ISNM
International Series of Numerical Mathematics
Volume 152

Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming

Ivo Nowak

Birkhäuser Verlag
Basel · Boston · Berlin

Author:

Ivo Nowak
Niebuhrstr. 63
D-10629 Berlin
e-mail: Dr.Ivo_Nowak@gmx.de

2000 Mathematics Subject Classification: 90C11, 90C26, 90C22, 90C46, 90C57, 90C59

A CIP catalogue record for this book is available from the Library of Congress, Washington D.C., USA
Bibliographic information published by Die Deutsche BibliothekDie Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

ISBN 3-7643-7238-9 Birkhäuser Verlag, Basel – Boston – Berlin

ISBN-10: 3-7643-7238-9 e-ISBN: 3-7643-7374-1
ISBN-13: 978-3-7643-7238-5

9 8 7 6 5 4 3 2 1 www.birkhauser.ch

Contents

Preface

Nonlinear optimization problems containing both continuous and discrete variables are called mixed integer nonlinear programs (MINLP). Such problems arise in many fields, such as process industry, engineering design, communications, and finance.

There is currently a huge gap between MINLP and mixed integer linear programming (MIP) solver technology. With a modern state-of-the-art MIP solver it is possible to solve models with millions of variables and constraints, whereas the dimension of solvable MINLPs is often limited by a number that is smaller by three or four orders of magnitude. It is theoretically possible to approximate a general MINLP by a MIP with arbitrary precision. However, good MIP approximations are usually much larger than the original problem. Moreover, the approximation of nonlinear functions by piecewise linear functions can be difficult and time-consuming.

In this book relaxation and decomposition methods for solving nonconvex structured MINLPs are proposed. In particular, a generic *branch-cut-and-price* (BCP) framework for MINLP is presented. BCP is the underlying concept in almost all modern MIP solvers. Providing a powerful decomposition framework for both sequential and parallel solvers, it made the success of the current MIP technology possible. So far generic BCP frameworks have been developed only for MIP, for example, COIN/BCP (IBM, 2003) and ABACUS (OREAS GmbH, 1999). In order to generalize MIP-BCP to MINLP-BCP, the following points have to be taken into account:

- A given (sparse) MINLP is reformulated as a block-separable program with linear coupling constraints. The block structure makes it possible to generate Lagrangian cuts and to apply Lagrangian heuristics.
- In order to facilitate the generation of polyhedral relaxations, nonlinear convex relaxations are constructed.
- The MINLP separation and pricing subproblems for generating cuts and columns are solved with specialized MINLP solvers.
- Solution candidates are computed via MINLP heuristics by using an NLP solver.

I started to work on these tasks in 1996 when I implemented a branch-and-bound algorithm for solving polynomial programs based on multivariate Bézier polynomials (Nowak, 1996). Since polynomial programs can be reformulated as all-quadratic programs, I got interested in semidefinite programming relaxations. At this time I learned from Werner Römisch and Krzysztof Kiwiel about Lagrangian decomposition in the context of stochastic programming. Motivated by both approaches, I started in 2000 to implement an object oriented library, called LaGO (Lagrangian Global Optimizer), for solving nonconvex mixed-integer all-quadratic programs (MIQQPs) based on Lagrangian decomposition and semidefinite relaxation. From 2001 until 2003, LaGO was extended in a project funded by the German Science Foundation to solve nonconvex MINLPs.

This book documents many of the theoretical and algorithmic advances that made the development of LaGO possible and that give suggestions for further improvements. The most important contributions are:

- Several estimates on the duality gap (Sections 3.4, 3.5 and 5.4).
- A new column generation method for generating polyhedral inner and outer approximations of general MINLPs (Section 4.3).
- A new decomposition-based method for solving the dual of general MIQQPs through eigenvalue computation (Section 5.3).
- A new lower bounding method for multivariate polynomials over simplices based on Bernstein–Bézier representations (Section 6.2).
- A new polynomial underestimator for general nonconvex multivariate black-box functions (Section 6.5).
- New locally exact cuts based on interval arithmetic (Section 7.1.3).
- Decomposition-based lower bounds and box-reduction techniques for MINLPs (Sections 7.3 and 7.4).
- Optimality cuts and global optimality criteria for quadratically constrained quadratic programs (QQPs) based on a new strong duality result (Chapter 8).
- A new adaptive method for simultaneously generating discretizations and computing relaxations of infinite dimensional MINLPs (Chapter 9).
- New deformation heuristics for MaxCut and MINLP (Chapter 11) based on convex relaxations.
- Rounding and partitioning heuristics for MINLP (Sections 12.1 and 12.2).
- A Lagrangian heuristic for MINLP (Section 12.4).
- The first BCP algorithm for general MINLPs (Chapter 13).
- The first finiteness proof for QQP branch-and-bound methods that use optimality cuts (Section 13.2).

- A tool for automatically generating a block-separable reformulation of a black-box MINLP (Sections 2.3.2 and 14.4.1).

The use of relaxation-based methods for solving practically relevant large-scale MINLPs is quite new, and the integration of the two well established areas, nonlinear and mixed integer optimization, does not belong to the "traditional" operation research areas yet. However, according to a recent paper on future perspectives of optimization (Grossmann and Biegler, 2002) this can change in the future.

This monograph can be used both as a research text and as an introduction into MINLP. It is subdivided into two parts. The first part provides some basic concepts and the second part is devoted to solution algorithms.

Chapters 1 and 2 give an introduction into structured MINLPs and discuss various ways of reformulating a MINLP to be block-separable. Chapters 3, 4, 5, 6, 7 are devoted to theory and computational methods for generating Lagrangian and convex relaxations. Chapters 8 and 9 present global optimality cuts and a new method for refining discretizations of infinite dimensional MINLPs.

Chapter 10 gives an overview on existing global optimization methods. Chapters 11 and 12 describe deformation, rounding-and-partitioning and Lagrangian heuristics. Chapter 13 presents branch-cut-and-price algorithms for general MINLPs.

Chapter 14 contains a short description of the MINLP solver LaGO. Appendices A and B discuss future perspectives on MINLP and describe the MINLP instances used in the numerical experiments.

Berlin, January 2005 Ivo Nowak

Acknowledgments

At this point, I would like to thank some people who accompanied me in the last six years.

First of all, I want to thank Werner Römisch who provided me with a very nice working atmosphere at the HU-Berlin and gave me all the freedom for my own research. He helped me with the DFG-project "Optimization of a Complex Energy Conversion Plant", and introduced me into Lagrangian decomposition. I want to thank Ignacio Grossmann, Christoph Helmberg and Nikolaos Sahinidis for their excellent reports. Special thanks go to my friend and coauthor Stefan Vigerske who was a driving force for the software implementation. I want to thank my coauthors Turang Ahadi-Oskui, Hernán Alperin, Frank Cziesla and George Tsatsaronis for many lively discussions and a wonderful cooperation in the DFG-project. Thanks go also to Alex Meeraus, Michael Bussieck and Franz Nelißen from *The GAMS Development Corporation* for helping to build a link between LaGO and GAMS and for inviting Stefan to Washington D.C. I would like to thank some people from the optimization community, in particular Mirjam Dür and Arnold Neumaier for carefully reading a first version of this book and making many useful comments, Jan Bisschop for some valuable advice and for lively discussions at Paragon Decision Technology B.V. in Haarlem, Krzysztof Kiwiel for making NOA available and for valuable remarks on nonsmooth optimization, Rainer Horst and Nguyen Thoai for inviting me to Trier and for discussions about global optimization, Andreas Grothey for pointing me to branch-and-price and Hans Mittelmann for submitting MINLP instances. I wish to thank also all the people from the applied math-department who helped me many times. I would like to thank the editors of the ISNM series for accepting this book and Birkhäuser Verlag for a very pleasant cooperation. I acknowledge the financial support from the German Research Foundation (DFG) under grant NO 421/2-1 and NO 421/2-3. Finally, I would like to thank my wife Elena for her love.

Notation

∇	gradient column vector.
∇^2	Hessian.
∇_x	gradient with respect to x.
$\partial f(x)$	subdifferential of f at x.
x^T	the transpose of the vector x.
x_I	sub-vector defined by $(x_i)_{i\in I}$.
$f_I(x)$	sub-function defined by $(f_i(x))_{i\in I}$.
$A_{I,J}$	sub-matrix defined by $(a_{ij})_{i\in I, j\in J}$.
$\|x\|$	Euclidian norm of the vector x.
$\langle x, y\rangle$	scalar product of vectors x and y.
$\langle A, B\rangle$	inner product of matrices A and B defined by trace AB.
$f(x;t)$	function depending on a variable x and a parameter t.
$\bar{f}(x;z)$	linear approximation to f at z evaluated at x.
$\|I\|$	cardinality of the index set I.
μ	dual point (Lagrangian multiplier).
$L(x;\mu)$	Lagrangian function.
$D(\mu)$	dual function.
$\mathcal{M}$	Lagrangian multiplier set.
$\mathbb{B}^n(\rho, x)$	n-dimensional ball with center x and radius ρ.
$\mathbb{B}^n(\rho)$	$\mathbb{B}^n(\rho, 0)$.
$\mathbb{B}(n)$	$\mathbb{B}^n(\sqrt{n}, 0)$.
$\mathbb{S}^n$	n-sphere.
Δ_n	standard simplex in $\mathbb{R}^n$.
$\lambda_1(A)$	smallest eigenvalue of a matrix A.
$A \succcurlyeq B$	Loewner order of symmetric matrices defined by $A \succcurlyeq B \Leftrightarrow A - B$ is positive semi-definite.
$\mathrm{conv}(S)$	convex hull of the set S.
$\mathrm{vert}(S)$	extreme points of the set S.
$\mathrm{int}\, S$	interior of the set S.
$\breve{S}$	convex relaxation of the set S.
$\hat{S}$	polyhedral outer approximation of the set S.
$\check{S}$	polyhedral inner approximation of the set S.

$\imath_\Omega(x)$	characteristic function.
$\mathrm{val}(P)$	optimal value of the optimization problem (P).
$\mathrm{sol}(P)$	solution set of the optimization problem (P).

Part I

Basic Concepts

Chapter 1

Introduction

In this chapter a general mixed integer nonlinear program (MINLP) is defined, and some structural properties are described that are used in solution algorithms. Furthermore, several applications of MINLPs are given. Finally, an outline of the proposed solution approach is given, and an illustrative example is presented to demonstrate some basic ideas.

1.1 The structured nonconvex mixed integer nonlinear program

A general nonconvex mixed-integer nonlinear program ($MINLP$) is defined by

$$\text{(MINLP)} \qquad \begin{array}{ll} \min & f(x,y) \\ \text{s.t.} & g(x,y) \leq 0 \\ & h(x,y) = 0 \\ & x \in [\underline{x}, \overline{x}] \\ & y \in [\underline{y}, \overline{y}] \quad \text{integer} \end{array}$$

where the vector of continuous variables x and the vector of discrete variables y are finite and f, g and h are general nonlinear functions. The value of the objective function at a global minimizer is called *optimal value* of the MINLP and is denoted by val($MINLP$). The set of all global minimizers is called the *solution set* and is denoted by sol($MINLP$). If f and g are convex and h is affine, a MINLP is called *convex*. Otherwise, it is called nonconvex. The acronym MINLP usually stands for convex MINLPs. Here, it is also used for nonconvex problems.

If the functions f, g and h are *block-separable*, i.e. if they can be represented as a sum of sub-functions depending on a common subset of variables, a MINLP is called block-separable. In Chapter 2 we show that *sparse* MINLPs, for which most of the entries of the Hessians of f, g and h are zero, can be reformulated as

block-separable programs. Block-separability is the key structural property that is exploited by decomposition-based solution methods.

Two important subclasses of MINLP are the mixed-integer quadratically constrained quadratic program (*MIQQP*), where f, g and h are quadratic forms, and the mixed-integer linear program (*MIP*), where f, g and h are affine functions. Analyzing 150 MINLP problems of the MINLPLib, a library of MINLP instances collected by GAMS (Bussieck et al., 2003a), showed that 85% of these problems are nonconvex, 50% are quadratic and 85% are block-separable[1]. This demonstrates that nonconvexity is an important issue, and it is worthwhile to use specialized algorithms for solving quadratic problems. The analysis shows also that a large number of problems have a natural block-separable structure, which is often related to components of the underlying model.

How difficult is it to solve a MINLP? From the theoretical point of view, MINLPs are NP-hard (Garey and Johnson, 1979; Murty, 1987; Vavasis, 1995). This means if $NP \neq P$ then, in the worst case, it is not possible to solve a MINLP in polynomial time. However, these theoretical considerations do not reflect the difficulty of solving the problem in terms of computing time. There is often a large computational gap between finding a solution by a heuristic and verifying its global optimality. It is very hard to say what makes a problem difficult. Is it the number of local solutions, the number and the type of nonlinearities or the size of the problem? Unlike from convex programming, the prediction of computing time in nonconvex programming is usually impossible. Numerical experiments with different kinds of solvers seem to be the only reliable measure of difficulty.

1.2 Applications

There are a vast number of MINLP applications in many areas, such as engineering design, computational chemistry, computational biology, communications and finance. Floudas (Floudas, 2000) gives an overview of many applications, including process synthesis, process design, process synthesis and design under uncertainty, molecular design, interaction of design, synthesis and control, process operations, facility location and allocation, facility planning and scheduling and topology of transportation networks. Other areas of interest are VLSI manufacturing, automobile and aircraft. More applications can be found in (Pintér, 1996; Grossmann and Sahinidis, 2002a; Grossmann and Sahinidis, 2002b).

MIQQP applications include all bilinear problems, for example pooling problems in petrochemistry (Visweswaran and Floudas, 1990), modularization of product sub-assemblies (Rutenberg and Shaftel, 1971) and special cases of structured stochastic games (Filar and Schultz, 1999). Other applications are packing problems studied in the book of Conway and Sloane (Conway and Sloane, 1993), minmax location problems (Phan-huy-Hao, 1982), chance-constrained problems

[1]The numbers are from July 2003 and subject to change, since the MINLPLib is growing quite fast.

in portfolio optimization (Demands and Tang, 1992; Phan-huy-Hao, 1982; Weintraub and Vera, 1991), fuel-mixture problems encountered in the oil industry (Phing et al., 1994), placement and layout problems in integrated circuit design (Al-Khayyal et al., 1995; Al-Khayyal and van Voorhis, 1996). Many hard combinatorial optimization problems are special cases of MIQQP, such as MaxCut, MaxClique or quadratic assignment.

Several MINLPs can be reformulated as MIQQPs, for example (fractional) polynomial programs. MIQQPs can also serve as approximations of MINLPs for computing solution estimates, as with the approximation of the molecular predicting problem by a quadratic assignment problem (Phillips and Rosen, 1994). Under mild assumptions it can be shown that every MINLP can be approximated by a MIQQP or by a MIP with arbitrary precision (Neumaier, 2001). Since MIP is currently the only class that can be solved reliably in high dimensions, many real-world problems are modeled as MIPs.

1.3 Outline of the solution approach

The core of the proposed approach for solving nonconvex MINLPs is a polyhedral outer approximation (R) of a given problem (P) that is computed by the following five steps:

1. A block-separable reformulation ($\mathrm{P}_{\mathrm{split}}$) of (P) is generated by partitioning the sparsity graph of (P).
2. A nonlinear convex underestimating relaxation ($\mathrm{C}_{\mathrm{under}}$) of ($\mathrm{P}_{\mathrm{split}}$) is constructed by replacing nonconvex functions with convex underestimators.
3. The bounding box of (P) is reduced and some binary variables are fixed by using convex constraints of ($\mathrm{C}_{\mathrm{under}}$).
4. Two reformulations ($\mathrm{C}_{\mathrm{ext}}$) and ($\mathrm{P}_{\mathrm{ext}}$) of ($\mathrm{C}_{\mathrm{under}}$) and of ($\mathrm{P}_{\mathrm{split}}$) are constructed that have linear coupling constraints.
5. A polyhedral outer and inner approximation (R) and (RMP) are computed from ($\mathrm{C}_{\mathrm{ext}}$) and ($\mathrm{P}_{\mathrm{ext}}$) by using a decomposition algorithm that generates cuts and so-called inner approximation points. For this, (convex) nonlinear constraints are linearized and MINLP separation and pricing subproblems are solved.

Four algorithms for generating solution candidates of the original problem that use a nonlinear convex relaxation ($\mathrm{C}_{\mathrm{under}}$) or a polyhedral relaxation (R) are proposed (see Chapters 11, 12 and 13). The first algorithm is a deformation heuristic that is based on iteratively deforming a convex relaxation into a box-constrained formulation of the original problem. During this transformation sample points are modified by applying a neighborhood search. The second algorithm is a rounding-and-partitioning heuristic based on rounding solutions of convex relaxations and

solving the nonconvex continuous subproblems by subsequently splitting off solution candidates. The third algorithm is a Lagrangian heuristic that generates solution candidates by combining inner approximation points. The fourth algorithm is a branch-cut-and-price algorithm that uses the aforementioned heuristics for computing upper bounds, and convex relaxations for computing lower bounds.

For the efficiency of the algorithms, it is important that a relaxation can be constructed quickly, and also that it is a good approximation of the given nonconvex problem. Both goals depend on the size of the blocks. Small blocks make it possible to compute convex underestimators, cuts and columns quickly, but they lead to larger duality gaps. Increasing the size of the blocks diminishes the duality gap, but makes it more difficult to compute the relaxation. A block-separable splitting-scheme described in Section 2.3 makes it possible to balance both goals.

1.4 An illustrative example

In order to explain some basic ideas of the proposed solution approach, a simple rounding heuristic for computing a solution candidate of the following MINLP is described:

$$\text{(P)} \qquad \min\{x_2 \mid g_1(x) \leq 0,\ g_2(x) \leq 0,\ x_1 \in \{0,1\},\ x_2 \in [0,1]\},$$

where g_1 is a complicated nonconvex function and g_2 is a convex function. The rounding heuristic consists of the following four steps, which are illustrated in Figure 1.1.

In the first step, a convex relaxation is constructed by replacing g_1 with a convex underestimator q_1 and relaxing the binary constraint by $x_1 \in [0,1]$. The resulting nonlinear convex relaxation, defined by

$$(\text{C}_{\text{under}}) \qquad \min\{x_2 \mid q_1(x) \leq 0,\ g_2(x) \leq 0,\ x_1 \in [0,1],\ x_2 \in [0,1]\},$$

is solved yielding the point x^1. In the second step, a polyhedral relaxation is generated by a linearization $\bar{g}_2$ of g_2 at x^1 and by the affine function $\bar{g}_1$ that is parallel to a linearization $\bar{q}_1$ of q_1 at x^1. The polyhedral relaxation

$$\text{(R)} \qquad \min\{x_2 \mid \bar{g}_1(x) \leq 0,\ \bar{g}_2(x) \leq 0,\ x_1 \in [0,1],\ x_2 \in [0,1]\},$$

is solved resulting in a point x^2. In the third step, the binary component of x^2 is rounded to x^3, and the polyhedral subproblem of (R) with fixed binary variables is solved giving a point x^4. Finally, a local minimizer x^5 of the NLP subproblem of (P) with fixed binary variables is computed using x^4 as a starting point. Note that without adding the cut $\bar{g}_1$, rounding of x^2 would lead to the wrong subproblem.

Instead of rounding x^2, we could also split the problem into two subproblems using a branch-and-bound algorithm.

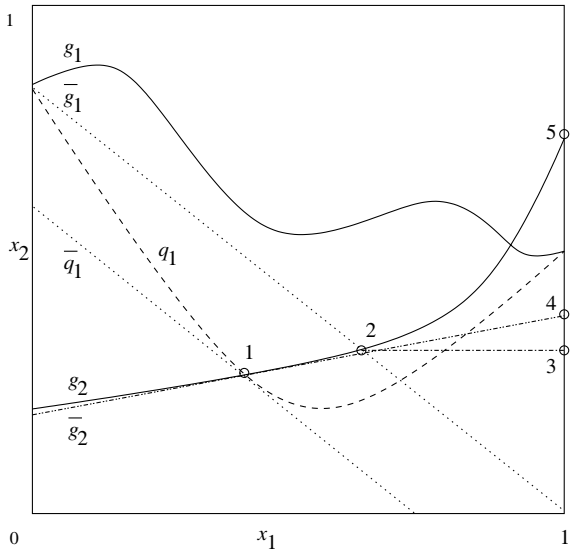

Figure 1.1: Basic steps of a rounding heuristic

Chapter 2

Problem Formulations

The formulation of a problem plays a central role in the solution strategy. Nemhauser and Wolsey (Nemhauser and Wolsey, 1988) wrote: "In integer programming, formulating a 'good' model is of crucial importance to solving the model." Automatic generation of favorable reformulations is now standard in many MIP codes (Bixby et al., 2000). It is used only recently in MINLP (Tawarmalani and Sahinidis, 2002; see also Section 14.4.1).

This chapter discusses various formulations of MINLPs that facilitate (i) the computation of local minimizers, (ii) the construction of convex underestimators (see Chapter 6) and (iii) the generation of valid cuts and columns (see Chapter 7). Particularly, it is shown how block-separable splitting-schemes with almost arbitrary block-sizes can be derived from sparse problems via partitions of the sparsity graph. As already mentioned in Section 1.3, these reformulations make it possible to balance two goals: (i) fast computation of underestimators, cuts and columns and (ii) small duality gaps.

2.1 The condensed formulation

A representation of a MINLP that uses as few (decision) variables as possible is called a *condensed formulation*. It is defined by

$$\text{(P)} \qquad \begin{array}{ll} \min & f(x,y) \\ \text{s.t.} & g(x,y) \leq 0 \\ & h(x,y) = 0 \\ & x \in [\underline{x}, \overline{x}] \\ & y \in [\underline{y}, \overline{y}] \quad \text{integer} \end{array}$$

where $x, \underline{x}, \overline{x} \in \mathbb{R}^{n_x}$, $y, \underline{y}, \overline{y} \in \mathbb{R}^{n_y}$ and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \mapsto \mathbb{R}$, $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \mapsto \mathbb{R}^{m_g}$ and $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \mapsto \mathbb{R}^{m_h}$ are piecewise twice-differentiable functions, that might be defined using if-then-else expressions. A continuous subproblem of (P) with a

fixed vector of integer variables $\hat{y}$ is defined by:

$$(\mathrm{P}[\hat{y}]) \qquad \begin{array}{ll} \min & f(x,\hat{y}) \\ \text{s.t.} & g(x,\hat{y}) \leq 0 \\ & h(x,\hat{y}) = 0 \\ & x \in [\underline{x},\overline{x}]. \end{array}$$

Since the objective and the constraint functions of (P[$\hat{y}$]) are piecewise twice-differentiable, a standard NLP solver can be used to compute local minimizers.

2.2 Smooth and disjunctive reformulations

In order to construct convex relaxations of (P), it is useful to reformulate (P) as a binary program with smooth objective and constraint functions. For this, several transformations are described in the following.

2.2.1 Integrality constraints

An *integrality constraint* $y \in \mathbb{Z} \cap [\underline{y},\overline{y}]$ with $\underline{y},\overline{y} \in \mathbb{Z}$ can be expressed through a vector $x \in \{0,1\}^N$ of binary variables by the following formula:

$$y = \underline{y} + x_1 + 2x_2 + \cdots + 2^{N-1}x_N$$

where N is the minimum number of binary variables needed. This minimum number is given by

$$N = 1 + \mathrm{trunc}\left(\frac{\log(\bar{y}-\underline{y})}{\log 2}\right)$$

where the trunc function truncates its real argument to an integer value. Note that this transformation is only efficient if N is not too large.

