1681, University of Wisconsin-Madison, Computer Sciences Department, 2010. URL <http://digital.library.wisc.edu/1793/60720>. (cited at pp. 142 and 152)

- M. Kiliç, J. Linderoth, J. Luedtke, and A. Miller. Strong branching inequalities for mixed integer nonlinear programs. Technical Report 1696, University of Wisconsin-Madison, Computer Sciences Department, 2011. URL <http://digital.library.wisc.edu/1793/60748>. (cited at pp. 142 and 152)
- K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46(1-3):105–122, 1990. doi:10.1007/BF01585731. (cited at p. 40)
- W. K. Klein Haneveld, L. Stougie, and M. H. van der Vlerk. Simple integer recourse models: convexity and convex approximations. *Mathematical Programming*, 108(2-3):435–473, 2006. doi:10.1007/s10107-006-0718-4. (cited at pp. 30, 31, and 32)
- T. Koch. *Rapid Mathematical Programming*. PhD thesis, Technische Universität Berlin, 2004. urn:nbn:de:0297-zib-8346. (cited at p. 166)
- T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010 – mixed integer programming library version 5. *Mathematical Programming Computation*, 3(2):103–163, 2011. doi:10.1007/s12532-011-0025-9. (cited at pp. 116 and 225)
- G. R. Kocis and I. E. Grossmann. Relaxation strategy for the structural optimization of nonconvex MINLP problems in process flow sheets. *Industrial & Engineering Chemistry Research*, 26(9):1869–1880, 1987. doi:10.1021/ie00069a026. (cited at p. 124)
- G. R. Kocis and I. E. Grossmann. Computational experience with DICOPT solving MINLP problems in process systems engineering. *Computers & Chemical Engineering*, 13(3):307–315, 1989. doi:10.1016/0098-1354(89)85008-2. (cited at p. 151)
- R. Kouwenberg. Scenario generation and stochastic programming models for asset liability management. *European Journal of Operational Research*, 134(2):279–292, 2001. doi:10.1016/S0377-2217(00)00261-7. (cited at p. 8)
- C. Küchler. *Stability, Approximation, and Decomposition in Two- and Multistage Stochastic Programming*. PhD thesis, Humboldt-Universität zu Berlin, 2009. (cited at pp. 44, 53, 63, 64, 67, 69, 70, 72, and 90)
- C. Küchler and S. Vigerske. Decomposition of multistage stochastic programs with recombining scenario trees. Stochastic Programming E-Print Series 2007-09, 2007. urn:nbn:de:kobv:11-10078946. (cited at pp. iv, 44, and 72)
- C. Küchler and S. Vigerske. Ein Dekompositionsverfahren für stochastische Optimierungsprobleme mit rekombinierenden Szenariobäumen. In Schultz and Wagner [2009], chapter 10, pages 201–226. ISBN 978-3-8258-1359-8. (cited at pp. iv and 44)
- C. Küchler and S. Vigerske. Numerical evaluation of approximation methods in stochastic programming. *Optimization*, 59(3):401–415, 2010. doi:10.1080/02331931003700756. (cited at pp. iv, 44, and 94)

## Bibliography

- D. Kuhn. *Generalized bounds for convex multistage stochastic programs*, volume 548 of *Lecture Notes in Economics and Mathematical Systems*. Springer, Berlin, 2005. doi:10.1007/b138260. (cited at p. 8)
- K. Kuipers. bnb, 2003. <http://www.mathworks.com/matlabcentral/fileexchange/95>. (cited at p. 150)
- A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960. (cited at pp. 117 and 119)
- G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993. doi:10.1016/0167-6377(93)90002-X. (cited at p. 30)
- J. Lee and S. Leyffer, editors. *Mixed Integer Nonlinear Programming*, volume 154 of *IMA Volumes in Mathematics and its Applications*. Springer, 2012. doi:10.1007/978-1-4614-1927-3. (cited at pp. 116, 278, 279, 282, 289, 291, and 302)
- T. Lehmann. *On Efficient Solution Methods for Mixed-Integer Nonlinear and Mixed-Integer Quadratic Optimization Problems*. PhD thesis, Universität Bayreuth, in preparation. (cited at p. 155)
- C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer Series in Statistics. Springer, 2009. doi:10.1007/978-0-387-78165-5. (cited at p. 7)
- S. Leyffer. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18(3):295–309, 2001. doi:10.1023/A:1011241421041. (cited at pp. 119 and 154)
- D. Li and X. Sun. *Nonlinear Integer Programming*, volume 84 of *International Series in Operations Research & Management Science*. Springer, 2005. doi:10.1007/0-387-32995-1. (cited at p. 117)
- L. Liberti. Reformulations in mathematical programming: automatic symmetry detection and exploitation. *Mathematical Programming*, 131(1-2):273–304, 2012. doi:10.1007/s10107-010-0351-0. (cited at p. 151)
- L. Liberti and C. Pantelides. Convex envelopes of monomials of odd degree. *Journal of Global Optimization*, 25(2):157–168, 2003. doi:10.1023/A:1021924706467. (cited at pp. 128 and 192)
- L. Liberti, S. Cafieri, and F. Tarissan. Reformulations in mathematical programming: A computational approach. In A. Abraham, A.-E. Hassanien, and P. Siarry, editors, *Foundations of Computational Intelligence Volume 3: Global Optimization*, volume 203 of *Studies in Computational Intelligence*, pages 153–234. Springer, New York, 2009. doi:10.1007/978-3-642-01085-9\_7. (cited at pp. 156 and 170)

- L. Liberti, N. Mladenović, and G. Nannicini. A recipe for finding good solutions to MINLPs. *Mathematical Programming Computation*, 3(4):349–390, 2011. doi:10.1007/s12532-011-0031-y. (cited at p. 145)
- Y. Lin and L. Schrage. The global solver in the LINDO API. *Optimization Methods & Software*, 24(4–5):657–668, 2009. doi:10.1080/10556780902753221. (cited at p. 153)
- J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999. doi:10.1287/ijoc.11.2.173. (cited at p. 136)
- J. T. Linderoth and S. J. Wright. Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications*, 24(2-3):207–250, 2003. doi:10.1023/A:1021858008222. (cited at p. 25)
- Lindo Systems, Inc. *What’sBest!* 10.0, 2009. URL <http://www.lindo.com>. (cited at p. 153)
- Lindo Systems, Inc. Lindo API 6.1, 2010. URL <http://www.lindo.com>. (cited at p. 153)
- M. Locatelli. Convex envelopes for quadratic and polynomial functions over polytopes. 2010. URL [http://www.optimization-online.org/DB\\_HTML/2010/11/2788.html](http://www.optimization-online.org/DB_HTML/2010/11/2788.html). (cited at p. 132)
- M. Locatelli and F. Schoen. On the convex envelopes and underestimators for bivariate functions. 2009. URL [http://www.optimization-online.org/DB\\_HTML/2009/11/2462.html](http://www.optimization-online.org/DB_HTML/2009/11/2462.html). (cited at p. 132)
- R. Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57–66, 2003. URL <http://www.coin-or.org>. doi:10.1147/rd.471.0057. (cited at pp. iii and 151)
- F. V. Louveaux and R. Schultz. *Stochastic Integer Programming*, chapter 4, pages 213–266. Volume 10 of Ruszczyński and Shapiro [2003], 2003. doi:10.1016/S0927-0507(03)10004-7. (cited at p. 21)
- M. E. Lübbecke. Column generation. In Cochran et al. [2010]. doi:10.1002/9780470400531.eorms0158. (cited at pp. 53 and 71)
- J. Luedtke, M. Namazifar, and J. Linderoth. Some results on the strength of relaxations of multilinear functions. *Mathematical Programming*, 136(2):325–351, 2012. doi:10.1007/s10107-012-0606-z. (cited at p. 144)
- G. Lulli and S. Sen. A branch-and-price algorithm for multistage stochastic integer programming with application to batch-sizing problems. *Management Science*, 50(6):786–796, 2004. doi:10.1287/mnsc.1030.0164. (cited at p. 40)