2.2.2 Disjunctive constraints

A *disjunctive constraint* is defined as $x \in \bigcup_{j=1}^{p} G_j$, where $G_j \subset \mathbb{R}^n$, $j = 1,\ldots,p$, are arbitrary disjunctive sets. Such a constraint can also be formulated as

$$\bigvee_{j=1}^{p} (x \in G_j)$$

where $\vee$ denotes the 'or'-operation. Disjunctive constraints can be used to reformulate if-then-else expressions and piecewise defined functions. Consider an if-then-else expression of the form

$$\textbf{if}\,(x \in G_1)\,\textbf{then}\,(x \in G_2)\,\textbf{else}\,(x \in G_3).$$

The above if-then-else expression can be reformulated equivalently as a disjunctive constraint by

$$(x \in G_1, x \in G_2) \vee (x \notin G_1, x \in G_3).$$

An inequality constraint $f(x) \leq 0$, where f is a piecewise defined function of the form

$$f(x) = f_k(x) \text{ for } x \in G_k, \quad k = 1, \ldots, p,$$

can be reformulated as a disjunctive constraint by

$$(x \in G_1, f_1(x) \leq 0) \vee \cdots \vee (x \in G_p, f_p(x) \leq 0).$$

2.2.3 Big-M constraints

Consider a disjunctive constraint

$$\bigvee_{j=1}^{p} (x \in G_j)$$

where $G_j = \{x \in X \mid f_j(x) \leq 0\}$, $X \subset \mathbb{R}^n$ is a bounded set and $f_j : \mathbb{R}^n \mapsto \mathbb{R}^{m_j}$, $j = 1, \ldots, p$, are smooth functions. The above disjunctive constraint can be described equivalently by the so-called *big-M constraint*

$$f_j(x) \leq \overline{f}_j(1 - y_j), \quad y \text{ is in SOS}, \quad x \in X,$$

where $\overline{f}_j$ is an upper bound of f_j over X and the binary vector $y \in \{0, 1\}^p$ is in the *special order set* (SOS) defined by

$$\sum_{j=1}^{p} y_j = 1.$$

The name big-M comes from using M as a notation for the upper bound $\overline{f}_j$.

2.2.4 The smooth binary formulation

By applying the above rules, we can reformulate the condensed problem (P) as the following *smooth binary problem*:

$$\begin{array}{lll} \min & h_0(x) & \\ \text{s.t.} & h_i(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \end{array} \tag{2.1}$$

where $\underline{x}, \overline{x} \in \mathbb{R}^n$, $B \subseteq \{1, \ldots, n\}$, and $h_i, i = 0, \ldots, m$, are twice-differentiable functions. For the sake of simplicity, we assume here that equality constraints in (P) are replaced by two inequality constraints, i.e. $h_i(x) = 0$ is replaced by $h_i(x) \leq 0$ and $-h_i(x) \leq 0$. It is also possible to add disjunctive constraints $\bigvee_{j=1}^{p}(x \in G_j)$ to (2.1). This results in a *hybrid formulation* (Vecchietti et al., 2003).

2.2.5 Block-separability

Problem (2.1) is called *block-separable*, if there exists a *partition* $\{J_1, \ldots, J_p\}$ of $\{1, \ldots, n\}$, i.e. $\bigcup_{k=1}^{p} J_k = \{1, \ldots, n\}$ and $J_i \cap J_k = \emptyset$ for $i \neq k$, and functions $h_{i,k} : \mathbb{R}^{n_k} \mapsto \mathbb{R}$ with $n_k = |J_k|$ such that

$$h_i(x) = \sum_{k=1}^{p} h_{i,k}(x_{J_k}), \quad i = 0, \ldots, m.$$

In other words, all Hessians of the functions h_i, $i = 0, \ldots, m$, have a common block-diagonal structure. If the size of all blocks J_k is 1, i.e. $J_k = \{k\}$ for $k = 1, \ldots, p$, (2.1) is called *separable*. In this case, the functions h_i have the form

$$h_i(x) = \sum_{k=1}^{n} h_{i,k}(x_k).$$

2.3 Block-separable splitting-schemes

We will now discuss splitting-schemes for transforming general sparse MINLPs into block-separable MINLPs. This technique goes back to 1956 (Douglas and Rachford, 1956) where it was used for partial differential equations. It is widely used in stochastic programming (Ruszczyński, 1997) and in combinatorial optimization (Guignard and Kim, 1987).

2.3.1 The sparsity graph

We define the *sparsity graph* of (2.1) as the graph $G_{\text{sparse}} = (V, E_{\text{sparse}})$ with the vertices $V = \{1, \ldots, n\}$ and the edges

$$E_{\text{sparse}} = \left\{ (i,j) \in V^2 \;\middle|\; \frac{\partial^2 h_l(x)}{\partial x_i \partial x_j} \neq 0 \text{ for some } l \in \{0, \ldots, m\} \text{ and } x \in [\underline{x}, \overline{x}] \right\}.$$

The sparsity graph can be used to detect a block structure of (2.1).

Observation 2.1. *Let J_k, $k = 1, \ldots, p$, be the connected components of G_{sparse}. Then (2.1) is block-separable with respect to J_k, $k = 1, \ldots, p$.*

2.3.2 MINLP splitting-schemes

If problem (2.1) is not block-separable or if it has some large blocks that should be subdivided into smaller blocks, (2.1) can be reformulated to be block-separable by

introducing extra variables and constraints. Let $J_1, \ldots, J_p$ be an arbitrary partition of the vertex set V. The set of nodes of $\bigcup_{l=k+1}^{p} J_l$ connected to J_k is defined by

$$R_k = \{i \in \bigcup_{l=k+1}^{p} J_l \mid (i,j) \in E_{\text{sparse}},\, j \in J_k\},$$

for $k = 1, \ldots, p$. The set R_k can be interpreted as the set of flows of a network problem connecting a component J_k with components J_l, where $k < l \leq p$. If (2.1) is block-separable with respect to the blocks $J_k, k = 1, \ldots, p$, then $R_k = \emptyset$. Otherwise, some R_k's will be non-empty. From the definition of R_k it follows that there exist functions $\tilde{h}_{i,k} : \mathbb{R}^{|J_k|} \times \mathbb{R}^{|R_k|} \to \mathbb{R}$ such that

$$h_i(x) = \sum_{k=1}^{p} \tilde{h}_{i,k}(x_{J_k}, x_{R_k}), \quad i = 0, \ldots, m.$$

Replacing x_{R_k} by a new variable $y_k \in \mathbb{R}^{|R_k|}$ defines the functions

$$\tilde{h}_i(x, y_1, \ldots, y_p) = \sum_{k=1}^{p} \tilde{h}_{i,k}(x_{J_k}, y_k), \quad i = 0, \ldots, m.$$

Remark 2.2. Assuming that a black-box function h_i is block-separable w.r.t. a partition $J_1, \ldots, J_p$, the functions $\tilde{h}_{i,k}$ can be defined by

$$\tilde{h}_{i,k}(x_{J_k}) = h_i(\tilde{x}^k) - \frac{p-1}{p} h_i(\hat{x}),$$

where $\tilde{x}^k_{J_k} = x_{J_k}$, $\tilde{x}^k_{J_l} = \hat{x}_{J_l}$ for $l \neq k$ and $\hat{x} \in [\underline{x}, \overline{x}]$ is an arbitrary fixed point[1]. Clearly, $\sum_{k=1}^{p} \tilde{h}_{i,k}(x_{J_k}) = h_i(x) + h_i(\hat{x}) - p\frac{p-1}{p} h_i(\hat{x}) = h_i(x)$.

Since $\tilde{h}_i(x, x_{R_1}, \ldots, x_{R_p}) = h_i(x)$, the following splitting-problem with $n + \sum_{k=1}^{p} |R_k|$ variables is equivalent to (2.1):

$$\begin{array}{lll} \min & \tilde{h}_0(x, y_1, \ldots, y_p) & \\ \text{s.t.} & \tilde{h}_i(x, y_1, \ldots, y_p) \leq 0, & i = 1, \ldots, m \\ & y_k = x_{R_k}, & k = 1, \ldots, p \\ & y_k \in [\underline{x}_{R_k}, \overline{x}_{R_k}], & k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary}. \end{array} \tag{2.2}$$

Problem (2.2) is block-separable with respect to the blocks (J_k, R_k), $k = 1, \ldots, p$. The constraints $y_k = x_{R_k}$ are called *copy constraints*.

Example 2.3. Consider the sparsity graph shown in Figure 2.1, and let $J_1 = \{3, 4, 7, 8\}$ and $J_2 = \{1, 2, 5, 6, \}$ be a partition of this graph. Then $R_1 = \{2, 6\}$ and $R_2 = \emptyset$. The splitting-problem (2.2) contains the new variables x_9 and x_{10} and the copy constraints: $x_2 = x_9$ and $x_6 = x_{10}$. The new blocks are $\tilde{J}_1 = \{3, 4, 7, 8, 9, 10\}$ and $\tilde{J}_2 = \{1, 2, 5, 6\}$.

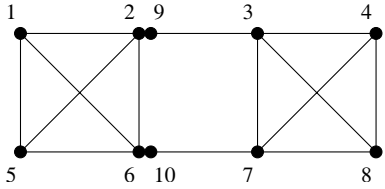

Figure 2.1: Partition of a sparsity graph into two components

The splitting-scheme (2.2) will be only efficient, if the cardinalities $|R_k|$, $k = 1, \ldots, p$, are not too large. To this end, the sparsity graph can be subsequently partitioned into two blocks at a time by solving, for example, a MinCut problem. The blocks can also be defined according to physically meaningful components of the given optimization model. In (Dentcheva and Römisch, 2002) such a decomposition is called geographical.

2.3.3 MIQQP splitting-schemes

Consider now the case where all functions in (2.1) are quadratic forms $h_i(x) = x^T A_i x + 2b_i^T x + c_i$. Since

$$x^T A_i x = \sum_{k=1}^{p} x_{J_k}^T A_{i,J_k,J_k} x_{J_k} + 2 \sum_{l=k+1}^{p} x_{J_k}^T A_{i,J_k,J_l} x_{J_l},$$

where $A_{J_k,J_l} \in \mathbb{R}^{(|J_k|,|J_l|)}$ denotes the submatrix $(a_{r,s})_{r \in J_k, s \in J_l}$, it follows that

$$x^T A_i x = \sum_{k=1}^{p} x_{J_k}^T A_{i,J_k,J_k} x_{J_k} + 2 \sum_{k=1}^{p} x_{J_k}^T A_{i,J_k,R_k} x_{R_k} \tag{2.3}$$

for $i = 0, \ldots, m$. Setting

$$\tilde{h}_i(x, y_1, \ldots, y_p) = \sum_{k=1}^{p} x_{J_k}^T A_{i,J_k,J_k} x_{J_k} + 2 \sum_{k=1}^{p} x_{J_k}^T A_{i,J_k,R_k} y_k + 2b_i^T x + c_i,$$

$i = 0, \ldots, m$, it follows that $\tilde{h}_i(x, x_{R_1}, \ldots, x_{R_p}) = h_i(x)$.

Block-angular structure

Problem (2.1) has a block-angular structure if the matrices are of the form

$$A_i = \begin{pmatrix} A_{i,1} & & & B_{i,1} \\ & \ddots & & \vdots \\ & & A_{i,p-1} & B_{i,p-1} \\ B_{i,1}^T & \ldots & B_{i,p-1}^T & A_{i,p} \end{pmatrix}.$$

[1]This definition is used in the current implementation of LaGO (see Chapter 14).

Problems with such a structure arise, for example, in process system engineering, telecommunications problems, network problems and stochastic programming. In (Ferris and Horn, 1998) it is demonstrated that many sparse optimization problems can be efficiently transformed into problems with block-angular structure. Automatic detection of block structure of sparse MIPs is discussed in (Martin, 1999).

Let $\mathcal{P} = \{J_1, \ldots, J_p\}$ be a partition of V according to the above block structure. Then $R_k = J_p$ for $k < p$ and $R_p = \emptyset$. The related splitting-scheme is block-separable with respect to p blocks with block sizes $n_1 + n_p, \ldots, n_{p-1} + n_p$, n_p. It follows that the number of additional variables in the splitting-scheme (2.2) is $(p-1)n_p$.

Band structure

Problem (2.1) has a band structure if the matrices have the form

$$A_i = \begin{pmatrix} A_{i,1} & B_{i,1} & & \\ B_{i,1}^T & \ddots & \ddots & \\ & \ddots & A_{i,p-1} & B_{i,p-1} \\ & & B_{i,p-1}^T & A_{i,p} \end{pmatrix}.$$

There are many methods for transforming sparse matrices into matrices with band structure. A main application of these algorithms is to reduce the fill-in of a Cholesky factorization.

Let $\mathcal{P} = \{J_1, \ldots, J_p\}$ be a partition of V according to the above block structure. Then $R_k = J_{k+1}$ for $k < p$ and $R_p = \emptyset$. The related splitting-scheme is block-separable with respect to p blocks with block sizes $n_1 + n_2, \ldots, n_{p-1} + n_p, n_p$. It follows that the number of additional variables in the splitting-scheme (2.2) is not greater than $\sum_{k=2}^{p} n_k = n - n_1$.

2.4 Separable reformulation of factorable programs

In general, it is not possible to produce separable reformulations using splitting-schemes. A partition of the vertex set V into blocks $J_k = \{k\}$ of cardinality 1 leads to empty sets $R_k = \emptyset$ if and only if (2.1) is fully separable. In order to reformulate (2.1) to be separable or to diminish the size of some blocks, a disaggregation technique presented in (Tawarmalani and Sahinidis, 1999) can be used, provided that all functions are *factorable*. A function is called factorable, if it is defined by taking recursive sums and products of univariate functions. Algorithm 2.1 decomposes a factorable function f into equations involving separable functions (Tawarmalani and Sahinidis, 1999). For example, the function $f(x) = \log(x_1)x_2$ is decomposed by this algorithm into $f(x) = \frac{1}{4}y_1^2 - \frac{1}{4}y_2^2$, $y_1 = y_3 + x_2$, $y_2 = y_3 - x_2$ and $y_3 = \log(x_1)$.

Algorithm `Reform` $f(x)$:
`if` $f(x) = g(x)/h(x)$ `return Fractional_Reform(g,h)`
`if` $f(x) = \Pi_{i=1}^{l} f_i(x)$
    `for` $i := 1$ `to` l `do`
        add variable y_{f_i}, and constraint y_{f_i} =`Reform` $(f_i(x))$
    `return Multilinear_Reform`$(\Pi_{i=1}^{l} y_{f_i})$
`if` $f(x) = \sum_{i=1}^{l} f_i(x)$
    add variable y_{f_i}, and constraints y_{f_i} =`Reform` $(f_i(x))$
    `return` $\sum_{i=1}^{l} y_{f_i}$
`if` $f(x) = g(h(x))$
    add variable y_h and constraint y_h=`Reform`$(h(x))$
    `return Reform`$(g(y_h))$

Algorithm `Multilinear_Reform`$(\Pi_{i=1}^{l} y_{r_i})$:
`for` $i := 2$ `to` l `do`
    add variable $y_{r_1,\ldots,r_i}$ and the constraint
    $y_{r_1,\ldots,r_i}$ =`Bilinear_Reform`$(y_{r_1,\ldots,r_{i-1}} y_{r_i})$
`return` $y_{r_1,\ldots,r_l}$

Algorithm `Fractional_Reform(g,h)`:
add variables y_f, y_g, y_h and constraints
y_g=`Reform`$(h(x))$, y_h=`Reform`$(h(x))$ and y_g=`Bilinear_Reform`$(y_f y_h)$
`return` y_f

Algorithm `Bilinear_Reform`$(y_i y_j)$:
`return Reform` $(\frac{1}{4}(y_i + y_j)^2 - \frac{1}{4}(y_i - y_j)^2)$

Algorithm 2.1: Procedure for decomposing a factorable function into separable expressions

2.5 Extended block-separable reformulation

In order to facilitate the generation of polyhedral relaxations, it is useful to generate an extended reformulation of (2.1) with linear coupling constraints (see Chapter 7). In other words, all constraint functions depending on variables of different blocks are affine, and all nonlinear constraint functions of the extended reformulation depend on variables of only one block. To this end, an inequality constraint

$$\sum_{k=1}^{p} h_{i,k}(x_{I_k}) \leq 0$$

of (2.1) is rewritten equivalently by

$$\sum_{k=1}^{p} t_{i,k} \leq 0, \quad g_{i,k}(x_{I_k}, t_{i,k}) := h_{i,k}(x_{I_k}) - t_{i,k} \leq 0, \quad k = 1, \ldots, p.$$

The new variable $t_{i,k}$ is in the interval $[\underline{t}_{i,k}, \overline{t}_{i,k}]$ with

$$\underline{t}_{i,k} = \min\{\breve{h}_{i,k}(x) \mid x \in [\underline{x}_{I_k}, \overline{x}_{I_k}]\} \quad \text{and} \quad \overline{t}_{i,k} = \sum_{j \neq k} -\underline{t}_{i,j},$$

where $\breve{h}_{i,k}$ is a *convex underestimator* of $h_{i,k}$ over $[\underline{x}_{I_k}, \overline{x}_{I_k}]$, i.e. $\breve{h}_{i,k}(x_{I_k}) \leq h_{i,k}(x_{I_k})$ for $x \in [\underline{x}_{I_k}, \overline{x}_{I_k}]$ and $\breve{h}_{i,k}$ is convex over $[\underline{x}_{I_k}, \overline{x}_{I_k}]$. Examples for convex underestimators are given in Chapter 6. It is also possible to compute $\underline{t}_{i,k}$ via a box reduction method and interval arithmetic (see Section 7.4).

Transforming all nonlinear constraints and the objective function of (2.1) according to the above representation yields the following *extended reformulation* with linear coupling constraints:

$$\begin{array}{lll} \min & c^T x + c_0 & \\ \text{s.t.} & Ax + b \leq 0 & \\ & g_{i,k}(x_{J_k}) \leq 0, & i \in M_k,\ k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \end{array} \tag{2.4}$$

where $c \in \mathbb{R}^n$, $c_0 \in \mathbb{R}$, $A \in \mathbb{R}^{(m,n)}, b \in \mathbb{R}^m$, $g_{i,k}$ are nonlinear functions and $M_k \subset \mathbb{N}$ are finite sets. For the sake of simplicity, we denote, as in (2.1), by $\{J_1, \ldots, J_p\}$ the blocks of (2.4).

Example 2.4. Consider the following separable program:

$$\begin{array}{ll} \min & \sin(x_1) + x_2 \\ \text{s.t.} & 2x_1 + 3x_2 \leq 0 \\ & x_1 + x_2^2 \leq 0. \end{array}$$

An extended reformulation with linear coupling constraints of the above problem is:

$$\begin{array}{ll} \min & t_{0,1} + x_2 \\ \text{s.t.} & 2x_1 + 3x_2 \leq 0 \\ & x_1 + t_{2,2} \leq 0 \\ & \sin(x_1) \leq t_{0,1} \\ & x_2^2 \leq t_{2,2} \end{array}$$

2.6 Other formulations

Apart from the reformulations described in this chapter, there are also other interesting reformulations.

1. A MIQQP containing only binary variables can be transformed into a MIP by the following procedure. Let $x_i, x_j \in \{0,1\}$. Then $x_{ij} = x_i x_j$ if and only if $x_{ij} \geq 0$, $x_{ij} \geq x_i + x_j - 1$, $x_{ij} \leq x_i$ and $x_{ij} \leq x_j$. Replacing the bilinear terms by these expressions, and using $x_i^2 = x_i$, yields an MIP reformulation of a MIQQP with $n^2/2$ additional variables and $3n^2/2$ additional constraints:

$$\begin{array}{lll} \min & \langle A_0, X \rangle + 2b_0^T x + c_0 & \\ \text{s.t.} & \langle A_i, X \rangle + 2b_i^T x + c_i \leq 0, & i = 1, \ldots, m \\ & x_{ij} \geq x_i + x_j - 1, & 1 \leq i < j \leq n \\ & x_{ij} \leq x_i, & 1 \leq i < j \leq n \\ & x_{ij} \leq x_j, & 1 \leq i < j \leq n \\ & x \in \{0,1\}^n,\ X \in \mathbb{R}_+^{(n,n)} & \end{array}$$

   where $\langle A, X \rangle = \sum_{i,j} a_{ij} x_{ij}$.

2. A recent cutting-plane algorithm for box-constrained nonconvex quadratic programs of the form $\min\{\frac{1}{2}x^T Q x + c^T x \mid x \in [0, e]\}$ is based on the following reformulation (Vandenbussche, 2003):

$$\max\{\tfrac{1}{2}c^T x + \tfrac{1}{2}\overline{\mu}^T e \mid (x, \underline{\mu}, \overline{\mu}) \in \text{LPS}\},$$

   where $\overline{\mu} \in \mathbb{R}^n$ and $\underline{\mu} \in \mathbb{R}^n$ are the dual variables for the constraints $x \leq e$ and $x \geq 0$ respectively, and LPS $\subset \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n$ is the set of Karush–Kuhn–Tucker points of the given problem.

3. Let $\min\{f(x) \mid g_j(x) \leq 0,\ j = 1, \ldots, m\}$ be a twice-differentiable nonconvex optimization problem. In (Kojima et al., 1999) the following reformulation of the above problem that contains a single nonconvex quadratic constraint is proposed:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_j(x) + \sigma_j(\|x\|^2 - t) \leq 0, \quad j = 1, \ldots, m \\ & -\|x\|^2 + t \leq 0 \end{array}$$

where $\sigma_j \geq -\min\{0, \min_{x \in S} \lambda_1(\nabla^2 g_j(x))\}$ for $j = 1, \ldots, m$, S denotes the feasible set and λ_1 denotes the minimum eigenvalue.

4. A MINLP can be formulated as a continuous optimization problem by replacing a binary constraint $x_j \in \{\underline{x}_j, \overline{x}_j\}$ by the quadratic equality constraint

$$(x_j - \underline{x}_j)(x_j - \overline{x}_j) = 0.$$

This formulation is used in Chapter 5 for deriving semidefinite relaxations.

5. Other reformulations are proposed in (Sherali and Adams, 1999) that are used in the so-called reformulation linearization technique (RLT). Moreover, in (Smith and Pantelides, 1999) it is shown that algebraic models are always reducible to bilinear, linear fractional and concave separable functions, provided they do not involve trigonometric functions.

Chapter 3

Convex and Lagrangian Relaxations

After a short introduction on the convexification of sets and functions, this chapter presents convex and Lagrangian relaxations of general MINLPs. The presented relaxations are compared and it is shown that Lagrangian relaxations are stronger than so-called convex underestimating-relaxations. Several dual-equivalent results are proven and estimates on the duality gap are given. Furthermore, the concept of augmented Lagrangians is introduced, which leads to a zero duality gap.

The roots of Lagrangian relaxation go back to Joseph Louis Lagrange (1736-1813) (Lagrange, 1797). It was presented in (Everett, 1963) for resource allocation problems. The reader is referred to (Geoffrion, 1974) and (Lemaréchal, 2001) for an introduction into this field. A comprehensive overview on duality theory is given in (Rockafellar and Wets, 1997).

The presented theory forms the background for the computation of a relaxation, which is the main tool in relaxation-based MINLP solution methods (see Chapters 11, 12 and 13).

3.1 Convexification of sets and functions

The following definitions and results on the convexification of sets and functions will be used in the subsequent sections. The intersection of all convex sets containing an arbitrary compact set G is called the *convex hull* of G and is denoted by $\text{conv}(G)$. There are two representations of the convex hull that are used in solution algorithms. The first one is based on supporting hyperplanes and the second one on extreme points. For a general compact set $G \subseteq \mathbb{R}^n$ and a vector $a \in \mathbb{R}^n$ we define the *support function* by

$$\sigma_G(a) = \sup_{x \in G} a^T x$$

and the related *supporting half-space* by

$$\{x \in \mathbb{R}^n \mid a^T x \leq \sigma_G(a)\}.$$

Observe that a support function does not distinguish a set from its closed convex hull.

Observation 3.1. *The convex hull of an arbitrary compact set $G \subset \mathbb{R}^n$ can be represented as*

$$\mathrm{conv}(G) = \{x \in \mathbb{R}^n \mid a^T x \leq \sigma_G(a) \text{ for all } a \in \mathbb{S}^n\} \tag{3.1}$$

where $\mathbb{S}^n$ denotes the n-sphere in $\mathbb{R}^n$.

The second characterization of $\mathrm{conv}(G)$ is based on the *extreme points* of G. A point $x \in G$ is called an extreme point of G if it cannot be represented by $x = (1-\lambda)y + \lambda z$, where $y, z \in G$, $y \neq z$, and $\lambda \in (0,1)$.

Observation 3.2. *Let* $\mathrm{vert}(G)$ *be the set of extreme points of a compact set G. Then*

$$\mathrm{conv}(G) \quad = \quad \bigcup \{\mathrm{conv}(V) \mid V \subseteq \mathrm{vert}(G) \text{ and } V \text{ is finite }\}.$$

If $\mathrm{vert}(G)$ *consists of finitely many points $w_1, \ldots, w_l$, then*

$$\mathrm{conv}(G) = \{\sum_{j=1}^{l} z_j w_j \mid z \in \Delta_l\} \tag{3.2}$$

where Δ_l denotes the standard simplex *in $\mathbb{R}^l$.*

Note that if G is an unbounded polyhedral set, its convex hull can be represented as a convex combination of extreme points and so-called extreme rays (Schrijver, 1986).

Next, the convexification of functions will be studied. A twice-differentiable function $f : X \mapsto \mathbb{R}$ is convex over a convex set $X \subseteq \mathbb{R}^n$ if the Hessian $\nabla^2 f(x)$ is positive semidefinite for all $x \in X$. It is called *strictly convex* over X if the Hessian $\nabla^2 f(x)$ is positive definite for all $x \in X$. A *convex underestimator* $\breve{f}$ of a function f over a set X is a convex function below f over X, i.e. $\breve{f}(x) \leq f(x)$ for all $x \in X$. The best convex underestimator of f over a convex set X is its *convex envelope* that is the supremum of all lower semi-continuous convex functions below f over X. The convex envelope over X is equivalent to the *biconjugate* $f_X^{**} = (f + \imath_X)^{**}$, where

$$f^*(y) = \sup_x [y^T x - f(x)]$$

is the *conjugate* of a function, $f^{**} = (f^*)^*$ and

$$\imath_X(x) = \begin{cases} 0 & \text{if } x \in X \\ +\infty & \text{otherwise} \end{cases}$$

is the indicator function (Hiriart-Urruty and Lemaréchal, 1993). The support function σ_S is also the *conjugate function* $(\imath_S)^*$ of $\imath_S$. In fact, convexity of f is not necessary for the conjugacy operation to make sense: f just needs to be finite at some point, and to have some affine minorant (Hiriart-Urruty and Lemaréchal, 1993).

Let $\mathrm{epi}_X(f) = \{(t,x) \mid t \geq f(x),\, x \in X\}$ be the *epi-graph* of a function f over a set X. For the epi-graph of a convex envelope f_X^{**} we have:

$$\mathrm{epi}_X(f_X^{**}) = \mathrm{conv}(\mathrm{epi}_X(f)).$$

For many functions, there exist analytic expressions of the convex envelope. For example, the convex envelope of the bilinear function $f(x) = x_1 \cdot x_2$ over the box $X = [-e, e]$ is the function $f_X^{**}(x) = \max\{-1 + x_1 + x_2, -1 - x_1 - x_2\}$. In (Tawarmalani and Sahinidis, 2002) so-called *convex extensions* are presented that provide representations of convex envelopes in terms of generating points.

3.2 Convex underestimating-relaxations

In this section convex relaxations obtained by replacing nonconvex functions by convex underestimators are studied. Typically, such relaxations can be computed quickly, but they are often not tight. Consider a MINLP of the form

$$\begin{array}{lll} \min & h_0(x) & \\ \text{s.t.} & h_i(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \end{array} \tag{3.3}$$

where h_i, $i = 0, \ldots, m$, are arbitrary functions and $B \subseteq \{1, \ldots, n\}$. A *convex underestimating-relaxation* of (3.3) is defined by

$$\begin{array}{lll} \min & \breve{h}_0(x) & \\ \text{s.t.} & \breve{h}_i(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}] & \end{array} \tag{3.4}$$

where $\breve{h}_i$ is a *convex underestimator* of h_i over $[\underline{x}, \overline{x}]$ of (3.3), i.e. $\breve{h}_i(x) \leq h_i(x)$ for all $x \in [\underline{x}, \overline{x}]$ and $\breve{h}_i$ is convex over $[\underline{x}, \overline{x}]$. Examples for convex underestimators are given in Chapter 6. An optimal convex underestimating-relaxation of (3.3) is defined by replacing all functions by their convex envelopes:

$$\begin{array}{lll} \min & h_0^{**}(x) & \\ \text{s.t.} & h_i^{**}(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}] & \end{array} \tag{3.5}$$

where h_i^{**} is the convex envelope of h_i over $[\underline{x}, \overline{x}]$. Since $\breve{h}_i(x) \leq h_i^{**}(x)$ for $x \in [\underline{x}, \overline{x}]$, we have

$$\mathrm{val}(3.4) \leq \mathrm{val}(3.5).$$

An exact convex relaxation of (3.3) can be obtained by formulating (3.3) as a box-constrained optimization problem by using the exact penalty function

$$P(x) = h_0(x) + \sum_{i=1}^{m} \delta_i \max\{0, h_i(x)\} + \sum_{i \in B} \gamma_i \max\{0, (x_i - \underline{x}_i)(x_i - \overline{x}_i)\}.$$

Assuming that the weights δ_i and γ_i are large enough, we have $\mathrm{val}(3.3) = \min_{x \in [\underline{x}, \overline{x}]} P(x)$. In this case, the convex relaxation

$$\begin{array}{ll} \min & P_X^{**}(x) \\ \text{s.t.} & x \in [\underline{x}, \overline{x}] \end{array} \tag{3.6}$$

with $X = [\underline{x}, \overline{x}]$ is exact (see Horst et al., 1995), i.e. $\mathrm{val}(3.3) = \mathrm{val}(3.6)$ and $\mathrm{conv}(\mathrm{sol}(3.3)) = \mathrm{sol}(3.6)$. From the above considerations there follows:

Lemma 3.3. *It holds that*

$$\mathrm{val}(3.3) = \mathrm{val}(3.6) \geq \mathrm{val}(3.5) \geq \mathrm{val}(3.4).$$

3.3 Lagrangian relaxation

This section examines Lagrangian relaxation and dual bounds of general optimization problems of the form:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x \in G \end{array} \tag{3.7}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ are continuous functions and $G \subseteq \mathbb{R}^n$ is an arbitrary set. It is clear that a general MINLP can be formulated as in (3.7) by replacing equality constraints by two inequality constraints and shifting integrality constraints to the set G. A *Lagrangian* to (3.7) is defined by

$$L(x; \mu) = f(x) + \mu^T g(x)$$

where the *dual point* μ is in $\mathbb{R}^m_+$. A *Lagrangian relaxation* of (3.7) is defined by

$$\begin{array}{ll} \inf & L(x; \mu) \\ \text{s.t.} & x \in G. \end{array} \tag{3.8}$$

A *dual function* related to (3.8) is given by its optimal value, i.e. $D(\mu) = \mathrm{val}(3.8)$. Since $L(x; \mu) \leq f(x)$ if x is feasible for (3.7) and $\mu \in \mathbb{R}^m_+$, we have

$$D(\mu) \leq \mathrm{val}(3.7) \text{ for all } \mu \in \mathbb{R}^m_+.$$

Thus, the values of the dual function at dual feasible points are lower bounds of the optimal value of (3.7). This provides a very valuable *quality measure* for any feasible solution x to (3.7) since $f(x) - \text{val}(3.7) \leq f(x) - D(\mu)$, for all $\mu \in \mathbb{R}^m_+$. The sharpest lower bound is given by the optimal value of the *dual problem*

$$\sup_{\mu \in \mathbb{R}^m_+} \quad D(\mu). \tag{3.9}$$

From the definition of the Lagrangian follows:

Observation 3.4 (weak duality). *It holds that* $\text{val}(3.7) - \text{val}(3.9) \geq 0$.

The expression $\text{val}(3.7) - \text{val}(3.9)$ is called *duality gap*. If problem (3.7) is nonconvex, the duality gap is usually greater than zero. However, for convex problems fulfilling a constraint qualification, the duality gap disappears (Bertsekas, 1995).

Proposition 3.5 (strong duality). *If* (3.7) *is convex and a constraint qualification (see Condition* 8.2 *in Section* 8.1*) is fulfilled, then* $\text{val}(3.7) = \text{val}(3.9)$.

In Section 4.1 several methods for solving dual problems based on evaluating the dual function D are studied. These methods are efficient if the dual function can be evaluated very fast, i.e. the Lagrangian relaxation (3.8) can be solved fast. This might be the case if (3.7) is block-separable. In this situation the Lagrangian problem (3.8) decomposes into several subproblems, which typically can be solved relatively fast.

3.4 Dual-equivalent convex relaxations

In the following, several convex relaxations are studied that are equivalent to a related dual problem. Furthermore, it is shown that dual relaxations are stronger than convex underestimating-relaxations. Consider a MINLP of the form

$$\begin{array}{lll} \min & h_0(x) & \\ \text{s.t.} & h_i(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \end{array} \tag{3.10}$$

where $h_i(x) = \sum_{k=1}^{p} h_{i,k}(x_{J_k})$. An extended reformulation of (3.10) as defined in Section 2.5 is given by:

$$\begin{array}{lll} \min & e^T t_0 & \\ \text{s.t.} & e^T t_i \leq 0, & i = 1, \ldots, m \\ & h_{i,k}(x_{J_k}) \leq t_{i,k}, & i = 0, \ldots, m,\ k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \end{array} \tag{3.11}$$

where $t_i = (t_{i,k})_{k=1,\ldots,p} \in \mathbb{R}^p$.

Lemma 3.6. *Let* (D) *and* (D_{ext}) *be the dual problems to* (3.10) *and* (3.11) *respectively. Then* $\text{val}(D) = \text{val}(D_{\text{ext}})$.

Proof. Let $X_k = \{x_{J_k} \in [\underline{x}_{J_k}, \overline{x}_{J_k}] \mid x_{B \cap J_k} \text{ binary}\}$. For the partial dual functions D_k and $D_{\text{ext},k}$ to (D) and (D_{ext}) we have

$$\begin{aligned} D_k(\mu) &= \min\{h_{0,k}(x) + \mu^T h_{1:m,k}(x) \mid x \in X_k\} \\ &= \min\{t_{0,k} + \mu^T t_{1:m,k} \mid h_{i,k}(x) \le t_{i,k},\, i = 0, \ldots, m,\, x \in X_k\} \\ &= D_{\text{ext},k}(\mu). \end{aligned}$$

Hence, the dual functions to (D) and (D_{ext}) are equivalent, thus proving the statement. □

In the same way as in the proof of Lemma 3.6 it can be shown that the dual problems to problem (3.10) and to the extended reformulation (2.4), as defined in Section 2.5, are equivalent. Consider now the extended reformulation (2.4) given in the form:

$$\min\{c^T x + c_0 \mid x \in G,\, Ax + b \le 0\} \tag{3.12}$$

where

$$G = \{x \in [\underline{x}, \overline{x}] \mid x_B \text{ binary and } g_{i,k}(x_{J_k}) \le 0,\, i \in M_k,\, k = 1, \ldots, p\}.$$

A *convex relaxation* of (3.12) is defined by

$$\min\{c^T x + c_0 \mid x \in \text{conv}(G),\, Ax + b \le 0\}. \tag{3.13}$$

The next lemma compares convex and Lagrangian relaxations.

Lemma 3.7. *Let* (D_{ext}) *be the dual of* (3.12) *and assume that for* (3.12) *a constraint qualification holds. Then* $\text{val}(D_{\text{ext}}) = \text{val}(3.13)$. *Let* $\hat{x}$ *be a solution point of the optimal Lagrangian relaxation to* (D_{ext}). *If the duality gap of* (D_{ext}) *is not zero, then* $\hat{x} \notin \text{sol}(3.13)$.

Proof. We have

$$\begin{aligned} \text{val}(D_{\text{ext}}) &= \max_{\mu \in \mathbb{R}^m_+} \min\{c^T x + c_0 + \mu^T (Ax + b) \mid x \in G\} \\ &= \max_{\mu \in \mathbb{R}^m_+} \min\{c^T x + c_0 + \mu^T (Ax + b) \mid x \in \text{conv}(G)\} \\ &= \min\{c^T x + c_0 \mid Ax + b \le 0,\, x \in \text{conv}(G)\} \\ &= \text{val}(3.13), \end{aligned}$$

where for the third equation strong duality was used. Let $\hat{x} \in G$ be a solution point of the optimal Lagrangian relaxation to (D_{ext}). If the duality gap of (D_{ext}) is not zero, then $A\hat{x} + b \not\le 0$, because otherwise $\hat{x}$ would be feasible for (3.12), and the duality gap would be zero. Since $\hat{x}$ is not feasible for (3.13), it cannot be a solution of (3.13). □

From Lemma 3.6 it follows that the dual to (3.10) and the convex relaxation (3.13) are equivalent. The next proposition shows that dual relaxations are stronger than convex underestimating-relaxations.

Lemma 3.8. *It holds that* $\text{val}(3.5) \leq \text{val}(3.13)$.

Proof. Let D^{**} and D be the dual functions to (3.5) and (3.10) respectively. Since $D^{**}(\mu) = \min_{x\in[\underline{x},\overline{x}]} h_0^{**}(x) + \sum_{i=1}^m \mu_i h_i^{**}(x)$ and $h_i^{**}(x) \leq h_i(x)$ for $x \in [\underline{x},\overline{x}]$, it follows that $D^{**}(\mu) \leq D(\mu)$ for all $\mu \in \mathbb{R}_+^m$. Hence, $\text{val}(3.5) \leq \text{val}(D)$. From Lemma 3.6 and Lemma 3.7 we have $\text{val}(\text{Dual}(3.10)) = \text{val}(3.13)$. This proves the statement. □

The following dual-equivalent convex relaxation to (3.7) is presented in (Feltenmark and Kiwiel, 2000):

$$\begin{array}{lll} \min & \displaystyle\sum_{j=1}^{n+1} z_j \cdot f(w_j) & \\ \text{s.t.} & \displaystyle\sum_{j=1}^{n+1} z_j \cdot g(w_j) \leq 0 & \\ & w_j \in G, & j = 1,\ldots,n+1 \\ & z \in \Delta_{n+1} & \end{array} \tag{3.14}$$

where $\Delta_{n+1} = \{z \in \mathbb{R}^{n+1} \mid e^T z = 1,\, z \geq 0\}$ is the standard simplex. The Lagrangian problem to (3.14) is:

$$\begin{array}{ll} \min & \displaystyle\sum_{j=1}^{n+1} z_j \cdot L(w_j;\mu) \\ & w_j \in G, \quad j = 1,\ldots,n+1 \\ & z \in \Delta_{n+1}, \end{array} \tag{3.15}$$

where $L(\cdot;\mu)$ is the Lagrangian to (3.7).

Lemma 3.9. *Let* (D) *be the dual problem to* (3.7) *and* D *be the related dual function. Then* $\text{val}(3.15) = D(\mu)$ *and* $\text{val}(D) = \text{val}(3.14)$.

Proof. The statement follows from

$$\begin{aligned} \text{val}(3.15) &= \min_{z\in\Delta_{n+1}} \sum_{j=1}^{n+1} z_j \cdot \min_{w_j\in G} L(w_j;\mu) \\ &= \min_{z\in\Delta_{n+1}} \sum_{j=1}^{n+1} z_j \cdot D(\mu) \\ &= D(\mu). \end{aligned}$$

□

Remark 3.10. A similar dual-equivalent problem can be formulated for the extended block-separable reformulation (2.4). Consider the problem:

$$\begin{aligned}
\min \quad & \sum_{k=1}^{p} \sum_{j=1}^{|J_k|+1} z_{k,j} \cdot (c_{J_k}^T w_{k,j} + c_0) \\
\text{s.t.} \quad & \sum_{k=1}^{p} \sum_{j=1}^{|J_k|+1} z_{k,j} \cdot (A_{J_k} w_{k,j} + b) \leq 0 \\
& w_{k,j} \in G_k, \quad j = 1, \ldots, |J_k| + 1, \, k = 1, \ldots, p \\
& z_k \in \Delta_{|J_k|+1}, \quad k = 1, \ldots, p
\end{aligned}$$

which can be written as:

$$\begin{aligned}
\min \quad & c^T x(w, z) + c_0 \\
\text{s.t.} \quad & Ax(w, z) + b \leq 0 \\
& w_{k,j} \in G_k, \quad j = 1, \ldots, |J_k| + 1, \, k = 1, \ldots, p \\
& z_k \in \Delta_{|J_k|+1}, \quad k = 1, \ldots, p
\end{aligned} \tag{3.16}$$

where $x(w, z) = \sum_{k=1}^{p} \sum_{j=1}^{|J_k|+1} z_{k,j} \cdot w_{k,j}$. In the same way as in the proof of Lemma 3.9, it can be shown that (3.16) is equivalent to the dual of (2.4). Based on formulation (3.16), a *column generation* method for solving dual problems of general MINLPs is presented in Section 4.3.

3.5 Reducing the duality gap

Reducing the duality gap is an important issue for improving Lagrangian and convex relaxations. This section discusses methods for reducing the duality gap by either reformulating the primal problem, or by changing the dualization, i.e. the definition of the dual problem. Consider an optimization problem of the form:

$$\begin{aligned}
\min \quad & f(x) \\
\text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m \\
& x \in G.
\end{aligned} \tag{3.17}$$

A well-known method for reducing a duality gap is the addition of valid cuts.

Lemma 3.11. *Denote by* (P') *a program that is obtained from* (3.17) *by adding the* valid cut $g_{m+1}(x) \leq 0$ *that does not change the feasible set of* (3.17)*. Then for the dual* (D') *of* (P') *we have* $\text{val}(D') \geq \text{val}(\text{Dual}(3.17))$.

Proof. Let $L(x; \mu)$ be the Lagrangian to (3.17) and let $L'(x; \mu) = L(x; \mu) + \mu_{m+1} g_{m+1}(x)$ be the Lagrangian to (P'). Then

$$\text{val}(D') = \max_{\mu \in \mathbb{R}_+^{m+1}} \min_{x \in G} L'(x; \mu) \geq \max_{\mu \in \mathbb{R}_+^{m} \times \{0\}} \min_{x \in G} L'(x; \mu) = \text{val}(\text{Dual}(3.17)). \qquad \square$$

Remark 3.12. Let (D') be defined as in Lemma 3.11. The dual-equivalent formulation (3.13) provides a tool to prove strict reduction of the duality gap, i.e. $\text{val}(D') > \text{val}(\text{Dual}(3.17))$. Denote by (C) and (C') the dual-equivalent relaxations to (Dual(3.17)) and (D') as defined in (3.13). Since $\text{val}(\text{Dual}(3.17)) = \text{val}(C)$ and $\text{val}(D') = \text{val}(C')$, we have the strict inequality $\text{val}(D') > \text{val}(\text{Dual}(3.17))$, if and only if $\text{val}(C') > \text{val}(C)$. The last inequality is fulfilled, if the inequality $g_{m+1}(x) \leq 0$ is violated for all $x \in \text{sol}(C)$. In the same way, strict reduction of the duality gap may be shown using the dual-equivalent formulation (3.14).

Examples for valid cuts of MINLPs are given in Section 7.1. The following observation shows that it is possible to close the duality gap by adding valid cuts.

Observation 3.13. *Assume that the inequality constraint* $\text{val}(3.17) - f(x) \leq 0$ *is added to* (3.17). *Then* $\text{val}(3.17) = \text{val}(\text{Dual}(3.17))$.

Proof. Choosing the Lagrangian multiplier corresponding to the inequality constraint $\text{val}(3.17) - f(x) \leq 0$ equal to 1 and setting the remaining Lagrangian multipliers 0 gives $L(x;\mu) = \text{val}(3.17)$ implying $\text{val}(\text{Dual}(3.17)) \geq \text{val}(3.17)$. Since $\text{val}(\text{Dual}(3.17)) \leq \text{val}(3.17)$, the statement is proven. □

Of course, this result is not very useful in practice since the optimal value val(3.17) is not known in advance. Consider now reformulations of (3.17) by shifting constraints to the set G.

Lemma 3.14. *Define a reformulation* (P') *of* (3.17) *by shifting the inequality constraint* $g_m(x) \leq 0$ *to* $G' = \{x \in G \mid g_k(x) \leq 0\}$. *Let* (D') *be the dual to* (P'). *Then, (i)* $\text{val}(D') \geq \text{val}(\text{Dual}(3.17))$, *and (ii)* $\text{val}(D') = \text{val}(\text{Dual}(3.17))$ *if* g_k *is convex and a constraint qualification holds.*

Proof. (i) Let $L'(\cdot;\mu)$ be the Lagrangian to (P'). We have

$$\begin{aligned} \text{val}(D') &= \max_{\mu \in \mathbb{R}_+^{m-1}} \min_{x \in G'} L'(x;\mu) \geq \max_{\mu_m \in \mathbb{R}_+} \max_{\mu \in \mathbb{R}_+^{m-1}} \min_{x \in G} L'(x;\mu) + \mu_m g_m(x) \\ &= \text{val}(\text{Dual}(3.17)). \end{aligned}$$

(ii) If g_m is convex, it follows from strong duality that the above inequality is an equality. □

In the same way, it can be proven that shifting equality constraints to G' may reduce the duality gap. The following result shows that the duality gap can be diminished by squaring linear equality constraints.

Lemma 3.15. *Define problem* (P') *by replacing a linear equality constraint* $b^T x = 0$ *of* (3.17) *by a quadratic equality constraint* $(b^T x)^2 = 0$. *Then* $\text{val}(\text{Dual}(P')) \geq \text{val}(\text{Dual}(3.17))$.

Proof. We use the dual-equivalent formulation (3.14). The constraints in (3.14) corresponding to $b^T x = 0$ and $(b^T x)^2 = 0$ read

$$\sum_{j=1}^{n+1} z_j b^T w_j = 0 \quad \text{and} \quad \sum_{j=1}^{n+1} z_j (b^T w_j)^2 = 0$$

respectively. The first constraint is equivalent to

$$\left(b^T \left(\sum_{j=1}^{n+1} z_j w_j \right) \right)^2 = 0.$$

From

$$0 = \sum_{j=1}^{n+1} z_j (b^T w_j)^2 \geq \left(\sum_{j=1}^{n+1} z_j b^T w_j \right)^2$$

the assertion follows. □

The next lemma shows that reformulating convex inequalities by semi-infinite linear constraints may increase the duality gap.

Lemma 3.16. *Define a reformulation* (P') *of* (3.17) *by replacing a convex constraint* $g_i(x) \leq 0$, $i \in \{1, \ldots, m\}$, *of* (3.17) *by the equivalent semi-infinite linear constraint*

$$\bar{g}_i(x; y) \leq 0, \quad \forall y \in \mathbb{R}^n, \tag{3.18}$$

where $\bar{g}_i(x; y) = g_i(y) + \nabla g_i(y)^T (x - y)$. *Then for the dual* (D') *to* (P') *we have* $\text{val}(\text{Dual}(3.17)) \geq \text{val}(D')$.