## Bibliography

- A. Mahajan and T. Munson. Exploiting second-order cone structure for global optimization. Technical Report ANL/MCS-P1801-1010, Argonne National Laboratory, 2010. URL [http://www.optimization-online.org/DB\\_HTML/2010/10/2780.html](http://www.optimization-online.org/DB_HTML/2010/10/2780.html). (cited at pp. 154 and 195)
- A. Mahajan, S. Leyffer, and C. Kirches. Solving mixed-integer nonlinear programs by QP-diving. Preprint ANL/MCS-P2071-0312, Argonne National Laboratory, 2012. URL [http://www.optimization-online.org/DB\\_HTML/2012/03/3409.html](http://www.optimization-online.org/DB_HTML/2012/03/3409.html). (cited at p. 154)
- C. D. Maranas and C. A. Floudas. A global optimization approach for Lennard-Jones microclusters. *The Journal of Chemical Physics*, 97(10):7667–7678, 1992. doi:10.1063/1.463486. (cited at p. 132)
- C. D. Maranas and C. A. Floudas. Finding all solutions of nonlinearly constrained systems of equations. *Journal of Global Optimization*, 7(2):143–182, 1995. doi:10.1007/BF01097059. (cited at pp. 127 and 179)
- C. D. Maranas and C. A. Floudas. Global optimization in generalized geometric programming. *Computers & Chemical Engineering*, 21(4):351–369, 1997. doi:10.1016/S0098-1354(96)00282-7. (cited at p. 141)
- H. Marchand, A. Martin, R. Weismantel, and L. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397–446, 2002. doi:10.1016/S0166-218X(01)00348-1. (cited at p. 142)
- A. Märkert and R. Gollmer. *User’s Guide to ddsip – A C Package for the Dual Decomposition of Two-Stage Stochastic Programs with Mixed-Integer Recourse*. University Duisburg-Essen, Campus Duisburg, 2008. URL <http://www.neos-server.org/neos/solvers/slp:ddsip/LP.html>. (cited at p. 40)
- A. Martin. *Integer programs with block structure*. Habilitation treatise, Zuse Institute Berlin, 1999. urn:nbn:de:0297-zib-3911. (cited at p. 162)
- MathWorks. *MATLAB User’s Guide*, 2009. URL <http://www.mathworks.com>. (cited at p. 150)
- G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976. doi:10.1007/BF01580665. (cited at pp. 126, 127, 128, and 129)
- C. A. Meyer and C. A. Floudas. Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes. *Journal of Global Optimization*, 29(2):125–155, 2004. doi:10.1023/B:JOGO.0000042112.72379.e6. (cited at p. 127)
- C. A. Meyer and C. A. Floudas. Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: Spline  $\alpha$ BB underestimators. *Journal of Global Optimization*, 32(2):221–258, 2005. doi:10.1007/s10898-004-2704-9. (cited at p. 132)

- R. Misener and C. A. Floudas. Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. *Mathematical Programming*, 136(1):155–182, 2012a. doi:10.1007/s10107-012-0555-6. (cited at p. 153)
- R. Misener and C. A. Floudas. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, to appear, 2012b. doi:10.1007/s10898-012-9874-7. (cited at p. 153)
- A. Mitsos, B. Chachuat, and P. I. Barton. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009. doi:10.1137/080717341. (cited at p. 156)
- M. Mönnigmann. Efficient calculation of bounds on spectra of Hessian matrices. *SIAM Journal on Scientific Computing*, 30(5):2340–2357, 2008. doi:10.1137/070704186. (cited at p. 178)
- R. E. Moore. *Interval Analysis*. Englewood Cliffs, NJ: Prentice Hall, 1966. (cited at p. 170)
- R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. Other Titles in Applied Mathematics. SIAM, 2009. doi:10.1137/1.9780898717716. (cited at pp. 170 and 172)
- J. J. Moré. Recent developments in algorithms and software for trust region methods. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming – The State of the Art*, pages 258–287. Springer, Berlin Heidelberg, 1983. doi:10.1007/978-3-642-68874-4\_11. (cited at p. 25)
- D. P. Morton. An enhanced decomposition algorithm for multistage stochastic hydroelectric scheduling. *Annals of Operations Research*, 64(1):211–235, 1996. doi:10.1007/BF02187647. (cited at p. 29)
- MOSEK Corporation. *The MOSEK optimization tools manual*, 6.0 edition, 2009. URL <http://www.mosek.com>. (cited at p. 155)
- B. A. Murtagh and M. A. Saunders. *MINOS 5.51 User’s Guide*. Department of Management Science and Engineering, Stanford University, 2003. Report SOL 83-20R. (cited at p. 205)
- G. Nannicini and P. Belotti. Rounding-based heuristics for nonconvex MINLPs. *Mathematical Programming Computation*, 4(1):1–31, 2012. doi:10.1007/s12532-011-0032-x. (cited at pp. 146 and 151)
- G. Nannicini, P. Belotti, and L. Liberti. A local branching heuristic for MINLPs. arXiv:0812.2188, 2009. (cited at pp. 145 and 151)
- G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, 1988. ISBN 0471359432. (cited at pp. 40 and 141)