Proof. We use again the dual-equivalent formulation (3.14). Formulating the constraint (3.18) as in (3.14) gives

$$\sum_{j=1}^{n+1} z_j \bar{g}_i(w_j; y) = \bar{g}_i \left(\sum_{j=1}^{n+1} w_j z_j; y \right) \leq 0 \quad \forall y \in \mathbb{R}^n,$$

which is equivalent to

$$g_i \left(\sum_{j=1}^{n+1} w_j z_j \right) \leq 0.$$

Since

$$g_i \left(\sum_{j=1}^{n+1} w_j z_j \right) \leq \sum_{j=1}^{n+1} g_i(w_j) z_j \leq 0,$$

the statement follows. □

3.6 Augmented Lagrangians

The Lagrangian dual problem introduces a duality gap that might be nonzero if the problem is nonconvex. In Section 3.5 it was shown that the gap can be closed only in particular cases. A general tool for closing duality gaps are augmented Lagrangians. Consider an optimization problem with equality constraints

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & h(x) = 0 \\ & x \in G, \end{array} \tag{3.19}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $h : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $G \subseteq \mathbb{R}^n$. The *augmented Lagrangian* to (3.19) is the function

$$L_{\text{aug}}(x; \mu) = L(x; \mu) + \rho \|h(x)\|^2,$$

where $L(x; \mu) = f(x) + \mu^T h(x)$ is the ordinary Lagrangian and $\rho > 0$ is a penalty parameter. Defining the dual function

$$D_{\text{aug}}(\mu) = \min_{x \in G} L_{\text{aug}}(x; \mu)$$

the augmented dual to (3.19) reads

$$(D_{\text{aug}}) \qquad \max_{\mu \in \mathbb{R}^m} D_{\text{aug}}(\mu).$$

In (Rockafellar, 1974) the following result is proven.

Proposition 3.17. *Assume that all functions in* (3.19) *are twice continuously differentiable. Further assume that* $\text{val}(D_{\text{aug}}) > -\infty$*, G is compact and that the second-order sufficient conditions (see Proposition* 8.2 *of Section* 8.1*) hold at the global solution x^* of* (3.19) *with multipliers μ^*. Then there exists a $\bar{\rho} > 0$ such that for all $\rho \geq \bar{\rho}$, μ^* is the global solution to (D_{aug}), x^* is the global solution to (3.19) for $\mu = \mu^*$, and* $\text{val}(D_{\text{aug}}) = \text{val}(3.19)$.

Problem $\min_{x \in G} L_{\text{aug}}(x; \mu)$ is not separable. Nevertheless, it is possible to take advantage of the separability of the original problem. Grothey presents a decomposition-based approach for computing local solutions of MINLPs based on the augmented Lagrangian dual problem (D_{aug}) (Grothey, 2001).

Chapter 4

Decomposition Methods

Decomposition methods solve large scale problems by splitting them into several smaller subproblems that are coupled through a master problem. Usually, the master problem is a simple problem that can be solved in high dimensions, while the subproblems contain the complicated constraints. Decomposition of optimization problems started with the so-called *Dantzig–Wolfe decomposition* of linear programs with block-angular structure (Dantzig and Wolfe, 1960; Dantzig and Wolfe, 1961). The method is linked to the dual simplex method, which is still one of the most efficient methods for solving linear programs (Bixby, 2001).

In the following, four decomposition principles are described: dual methods, primal cutting-plane methods, column generation and Benders decomposition. The approaches differ mainly in the definition of the master problem. We do not discuss *cross decomposition* that is an integration of Lagrangian and Benders decomposition.

A main issue of this chapter is the description of a new column generation method for computing inner and outer polyhedral relaxations of general MINLPs, which makes it possible to compute and update high quality dual bounds in branch-cut-and-price methods (see Chapter 13).

4.1 Lagrangian decomposition — dual methods

Consider a block-separable optimization problem of the form

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & x \in G \end{array} \tag{4.1}$$

where the functions $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ and the set $G \subset \mathbb{R}^n$ are block-separable, i.e. there exists a partition $\{J_1, \ldots, J_p\}$ of $\{1, \ldots, n\}$ such that

$f(x) = \sum_{k=1}^{p} f_k(x_{J_k})$, $g(x) = \sum_{k=1}^{p} g_k(x_{J_k})$ and $G = \{x \in \mathbb{R}^n \mid x_{J_k} \in G_k,\ k = 1, \ldots, p\}$. Let

$$L(x; \mu) = f(x) + \mu^T g(x)$$

be the Lagrangian function to (4.1) and

$$D(\mu) = \inf_{x \in G} L(x; \mu)$$

be the related dual function. Then the Lagrangian dual problem to (4.1) reads

$$\sup_{\mu \in \mathbb{R}^m_+} D(\mu). \tag{4.2}$$

Since (4.1) is block-separable, a Lagrangian relaxation decomposes into p *partial Lagrangian problems*

$$\begin{array}{ll} \inf & L_k(x_{J_k}; \mu) \\ \text{s.t.} & x_{J_k} \in G_k \end{array} \tag{4.3}$$

where

$$L_k(x_{J_k}; \mu) = f_k(x_{J_k}) + \sum_{i=1}^{m} \mu_i g_{i,k}(x_{J_k})$$

is a *partial Lagrangian* function related to the k-th variable block. Let $D_k(\mu) = \text{val}(4.3)$ be a partial dual function. Then

$$D(\mu) = \sum_{k=1}^{p} D_k(\mu).$$

This simplification is called *Lagrangian decomposition*. It was a main motive for applying Lagrangian relaxation (Dantzig and Wolfe, 1960; Flippo and Kan, 1993; Thoai, 1997). It is mainly used in mixed-integer linear programming. The following lemma describes two properties of the dual function that are exploited in dual solution methods.

Lemma 4.1. (i) *The domain* dom D *of the dual function* D *is convex and* D *is concave over* dom D.

(ii) *Let* $\lambda \in \operatorname{dom} D$ *be a given dual point. Then for all* $x_\lambda \in \operatorname{Argmin}_{x \in G} L(x; \lambda)$ *the vector* $g(x_\lambda)$ *is a supergradient of* D *at* λ, *i.e.*

$$D(\mu) \leq D(\lambda) + g(x_\lambda)^T(\mu - \lambda), \quad \forall \mu \in \mathbb{R}^m.$$

Proof. (i) For any x, μ, λ, and $t \in [0, 1]$, we have

$$L(x; t\mu + (1-t)\lambda) = tL(x; \mu) + (1-t)L(x; \lambda).$$

Taking the infimum over all $x \in G$, we obtain

$$\inf_{x\in G} L(x; t\mu + (1-t)\lambda) \geq t \inf_{x\in G} L(x;\mu) + (1-t) \inf_{x\in G} L(x;\lambda)$$

or

$$D(t\mu + (1-t)\lambda) \geq tD(\mu) + (1-t)D(\lambda) > -\infty.$$

(ii) Let $\lambda \in \operatorname{dom} D$. It holds that

$$\begin{aligned} D(\mu) &= L(x_\mu;\mu) \\ &\leq L(x_\lambda;\mu) \\ &= f(x_\lambda) + \mu^T g(x_\lambda) \\ &= f(x_\lambda) + \lambda^T g(x_\lambda) + \mu^T g(x_\lambda) - \lambda^T g(x_\lambda) \\ &= D(\lambda) + g(x_\lambda)^T(\mu - \lambda). \end{aligned}$$

Thus

$$D(\mu) - D(\lambda) \leq g(x_\lambda)^T(\mu - \lambda).$$

This proves that $g(x_\lambda)$ is a supergradient of $D(\cdot)$ at λ. □

In the following, three dual solution methods based on function and subgradient evaluations of the dual function are discussed.

4.1.1 Subgradient methods

The simplest method for solving the dual problem (4.2) of (4.1) is the *subgradient method*. Let $\{\alpha^j\}_{j\in\mathbb{N}}$ be a sequence with $\alpha^j \geq 0$. Denote the projection of a point $\mu \in \mathbb{R}^m$ onto $\mathbb{R}^m_+$ by $\Pi(\mu)$. A subgradient algorithm computes a sequence of dual points $\{\mu^j\}$ according to Algorithm 4.1.

Choose a start vector $\mu^1 \in \mathbb{R}^m_+$.

for $j = 1, \ldots, l$

Set $g^j = g(\mu^j)$.

Set $\mu^{j+1} = \Pi\left(\mu^j - \alpha^j g^j / \|g^j\|\right)$.

end for

Algorithm 4.1: Subgradient method

Note that Algorithm 4.1 is not necessarily convergent, since the supergradient is not necessarily a direction of ascent with respect to D. It is, however, a descent direction with respect to the Euclidean distance to the set of optimal solutions. The following result concerning the convergence of $\{\mu^j\}$ is proven in (Polyak, 1993).

Proposition 4.2. *Assume that the solution set* sol(4.2) *is non-empty and bounded. Then for any sequence of step-length* $\{\alpha^j\}$ *fulfilling*

$$\alpha^j \to 0, \quad \sum_{j=1}^{\infty} \alpha^j = \infty,$$

the sequence $\{\mu^j\}$ *has all limit points in* sol(4.2)*.*

In practice, this rule gives slow convergence, and there are also theoretical results that bound the rate of convergence. Nevertheless, choosing the step-length rule according to the *divergent series rule* is a common practice (Takriti et al., 1996). Choosing the step-length according to the *geometric* series rule, $\alpha^j = q_0 q_1^j$, may yield the so-called *geometric rate of convergence* of the distance from μ^j to a solution μ^* of (4.2), but this requires careful selection of the parameters q_0, q_1 (Goffin, 1977). A very popular step-length rule is the Polyak II rule:

$$\alpha^j = \theta^j (D(\mu^j) - D_{lev}^j)/\|g^j\|$$

where D_{lev}^j is a *level* to aim at, usually an underestimate of the optimal value, val(4.1), and $0 < \delta \leq \theta^j$. Convergence of the iterates $\{\mu^j\}$ to μ^* is ensured if $D_{lev}^j \to$ val(4.2) Polyak, 1987.

4.1.2 Dual cutting-plane methods

A further development of the subgradient method is the *dual cutting-plane method* shown in Algorithm 4.2 that uses the function and subgradient information of all previous steps. In each iteration, it maximizes a polyhedral approximation of the dual function

$$\hat{D}^j(\mu) = \min\{D(\mu^i) + g(\mu^i)^T(\mu - \mu^i) \mid 1 \leq i \leq j\}. \tag{4.4}$$

Algorithm 4.2 has similar convergence properties as the subgradient method (Bertsekas, 1995).

Proposition 4.3. *Assume that* $\operatorname{argmax}_{\mu \in \mathbb{R}_+^m} \hat{D}^{j_0}(\mu) \neq \emptyset$ *for some* $j_0 \in \mathbb{N}$*, and that* $\{g(x_{\mu^j})\}$ *is a bounded sequence. Then every limit point of a sequence* $\{\mu^j\}$ *generated by Algorithm* 4.2 *is a dual-optimal point. Moreover, if the dual function* D *is polyhedral, then the dual cutting-plane method terminates finitely; that is, for some* j*,* μ^j *is a dual optimal solution.*

Choose a start vector $\mu^1 \in \mathbb{R}^m_+$.

for $j = 1, \ldots, l$

Compute $g(\mu^j)$ and $D(\mu^j)$ and set $\mu_{j+1} = \underset{\mu \in \mathbb{R}^m_+}{\operatorname{argmax}} \hat{D}^j(\mu)$.

end for

Algorithm 4.2: Dual cutting-plane algorithm

Proof. Since $g^j = g(x_{\mu^j})$ is a supergradient of D at μ^j, we have $D(\mu^j) + (\mu - \mu^j)^T g^j \geq D(\mu)$ for all $\mu \in \mathbb{R}^m_+$. Hence

$$\hat{D}^j(\mu^j) \geq \hat{D}^j(\mu) \geq D(\mu), \quad \forall \mu \in \mathbb{R}^m_+. \tag{4.5}$$

Suppose that a subsequence $\{\mu^k\}_{k \in K}$ converges to μ^*. Then, since $\mathbb{R}^m_+$ is closed, we have $\mu^* \in \mathbb{R}^m_+$, and by using the above inequality, we obtain for all k and $i < k$,

$$D(\mu^i) + (\mu^k - \mu^i)^T g^i \geq \hat{D}^k(\mu^k) \geq D^k(\mu^*) \geq D(\mu^*).$$

From the upper-semicontinuity of D it follows that

$$\lim_{i \to \infty} \sup_{i \in K} D(\mu^i) \leq D(\mu^*).$$

Since the subgradient sequence $\{g^i\}$ is bounded, we have

$$\lim_{i,k \to \infty, i,k \in K} (\mu^k - \mu^i)^T g^i = 0.$$

Hence,

$$D(\mu^*) \geq \limsup_{k \to \infty, k \in K} \hat{D}^k(\mu^k) \geq \liminf_{k \to \infty, k \in K} \hat{D}^k(\mu^k) \geq D(\mu^*)$$

and therefore $\lim_{k \to \infty, k \in K} \hat{D}^k(\mu^k) = D(\mu^*)$. From (4.5) it follows that $D(\mu^*) \geq D(\mu)$, $\forall \mu \in \mathbb{R}^m_+$. □

Remark 4.4. If (4.1) is block-separable, the polyhedral model $\hat{D}^j$ can be replaced in Algorithm 4.2 by the following modified polyhedral model :

$$\tilde{D}^j(\mu) = \sum_{k=1}^{p} \min\{\overline{D}^i_k(\mu) \mid 1 \leq i \leq j\}$$

where

$$\overline{D}^i_k(\mu) = L_k(w^i_k; \mu^i) + \sum_{l=1}^{m} g_{l,k}(w^i_k) \cdot (\mu_l - \mu^i_l)$$

and $w_k^i = \text{argmin}_{x \in G_k} L_k(x; \mu^i)$. Since $D(\mu) \leq \tilde{D}^j(\mu) \leq \hat{D}^j(\mu)$, it can be shown in the same way as in the proof of Proposition 4.3 that the resulting algorithm has the same convergence properties as Algorithm 4.2.

4.1.3 Proximal bundle methods

In this section we discuss the proximal bundle method of (Kiwiel, 1990) that uses a polyhedral model of the dual function which is penalized using a quadratic term (see Algorithm 4.3). The method generates a sequence $\{\mu^j\}_{j=1}^{\infty}$ and trial points $\lambda^j \in \mathbb{R}_+^m$ for evaluating supergradients $g^j = g(x_{\lambda^j})$ of D and its linearizations

$$\bar{D}^j(\mu) = D(\lambda^j) + (g^j)^T(\mu - \lambda^j) \geq D(\mu),$$

starting from an arbitrary point $\mu^1 = \lambda^1 \in \mathbb{R}_+^m$. Iteration j uses the polyhedral model

$$\hat{D}^j(\mu) = \min\{\bar{D}^i(\mu) \mid i \in J^j\}$$

with $J^j \subset \{1, \ldots, j\}$ for finding

$$\lambda^{j+1} = \text{argmax}\{\hat{D}^j(\mu) - \frac{u^j}{2}\|\mu - \mu^j\|^2 \mid \mu \in \mathbb{R}_+^m\} \tag{4.6}$$

where $u^j > 0$ is a proximity weight. An ascent step $\mu^{j+1} = \lambda^{j+1}$ occurs if λ^{j+1} is significantly better than μ^j measured by

$$D(\lambda^{j+1}) \geq D(\mu^j) + \kappa \cdot (\bar{D}^j(\lambda^{j+1}) - D(\mu^j))$$

where $\kappa \in (0, 1)$ is a fixed Armijo-like parameter. Otherwise, a null step $\mu^{j+1} = \mu^j$ improves the next model $\hat{D}^{j+1}$ with the new linearization $\bar{D}^{j+1}$.

The following convergence result of Algorithm 4.3 is proven in (Kiwiel, 1990):

Proposition 4.5. *Either* $\mu^j \to \bar{\mu} \in \text{sol}(4.2)$ *or* $\text{sol}(4.2) = \emptyset$ *and* $\|\mu^j\| \to +\infty$. *In both cases* $D(\mu^j) \uparrow \sup_{\mathbb{R}_+^m} D$.

Remark 4.6. Dual iteration points of the proximal bundle algorithm are related to solutions of the convexified problem (3.14) defined in Section 3.4. Let x_{μ^j} be the solution of the Lagrangian problem for computing a subgradient $g^j = g(x_{\mu^j})$ and z^j be the dual solution of the quadratic program (4.6). In (Feltenmark and Kiwiel, 2000) it is proven that each accumulation point of the sequence (z^j, x_{μ^j}) generated by Algorithm 4.3 solves the dual-equivalent convex program (3.14). The result is particularly interesting in the context of Lagrangian heuristics. If the original problem is a linear program, then $\bar{x}^j = \sum_{i \in J^j} z^i x_{\mu^i}$ is an estimate for a primal solution point.

Parameters: $u^1 > 0$ and $\kappa \in (0,1)$

Choose a start vector $\mu^1 \in \mathbb{R}^m_+$, set $\lambda^1 = \mu^1$.

Compute $D(\mu^1)$ and a supergradient g^1 of D at μ^1.

for $j = 1, \ldots, l$

Solve (4.6) obtaining λ^{j+1}.

if $\hat{D}^j(\lambda^{j+1}) - D(\mu^j) < \epsilon$: **stop**.

if $D(\lambda^{j+1}) \geq D(\mu^j) + \kappa \cdot (\bar{D}^j(\lambda^{j+1}) - D(\mu^j))$:
Set $\mu^{j+1} = \lambda^{j+1}$ (serious step).

else: Set $\mu^{j+1} = \mu^j$ (null-step).

Compute $D(\mu^j)$ and a supergradient g^j of D at μ^j.

Choose $J^{j+1} \subset J^j \cup \{j+1\}$ and update u^{j+1}.

end for

Algorithm 4.3: Proximal bundle algorithm

If the number of coupling constraints is high it is important that the box-constrained quadratic program (4.6) is solved efficiently. For this, a direct method, as in (Kiwiel, 1994b; Kiwiel, 1994a), or an iterative method, such as the conjugate gradient method, can be used. A simplified bundle method that makes use of near optimal solutions of a Lagrangian relaxation is described in (Zhao and Luh, 2002). A good overview of bundle methods is given in (Hiriart-Urruty and Lemaréchal, 1993).

4.2 Primal cutting-plane methods

Primal cutting-plane methods solve a dual-equivalent convex relaxation by generating a polyhedral outer approximation of the feasible set, which is improved successively by adding valid cuts. Consider the following dual-equivalent semiinfinite formulation:

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & a^T x_{J_k} \leq \sigma_{G_k}(a), \quad a \in \mathbb{S}^{|J_k|},\ k = 1, \ldots, p, \end{array} \tag{4.7}$$

where $\mathbb{S}^n$ is the n-dimensional sphere and $\sigma_{G_k}(a) = \max_{x \in G_k} a^T x_{J_k}$ is the support

function. Related to (4.7) a *linear master program* is defined:

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & a^T x_{J_k} \leq \overline{a}, \quad (a, \overline{a}) \in N_k^j,\ k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}] \end{array} \tag{4.8}$$

where $N_k^j \subset \mathbb{S}^{|J_k|} \times \mathbb{R}$ is a finite set. The cutting-plane method described in Algorithm 4.4 generates in the j-th iteration *valid cuts* of the form

$$a^T x_{J_k} \leq \overline{a}, \tag{4.9}$$

where $k \in \{1, \ldots, p\}$, $(a, \overline{a}) \in \mathbb{S}^{|J_k|} \times \mathbb{R}$ and $\overline{a} \geq \sigma_{G_k}(a)$.

Set $N_k^1 = \emptyset$, $k = 1, \ldots, p$.

for $j = 1, \ldots, l$

Compute a solution x^j of (4.8).

Update N_k^{j+1} by adding cutting-planes of the form (4.9) to N_k^j for $k = 1, \ldots, p$.

end for

Algorithm 4.4: Cutting-plane algorithm

For the next convergence result the distance of point $x \in \mathbb{R}^{|J_k|}$ to the set G_k is defined by:

$$\text{dist}_k(x) = \max\{a^T x - \sigma_{G_k}(a) \mid a \in \mathbb{S}^{|J_k|}\}.$$

Proposition 4.7. *Let $\{x^j\}$ be a sequence generated by Algorithm 4.4. Define the maximum violation related to the constraints of the k-th block at iteration $j+1$ at x^j by*

$$d_k^j = \{0, \max_{(a,\overline{a}) \in N_k^{j+1}} a^T x_{J_k}^j - \sigma_{G_k}(a)\}.$$

If there exists $\delta > 0$ such that, for each $i \in \mathbb{N}$ and $k \in \{1, \ldots, p\}$, there exists $j \geq i$ with $d_k^j \geq \delta \cdot \text{dist}_k(x_{J_k}^j)$, then $\{x^j\}$ converges towards a solution of (4.7).

Proof. Since $\{x^j\}$ is bounded, there exists a subsequence of $\{x^j\}$ converging to a point $\bar{x}$. From the above assumption it follows that $\text{dist}_k(x_{J_k}^j) \to 0$ for $j \to \infty$ and $k \in \{1, \ldots, p\}$. Hence, $\bar{x}$ is feasible for (4.7) showing $c^T \bar{x} + c_0 \geq \text{val}(4.7)$. From $\text{val}(4.8) \leq \text{val}(4.7)$ it follows that $c^T \bar{x} + c_0 \leq \text{val}(4.7)$. This proves the statement. □

Remarks.

1. Problem (4.7) is a *linear semi-infinite program* (SIP) and can be solved by any SIP method. For example, in (Reemtsen, 1994) a Kelley–Cherney–Goldstein (KCG) cutting-plane algorithm is proposed that adds, in the j-th iteration, the most violated constraint to the master program. However, the determination of the most violated constraint for problem (4.7) is a nonconvex MINLP with respect to the variables $(x_{J_k}, a) \in \mathbb{R}^{|J_k|} \times \mathbb{R}^{|J_k|}$, which can be very difficult to solve.

2. The following example shows that Algorithm 4.4 does not converge necessarily towards a solution of the dual problem (4.2), if the following Lagrangian cuts are used:
$$L_k(x; \mu^j) \geq D_k(\mu^j),$$
where $D_k(\mu) = \min_{x \in G_k} L_k(x; \mu)$ is the k-th partial dual function,
$$L_k(x_{J_k}; \mu) = (c_{J_k} + A_{J_k}^T \mu)^T x_{J_k}$$
is the k-th partial Lagrangian to (4.7), and μ^j is a dual solution point of the master problem (4.8).

   Example 4.8. Consider the optimization problem:
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & a^T x = 0 \\ & x \in G \subset [\underline{x}, \overline{x}] \subset \mathbb{R}^2, \end{array} \tag{4.10}$$
   where $c, a \in \mathbb{R}^2$ are linearly independent. A polyhedral relaxation of this problem with one Lagrangian cut is defined by
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & a^T x = 0 \\ & L(x; \mu^1) \geq D(\mu^1) \end{array} \tag{4.11}$$
   where $L(x; \mu) = c^T x + \mu a^T x$ is a Lagrangian, $D(\mu)$ is the related dual function to (4.10), and μ^1 is an arbitrary dual point with $D(\mu^1) < \text{val(Dual(4.10))}$. Let (μ^2, τ) be a dual solution to (4.11), where μ^2 corresponds to the first and τ to the second constraint of (4.11). From the first-order optimality condition for (4.11) it follows that:
$$c + \mu^2 a + \tau(c + \mu^1 a) = 0.$$
   Hence, $\tau = -1$ and $\mu^2 = \mu^1$, which shows that the Lagrangian cut $L(x; \mu^2) \geq D(\mu^2)$ is equivalent to the given Lagrangian cut and does not improve (4.11).

3. Instead of using solution points of the master problem (4.8) as trial points, it is possible to use centers. This leads to so-called central cutting-plane methods (Elzinga and Moore, 1975; Goffin and Vial, 1999).

4. It is possible to use a MIP master program, instead of a LP master program (Duran and Grossmann, 1986). In (Quesada and Grossmann, 1992) it is shown that the MIP master program can be solved efficiently by updating a branch-and-bound tree.

5. Consider a nonlinear convex relaxation of the form

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in \text{conv}(G_k), \quad k = 1, \ldots, p, \end{array} \tag{4.12}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a nonlinear convex function that is not block-separable. Problem (4.12) can be solved by a decomposition-based cutting-plane algorithm via the NLP master problem:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax + b \leq 0 \\ & a^T x_{J_k} \leq \overline{a}, \quad (a, \overline{a}) \in N_k^j,\ k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}] \end{array} \tag{4.13}$$

where $(a, \overline{a})$ is defined as in (4.9). The convergence of such an algorithm can be shown under the same assumptions as in Proposition 4.7.

4.3 Column generation

This section describes a *column generation method*, also called *Dantzig–Wolfe decomposition method*, for solving the dual-equivalent convex relaxation:

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in \text{conv}(G_k), \quad k = 1, \ldots, p \end{array} \tag{4.14}$$

by producing inner and outer approximations of $\text{conv}(G_k)$. This technique has three main advantages: (i) It is possible to fix Lagrangian subproblems that are 'explored'; (ii) It is possible to work with near-optimal solutions of Lagrangian subproblems; (iii) It is easy to update relaxations after branching operations in branch-cut-and-price algorithms. As a result, the method makes it possible to compute and update dual bounds in branch-and-bound algorithms efficiently.

4.3.1 A simple column generation method

In the following, a simple *column generation method* for solving (4.14) is described that subsequently produces inner approximations of $\text{conv}(G_k)$. Let $\overline{W}_k = \text{vert}(G_k)$

be the extreme points of the set G_k, and $\overline{W} = (\overline{W}_1, \ldots, \overline{W}_p)$. From Observation 3.2 (Section 3.1) we know that

$$\operatorname{conv}(G_k) = \{ \sum_{w \in \overline{W}_k} z_w \cdot w \mid \sum_{w \in \overline{W}_k} z_w = 1,\ z_w \geq 0,\ w \in \overline{W}_k \}.$$

For a finite set $W = (W_1, \ldots, W_p)$ with $W_k \subset \mathbb{R}^{|J_k|}$ and a point $z \in \times_{k=1}^{p} \mathbb{R}^{|W_k|}$, the product $x = W \bullet z$ is defined by $x_{J_k} = \sum_{w \in W_k} z_w \cdot w$. Replacing x with $\overline{W} \bullet z$ in (4.14) yields the following *extensive formulation*, which is equivalant to (4.14):

$$\begin{array}{ll} \min & c^T \overline{W} \bullet z + c_0 \\ \text{s.t.} & A\overline{W} \bullet z + b \leq 0 \\ & \sum_{w \in \overline{W}_k} z_w = 1, \quad k = 1, \ldots, p \\ & z \geq 0. \end{array} \tag{4.15}$$

Since it is usually too difficult to solve the extensive formulation (4.15), if the number of extreme points $|\overline{W}_k|$ is very large, the following *restricted master problem* (RMP) is considered:

$$\begin{array}{ll} \min & c^T W^j \bullet z + c_0 \\ \text{s.t.} & AW^j \bullet z + b \leq 0 \\ & \sum_{w \in W_k^j} z_w = 1, \quad k = 1, \ldots, p \\ & z \geq 0 \end{array} \tag{4.16}$$

where $W_k^j \subseteq \operatorname{conv}(G_k)$ and $W^j = (W_1^j, \ldots, W_p^j)$ is a finite set. The elements of W_k^j are called *inner approximation points* (see Figure 4.1). The set W^j is called *admissible* if the related RMP (4.16) is feasible.

Observation 4.9. *From Remark* 3.10 *(Section* 3.4*) it follows that there exist finite sets $W_k^* \subseteq \overline{W}_k$ with $|W_k^*| \leq |J_k| + 1$ such that $\overline{W}$ can be replaced with $W^* = (W_1^*, \ldots, W_p^*)$ without changing the optimal value of* (4.15)*. Hence, if $W_k^* \subseteq W_k^j$ for $k = 1, \ldots, p$, then* (4.14) *and* (4.16) *are equivalent.*

Algorithm 4.5 describes a column generating method that alternatively solves the RMP (4.16) and the Lagrangian subproblems:

$$\min\{L_k(x; \mu^j) \mid x \in G_k\} \tag{4.17}$$

for $k = 1, \ldots, p$.

The next lemma shows that dual cutting-plane and column generation methods are equivalent.

Lemma 4.10. *Let μ^j be the optimal dual point of the RMP* (4.16) *and $\tilde{D}^j$ be the polyhedral model defined in Remark* 4.4 *used in the j-th iteration of the dual cutting-plane method described in Algorithm* 4.2*. Then $\mu^j = \operatorname{argmax}_{\mu \in \mathbb{R}_+^m} \tilde{D}^j(\mu)$.*

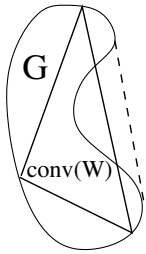

Figure 4.1: Inner approximation of conv(G)

Initialize $W_k^1 \subset \text{conv}(G_k)$, $k \in \{1, \ldots, p\}$, such that (4.16) is feasible.

for $j = 1, \ldots, l$

Compute dual and primal solutions μ^j and z^j of (4.16).

for $k = 1, \ldots, p$

Compute a solution w_k of (4.17).

Set $W_k^{j+1} = W_k^j \cup \{w_k\}$.

end for

end for

Algorithm 4.5: Column generation method

Proof. The polyhedral model $\tilde{D}^j$ can be written in the form

$$\tilde{D}^j(\mu) = c_0 + \mu^T b + \sum_{k=1}^{p} \min\{\overline{D}_k^i(\mu) \mid 1 \le i \le j\}$$

where $\overline{D}_k^i(\mu) = c_{J_k}^T w_k^i + \mu^T A_{J_k} w_k^i$, w_k^i is a solution point of the k-th Lagrangian subproblem at the i-th iteration of Algorithm 4.2, and $A_{J_k} = (a_{ij})_{i \in \{1,\ldots,n\}, j \in J_k}$. Then

$$\begin{aligned}
\max_{\mu \in \mathbb{R}_+^m} \tilde{D}^j(\mu) &= \max_{\mu \in \mathbb{R}_+^m} c_0 + \mu^T b + \sum_{k=1}^{p} \min_{z_k \in \Delta_j} \sum_{i=1}^{j} z_{k,i} (c_{J_k}^T w_k^i + \mu^T A_{J_k} w_k^i) \\
&= \max_{\mu \in \mathbb{R}_+^m} \min_{z_1, \ldots, z_p \in \Delta_j} c^T W^j \bullet z + c_0 + \mu^T (A W^j \bullet z + b) \\
&= \min_{z_1, \ldots, z_p \in \Delta_j} \{c^T W^j \bullet z + c_0 \mid A W^j \bullet z + b \le 0\} \\
&= \text{val}(4.16)
\end{aligned}$$

□

From the convergence of the dual cutting-plane method (Proposition 4.3) follows:

Proposition 4.11. *Algorithm* 4.5 *generates a sequence* $\{x^j\} = \{W^j \bullet z^j\}$ *that converges finitely towards an* ϵ*-solution of* (4.14).

Proof. Since W^1 is admissible, W^j is admissible for $j \geq 1$. Hence, the RMP (4.16) is always feasible, and from Lemma 4.10 and Proposition 4.3 the statement follows. □

4.3.2 Initializing the RMP

Algorithm 4.5 has to be initialized by admissible inner approximation points $W_k^1 \subset \text{conv}(G_k)$ such that the RMP (4.16) is feasible, i.e.

$$\text{conv}(W_k^1) \cap \{x_{J_k} \mid Ax + b \leq 0\} \neq \emptyset \tag{4.18}$$

for $k \in \{1, \ldots, p\}$. In general, finding points that fulfill (4.18) is a non-trivial task. Algorithm 4.6 and Algorithm 4.7 describe a two-phase procedure for initializing inner approximation points.

Input: a point $x \in [\underline{x}, \overline{x}]$

for $k = 1, \ldots, p$

Set $W_k^1 = \emptyset$ and $K = J_k \setminus \{i : x_i \in \{\underline{x}_i, \overline{x}_i\}\}$.

while $K \neq \emptyset$

Set $w = (\text{round}(x, K))_{J_k}$ and $W_k^1 = W_k^1 \cup \{w\}$.

Set $x = w + t(x - w)$ such that x is on a j-face of $[\underline{x}_{J_k}, \overline{x}_{J_k}]$ with $j \in K$.

Set $K = K \setminus \{i : x_i \in \{\underline{x}_i, \overline{x}_i\}\}$.

end while

Set $W_k^1 = W_k^1 \cup \{x\}$.

end for

Algorithm 4.6: Computing admissible vertices

Finding admissible vertices

In the first phase, described in Algorithm 4.6, vertices of the interval $[\underline{x}_{J_k}, \overline{x}_{J_k}]$ are computed whose convex hull contains a given trial point $\hat{x}_{J_k}$, where $\hat{x}$ fulfills the

coupling constraints $A\hat{x} + b \leq 0$. The algorithm uses the rounding operation with respect to an index set $K \subseteq \{1, \ldots, n\}$ defined by

$$\text{round}(x, K)_i = \begin{cases} \overline{x}_i & \text{for } i \in K \text{ and } x_i > 0.5(\underline{x}_i + \overline{x}_i) \\ \underline{x}_i & \text{for } i \in K \text{ and } x_i \leq 0.5(\underline{x}_i + \overline{x}_i) \\ x_i & \text{else.} \end{cases}$$

In each iteration of the algorithm a point x is computed that is a convex combination of the previous point x and the current inner approximation point w. From this follows:

Lemma 4.12. *For a given point $x \in [\underline{x}, \overline{x}]$ Algorithm 4.6 computes vertices $W_k^1 \subseteq \text{vert}([\underline{x}_{J_k}, \overline{x}_{J_k}])$ with $|W_k^1| \leq |J_k| + 1$ and $x_{J_k} \in \text{conv}(W_k^1)$.*

If $\text{vert}(G_k) = \text{vert}([\underline{x}_{J_k}, \overline{x}_{J_k}])$, such as in unconstrained binary programming, Algorithm 4.6 generates admissible inner approximation points.

The general case

In the general case, the following *constraint satisfaction* problem has to be solved:

$$\begin{array}{ll} \text{find} & (z, W) \\ \text{s.t.} & AW \bullet z + b \leq 0 \\ & W_k \subset \text{conv}(G_k), \quad k = 1, \ldots, p \end{array}$$

where $z \in \times_{k=1}^{p} \mathbb{R}^{|J_k|+1}$, $W = (W_1, \ldots, W_p)$ and $|W_k| = |J_k| + 1$. Since solving this problem directly by using a constraint satisfaction method may be too difficult, Algorithm 4.7 is used. This method uses an auxiliary LP to find new inner approximation points. In addition, the following polyhedral outer approximation is generated:

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in \hat{G}_k, \quad k = 1, \ldots, p \end{array} \tag{4.19}$$

where $\hat{G}_k \supseteq G_k$ is a polyhedron. There are three reasons for using (4.19). First, it is used as an LP-relaxation in branch-and-bound methods (see Chapter 13). Second, it helps to solve Lagrangian subproblems. And finally, it is used to check the feasibility of problem (4.14) by checking the feasibility of (4.19).

Let $W_k^1 \subset [\underline{x}_{J_k}, \overline{x}_{J_k}]$ be an arbitrary initial set of inner approximation points. The points of W_k^1 can be generated by Algorithm 4.6, they can be solutions of Lagrangian subproblems, or they can be computed in a branching operation of a BCP-Algorithm, as described in Section 13.4.2. At the beginning of the procedure, the points w of W_k^1 that are infeasible are projected onto G_k in the direction of $w - \overline{w}_k$, where $\overline{w}_k = \frac{1}{|W_k^1|} \sum_{w \in W_k^1} w$ is the midpoint of $\text{conv}(W_k^1)$. This is done by solving the subproblem:

$$\max_x \{(w - \overline{w}_k)^T x \mid w \in G_k\}. \tag{4.20}$$

The resulting inner approximation points might not be admissible, since it is possible that some inner approximation points are identical after the projection onto G_k. In this case, the method tries to find a point that fulfills the coupling constraints $Ax + b \leq 0$ and which has a minimum distance from the convex hull of the current inner approximation points W^1. For this, the following auxiliary LP is solved:

$$\begin{array}{lll} \min & s^T y & \\ \text{s.t.} & A(W^1 \bullet z + y) + b \leq 0 & \\ & W^1 \bullet z + y \in \hat{G}_k, & k = 1, \ldots, p \\ & \sum_{w \in W_k^1} z_w = 1, & k = 1, \ldots, p \\ & s_i y_i \geq 0, & i = 1, \ldots, n \\ & z \geq 0 & \end{array} \tag{4.21}$$

where $y \in \mathbb{R}^n$ and the sign vector $s \in \{-1, 1\}^n$ is defined by

$$s_i = \operatorname{sign}(\hat{x}_i - \overline{w}_i), \tag{4.22}$$

$\overline{w}_{J_k} = \frac{1}{|W_k^1|} \sum_{w \in W_k^1} w$ and $\hat{x}$ is a solution of (4.19). Since $\overline{w} = W^1 \bullet \overline{z}$ with $\overline{z}_w = \frac{1}{|W_k^1|}$ for $w \in W_k^1$, it follows that the point $(y, \overline{z})$ with $y = \hat{x} - \overline{w}$ is feasible for (4.21), if (4.19) is feasible.

Let (y, z) be a solution of (4.21). Then $x = W^1 \bullet z + y$ is feasible for (4.19) and has a minimum distance from the convex hull of W^1. If $y = 0$, W^1 is admissible.

If $y_{J_k} \neq 0$, we have $\operatorname{conv}(W_k^1) \cap \{x_{J_k} \mid Ax + b \leq 0\} = \emptyset$. In this case, a cut that separates $\hat{x}_{J_k}$ and $\operatorname{conv}(W_k^1)$ is constructed for finding a new inner approximation point or for checking whether (4.14) is feasible. Let $\tilde{w}_k$ be the solution of

$$\min\{\|w - \hat{x}_{J_k}\|^2 \mid w \in \operatorname{conv}(W_k^1)\}$$

where $\hat{x}$ is a solution of (4.19). Note that $\tilde{w}_k$ can be computed by $\tilde{w}_k = W_k^1 \bullet \tilde{z}_{I_k}$, where $\tilde{z}_{I_k}$ is a solution of the quadratic program:

$$\begin{array}{ll} \min & \|W_k \bullet z_{I_k} - \hat{x}_{J_k}\|^2 \\ & z_{I_k} \in \Delta_{|I_k|}. \end{array} \tag{4.23}$$

Define $a_k = \hat{x}_{J_k} - \tilde{w}_k$ and $d_k = \frac{1}{2}(\hat{x}_{J_k} + \tilde{w}_k)$. Then the hyperplane

$$a_k^T (x - d_k) = 0 \tag{4.24}$$

separates $\hat{x}_{J_k}$ and $\operatorname{conv}(W_k^1)$, i.e. $a_k^T(\hat{x}_{J_k} - d_k) > 0$ and $a_k^T(x - d_k) < 0$ for all $x \in \operatorname{conv}(W_k^1)$.

In order to find a new inner approximation point, the following subproblem is solved (see Figure 4.2):

$$\max\{a_k^T x \mid a_k^T (x - d_k) \geq 0,\, x \in G_k\}. \tag{4.25}$$

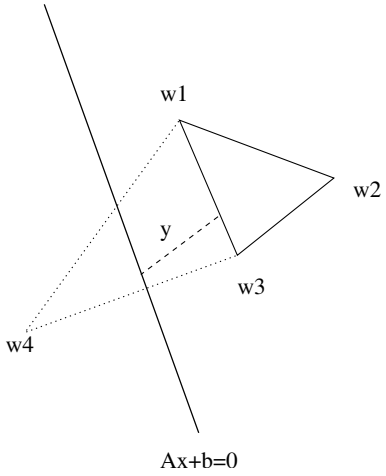

Figure 4.2: Finding a new admissible inner approximation point

If (4.25) is feasible, a solution w_k of (4.25) is added to W_k^1, and the cut

$$a_k^T x \leq \text{val}(4.25) \tag{4.26}$$

is added to $\hat{G}_k$. Otherwise, the cut

$$a_k^T (x - d_k) \leq 0 \tag{4.27}$$

is added to $\hat{G}_k$. The procedure is repeated as long as either W^1 is admissible, i.e. $y = 0$, or (4.19) is infeasible.

Remark 4.13. For finding initial admissible inner approximation points, it is not necessary to solve the subproblems (4.25) exactly. It is sufficient to find a feasible point $w \in G_k$ and an upper bound of the optimal value of (4.25).

Remark 4.14. Algorithm 4.7 works well in practice, but it does not guarantee to find always admissible inner approximation points. The initialization problem can be simplified by considering the following block-separable reformulation of the given MINLP (2.1) with additional slack variables $y \in \mathbb{R}^p$:

$$\begin{array}{lll} \min & c^T x + c_0 + \delta e^T y & \\ & Ax + b \leq 0 & \\ & g_{i,k}(x_{J_k}) \leq y_k, & i \in M_k,\ k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \\ & y \geq 0 & \end{array} \tag{4.28}$$

where the penalty parameter $\delta > 0$ is sufficiently large. Let $\hat{x}$ be a trial point

fulfilling the coupling constraints. Furthermore, choose $W_k \subset [\underline{x}_{J_k}, \overline{x}_{J_k}]$ such that $x_{J_k} \in \mathrm{conv}(W_k)$, for example, by using Algorithm 4.6. Then the sets

$$\tilde{W}_k = \{(w, y_w^k) \mid w \in W_k\},$$

where $y_w^k = \max\{0, \max_{i \in M_k} g_{i,k}(w)\}$ are admissible inner approximation points for (4.28).

Input: Initial inner approximation points $W_k^1 \subset [\underline{x}_{J_k}, \overline{x}_{J_k}]$, $k \in \{1, \ldots, p\}$

Initialize $\hat{G}_k \supseteq G_k$ in (4.19), for example, by using Algorithm 7.1.

for $k = 1, \ldots, p$ and $w \in W_k^1 \setminus G_k$: Project w onto G_k by solving (4.20).

Compute a solution $\hat{x}$ of (4.19).

repeat

Initialize the vector s of (4.21) as in (4.22) w.r.t. $\hat{x}$.

Compute dual and primal solutions μ, y and z of (4.16), and delete inner approximation points $w \in W_k^1$ if $z_w = 0$.

if $y = 0$: **stop** (W^1 is admissible).

Compute a solution $\hat{x}$ of (4.19).

if (4.19) is not feasible: **stop** ((4.14) is not feasible).

for $k = 1, \ldots, p$: **if** $y_{J_k} \neq 0$:

if (4.25) is feasible: Add a solution w_k of (4.25) to W_k^1, and add the cut (4.26) to $\hat{G}_k$.

else: Add the cut (4.27) to $\hat{G}_k$.

end for

end repeat

Algorithm 4.7: Initialization of admissible inner approximation points

4.3.3 An improved column generation method

In decomposition methods, there should be a possibility to check the quality of an inner or outer approximation of a subproblem by computing an approximation

error. If the approximation error is relatively small, the related subproblem will not be solved for a while. Such a mechanism prevents the decomposition method from generating identical Lagrangian solutions repeatedly. This is particularly important if the Lagrangian subproblems are difficult to solve, which is often the case in MINLP. Moreover, the approximation error can be used as a stopping criterion.

In primal cutting-plane methods, it is possible to check if a subproblem is 'explored' by computing the partial dual function related to the master problem (4.8):

$$\hat{D}_k(\mu^j) = \min\{L_k(x;\mu^j) \mid a^T x \leq \overline{a}, (a,\overline{a}) \in N_k^j, x \in [\underline{x}_{J_k}, \overline{x}_{J_k}]\}.$$

It holds that

$$\hat{D}_k(\mu^j) \leq D_k(\mu^j) = \min_{x \in G_k} L_k(x;\mu^j).$$

If $\hat{D}_k(\mu^j) = D_k(\mu^j)$, a subproblem is said to be explored, and a cutting-plane method could stop (for a while) to generate cuts for this subproblem. However, this test requires some extra computational effort, since the evaluation of the dual function comes at a price.

In column generation, a similar test can be performed without any extra computational effort. Consider the partial dual function related to the RMP (4.16) given by:

$$\begin{aligned} \check{D}_k(\mu) &= \min\{L_k((W^j \bullet z)_{J_k};\mu) \mid \sum_{w \in W_k^j} z_w = 1,\, z_{W_k^j} \geq 0\} \\ &= \min\{L_k(x;\mu) \mid x \in \operatorname{conv}(W_k^j)\}. \end{aligned}$$

We define the k-th *reduced cost* by

$$R_k(\mu) = D_k(\mu) - \check{D}_k(\mu). \tag{4.29}$$

The related Lagrangian problem is called the *pricing problem*. Since $R_k(\mu) \leq 0$, the optimal value of a Lagrangian subproblem of the restricted master-problem (4.16) is an upper bound of the optimal value of the Lagrangian subproblem of the original problem (4.14). By using reduced costs, it is possible to check if a subproblem is explored, and to stop generating columns for this subproblem for a while. Moreover, it holds that

$$e(\hat{\mu}^j) = -\sum_{k=1}^{p} R_k(\hat{\mu}^j) \leq \text{val}(4.16) - \text{val}(4.14) \geq 0,$$

where μ^j is a dual solution point to (4.16). Hence, the column generation method can be stopped if $e(\mu^j)$ is small enough.

Algorithm 4.5 describes an improved column generation method that fixes 'explored' subproblems according to the above considerations. In order to accelerate the computation of Lagrangian solutions, the method updates the polyhedral outer approximation (4.19).

Proposition 4.15. *Algorithm* 4.8 *generates a sequence* $\{x^j\} = \{W^j \bullet z^j\}$ *that converges finitely towards an* ϵ*-solution of* (4.14).

Proof. Denote by $\{\mu^j\}$ the subsequence of dual solution points of (4.16) where the complete dual function D is evaluated, i.e. $K_{\text{free}} = \{1, \ldots, p\}$. Note that D is evaluated after at least $r_{\max}$ iterations. Similar as in Lemma 4.10, we define the corresponding polyhedral cutting-plane model by

$$\check{D}^j(\mu) = c_0 + \mu^T b + \sum_{k=1}^{p} \min\{\overline{D}_k^i(\mu) \mid i \in K_k^j\}$$

where $\overline{D}_k^i(\mu) = c_{J_k}^T w_k^i + \mu^T A_{J_k} w_k^i$, the point w_k^i is a minimizer of the k-th Lagrangian subproblem of the i-th iteration of Algorithm 4.8, and K_k^j are the iteration numbers $\leq j$ where the k-th Lagrangian subproblem was solved. Since

$$D(\mu) \leq \check{D}^j(\mu) \leq \tilde{D}^j(\mu) \leq \hat{D}^j(\mu),$$

where $\tilde{D}^j$ is defined as in Lemma 4.10 and $\hat{D}^j$ is defined as in Proposition 4.11, the convergence of Algorithm 4.8 can be proved in the same way as in Proposition 4.11. □

Input: inner approximation points $W_k^1 \subset \text{conv}(G_k)$ and polyhedra $\hat{G}_k \supset G_k$, $k \in \{1, \ldots, p\}$

Set $K_{\text{free}} = \{1, \ldots, p\}$ and $r = 0$.

for $j = 1, \ldots, l$

Compute dual and primal solutions μ^j and z^j of (4.16).

Delete $w \in W^j$ if $z_w^j = 0$ and set $W^{j+1} = W^j$.

for $k \in K_{\text{free}}$:

Compute a solution w_k of (4.17) using $\hat{G}_k$ and add a related Lagrangian cut to $\hat{G}_k$.