## Bibliography

- G. L. Nemhauser, M. W. P. Savelsbergh, and G. S. Sigismondi. MINTO, a Mixed INTEger Optimizer. *Operations Research Letters*, 15(1):47–58, 1994. doi:10.1016/0167-6377(94)90013-2. (cited at p. 152)
- I. P. Nenov, D. H. Fylstra, and L. V. Kolev. Convexity determination in the Microsoft Excel solver using automatic differentiation techniques. Extended abstract, Frontline Systems Inc., 2004. URL <http://www.autodiff.org/ad04/abstracts/Nenov.pdf>. (cited at p. 178)
- A. Neumaier. *Interval Methods for Systems of Equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1990. ISBN 052133196X. (cited at p. 170)
- A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. In *Acta Numerica*, volume 13, chapter 4, pages 271–369. Cambridge University Press, 2004. doi:10.1017/S0962492904000194. (cited at p. 116)
- W. Neun, T. Sturm, and S. Vigerske. Supporting global numerical optimization of rational functions by generic symbolic convexity tests. In V. P. Gerdt, W. Koepf, E. W. Mayr, and E. H. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, volume 6244 of *Lecture Notes in Computer Science*, pages 205–219. Springer, 2010. ISSN 978-3-642-15273-3. doi:10.1007/978-3-642-15274-0\_19. (cited at pp. v, 178, and 181)
- J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2nd edition, 2006. ISSN 978-0-387-30303-1. (cited at p. 116)
- P. Nørgård, G. Giebel, H. Holttinen, and A. Petterteig. Fluctuations and predictability of wind and hydropower. Technical report, WILMAR, Risø National Laboratory, 2004. URL <http://www.wilmar.risoe.dk/Results.htm>. (cited at p. 73)
- I. Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*, volume 152 of *International Series of Numerical Mathematics*. Birkhäuser Verlag, Basel, 2005. doi:10.1007/3-7643-7374-1. (cited at pp. iii, 117, and 191)
- I. Nowak and S. Vigerske. Adaptive discretization of convex multistage stochastic programs. *Mathematical Methods of Operations Research*, 65(2):361–383, 2007. doi:10.1007/s00186-006-0124-y. (cited at p. iii)
- I. Nowak and S. Vigerske. LaGO: a (heuristic) branch and cut algorithm for nonconvex MINLPs. *Central European Journal of Operations Research*, 16(2):127–138, 2008. doi:10.1007/s10100-007-0051-x. (cited at pp. iii, 122, 141, and 153)
- I. Nowak, H. Alperin, and S. Vigerske. LAGO - an object oriented library for solving MINLPs. In C. Bliet, C. Jermann, and A. Neumaier, editors, *Global Optimization and Constraint Satisfaction*, volume 2861 of *Lecture Notes in Computer Science*, pages 32–42. Springer, 2003. doi:10.1007/978-3-540-39901-8\_3. (cited at p. iii)

- M. P. Nowak and W. Römisch. Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. *Annals of Operations Research*, 100(1-4):251–272, 2000. doi:10.1023/A:1019248506301. (cited at pp. 21, 22, and 102)
- L. Ntaimo and S. Sen. The million-variable “march” for stochastic combinatorial optimization. *Journal of Global Optimization*, 32(3):385–400, 2005. doi:10.1007/s10898-004-5910-6. (cited at p. 33)
- L. Ntaimo and S. Sen. A branch-and-cut algorithm for two-stage stochastic mixed-binary programs with continuous first-stage variables. *International Journal of Computational Science and Engineering*, 3(3):232–241, 2008a. doi:10.1504/IJCSE.2007.017829. (cited at pp. 33 and 37)
- L. Ntaimo and S. Sen. A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, 40(3):299–319, 2008b. doi:10.1007/s10589-007-9085-1. (cited at p. 33)
- M. G. Osanloo, J. Gholamnejad, and B. Karimi. Long-term open pit mine production planning: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, 22(1):3–35, 2008. doi:10.1080/17480930601118947. (cited at p. 206)
- T. Pennanen. Epi-convergent discretizations of multistage stochastic programs via integration quadratures. *Mathematical Programming*, 116(1-2):461–479, 2009. doi:10.1007/s10107-007-0113-9. (cited at p. 8)
- M. V. F. Pereira and L. M. V. G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, 1991. doi:10.1007/BF01582895. (cited at p. 29)
- L. Perron and M. A. Trick, editors. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008*, volume 5015 of *Lecture Notes in Computer Science*, 2008. Springer. doi:10.1007/978-3-540-68155-7. (cited at p. 275)
- M. E. Pfetsch, A. Fügenschuh, B. Geißler, N. Geißler, R. Gollmer, B. Hiller, J. Humpola, T. Koch, T. Lehmann, A. Martin, A. Morsi, J. Rövekamp, L. Schewe, M. Schmidt, R. Schultz, R. Schwarz, J. Schweiger, C. Stangl, M. C. Steinbach, S. Vigerske, and B. M. Willert. Validation of nominations in gas network optimization: Models, methods, and solutions. ZIB-Report 12-41, Zuse Institute Berlin, 2012. urn:nbn:de:0297-zib-16531. (cited at pp. v, 166, and 205)
- G. C. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89(2):251–271, 2001. doi:10.1007/PL00011398. (cited at p. 8)

## Bibliography

- G. C. Pflug and A. Pichler. A distance for multi-stage stochastic optimization models. *SIAM Journal on Optimization*, 22(1):1–23, 2012. doi:10.1137/110825054. (cited at p. 7)
- G. C. Pflug and W. Römisch. *Modeling, Measuring and Managing Risk*. World Scientific, Singapore, 2007. ISBN 9812707409. (cited at p. 3)
- J. D. Pintér, editor. *Global Optimization: Scientific and Engineering Case Studies*, volume 85 of *Nonconvex Optimization and Its Applications*. Springer, 2006. doi:10.1007/0-387-30927-6. (cited at p. 115)
- A. Prékopa. *Stochastic Programming*, volume 324 of *Mathematics and Its Applications*. Kluwer, Dordrecht, 1995. ISBN 978-0-7923-3482-8. (cited at p. 3)
- A. Qualizza, P. Belotti, and F. Margot. Linear programming relaxations of quadratically constrained quadratic programs. In Lee and Leyffer [2012], pages 407–426. doi:10.1007/978-1-4614-1927-3\_14. (cited at p. 143)
- 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. doi:10.1016/0098-1354(92)80028-8. (cited at p. 122)
- I. Quesada and I. E. Grossmann. Global optimization algorithm for heat exchanger networks. *Industrial & Engineering Chemistry Research*, 32(3):487–499, 1993. doi:10.1021/ie00015a012. (cited at p. 141)
- I. Quesada and I. E. Grossmann. A global optimization algorithm for linear fractional and bilinear programs. *Journal of Global Optimization*, 6:39–76, 1995. doi:10.1007/BF01106605. (cited at p. 141)
- S. T. Rachev. *Probability Metrics and the Stability of Stochastic Models*. Wiley, Chichester, 1991. ISBN 0471928771. (cited at pp. 7 and 19)
- S. T. Rachev and W. Römisch. Quantitative stability in stochastic programming: The method of probability metrics. *Mathematics of Operations Research*, 27(4):792–818, 2002. doi:10.1287/moor.27.4.792.304. (cited at p. 9)
- S. Ramazan. The new fundamental tree algorithm for production scheduling of open pit mines. *European Journal of Operational Research*, 177(2):1153–1166, 2007. doi:10.1016/j.ejor.2005.12.035. (cited at p. 206)
- A. D. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10(4):425–437, 1997. doi:10.1023/A:1008217604285. (cited at p. 144)
- R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis*, volume 317 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1998. doi:10.1007/978-3-642-02431-3. (cited at pp. 17 and 126)

- M. Roelofs and J. Bisschop. *AIMMS 3.9 – The Language Reference*. Paragon Decision Technology B.V., Haarlem, The Netherlands, 2009. (cited at pp. 147 and 150)
- W. Römisch. Stability of stochastic programming problems. In Ruszczyński and Shapiro [2003], chapter 8, pages 483–554. doi:10.1016/S0927-0507(03)10008-4. (cited at pp. 7, 14, 16, 18, and 89)
- W. Römisch. Scenario generation. In Cochran et al. [2010]. doi:10.1002/9780470400531.eorms0745. (cited at p. 7)
- W. Römisch and S. Vigerske. Quantitative stability of fully random mixed-integer two-stage stochastic programs. *Optimization Letters*, 2(3):377–388, 2008. doi:10.1007/s11590-007-0066-1. (cited at pp. iv and 9)
- W. Römisch and S. Vigerske. Recent progress in two-stage mixed-integer stochastic programming with applications in power production planning. In S. Rebennack, P. M. Pardalos, M. V. F. Pereira, and N. A. Iliadis, editors, *Handbook of Power Systems I*, pages 177–208. Springer, 2010. doi:10.1007/978-3-642-02493-1\_8. (cited at pp. iv and 22)
- E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19(4):534–541, 2007. doi:10.1287/ijoc.1060.0189. (cited at p. 204)
- L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007. doi:10.1016/j.orl.2006.11.004. (cited at p. 71)
- A. Ruszczyński. A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35(3):309–333, 1986. doi:10.1007/BF01580883. (cited at pp. 24 and 71)
- A. Ruszczyński. *Decomposition Methods*, chapter 3, pages 141–211. Volume 10 of Ruszczyński and Shapiro [2003], 2003. doi:10.1016/S0927-0507(03)10003-5. (cited at pp. 21, 23, 25, 28, 50, and 57)
- A. Ruszczyński and A. Shapiro, editors. *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. Elsevier, Amsterdam, 2003. ISBN 978-0-444-50854-6. (cited at pp. 3, 23, 26, 61, 297, 303, and 305)
- H. S. Ryoo and N. V. Sahinidis. Analysis of bounds for multilinear functions. *Journal of Global Optimization*, 19(4):403–424, 2001. doi:10.1023/A:1011295715398. (cited at p. 144)
- M. W. P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *INFORMS Journal on Computing*, 6:445–454, 1994. doi:10.1287/ijoc.6.4.445. (cited at pp. 138 and 140)