Set $W_k^{j+1} = W_k^{j+1} \cup \{w_k\}$ and $r = r + 1$.

if $R_k(\mu^j)$ is small **or** $w_k \in W_k^j$: Set $K_{\text{free}} = K_{\text{free}} \setminus \{k\}$.

if $K_{\text{free}} = \emptyset$ **or** $r = r_{\max}$: Set $K_{\text{free}} = \{1, \ldots, p\}$ and $r = 0$.

end for

Algorithm 4.8: Improved column generation method

Remark 4.16. Note that Lagrangian solutions that are computed by a subgradient type algorithm can be added to the initial sets W_k^1, $k = 1, \ldots, p$. The resulting method is a hybrid subgradient column generation method.

Remark 4.17. The optimal value of the polyhedral outer approximation (4.19) that is generated by Algorithm 4.8 can be used as a lower bounding method. Moreover, an LP-estimate for the reduced cost $R_k(\mu)$ is given by $\tilde{R}_k(\mu) = \check{D}_k(\mu) - \hat{D}_k(\mu)$, where $\hat{D}_k$ and $\check{D}_k$ are the k-th partial dual functions to the polyhedral outer and inner approximations (4.19) and (4.16) respectively. If $\tilde{R}_k(\mu) = 0$ then $R_k(\mu) = 0$. Hence, $\tilde{R}_k(\mu)$ can be used to detect explored subproblems. The use of $\tilde{R}_k(\mu)$ makes it possible to generate and update a polyhedral outer approximation without solving all subproblems. This may be more efficient than evaluating the (complete) dual function (see Sections 7.3.4 and 13.4.2).

4.4 Benders decomposition

In Benders decomposition, it is assumed that after fixing some (coupling) variables of an optimization problem, the resulting problem is decomposed into subproblems. Consider an optimization problem of the form:

$$\text{(P)} \qquad \begin{array}{ll} \min & \displaystyle\sum_{k=1}^{p} f_k(x_{J_k}, y) \\ \text{s.t.} & g_k(x_{J_k}, y) \le 0, \quad k = 1, \ldots, p \\ & y \in Y \end{array}$$

where Y is an arbitrary set. Related to (P), we define the subproblem with a fixed y-variable, $\hat{y} \in Y$:

$$\text{(P}[\hat{y}]\text{)} \qquad \begin{array}{ll} \min & \displaystyle\sum_{k=1}^{p} f_k(x_{J_k}, \hat{y}) \\ \text{s.t.} & g_k(x_{J_k}, \hat{y}) \le 0, \quad k = 1, \ldots, p. \end{array}$$

Let $v(y) = \text{val}(P[y])$ be the optimal value function to $(P[\hat{y}])$ and define the k-th subproblem

$$\text{(P}_k[\hat{y}]\text{)} \qquad \begin{array}{ll} \min & f_k(x_{J_k}, y) \\ \text{s.t.} & g_k(x_{J_k}, y) \le 0. \end{array}$$

Then the optimal value function can be formulated as

$$v(y) = \sum_{k=1}^{p} \text{val}(P_k[y]),$$

and the *master problem* reads

$$\text{val}(P) = \min_{y \in \hat{Y}} v(y) \tag{4.30}$$

where

$$\hat{Y} = \{y \in Y \mid (P[y]) \text{ is feasible}\}.$$

If (P) is a general nonconvex MINLP, then $v(y)$ is a non-differentiable nonconvex function and can be optimized by any non-differentiable optimization method. In (Grothey, 2001) a decomposition-based method for computing local solutions of (4.30) is proposed.

Chapter 5

Semidefinite Relaxations

The success of interior point methods in linear programming has led to the development of interior point methods for semidefinite programming (SDP). Such methods usually require few iterations to produce high quality solutions. Often however, one iteration is quite expensive, since it is not easy to exploit sparsity (Benson et al., 2000). Other approaches for solving SDP include the nonlinear programming approach (Burer and Monteiro, 2001), and the spectral bundle method (Helmberg and Kiwiel, 2002; Helmberg and Rendl, 2000) that is based on an eigenvalue representation of the dual function. Shor may have been the first to study the dual of all-quadratic programs and to propose an eigenvalue approach for solving the dual (Shor, 1987; Shor, 1992; Shor, 1998). Lagrangian relaxation of all-quadratic optimization problems is studied in (Lemaréchal and Oustry, 1999). For an overview of the state-of-the-art SDP methods and applications, the reader is referred to (Wolkowicz et al., 2000; Helmberg, 2000).

After a short introduction into semidefinite and convex relaxations of all-quadratic programs (QQPs), this chapter presents a novel approach for solving the dual of general block-separable mixed-integer all-quadratic programs (MIQQPs) via eigenvalue optimization (Nowak, 2004). The approach is based on a dual-equivalent reformulation of a general QQP, which makes it possible to formulate the dual function as a block-separable eigenvalue function. Numerical results for random MIQQPs show that the proposed eigenvalue approach allows a fast computation of near optimal dual solutions.

5.1 Semidefinite and Lagrangian relaxations

Consider a general nonconvex MIQQP of the form:

$$\text{(MIQQP)} \qquad \begin{array}{lll} \min & q_0(x) & \\ \text{s.t.} & q_i(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \end{array}$$

where $q_i(x) = x^T A_i x + 2a_i^T x + d_i$, $A_i \in \mathbb{R}^{(n,n)}$ is symmetric, $a_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $i = 0, \ldots, m$. Furthermore, $\underline{x}, \overline{x} \in \mathbb{R}^n$ and $B \subset \{1, \ldots, n\}$. Problem (MIQQP) can be reformulated as an all-quadratic program by replacing the box constraints $x_j \in [\underline{x}_j, \overline{x}_j]$, $j \in C = \{1, \ldots, n\} \setminus B$, by

$$(x_j - \underline{x}_j)(x_j - \overline{x}_j) \leq 0,$$

and the binary constraints $x_j \in \{\underline{x}_j, \overline{x}_j\}$, $j \in B$, by

$$(x_j - \underline{x}_j)(x_j - \overline{x}_j) = 0.$$

This defines the following QQP

$$\text{(Q)} \qquad \begin{array}{lll} \min & q_0(x) & \\ \text{s.t.} & q_i(x) \leq 0, & i = 1, \ldots, m \\ & r_B(x) = 0 & \\ & r_C(x) \leq 0 & \end{array}$$

where

$$r(x) = \text{Diag}(x - \underline{x})(x - \overline{x}). \tag{5.1}$$

Let $q(x) = (q_1(x), \ldots, q_m(x))^T$. By introducing the Lagrangian function

$$L(x; \mu) = q_0(x) + (\mu^q)^T q(x) + (\mu^r)^T r(x)$$

and the Lagrangian multiplier set

$$\mathcal{M} = \{\mu = (\mu^q, \mu^r) \in \mathbb{R}^m_+ \times \mathbb{R}^n \mid \mu^r_C \geq 0\},$$

a Lagrangian dual of (Q) is formulated by

$$\text{(D)} \qquad \begin{array}{ll} \max & D(\mu) \\ \text{s.t.} & \mu \in \mathcal{M} \end{array}$$

where $D(\mu) = \inf_{x \in \mathbb{R}^n} L(x; \mu)$ is the dual function. Since (Q) contains the quadratic box constraints $r(x) \leq 0$, it can be shown that $\text{val}(D) > -\infty$.

Remark 5.1 (zero duality gap). The duality gap of (D) is studied in (Anstreicher and Wolkowicz, 1998) for special cases. If problem (Q) is convex and satisfies a Slater condition there is no duality gap. For the trust region problem with one ellipsoid constraint, the duality gap is also known to be zero (see Section 5.4.1). However, in the presence of two ellipsoid constraints, a nonzero duality gap can occur. Shor proved in (Shor, 1992; Shor, 1998) that problem (Q) has a nonzero duality gap, if and only if the objective function of an equivalent unconstrained polynomial programming problem can be represented as a sum of squares of other polynomials. In general however, it is not known how to compute the polynomials.

Interestingly, the dual of the all-quadratic program (Q) is equivalent to the following semidefinite program:

$$\text{(SDP)}\qquad \begin{array}{lll} \min & \langle A_0, X\rangle + 2a_0^T x + d_0 & \\ \text{s.t.} & \langle A_i, X\rangle + 2a_i^T x + d_i \le 0, & i = 1,\dots,m \\ & X_{ii} - x_i(\underline{x}_i + \overline{x}_i) + \underline{x}_i\overline{x}_i = 0, & i \in B \\ & X_{ii} - x_i(\underline{x}_i + \overline{x}_i) + \underline{x}_i\overline{x}_i \le 0, & i \in C \\ & X \succcurlyeq x \cdot x^T & \end{array}$$

where $\langle A, X\rangle = \operatorname{tr} AB$. The dual of (SDP) is the program:

$$\text{(DSDP)}\qquad \begin{array}{ll} \max & t \\ \text{s.t.} & \begin{pmatrix} A(\mu) & a(\mu) \\ a(\mu)^T & d(\mu) - t \end{pmatrix} \succcurlyeq 0 \\ & \mu \in \mathcal{M}, \quad t \in \mathbb{R} \end{array}$$

where $A(\mu) = A_0 + \sum_{i=1}^m \mu_i^q A_i + 2\operatorname{Diag}(\mu^r)$, $a(\mu) = a_0 + \sum_{i=1}^m \mu_i^q a_i + 2\operatorname{Diag}(\mu^r)(\underline{x}+\overline{x})$ and $d(\mu) = d_0 + \sum_{i=1}^m \mu_i^q d_i + \underline{x}^T \operatorname{Diag}(\mu^r)\overline{x}$. The following result is proven in (Lemaréchal and Oustry, 1999).

Lemma 5.2 (strong duality). *Assume that a primal or dual Slater condition is fulfilled, i.e. there exists $\mu \in \mathcal{M}$ such that $A(\mu)$ is positive definite, or there exists a primal feasible pair (X, x) such that $X - x \cdot x^T$ is positive definite. Then*

$$\operatorname{val}(D) = \operatorname{val}(DSDP) = \operatorname{val}(SDP).$$

The next corollary gives a further equivalent formulation of the dual problem (D).

Corollary 5.3. *Let $I_{\text{lin}} \subset \{1,..,m\}$ and $I_q \subset \{1,..,m\}$ be the index sets of linear constraints and quadratic constraints of problem (MIQQP), respectively. We define the Lagrangian with respect to quadratic constraints*

$$L_q(x,\mu) = q_0(x) + \sum_{i\in I_q} \mu_i^q q_i(x) + (\mu^r)^T r(x)$$

and the feasible set with respect to linear constraints by

$$S_{\text{lin}} = \{x \in \mathbb{R}^n : q_i(x) \le 0, i \in I_{\text{lin}}\}.$$

Since $\operatorname{val}(D) > -\infty$, we get from strong duality:

$$\operatorname{val}(D) = \max\{\min_{x\in S_{\text{lin}}} L_q(x,\mu) \mid \mu_{I_q}^q \ge 0,\ \mu_C^r \ge 0,\ \nabla_x^2 L_q(x,\mu) \succcurlyeq 0\}. \tag{5.2}$$

Remark 5.4. Since $D(\mu) > -\infty$ if and only if $\nabla^2 L(\cdot;\mu)$ is positive semidefinite, the dual (D) contains a hidden semidefinite constraint. This implies that for all

$\hat{\mu} \in \operatorname{dom} D$ the function $L(\cdot;\hat{\mu})$ is a convex underestimator of q_0 over the feasible set of (MIQQP), and

$$\min\{L(x;\hat{\mu}) \mid x \in [\underline{x},\overline{x}]\}$$

is a convex relaxation of (Q) in this case. Note that if $\hat{\mu}$ is an optimal dual point, this convex relaxation is stronger than the convex underestimating-relaxation (3.4) obtained by replacing the functions q_i in (Q) by α-underestimators defined in Section 6.3.

5.2 Block-separable reformulation

Assuming that problem (Q) is block-separable, it is shown that (Q) can be reformulated in such a way that all variables are bounded by -1 and 1, and all linear terms $b_i^T x$ of the functions q_i in (Q) disappear. This formulation allows us to formulate the Lagrangian dual problem as a block-separable eigenvalue optimization problem, which can be solved efficiently. The transformation is carried out in two steps.

Note that problem (Q) is block-separable if there exists a partition $\mathcal{P} = \{J_1,\ldots,J_p\}$ of $\{1,\ldots,n\}$ with $\bigcup_{k=1}^p J_k = \{1,\ldots,n\}$ and $J_i \cap J_k = \emptyset$ if $i \neq k$, such that

$$q_i(x) = c_i + \sum_{k=1}^{p} q_{i,k}(x_{J_k}), \tag{5.3}$$

where $q_{i,k}(x_{J_k}) = \langle x_{J_k}, A_{i,J_k,J_k} x_{J_k}\rangle + 2b_{i,J_k}^T x_{J_k}$ for $i = 0,\ldots,m$. We denote by $n_k = |J_k|$ the size of a block J_k.

Let $u = \frac{1}{2}(\overline{x}+\underline{x})$ be the center and $w = \frac{1}{2}(\overline{x}-\underline{x})$ be the half-diameter vector of the interval $[\underline{x},\overline{x}]$ respectively. The affine transformation $\theta(x) = \operatorname{Diag}(w)x + u$ maps the interval $[-e,e]$ onto $[\underline{x},\overline{x}]$. In the first step of the transformation, called *standardization*, the variables x of (Q) are replaced with $\theta(x)$. The transformed quadratic forms take the form

$$\hat{q}_i(x) = q_i(\theta(x)) = \langle x, \hat{A}_i x\rangle + 2\hat{b}_i^T x + \hat{c}_i, \quad i = 0,\ldots,m, \tag{5.4}$$

where $\hat{A}^i = WA^iW$, $\hat{b}^i = Wb^i + WA^iu$, $\hat{c}_i = u^TAu + 2u^Tb^i + c_i$ and $W = \operatorname{Diag}(w)$. In the second step of the transformation, the problem is *homogenized* by replacing linear terms $\hat{b}_{i,J_k}^T x_{J_k}$ by quadratic terms $x_{n+k}\cdot\hat{b}_{i,J_k}^T x_{J_k}$ and adding constraints $x_{n+k}^2 - 1 = 0$. This gives the problem

$$(\tilde{Q}) \qquad \begin{array}{lll} \min & \tilde{q}_0(x) & \\ \text{s.t.} & \tilde{q}_i(x) \le 0, & i = 1,\ldots,m \\ & x_j^2 - 1 \le 0, & j \in C \\ & x_j^2 - 1 = 0, & j \in B \cup \{n+1,\ldots,n+p\} \end{array}$$

where $\tilde{q}_i(x) = \hat{c}_i + \sum_{k=1}^{p} \tilde{q}_{i,k}(x_{\tilde{J}_k})$, $\tilde{q}_{i,k}(x_{\tilde{J}_k}) = \langle x_{J_k}, \hat{A}_{i,J_k,J_k} x_{J_k}\rangle + 2x_{n+k}\hat{b}_{i,J_k} x_{J_k}$,

and $\tilde{J}_k = J_k \cup \{n+k\}$. Obviously, $\tilde{q}_{i,k}(x) = q_{i,k}(\hat{x})$, if $x_{1:n_k} = \hat{x}$ and $x_{n_k+1} = 1$ or $x_{1:n_k} = -\hat{x}$ and $x_{n_k+1} = -1$. Therefore, the optimal values of $(\tilde{Q})$ and (Q) coincide. Since each additional variable can be 1 or -1, the number of solutions of $(\tilde{Q})$ is 2^p times larger than of (Q).

5.3 Eigenvalue representation of the dual function

It is shown that the dual function to $(\tilde{Q})$ can be represented in terms of eigenvalues. Define a partial Lagrangian related to $(\tilde{Q})$ by

$$\begin{aligned}\tilde{L}_k(x_{\tilde{J}_k};\mu) &= \tilde{q}_{0,k}(x_{\tilde{J}_k}) + (\mu^q)^T \tilde{q}_k(x_{\tilde{J}_k}) + (\mu^r_{\tilde{J}_k})^T(\mathrm{Diag}(x_{\tilde{J}_k})x_{\tilde{J}_k} - e)\\ &= x^T_{\tilde{J}_k}\tilde{A}_k(\mu)x_{\tilde{J}_k} - e^T\mu^r_{\tilde{J}_k}\end{aligned}$$

where $\tilde{q}_k(x) = (\tilde{q}_{1,k}(x), \ldots, \tilde{q}_{m,k}(x))^T$,

$$\tilde{A}_k(\mu) = \begin{pmatrix} \hat{A}_k(\mu) & \hat{b}_k(\mu) \\ \hat{b}_k(\mu)^T & \mu^r_{n+k} \end{pmatrix}, \tag{5.5}$$

$\hat{A}_k(\mu) = \hat{A}_{0,J_k,J_k} + \sum_{i=1}^m \mu^q_i \hat{A}_{i,J_k,J_k} + \mathrm{Diag}(\mu^r_{J_k})$, $\hat{b}_k(\mu) = \hat{b}_{0,J_k} + \sum_{i=1}^m \mu^q_i \hat{b}_{i,J_k}$, and $\hat{A}_i, \hat{b}_i$ are defined as in (5.4). Hence, the related partial dual function is the eigenvalue function

$$\tilde{D}_k(\mu) = \min_{x \in I\!B(n_k+1)} \tilde{L}_k(x;\mu) = (n_k+1)\cdot \min\{0, \lambda_1(\tilde{A}_k(\mu))\} - e^T\mu^r_{\tilde{J}_k}$$

where $I\!B(n)$ denotes a zero-centered ball in $\mathbb{R}^n$ with radius $\sqrt{n}$. Defining the Lagrangian dual function to $(\tilde{Q})$ by

$$\tilde{D}(\mu) = \hat{c}(\mu) + \sum_{k=1}^p \tilde{D}_k(\mu),$$

where $\hat{c}(\mu) = \hat{c}_0 + \sum_{i=1}^m \mu^q_i \hat{c}_i$, a dual problem to $(\tilde{Q})$ can be formulated as the eigenvalue optimization problem

$$(\tilde{D}) \qquad \begin{array}{ll} \max & \tilde{D}(\mu) \\ \text{s.t.} & \mu \in \tilde{\mathcal{M}} \end{array}$$

with

$$\tilde{\mathcal{M}} = \{(\mu^q, \mu^r) \in \mathbb{R}^m \times \mathbb{R}^{n+p} \mid \mu^q \geq 0,\ \mu^r_C \geq 0\}.$$

A similar eigenvalue optimization problem was used in Rendl and Wolkowicz, 1997 for solving the trust region problem and in Helmberg, 2000 for unconstrained quadratic 0-1 programming.

5.4 Duality results and convex relaxation

In this section Lagrangian dual problems related to the all-quadratic problems (Q) and ($\tilde{Q}$) are formulated and analyzed. In particular, it is proven that the dual problems to (Q) and ($\tilde{Q}$) are equivalent. The proof is a straightforward generalization of a dual-equivalent result in (Poljak et al., 1995) on quadratic binary programs. In (Poljak et al., 1995) the problem is dualized with respect to a full-dimensional sphere, whereas here the problem is dualized with respect to a Cartesian product of low-dimensional balls.

5.4.1 The trust region problem

In order to prove the equivalence of (D) and ($\tilde{D}$), we need some results on the *trust region problem* defined by

$$\text{(T)} \qquad \begin{array}{ll} \min & q(x) \\ \text{s.t.} & x \in I\!B(n) \end{array}$$

where $q(x) = x^T Bx + 2b^T x$, $B \in \mathbb{R}^{(n,n)}$ and $b \in \mathbb{R}^n$. The dual of (T) is

$$\text{(DT)} \qquad \max_{\sigma \in \mathbb{R}_+} \inf_{x \in \mathbb{R}^n} q(x) + \sigma(\|x\|^2 - n).$$

Problem (T) is one of the few nonconvex all-quadratic optimization problems having a zero duality gap (Stern and Wolkowicz, 1995), i.e.

$$\text{val}(T) = \text{val}(DT). \tag{5.6}$$

If $b = 0$, then (T) is an eigenvalue problem and it holds that $\text{val}(T) = n \cdot \min\{0, \lambda_1(B)\}$. Consider now the case $b \neq 0$. By replacing $b^T x$ with $x_{n+1} \cdot b^T x$, where $x_{n+1}^2 = 1$, we get the following homogenized formulation of (T) with $n+1$ variables and an additional equality constraint

$$\text{($\tilde{T}$)} \qquad \begin{array}{ll} \min & x_{1:n}^T B x_{1:n} + 2x_{n+1} b^T x_{1:n} \\ \text{s.t.} & \|x\|^2 \leq n+1 \\ & x_{n+1}^2 = 1. \end{array}$$

Clearly, we have $\text{val}(T) = \text{val}(\tilde{T})$. Dualization of ($\tilde{T}$) with respect to the ball $I\!B(n+1)$ gives the dual problem

$$\text{(D$\tilde{T}$)} \qquad \max_{\mu \in \mathbb{R}_+} (n+1) \cdot \min\{0, \lambda_1(C(\mu))\} - \mu,$$

where $C(\mu) = \begin{pmatrix} B & b \\ b^T & \mu \end{pmatrix}$.

Lemma 5.5. *It holds that* $\text{val}(\tilde{T}) = \text{val}(T) = \text{val}(D\tilde{T})$.

Proof. This was proven in (Rendl and Wolkowicz, 1997):

$$\begin{aligned}
\min_{\|x\|^2\le n} q(x) &= \max_{\mu\in\mathbb{R}} \min_{\substack{\|x\|^2\le n\\ y^2=1}} x^TBx + 2yb^Tx + \mu(y^2-1)\\
&\ge \max_{\mu\in\mathbb{R}} \min_{\|x\|^2+y^2\le n+1} x^TBx + 2yb^Tx + \mu(y^2-1)\\
&\ge \max_{\substack{\mu\in\mathbb{R}\\ \sigma\in\mathbb{R}_+}} \inf_{\substack{x\in\mathbb{R}^n\\ y\in\mathbb{R}}} x^TBx + 2yb^Tx + \mu(y^2-1) + \sigma(\|x\|^2+y^2-n-1)\\
&= \max_{\sigma\in\mathbb{R}_+} \inf_{\substack{x\in\mathbb{R}^n\\ y^2=1}} x^TBx + 2yb^Tx + \sigma(\|x\|^2-n)\\
&= \min_{\|x\|^2\le n} x^TBx + 2b^Tx.
\end{aligned}$$

□

Lemma 5.6. *Let $\bar{\mu}$ be a solution of $(D\tilde{T})$. Then $\sigma^* = -\min\{0, \lambda_1(C(\bar{\mu}))\}$ solves (DT).*

Proof. Let

$$\tilde{L}(x;\sigma,\mu) = x^T(C(\mu)+\sigma I)x - \mu - (n+1)\sigma$$

be the Lagrangian of $(\tilde{T})$ and

$$\tilde{D}(\sigma,\mu) = \inf_{x\in\mathbb{R}^{n+1}} \tilde{L}(x;\sigma,\mu)$$

be the corresponding dual function, which can be formulated in closed form as

$$\tilde{D}(\sigma,\mu) = \begin{cases} -\mu-(n+1)\sigma & \text{if } C(\mu)+\sigma I \succcurlyeq 0\\ -\infty & \text{else .}\end{cases}$$

For a dual solution point $(\bar{\sigma},\bar{\mu}) \in \operatorname{Argmax}_{\sigma\in\mathbb{R}_+,\mu\in\mathbb{R}} \tilde{D}(\sigma,\mu)$, it follows from the closed form that $\bar{\sigma} = -\min\{0,\lambda_1(C(\bar{\mu}))\}$. From Lemma 5.5 we have $\tilde{D}(\bar{\sigma},\bar{\mu}) = \operatorname{val}(\tilde{T})$. Hence, the solution set of $(\tilde{T})$ is in $\operatorname{Argmin}_{x\in\mathbb{R}^{n+1}} \tilde{L}(x;\bar{\sigma},\bar{\mu})$. This proves

$$\operatorname{val}(T) = \min_{x\in\mathbb{R}^{n+1}, x_{n+1}=1} \tilde{L}(x;\bar{\sigma},\bar{\mu}) = \min_{x\in\mathbb{R}^n} L(x;\bar{\sigma}).$$

□

5.4.2 Dual-equivalence

Based on strong duality of the trust region problem, the following dual-equivalence result can be proven.