## Bibliography

- A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. *Mathematical Programming*, 124(1-2):383–411, 2010. doi:10.1007/s10107-010-0371-9. (cited at p. 143)
- A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Mathematical Programming*, 130(2):359–413, 2011. doi:10.1007/s10107-010-0340-3. (cited at p. 144)
- H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33(4):541–562, 2005. doi:10.1007/s10898-005-0937-x. (cited at pp. 140, 167, 170, 173, and 191)
- M. Schlüter and M. Gerdt. The oracle penalty method. *Journal of Global Optimization*, 47(2):293–325, 2009. doi:10.1007/s10898-009-9477-0. (cited at p. 154)
- M. Schlüter, M. Gerdt, and J.-J. Rückmann. A numerical study of MIDACO on 100 MINLP benchmarks. *Optimization*, 61(7):873–900, 2012. doi:10.1080/02331934.2012.668545. (cited at p. 154)
- L. Schrage. *Optimization Modeling with LINGO*. Lindo Systems, Inc., 2008. URL <http://www.lindo.com>. (cited at pp. 147 and 153)
- A. Schrijver. *Theory of linear and integer programming*. Wiley & Sons, Inc., Chichester, 1986. ISBN 978-0-471-98232-6. (cited at p. 53)
- R. Schultz. Rates of convergence in stochastic programs with complete integer recourse. *SIAM Journal of Optimization*, 6(4):1138–1152, 1996. doi:10.1137/S1052623494271655. (cited at pp. iv, 9, 10, 11, 13, 14, 19, and 20)
- R. Schultz. Stochastic programming with integer variables. *Mathematical Programming*, 97(1-2):285–309, 2003. doi:10.1007/s10107-003-0445-z. (cited at p. 21)
- R. Schultz and H.-J. Wagner, editors. *Innovative Modellierung und Optimierung von Energiesystemen*, volume 26 of *Umwelt- und Ressourcenökonomik*. LIT Verlag, 2009. ISBN 978-3-8258-1359-8. (cited at pp. iii, 286, and 295)
- R. Schultz, L. Stougie, and M. H. van der Vlerk. Solving stochastic programs with integer recourse by enumeration: a framework using Gröbner basis reductions. *Mathematical Programming*, 83(1-3):229–252, 1998. doi:10.1007/BF02680560. (cited at pp. 15 and 30)
- SCICON Ltd. *SCICONIC User Guide Version 1.40*. Scicon Ltd., Milton Keynes, UK, 1989. (cited at p. 146)
- J. K. Scott, M. D. Stuber, and P. I. Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51(4):569–606, 2011. doi:10.1007/s10898-011-9664-7. (cited at p. 130)

- S. Sen. Algorithms for stochastic mixed-integer programming models. In K. Aardal, G. L. Nemhauser, and R. Weismantel, editors, *Handbook of Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 515–558. North-Holland Publishing Co., 2005. doi:10.1016/S0927-0507(05)12009-X. (cited at p. 21)
- S. Sen and J. L. Hige. The  $C^3$  theorem and a  $D^2$  algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20, 2005. doi:10.1007/s10107-004-0566-z. (cited at pp. 30, 33, 34, 35, 36, and 37)
- S. Sen and H. D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006. doi:10.1007/s10107-005-0592-5. (cited at pp. 33 and 37)
- A. Shapiro. Monte carlo sampling methods. In Ruszczyński and Shapiro [2003], chapter 6, pages 353–425. doi:10.1016/S0927-0507(03)10006-0. (cited at pp. 7, 25, and 89)
- A. Shapiro. Inference of statistical bounds for multistage stochastic programming problems. *Mathematical Methods of Operations Research*, 58(1):57–68, 2003b. doi:10.1007/s001860300280. (cited at p. 8)
- A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011. doi:10.1016/j.ejor.2010.08.007. (cited at p. 29)
- 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. doi:10.1023/A:1008241411395. (cited at pp. 134 and 138)
- H. D. Sherali and W. P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, volume 31 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers, 1999. ISBN 978-0-7923-5487-1. (cited at p. 143)
- H. D. Sherali and B. M. P. Fraticelli. Enhancing RLT relaxations via a new class of semidefinite cuts. *Journal of Global Optimization*, 22(1):233–261, 2002. doi:10.1023/A:1013819515732. (cited at p. 143)
- Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch. ParaSCIP: a parallel extension of SCIP. In C. Bischof, H.-G. Hegering, W. E. Nagel, and G. Wittum, editors, *Competence in High Performance Computing 2010*, pages 135–148. Springer, 2012. doi:10.1007/978-3-642-24025-6\_12. (cited at p. 164)
- N. Z. Shor. Quadratic optimization problems. *Soviet Journal of Computer and Systems Sciences*, 25(6):1–11, 1987. (cited at p. 143)
- E. M. B. Smith and C. C. Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21(suppl.):S791–S796, 1997. doi:10.1016/S0098-1354(97)87599-0. (cited at p. 130)