Proposition 5.7. *The dual problems (D) and $(\tilde{D})$ have the same optimal value.*

Proof. Since (Q) is block-separable, i.e. (5.3) holds, the dual function D decomposes into

$$D(\mu) = c(\mu) + \sum_{k=1}^{p} D_k(\mu),$$

with $c(\mu) = c_0 + \sum_{i=1}^m \mu_i^q c_i$ and $D_k(\mu) = \min_{x \in \mathbb{R}^{n_k}} L_k(x;\mu)$ where

$$L_k(x_{J_k};\mu) = q_{0,k}(x_{J_k}) + (\mu^q)^T q_k(x_{J_k}) + (\mu^r_{J_k})^T r_{J_k}(x)$$

with $q_k(x) = (q_{1,k}(x), \ldots, q_{m,k}(x))^T$. We define the standardized partial Lagrangian

$$\hat{L}_k(x;\mu) = x^T \hat{A}_k(\mu) x + 2\hat{b}_k(\mu)^T x - (\mu^r_{J_k})^T e$$

according to (5.5), and the related partial dual function by

$$\hat{D}_k(\mu) = \inf_{x \in I\!B(n_k)} \hat{L}_k(x;\mu). \tag{5.7}$$

We denote by $e_{J_k} \in \mathbb{R}^n$ the characteristic vector of a partition element J_k defined by $e_{J_k,j} = \begin{cases} 1 & \text{for } j \in J_k \\ 0 & \text{else} \end{cases}$. From strong duality of the trust-region problem (5.6) it follows that

$$\begin{aligned} \hat{D}_k(\mu) &= \min_{x \in I\!B(n_k)} \hat{L}_k(x;\mu) \\ &= \max_{t \in \mathbb{R}_+} \inf_{x \in \mathbb{R}^{n_k}} \hat{L}_k(x;\mu) + t \cdot (\|x\|^2 - n_k) \\ &= \max_{t \in \mathbb{R}_+} \inf_{x \in \mathbb{R}^{n_k}} \hat{L}_k(x;\mu^q, \mu^r + t \cdot e_{J_k}). \end{aligned}$$

From Lemma 5.5 we have

$$\begin{aligned} \hat{D}_k(\mu) &= \min_{x \in I\!B(n_k)} x^T \hat{A}_k(\mu) x + 2\hat{b}_k(\mu)^T x - (\mu^r_{J_k})^T e \\ &= \max_{t \in \mathbb{R}} \inf_{x \in \mathbb{R}^{n_k+1}} x^T \tilde{A}_k(\mu^q, \mu^r + t e_{n+k}) x - (\mu^r_{\tilde{J}_k})^T e \\ &= \max_{t \in \mathbb{R}} \tilde{D}_k(\mu^q, \mu^r + t e_{n+k}). \end{aligned}$$

Hence,

$$\begin{aligned} \mathrm{val}(D) &= \max_{\mu \in \mathcal{M}} c(\mu) + \sum_{k=1}^p \inf_{x \in \mathbb{R}^{n_k}} L_k(x;\mu) \\ &= \max_{\mu \in \mathcal{M}} \hat{c}(\mu) + \sum_{k=1}^p \max_{t \in \mathbb{R}_+} \inf_{x \in \mathbb{R}^{n_k}} \hat{L}_k(x;\mu^q, \mu^r + t e_{J_k}) \\ &= \max_{\mu \in \mathcal{M}} \hat{c}(\mu) + \sum_{k=1}^p \hat{D}_k(\mu) \\ &= \max_{\mu \in \mathcal{M}} \hat{c}(\mu) + \sum_{k=1}^p \max_{t \in \mathbb{R}} \tilde{D}_k(\mu^q, \mu^r + t e_{n+k}) \\ &= \mathrm{val}(\tilde{D}). \end{aligned}$$

□

The next lemma shows how convex relaxations for (Q) can be obtained from feasible dual points of $(\tilde{D})$.

Lemma 5.8. *Let $\tilde{\mu} \in \tilde{\mathcal{M}}$ be a feasible dual point of $(\tilde{D})$ and define $\mu \in \mathcal{M}$ by $\mu^q = \tilde{\mu}^q$ and $\mu_j^r = \tilde{\mu}_j^r + t_k$ with $t_k = \min\{0, \lambda_1(\tilde{A}_k(\tilde{\mu}))\}$ for $j \in J_k$, $k = 1, \ldots, p$. Then:*

(i) *$\tilde{D}(\tilde{\mu}) \leq D(\mu)$ and $L(\cdot; \mu)$ is convex.*

(ii) *If $\tilde{\mu}$ is a solution of $(\tilde{D})$, then μ is a solution of (D).*

Proof. (i) From Lemma 5.5 and 5.6 it follows that

$$\tilde{D}_k(\tilde{\mu}) = \min_{x \in \mathbb{R}^{n_k+1}} \tilde{L}_k(x; \tilde{\mu}^q, \mu^r + t_k e_{\tilde{J}_k}) \leq \min_{x \in \mathbb{R}^{n_k}} \hat{L}_k(x; \mu) = D_k(\mu).$$

Hence, $\tilde{D}(\tilde{\mu}) \leq D(\mu)$.
Statement (ii) follows from (i) and Proposition 5.7. □

5.4.3 Modifications

Several simplifications of the dual problem $(\tilde{D})$ are possible.

Remark 5.9. If all variables of a block J_k are binary, i.e. $J_k \subseteq B$, we can dualize the related partial Lagrangian function with respect to the sphere $\partial\mathbb{B}(n_k)$. This simplifies the dual problem $(\tilde{D})$, since the number of dual constraints is reduced. We show that this modification does not change $\mathrm{val}(\tilde{D})$. To see this, we consider the modified partial dual function of $\tilde{D}$ defined by

$$\bar{D}_k(\mu) = (n_k + 1) \cdot \lambda_1(\tilde{A}^k(\mu)) - (\mu^r_{\tilde{J}_k})^T e.$$

Since $\lambda_1(\tilde{A}^k(\mu^q, \mu^r + t \cdot e_{\tilde{J}_k})) = \lambda_1(\tilde{A}^k(\mu)) + t(n_k + 1)$ and $(\mu^r_{\tilde{J}_k} + t \cdot e_{\tilde{J}_k})^T e = (\mu^r_{\tilde{J}_k})^T e + t(n_k + 1)$ for all $t \in \mathbb{R}$, it holds that

$$\bar{D}_k(\mu) = \tilde{D}_k(\mu^q, \mu^r + t \cdot e_{\tilde{J}_k}).$$

For $t = \min\{0, -\lambda_1(\tilde{A}^k(\mu))\}$ we have $\lambda_1(\tilde{A}^k(\mu^q, \mu^r + t \cdot e_{\tilde{J}_k})) \geq 0$ and therefore $\bar{D}_k(\mu^q, \mu^r + t \cdot e_{\tilde{J}_k}) = \tilde{D}_k(\mu^q, \mu^r + t \cdot e_{\tilde{J}_k})$, which implies that $\mathrm{val}(\tilde{D})$ is not changed.

Remark 5.10. A further simplification can be made in the case $b_{i,J_k} = (b_i)_{j \in J_k} = 0$ for $i = 0, \ldots, m$. In this case, the trust region problem (5.7) is an eigenvalue problem and it holds that

$$\hat{D}^k(\mu) = n_k \cdot \min\{0, \lambda_1(\hat{A}^k(\mu))\} - (\mu^r_{J_k})^T e.$$

From Lemma 5.5 it follows that $\tilde{D}^k$ can be replaced with $\hat{D}^k$ without changing $\mathrm{val}(\tilde{D})$.

Remark 5.11. If $A^i_{J_k,J_k}$ is zero for $i = 0, \ldots, m$, the related Lagrangian problem is linear and therefore separable with respect to all variables of this block. Hence, we can assume $J_k = \{j_k\}$, i.e. $I\!B(n_k) = [-1, 1]$. Then

$$\begin{aligned} \min_{x\in[-1,1]} \hat{L}_k(x;\mu) &= \min_{x\in[-1,1]} 2\hat{b}_k(\mu)^T x - (\mu^r_{J_k})^T e \\ &= 2\min\{\hat{b}_{j_k}(\mu)\underline{x}_{j_k}, \hat{b}_{j_k}(\mu)\overline{x}_{j_k}\} - (\mu^r_{J_k})^T e. \end{aligned}$$

If (Q) is a MIP, this yields the traditional linear relaxation.

5.4.4 Influence of decomposition on the dual function

Denote by $\tilde{D}_0$ the dual function $\tilde{D}$ of $(\tilde{Q})$ defined with respect to the trivial partition $\mathcal{P}_0 = \{V\}$ with $V = \{1, \ldots, n\}$. From Lemma 5.7 it follows that the optimal values related to $\tilde{D}_0$ and $\tilde{D}$ are the same. However, the dual values $\tilde{D}_0(\mu)$ and $\tilde{D}(\mu)$ at a dual point $\mu \in \tilde{\mathcal{M}}$ can be different. Let $\tilde{L}(x;\mu) = \hat{c}(\mu) + \sum_{k=1}^p \tilde{L}_k(x_{\tilde{J}_k};\mu)$ be the Lagrangian related to $(\tilde{Q})$ and $X = \{x \in \mathbb{R}^{n+p} \mid x_{\tilde{J}_k} \in I\!B(1+n_k),\ k = 1, \ldots, p\}$. Since $X \subseteq I\!B(n+p)$, we have

$$\tilde{D}_0(\mu) = \min_{x\in I\!B(n+p)} \tilde{L}(x;\mu) \le \min_{x\in X} \tilde{L}(x;\mu) = \tilde{D}(\mu).$$

The following example shows that the above inequality can be strict.

Example 5.12. Consider the MaxCut problem

$$\min\{x^T A x \mid x \in \{-1, 1\}^n\},$$

where A is a block-diagonal matrix consisting of sub-matrices $A_k \in \mathbb{R}^{(n_k,n_k)}$, $k = 1, \ldots, p$. Assuming $\lambda_1(A_1) < \lambda_1(A_j)$ for $j > 1$, it follows that

$$\tilde{D}_0(0) = n \cdot \lambda_1(A) < \sum_{k=1}^p n_k \lambda_1(A_k) = \tilde{D}(0).$$

This demonstrates that decomposition not only facilitates the evaluation of the dual function, but also improves the initial dual bound $\tilde{D}(0)$ (see Section 5.6). On the other hand, if a splitting-scheme is used, decomposition can worsen the dual bound $\tilde{D}(0)$. In (Lemaréchal and Renaud, 2001) it is shown:

Lemma 5.13. *Let (D) and* Dual(2.2) *be the Lagrangian dual of the original problem (Q) and the splitting-scheme* (2.2)*, as defined in Section* 2.3*, respectively. Then* $\mathrm{val}(\mathrm{Dual}(2.2)) \le \mathrm{val}(D)$.

The results of Section 5.6 demonstrate that this inequality can be strict.

5.5 Solving the Lagrangian dual problem ($\tilde{D}$)

The dual problem ($\tilde{D}$) is a convex non-differentiable optimization problem. It can be solved by many methods (Hiriart-Urruty and Lemaréchal, 1993). Here, the proximal bundle code NOA 3.0 (Kiwiel, 1994b) of Kiwiel described in (Kiwiel, 1990) is used for maximizing the dual function $\tilde{D}$. Supergradients of the dual function ($\tilde{D}$) are computed according to the following lemma:

Lemma 5.14. *For a given dual point $\mu \in \tilde{\mathcal{M}}$ let v_k be a (normalized) minimum eigenvector of $\tilde{A}_k(\mu)$. Define $x \in \mathbb{R}^{n+p}$ by $x_{\tilde{J}_k} = \sqrt{n_k+1} \cdot v_k$ for $k = 1, \ldots, p$. Then the point $g = (g_1, g_2) \in \mathbb{R}^m \times \mathbb{R}^{n+p}$ defined by $g_{1,i} = \tilde{q}_i(x)$ for $i = 1, \ldots, m$ and $g_{2,j} = x_j^2 - 1$ for $j = 1, \ldots, n+p$ is a supergradient of $\tilde{D}(\mu)$ at μ.*

Proof. We apply the subgradient formula of Lemma 4.1. Let $\tilde{L}_k(\cdot;\mu)$ be a partial Lagrangian to ($\tilde{Q}$). From the definition of x it follows that $x_{\tilde{J}_k} \in \operatorname{Argmin}_{y \in I\!B(n_k+1)} \tilde{L}_k(y;\mu)$. Hence, $x \in \operatorname{Argmin}_{y \in X} \tilde{L}(y;\mu)$, where $\tilde{L}$ and X are defined as in Section 5.4.4. This proves the statement according to Lemma 4.1. □

The evaluation of the dual function $\tilde{D}$ is implemented with the modifications of Remarks 5.9, 5.10 and 5.11, and the supergradient formula of Lemma 5.14 as part of the software package LaGO (see Chapter 14). For the computation of a minimum eigenvalue and minimum eigenvector, two algorithms are used. The first algorithm is an implicit symmetric QL-method from the EISPACK-library NETLIB, 1973, used if the dimension of the matrix is less than or equal to 50. If the dimension is greater than 50, the Lanczos method ARPACK++ (Gomes and Sorensen, 1997) is used.

The following parameters of the proximal bundle method NOA are used: bundle size = 50, line-search decrease = 0.1, QP weight = 10.0 and feasibility tolerance = 0.1. As a stopping criterion for the bundle method either the optimality tolerance is set equal to 10^{-3}, or the method stops if a *measure of relative improvement* is smaller than a given tolerance. In particular,

$$\delta_s^j = \frac{\tilde{D}(\mu^{s(j+1)}) - \tilde{D}(\mu^{sj})}{|\tilde{D}(\mu^0)| + 1},$$

is defined, and the iteration stops, if

$$\delta_s^j < \rho \cdot \delta_s^{j_{max}} \tag{5.8}$$

where $\{\mu^j\}$ is the sequence of dual points generated by the bundle method at serious steps, $\delta_s^{j_{max}} = \max\{\delta_s^0, \ldots, \delta_s^j\}$, with $\rho = 0.4$ and $s = 10$.

5.6 Numerical results

In order to study the influence of decomposition, numerical experiments with random MIQQP instances were made. All results were obtained on a machine that has a 1.8 GHz-Pentium IV processor with a LINUX system.

5.6.1 Block structure

In the first experiment, decomposition-based bounds computed by the QL-method and non-decomposition-based bounds computed by the full-dimensional Lanczos method are compared. Block-separable random MIQQPs are produced using Algorithm B.1 described in Appendix B.2 with parameters n, the number of variables, m, the number of constraints, and l, the block size. For a given set of input parameters (n, m, l), five random MIQQPs are produced.

For each instance two dual problems of the form $(\tilde{D})$ related to the partitions $\mathcal{P}_1 = \{J_1, \ldots, J_p\}$ and $\mathcal{P}_2 = \{V\}$ with $V = \{1, \ldots, n\}$ are generated. The first dual problem is called (D_1) and the second (D_2). The corresponding dual functions are denoted by D_1 and D_2 respectively. From Lemma 5.7 it is known that $\mathrm{val}(D_1) = \mathrm{val}(D_2)$.

First, a dual value $D_1(\hat{\mu})$ is computed by using the previously described bundle method with the stopping criterion (5.8). Then D_2 is maximized and the iteration stops if D_2 reaches the value $D_1(\hat{\mu})$. Furthermore, the initial relative error

$$\kappa_i^0 = \frac{\mathrm{val}(D_2) - D_i(0)}{|\,\mathrm{val}(D_2)| + 1}, \quad i \in \{1, 2\}, \tag{5.9}$$

is calculated, where the optimal value of (D_2) is computed by using the previously described bundle method with an optimality tolerance 10^{-3}. For different input parameters of Algorithm B.1, Tables 5.1 and 5.2 show :

- the fraction t_2/t_1 where t_1 and t_2 is the average time in seconds for solving (D_1) and (D_2) respectively,
- the time t_1,
- the fraction κ_2^0/κ_1^0 where κ_i^0 is the average initial relative error (5.9).

It can be seen from the tables that the decomposition scheme accelerates the running time by magnitudes. The acceleration is particularly large if the number of constraints is high. This is due to the increased cost for the matrix-vector multiplication used in the Lanczos algorithm. Moreover, the results show that $\kappa_1^0 < \kappa_2^0$ (see also Figure 5.1).

Decomposition also makes the dual solution method more stable. Convergence problems of the Lanczos method were observed when the optimality tolerance of the dual solver was small. It is well-known that the performance of the

	block-size $l = 10$			block-size $l = 20$		
n	t_2/t_1	t_1	κ_2^0/κ_1^0	t_2/t_1	t_1	κ_2^0/κ_1^0
200	312.526	0.392	7.22114	85.7879	0.594	5.81284
400	1544.22	0.768	10.5006	271.037	1.234	8.79377
600	3551.09	1.204	12.8053	563.391	1.818	11.3668
800	4243.39	1.656	15.5317	861.217	2.428	12.9469
1000	6546.61	2.068	17.3226	1279.55	3.226	14.7185

Table 5.1: number of constraints $m = 0$

	block-size $l = 10$			block-size $l = 20$		
m	t_2/t_1	t_1	κ_2^0/κ_1^0	t_2/t_1	t_1	κ_2^0/κ_1^0
0	53.7087	0.206	4.63817	21.9728	0.294	3.72246
4	159.35	0.24	4.84415	38.9673	0.428	3.6699
8	135.229	0.376	4.52294	37.0607	0.626	3.41876
12	132.924	0.472	4.40023	29.1492	0.764	3.51218
16	157.272	0.766	4.33168	47.5457	1.378	3.4816
20	166.995	0.85	4.19541	56.2844	1.568	3.44

Table 5.2: dimension $n = 200$

Lanczos method depends on how well the eigenvalues are separated. In (Helmberg and Rendl, 2000) it is demonstrated that eigenvalues cluster in eigenvalue optimization, causing numerical instabilities of the Lanczos method. In contrast, the QL-method is very stable.

5.6.2 Network structure

In order to study splitting-schemes, random MaxCut problems of the form

$$\min\{x^T A x \mid x \in \{-1, 1\}^n\},$$

are generated, where $A \in \mathbb{R}^{(n,n)}$ is the sparse matrix

$$A = \begin{pmatrix} A_1 & B_1 & 0 & B_p \\ B_1^T & \ddots & \ddots & 0 \\ 0 & \ddots & A_{p-1} & B_{p-1} \\ B_p^T & 0 & B_{p-1}^T & A_p \end{pmatrix}.$$

The sub-matrices $A_k \in \mathbb{R}^{(l,l)}$, $k = 1, \ldots, p$, are dense with a block-size $l = n/p$. The sub-matrices $B_k \in \mathbb{R}^{(l,l)}$ are sparse with nonzero entries at $(l - i, i)$, $i = 1, \ldots, s$, where $s \in \{0, \ldots, l\}$ is a given flow size. The resulting sparsity graph has a ring topology with p components that are each connected by s arcs. All

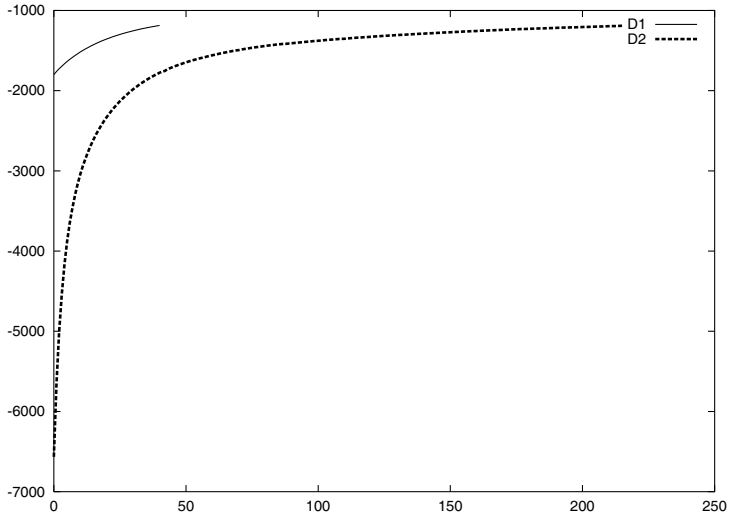

Figure 5.1: Dual values of D_1 and D_2 at serious steps, where $(n, m, l) = (200, 0, 10)$, showing that $D_2(\mu) < D_1(\mu)$

nonzero components of A are uniformly distributed random numbers in $[-10, 10]$. For a given MaxCut problem, a splitting-scheme (13.6) is generated, as described in Section 13.5.1, by using the partition $\mathcal{P} = \{J_1, \ldots, J_p\}$ with $J_k = \{(k-1)l + 1, \ldots, k \cdot l\}$, $k = 1, \ldots p$.

For the splitting-scheme as well as for the original MaxCut problem, dual problems of the form $(\tilde{D})$ are constructed, which are called (D_1) and (D_2) respectively. As in the previous experiment, five random MaxCut problems for a given set of input parameters (n, p, s) are produced, and first a dual value of $D_1(\hat{\mu})$ and then a dual value $D_2(\tilde{\mu}) \simeq D_1(\hat{\mu})$ is computed by using the bundle method NOA with the parameters previously described. Tables 5.3 and 5.4 show

- the fraction t_2/t_1 where t_1 and t_2 is the average time in seconds for solving (D_1) and (D_2) respectively,
- the time t_1,
- the fraction κ_2^0/κ_1^0 where κ_i^0 is the average initial relative error (5.9),
- the average percentage relative difference of the optimal dual values of (D_1) and (D_2)

$$\kappa_d = \frac{\text{val}(D_2) - \text{val}(D_1)}{|\,\text{val}(D_2)| + 1}.$$

The results demonstrate in most cases that the splitting-scheme accelerates the evaluation of the dual considerably. However, in the last experiment the computation without decomposition was faster. It can also be seen that for these instances the relative difference of the optimal dual values κ_d is not zero (see Section 5.4.4). Moreover, in most cases the fraction κ_2^0/κ_1^0 was greater than 1 for $s = 2$ and smaller than 1 for $s = 4$ (see also Figure 5.2).

	block-size $l = 10$				block-size $l = 20$			
n	t_2/t_1	t_1	κ_2^0/κ_1^0	$100 \cdot \kappa_d$	t_2/t_1	t_1	κ_2^0/κ_1^0	$100 \cdot \kappa_d$
200	8.212	2.702	0.969	0.468	2.754	2.026	1.304	6.362
400	7.182	6.264	1.008	0.953	4.391	5.288	1.483	6.719
600	6.986	12.506	1.228	0.827	3.536	8.426	1.648	7.528
800	6.963	20.246	1.238	0.627	4.740	12.826	1.699	7.209
1000	9.214	29.322	1.197	0.601	5.227	16.876	1.694	7.337

Table 5.3: flow-size $s = 2$

	block-size $l = 10$				block-size $l = 20$			
n	t_2/t_1	t_1	κ_2^0/κ_1^0	$100 \cdot \kappa_d$	t_2/t_1	t_1	κ_2^0/κ_1^0	$100 \cdot \kappa_d$
200	1.928	3.756	0.634	2.185	0.256	7.38	0.485	0.801
400	1.977	11.532	0.711	3.398	0.463	18.394	0.434	2.242
600	1.986	22.364	0.755	3.723	0.441	34.216	0.578	1.941
800	2.261	36.732	0.892	3.608	0.513	52.098	0.614	3.390
1000	2.107	56.102	0.724	3.699	0.376	73.864	0.539	2.224

Table 5.4: flow-size $s = 4$

5.7 Computing relaxations of mixed linear quadratic programs

If a QQP contains many linear constraints, the solution of the dual problem $(\tilde{D})$ might be quite expensive. We describe a two-stage procedure for generating a convex relaxation that solves a QP in the first stage and computes an SDP-relaxation in the second stage. Consider the following QQP with mixed quadratic and linear constraints:

$$\begin{array}{lll} \min & q_0(x) & \\ \text{s.t.} & q_i(x) \leq 0, & i = 1, \ldots, m \\ & Ax + b \leq 0. & \end{array} \tag{5.10}$$

Define a Lagrangian for (5.10) with respect to quadratic constraints by

$$L_q(x; \mu) = q_0(x) + \mu^T q(x).$$

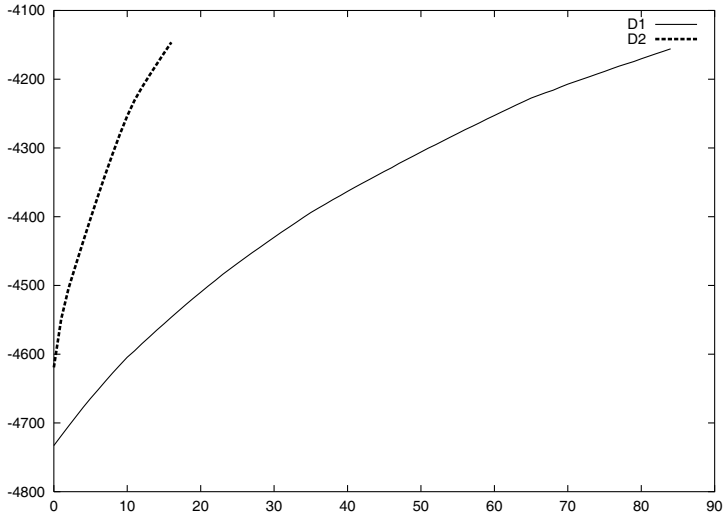

Figure 5.2: Dual values of D_1 and D_2 at serious steps, where $(n, s, l) = (100, 2, 10)$, showing that $D_2(\mu) > D_1(\mu)$

Algorithm 5.1 solves the dual of (5.10) by alternatively computing solutions of the following two subproblems:

$$\begin{array}{ll} \min & L_q(x; \mu^{j-1}) \\ \text{s.t.} & Ax + b \leq 0 \end{array} \tag{5.11}$$

and

$$\max_{\mu \in \mathbb{R}^m_+} \min_{x \in \mathbb{R}^n} L_q(x; \mu) + (\tau^j)^T (Ax + b) \tag{5.12}$$

where $\tau^j \in \mathbb{R}^p_+$ is a dual point related to the constraint $Ax + b \leq 0$ of (5.11).