## Bibliography

- E. M. B. Smith and C. C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimization of nonconvex MINLPs. *Computers & Chemical Engineering*, 23(4-5):457–478, 1999. doi:10.1016/S0098-1354(98)00286-5. (cited at pp. 130 and 141)
- I. Solberg. fminconset, 2000. URL <http://www.mathworks.com/matlabcentral/fileexchange/96>. (cited at p. 153)
- R. A. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86(3):515–532, 1999. doi:10.1007/s101070050103. (cited at p. 142)
- D. J. Swider and C. Weber. The costs of wind’s intermittency in germany: application of a stochastic electricity market model. *European Transactions on Electrical Power*, 17(2):151–172, 2007. doi:10.1002/etep.125. (cited at p. 102)
- D. J. Swider, P. Vogel, and C. Weber. Stochastic model for the european electricity market and the integration costs for wind power. GreenNet Report on WP 6, 2004. (cited at p. 102)
- M. Tawarmalani and N. V. Sahinidis. Semidefinite relaxations of fractional programs via novel convexification techniques. *Journal of Global Optimization*, 20(2):133–154, 2001. doi:10.1023/A:1011233805045. (cited at pp. 127 and 128)
- M. Tawarmalani and N. V. Sahinidis. Convex extensions and envelopes of lower semi-continuous functions. *Mathematical Programming*, 93(2):247–263, 2002a. doi:10.1007/s10107-002-0308-z. (cited at pp. 126 and 150)
- M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, volume 65 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers, 2002b. ISBN 978-1-4020-1031-6. (cited at pp. 115, 116, 127, 128, 129, 138, and 141)
- 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. doi:10.1007/s10107-003-0467-6. (cited at p. 138)
- M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005. doi:10.1007/s10107-005-0581-8. (cited at p. 150)
- M. Tawarmalani, J.-P. Richard, and K. Chung. Strong valid inequalities for orthogonal disjunctions and bilinear covering sets. *Mathematical Programming*, 124(1):481–512, 2010. doi:10.1007/s10107-010-0374-6. (cited at pp. 131 and 144)
- M. Tawarmalani, J.-P. Richard, and C. Xiong. Explicit convex and concave envelopes through polyhedral subdivisions. *Mathematical Programming*, 138(1-2):531–577, 2013. doi:10.1007/s10107-012-0581-4. (cited at p. 132)

- J. Tind and L. A. Wolsey. An elementary survey of general duality theory in mathematical programming. *Mathematical Programming*, 21(1):241–261, 1981. doi:10.1007/BF01584248. (cited at p. 33)
- S. Trukhanov, L. Ntaimo, and A. Schaefer. Adaptive multicut aggregation for two-stage stochastic linear programs with recourse. *European Journal of Operational Research*, 206(2):395–406, 2010. doi:10.1016/j.ejor.2010.02.025. (cited at p. 23)
- Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Martí. Scatter search and local NLP solvers: A multistart framework for global optimization. *INFORMS Journal on Computing*, 19(3):328–340, 2007. doi:10.1287/ijoc.1060.0175. (cited at p. 155)
- M. H. van der Vlerk. Convex approximations for complete integer recourse models. *Mathematical Programming*, 99(2):297–310, 2004. doi:10.1007/s10107-003-0434-2. (cited at pp. 30 and 32)
- M. H. van der Vlerk. Stochastic integer programming bibliography, 2007. URL <http://www.eco.rug.nl/mally/biblio/SIP.HTML>. (cited at p. 21)
- M. H. van der Vlerk. Convex approximations for a class of mixed-integer recourse models. *Annals of Operations Research*, 177(1):139–151, 2010. doi:10.1007/s10479-009-0591-7. (cited at pp. 30 and 32)
- R. M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17(4):638–663, 1969. doi:10.1137/0117061. (cited at pp. 22 and 23)
- R. J. Vanderbei and D. F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13(1-3):231–252, 1999. doi:10.1023/A:1008677427361. (cited at p. 154)
- S. A. Vavasis. Complexity issues in global optimization: A survey. In Horst and Pardalos [1995], pages 27–41. ISBN 978-0-7923-3120-9. (cited at p. 116)
- A. Vecchiotti and I. E. Grossmann. LOGMIP: A disjunctive 0-1 nonlinear optimizer for process system models. *Computers & Chemical Engineering*, 23(4-5):555–565, 1999. doi:10.1016/S0098-1354(98)00293-2. (cited at p. 156)
- J. P. Vielma, S. Ahmed, and G. L. Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs. *INFORMS Journal on Computing*, 20(3):438–450, 2008. doi:10.1287/ijoc.1070.0256. (cited at p. 236)
- S. Vigerske. Adaptive Diskretisierung stochastischer Optimierungsprobleme. Master’s thesis, Humboldt-Universität zu Berlin, 2005. (cited at p. iii)
- S. Vigerske. *RecombTree - user’s manual*. Humboldt-Universität zu Berlin, Germany, 2011. (cited at p. 70)

## Bibliography

- J. Viswanathan and I. E. Grossmann. A combined penalty function and outer-approximation method for MINLP optimization. *Computers & Chemical Engineering*, 14(7):769–782, 1990. doi:10.1016/0098-1354(90)87085-4. (cited at p. 124)
- X.-H. Vu, H. Schichl, and D. Sam-Haroud. Interval propagation and search on directed acyclic graphs for numerical constraint solving. *Journal of Global Optimization*, 45(4):499–531, 2009. doi:10.1007/s10898-008-9386-7. (cited at pp. 167, 172, 173, and 174)
- A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. URL <http://projects.coin-or.org/Ipopt>. doi:10.1007/s10107-004-0559-y. (cited at pp. 151, 155, 166, and 202)
- H.-J. Wagner. Wind energy and present status in germany. In N. Bansal and J. Mathur, editors, *Wind Energy Utilization : An Appraisal Of Projects In India And Germany*, pages 48–74. Anamaya Publishers, New Delhi, 2002. ISBN 8188342033. (cited at p. 102)
- D. W. Walkup and R. J.-B. R. Wets. Lifting projections of convex polyhedra. *Pacific Journal of Mathematics*, 28(2):465–475, 1969. doi:10.2140/pjm.1969.28.465. (cited at p. 11)
- C. Weber. *Uncertainty in the Electric Power Industry: Methods and Models for Decision Support*, volume 77 of *International Series in Operations Research & Management Science*. Springer, 2005. doi:10.1007/b100484. (cited at pp. 103 and 104)
- T. Westerlund and K. Lundquist. Alpha-ECP, version 5.04. an interactive MINLP-solver based on the extended cutting plane method. Technical Report 01-178-A, Process Design Laboratory, Åbo Akademi University, Åbo, Finland, 2003. URL <http://www.abo.fi/~twesterl1/A-ECPManual.pdf>. (cited at pp. 122 and 150)
- T. Westerlund and F. Pettersson. An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19(suppl.):131–136, 1995. doi:10.1016/0098-1354(95)87027-X. (cited at p. 121)
- T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3(3):253–280, 2002. doi:10.1023/A:1021091110342. (cited at pp. 124 and 150)
- L. A. Wolsey. Integer programming duality: Price functions and sensitivity analysis. *Mathematical Programming*, 20(1):173–195, 1981. doi:10.1007/BF01589344. (cited at p. 33)
- K. Wolter. Implementation of cutting plane separators for mixed integer programs. Master’s thesis, Technische Universität Berlin, 2006. (cited at pp. 142 and 164)
- World Wind Energy Association. World Wind Energy Report 2010, 2011. URL <http://www.wwindea.org>. (cited at p. 101)

- S. Wruck. Stochastische Optimierung von Energieerzeugung und Energieübertragungssystemen. Master's thesis, Humboldt-Universität zu Berlin, 2009. (cited at p. iv)
- G. Xu, S. Tsoka, and L. G. Papageorgiou. Finding community structures in complex networks using mixed integer optimisation. *The European Physical Journal B*, 60(2): 231–239, 2007. doi:10.1140/epjb/e2007-00331-0. (cited at p. 230)
- D. B. Yudin and A. S. Nemirovski. Informational complexity and efficient methods for solving complex extremal problems. *Ekonomika i Matematicheskie Metody*, 12: 357–369, 1976. Translated in *Matekon: Translations of Russian and East European Mathematical Economics*, 13:25–45, 1977. (cited at p. 116)
- J. M. Zamora and I. E. Grossmann. A global MINLP optimization algorithm for the synthesis of heat exchanger networks with no stream splits. *Computers & Chemical Engineering*, 22(3):367–384, 1998. doi:10.1016/S0098-1354(96)00346-8. (cited at p. 127)
- J. M. Zamora and I. E. Grossmann. A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. *Journal of Global Optimization*, 14(3):217–249, 1999. doi:10.1023/A:1008312714792. (cited at p. 128)