Proposition 5.15. *The sequence* $\{(\mu^j, \tau^j)\}$ *generated by Algorithm* 5.1 *converges towards a dual solution of* (5.10).

Proof. Let $D(\mu, \tau) = \min_{x \in \mathbb{R}^n} L_q(x; \mu) + \tau^T (Ax + b)$ be the dual function to (5.10). Since the sequence $\{D(\mu^j, \tau^j)\}$ is bounded and monotone, there exists a subsequence (μ^j, τ^j) converging towards a dual point (μ^*, τ^*). From the upper-semicontinuity of the dual function D, it follows that $D(\mu^*, \tau^*) \geq D(\mu^*, \tau)$ for all $\tau \in \mathbb{R}^p_+$ and $D(\mu^*, \tau^*) \geq D(\mu, \tau^*)$ for all $\mu \in \mathbb{R}^m_+$. Since D is concave, it follows that $D(\mu^*, \tau^*) \geq D(\mu, \tau)$ for all $(\mu, \tau) \in \mathbb{R}^m_+ \times \mathbb{R}^p_+$. This proves the statement. $\square$

Set $\mu^0 = 0$.

for $j = 1, \ldots, l$

Compute a dual solution τ^j of the QP (5.11) corresponding to $Ax + b \leq 0$.

Solve the dual problem (5.12) obtaining a dual point μ^j corresponding to $q_i(x) \leq 0$, $i = 1, \ldots, m$.

end for

Algorithm 5.1: Two-phase dual method for mixed linear quadratic programs

Chapter 6

Convex Underestimators

In Section 3.2 we looked at nonlinear convex underestimating-relaxations of MINLPs that are based on replacing nonconvex functions of the original problem with convex underestimators. In order to be efficient for branch-and-bound methods, the underestimators should be tight and cheap. This chapter presents several methods for generating convex underestimators. In particular, a recent underestimating-technique based on Bézier polynomials (Nowak, 1996), and a new sampling method for constructing polynomial underestimators of general nonlinear multivariate black-box functions (Nowak et al., 2003) are presented.

6.1 Interval arithmetic

Constant functions are the simplest type of convex underestimators. Such underestimators can be computed efficiently by using *interval arithmetic* (Moore, 1979; Neumaier, 1990), which is a natural generalization of the standard arithmetic for intervals. If $X = [\underline{x}, \overline{x}]$ and $Y = [\underline{y}, \overline{y}]$ are two intervals in $\mathbb{R}^n$, we define, for any binary operator $\circ$, that

$$X \circ Y = \square\{x \circ y \mid x \in X, y \in Y\},$$

whenever the right-hand side is defined, where

$$\square S = [\inf S, \sup S]$$

denotes the *interval hull* of a set in $\mathbb{R}^n$, i.e. the tightest interval containing S. A monotonicity argument yields

$$X + Y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}],$$

$$X - Y = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$$

for addition and subtraction and

$$X * Y = \square\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\},$$

$$X/Y = \square\{\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\}, \quad \text{if} \quad 0 \notin Y$$

for multiplication and division. We also define elementary functions $\varphi \in \{\text{sqr, sqrt, exp, log, sin, cos, } \dots\}$ of an interval X by

$$\varphi(X) = \square\{\varphi(x) \mid x \in X\}$$

whenever the right-hand side is defined. Depending on the monotonicity properties of φ, the interval $\varphi(X)$ can be computed from the value of φ at the endpoints of X and the interior extremal values. The interval evaluation $f(X)$ of some expression f often overestimates $\text{range}(f, X) = \{f(x) \mid x \in X\}$. Under very mild conditions (Neumaier, 1990) the evaluation satisfies

$$f(X) \subset \text{range}(f, X) + O(\epsilon), \quad \text{if } \overline{x} - \underline{x} = O(\epsilon).$$

This is called the *linear approximation property* of simple interval evaluation. Better enclosures of order $O(\epsilon^2)$ can be obtained by so-called *centered forms* (Neumaier, 1990)

$$f(x) \in f(\xi) + \nabla f(X)(x - \xi) \quad \text{if } x, \xi \in X. \tag{6.1}$$

In addition to bounds on expressions, interval arithmetic provides criteria for verifying solutions of nonlinear systems of equations. It can also be used for convexifying functions (see Section 6.3). In (Krawczyk, 1969) a criterion for checking if a nonlinear system of equations $F(x) = 0$ contains any solutions in an interval X is proposed. Multiplying the vector version of (6.1) by a matrix C and adding x defines the *Krawczyk operator*:

$$K(x) = \xi + CF(\xi) + (C\nabla F(x) - I)(x - \xi).$$

For $\xi \in X$ the operator has the following properties:
(i) Any zero $x \in X$ of F lies in $X \cap K(X, \xi)$.
(ii) If $K(x) = \emptyset$ then X contains no zero of F.
(iii) If $K(x) \in \text{int}\, X$ then X contains a unique zero of F.

Property (iii) can be used to eliminate regions around a local minimizer. Properties (i) and (ii) are useful for box reduction or box elimination. They are used in the global optimization software packages GlobSol (Kearfott, 1996) and Numerica (Van Hentenryck, 1997).

Interval arithmetic can also be used to check convexity. Let $f : X \mapsto \mathbb{R}$ be continuously twice-differentiable on the compact interval X, and let H be a symmetric interval matrix with $\nabla^2 f(x) \in H$ (component-wise) for all $x \in X$. If some symmetric matrix $H_0 \in H$ is positive definite, and all symmetric matrices in H are nonsingular, then they are all positive definite and f is uniformly convex in X.

6.2 Bézier polynomials

Convex underestimators of multivariate polynomials can be obtained in a natural way from Bernstein–Bézier-representations using the so-called convex hull property. Based on this idea, in (Nowak, 1996) the first branch-and-bound algorithm for global optimization of polynomial programs that uses Bernstein–Bézier lower bounds was developed.

Let $i = (i_1, \ldots, i_n)^T \in \mathbb{N}^n$ be a multi-index, $x^i = \prod_{k=1}^n x_k^{i_k}$, $|i| = \sum_{k=1}^n i_k$ and $i! = \prod_{k=1}^n i_k!$. Denote by $S = \mathrm{conv}(\{v_1, \ldots, v_{n+1}\}) \subset \mathbb{R}^n$ a simplex with vertices $v_i \in \mathbb{R}^n$. Any point $x \in S$ can be expressed uniquely in terms of *barycentric coordinates* $y \in \Delta_{n+1}$ by:

$$x = x_S(y) = \sum_{i=1}^{n+1} y_i v_i,$$

where $\Delta_{n+1} \subset \mathbb{R}^{n+1}$ is the standard simplex. Consider a multivariate polynomial of degree r defined by

$$p(x) = \sum_{|i| \le r} a_i x^i.$$

The Bézier-representation of p over a simplex S is given by

$$p(x_S(y)) = \sum_{|i|=r} b_i \cdot B_i^r(y)$$

where $B_i^r(y) = \frac{r!}{i!} \cdot y^i$ are *Bernstein polynomials* and b_i are B-points. The B-points b_i can be computed easily from intermediate points generated by *de Casteljau's algorithm* (Farin, 1986). Since $B_i^r(y) \ge 0$ and $\sum_{|i|=r} B_i^r(y) = 1$ for all $y \in \Delta_{n+1}$, we have the convex-hull property

$$\left\{ \begin{pmatrix} x \\ p(x) \end{pmatrix} \mid x \in S \right\} \subseteq \tilde{P}_{\mathrm{bez}} = \mathrm{conv} \left\{ \begin{pmatrix} i/r \\ b_i \end{pmatrix} \mid |i| = r \right\}.$$

Hence, $\underline{v} = \min_{|i|=r} b_i$ is a lower bound on $p(x)$ over a simplex S. This lower bound is used in (Nowak, 1996).

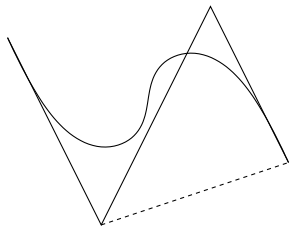

Figure 6.1: Polynomial and Bézier polygon

The following result on the quadratic convergence of B-points is shown in (Dahmen, 1986; Cohen and Schumaker, 1985):

Lemma 6.1. *Let $\mathcal{P} = \{S_1, ..., S_l\}$ be a simplicial partition of a polyhedron $\hat{P} \subset \mathbb{R}^n$ and $p: \mathbb{R}^n \mapsto \mathbb{R}$ be a polynomial of degree r. There exists a constant c that depends only on the restriction of p on $\hat{P}$ such that*

$$|b_{S_j,i} - p(x_{S_j}(i/r))| \leq c \cdot (\mathrm{diam} S_j)^2$$

where $|i| = r$, $1 \leq j \leq l$ and diam$S$ *denotes the diameter of S.*

From Lemma 6.1 it follows that Bézier-bounds are consistent (see Section 13.3.1).

Similarly, a convex underestimator of a multivariate polynomial over an interval $[0, e]$ can be obtained from its Bézier-representation. To this end, consider a multivariate polynomial with degree $l = (l_1, \ldots, l_n)$ of the form

$$p(x) = \sum_{i=0}^{l} a_i x^i.$$

The Bernstein–Bézier representation of p over the standard interval $[0, e]$ is defined by

$$p(x) = \sum_{i=0}^{l} b_i \cdot B_i(x)$$

where B_i are Bernstein polynomials defined by $B_i(x) = \begin{pmatrix} l \\ i \end{pmatrix} x^i (1-x)^{l-i}$ and $\begin{pmatrix} l \\ i \end{pmatrix} = \prod_{k=1}^{n} \begin{pmatrix} l_k \\ i_k \end{pmatrix}$. Since Bernstein polynomials are nonnegative over $[0, e]$ and form a partition of 1, we have the convex hull property

$$\left\{ \begin{pmatrix} x \\ p(x) \end{pmatrix} \mid x \in [0, e] \right\} \subseteq P_{\mathrm{bez}} = \mathrm{conv} \left\{ \begin{pmatrix} i/l \\ b_i \end{pmatrix} \mid 0 \leq i \leq l \right\}.$$

From this property we get the following piecewise linear convex underestimator, which is illustrated in Figure 6.1,

$$\underline{p}(x) = \min\{y \mid (x, y) \in P_{\mathrm{bez}}\}.$$

Based on this underestimator, in (Garloff et al., 2002; Garloff and Smith, 2003) an efficient method for computing affine underestimators of multivariate polynomials over an interval is proposed.

6.3 α-underestimators

Adjiman and Floudas (Adjiman and Floudas, 1997) proposed a method to convexify a continuously twice-differentiable function f by adding the quadratic form $\alpha^T r(x)$ to f, where $\alpha \in \mathbb{R}^n$ and

$$r(x) = \mathrm{Diag}(x - \underline{x})(x - \overline{x}). \tag{6.2}$$

The resulting function $\breve{f}(x) = f(x) + \alpha^T r(x)$ is called a *convex α-underestimator* of f over $[\underline{x}, \overline{x}]$, if the Hessian $\nabla^2 \breve{f}(x) = \nabla^2 f(x) + 2\,\mathrm{Diag}(\alpha)$ is positive semidefinite over $[\underline{x}, \overline{x}]$, and $\alpha \geq 0$. Since $\alpha^T r(x)$ is zero at the vertices of $[\underline{x}, \overline{x}]$, $\breve{f}$ coincides with f at those points. An optimal convex α-underestimator can be computed by solving the program

$$\max_{\alpha \in \mathcal{A}} \min_{x \in [\underline{x}, \overline{x}]} f(x) + \alpha^T r(x)$$

where

$$\mathcal{A} = \{\alpha \in \mathbb{R}^n \mid \alpha \geq 0,\ \nabla^2 f(x) + 2\,\mathrm{Diag}(\alpha) \succcurlyeq 0\, \forall x \in [\underline{x}, \overline{x}]\}.$$

Since finding α that solves such a program is usually too difficult, the following method is used in Adjiman et al., 1998.

Lemma 6.2 (scaled Gerschgorin theorem). *For any vector $d > 0$ and a symmetric interval matrix $A = [\underline{A}, \overline{A}]$, the vector α is defined as*

$$\alpha_i = \max\{0, -\frac{1}{2}(\underline{a}_{ij} - \sum_{j \neq i} |a|_{ij} \frac{d_j}{d_i})\}$$

where $|a|_{ij} = \max\{|\underline{a}_{ij}|, |\overline{a}_{ij}|\}$. Then for all $A \in [\underline{A}, \overline{A}]$, the matrix $A + 2\,\mathrm{Diag}(\alpha)$ is positive semi-definite.

From this it follows immediately:

Corollary 6.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice-differentiable function and $A = [\underline{A}, \overline{A}]$ be the interval Hessian of f at $[\underline{x}, \overline{x}]$, i.e. $\underline{A} \leq \nabla^2 f(x) \leq \overline{A}$ for all $x \in [\underline{x}, \overline{x}]$. Then for $\alpha \in \mathbb{R}^n$ computed as in Lemma 6.2 the function*

$$\underline{f}(x) = f(x) + \alpha^T r(x)$$

is a convex underestimator of f over $[\underline{x}, \overline{x}]$, i.e. $\underline{f}$ is convex on $[\underline{x}, \overline{x}]$ and $\underline{f}(x) \leq f(x)$ for all $x \in [\underline{x}, \overline{x}]$, where r is defined in (6.2).

Note that the vector α computed in Corollary 6.3 is not necessarily zero if f is convex over $[\underline{x}, \overline{x}]$. On the other hand, if $\alpha = 0$ then f is convex over $[\underline{x}, \overline{x}]$.

In (Nowak et al., 2003) a heuristic α-underestimator is proposed by computing the vector α by using a sampling technique. This method can be applied to black-box functions, which are provided, for example, by the modeling systems

GAMS (GAMS, 2003) and AMPL (Fourer et al., 1993). Let $w = \overline{x} - \underline{x}$ be the diameter vector of the interval, and

$$\rho = \min_{x \in S} \lambda_1(\mathrm{Diag}(w) \nabla^2 f(x) \, \mathrm{Diag}(w))$$

be a guess for the minimum eigenvalue of the Hessian of f over the box $[\underline{x}, \overline{x}]$, where $S \subset [\underline{x}, \overline{x}]$ is a finite sample set. The scaling by $\mathrm{Diag}(w)$ comes from the affine transformation $\theta(x) = \mathrm{Diag}(w)x + \underline{x}$ that maps the standard interval $[0, e]$ onto $[\underline{x}, \overline{x}]$. Then α is computed according to

$$\alpha = \frac{1}{2} \max\{0, -\rho\} \, \mathrm{Diag}(w)^{-2} e.$$

6.4 CGU-underestimators

Phillips, Rosen and Walke (Phillips et al., 1995) proposed the following heuristic method for approximating the convex envelope of a nonlinear function f by a quadratic function. Here, their method is modified slightly to reduce the absolute value of the smallest eigenvalue of the obtained quadratic underestimator. Let $S \subset [\underline{x}, \overline{x}]$ be a finite sample set, and define the quadratic function

$$q(x; a, b, c) = c + 2b^T x + x^T \, \mathrm{Diag}(a) x,$$

where $a, b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Then $q(\cdot; a, b, c)$ is convex, if and only if $a \geq 0$. The tightest quadratic convex underestimator $q(\cdot; a, b, c)$ over the set S is called *CGU-underestimator*, which stands for convex global underestimator. It is provided by the program

$$\text{(CGU)} \qquad \begin{array}{lll} \min_{a,b,c} & \displaystyle\sum_{x \in S} f(x) - q(x; a, b, c) + \delta e^T a & \\ \text{s.t.} & f(x) \geq q(x; a, b, c), & \forall x \in S \\ & a \geq 0, & \end{array}$$

where $\delta > 0$ is a small penalty parameter. Since q depends linearly on a, b, c, problem (CGU) is a linear program. The term $\delta e^T a$ reduces the absolute value of the smallest eigenvalue of $\mathrm{Diag}(a)$ in the case where (CGU) is degenerated. The quality of a CGU-underestimator depends strongly on the sample set S. In general, it cannot be guaranteed that the CGU-underestimator is a true underestimator over $[\underline{x}, \overline{x}]$.

6.5 Convexified polynomial underestimators

A further development of the CGU-underestimator is the sampling method presented in (Nowak et al., 2003). Similarly as the CGU-underestimator, this method requires only function evaluations, and can be therefore applied to black-box functions for which no analytic expressions are known.

Instead of constructing the convex underestimator directly, a two-step approach is proposed. In the first step, a given function f is underestimated by a (possibly nonconvex) multivariate polynomial p. In the second step, p is convexified by either an α-underestimator (Section 6.3) or a Bézier-underestimator (Section 6.2).

The direct application of the α-underestimator technique to the original function would also give a convex underestimator. However, the proposed *polynomial underestimator* is often tighter because the α-convexification depends only on the curvature of the function and not on the function behavior. For more clarification see the example in Figure 6.2, where f is the original function, $\breve{f}$ the α-convexification of f, q the polynomial underestimator, and $\breve{q}$ the α-convexification of q.

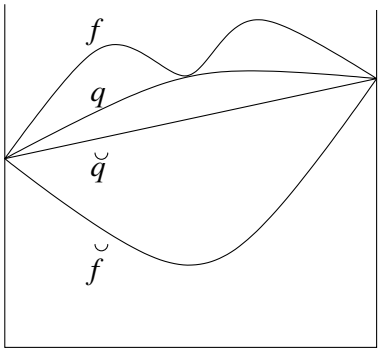

Figure 6.2: α-underestimator versus the convexification of the polynomial underestimator.

Given a nonconvex function $f : \mathbb{R}^n \mapsto \mathbb{R}$, a polynomial underestimator p is constructed over $[\underline{x}, \overline{x}]$ defined by

$$p(x; a) = \sum_{i \in I} a_i x^i \tag{6.3}$$

where $I \subset \mathbb{N}^n$ is a finite set and the multivariate monomial x^i is defined as in Section 6.2. The degree of the polynomial p is the number $d = \max_{i \in I} |i|$. In the numerical experiments shown in this book $d = 2$ is used. Polynomials of a degree of higher than 2 can be reformulated to be quadratic using additional variables and quadratic functions, for example, $x_i x_j x_k$ can be replaced by $x_k x_l$ with the addition of the variable $x_l \in [\underline{x}_l, \overline{x}_l]$ and the constraint $x_l = x_i x_j$. The bounds for the new variables can be computed by using the bounds on the original variables.

The index set I in (6.3) is chosen according to the sparsity pattern of f, i.e. the Hessians $\nabla^2 p$ and $\nabla^2 f$ have the same zero entries. In order to determine the coefficients $a_i, i \in I$, of the polynomial underestimator (6.3), the following

program is solved:

$$\begin{array}{ll} \min_a & \sum_{j=1}^{r} |f(x_j) - p(x_j; a)| \\ \text{s.t.} & p(x_j; a) \leq f(x_j), \qquad j = 1, \ldots, r \end{array}$$

where $x_j \in [\underline{x}, \overline{x}], j = 1, \ldots, r$, are sample points. This problem can be formulated equivalently as the linear program:

$$\begin{array}{ll} \min_{a,t} & \sum_{j=1}^{r} f(x_j) - t_j \\ \text{s.t.} & f(x_j) \geq t_j \geq p(x_j; a), \quad j = 1, \ldots, r. \end{array} \tag{6.4}$$

The quality of the polynomial underestimator p of f obtained by solving (6.4) depends strongly on the sample set $S = \{x_1, \ldots, x_r\}$. If f is concave and S includes the set of vertices $\text{vert}([\underline{x}, \overline{x}])$, the underestimator p is rigorous, i.e. $p(x) \leq f(x)$ for all $x \in [\underline{x}, \overline{x}]$. A possible definition of the sample set is $S = \text{vert}([\underline{x}, \overline{x}]) \cup S_{\min} \cup \{(\underline{x} + \overline{x})/2\}$, where $S_{\min}$ is the set of minimizers $S_{\min}$ of f over $[\underline{x}, \overline{x}]$. This definition of S guarantees that the global minima of p and f over $[\underline{x}, \overline{x}]$ are identical. However, it does not guarantee that p is a true underestimator. Since the nonlinear convex underestimating-relaxation (3.4) is only used as a means for constructing a polyhedral relaxation (see Section 7.2), we do not need to produce true underestimators. If it is not sure that an underestimator is rigorous, it can be replaced by Knapsack cuts (see Section 7.1.2).

6.5.1 Rigorous underestimators

Rigorous polynomial underestimators can be computed using interval arithmetic. Given a polynomial underestimator p of f constructed by the above sampling technique, interval arithmetic is used to determine an interval $[\underline{\delta}, \overline{\delta}]$ containing the function values $\delta(x) = f(x) - p(x)$ for $x \in [\underline{x}, \overline{x}]$. In order to avoid clustering effects, the interval should be computed by a central form (Neumaier, 2004). Then

$$\underline{p}(x) = p(x) + \underline{\delta}$$

is a rigorous polynomial underestimator of f over $[\underline{x}, \overline{x}]$. If the approximation error $\delta(x)$ is small over $[\underline{x}, \overline{x}]$ and the lower bound $\underline{\delta}$ is tight, then $\underline{p}$ is a tight underestimator.

6.5.2 Restricted sampling

If the function f is highly nonlinear, the approximation error $\delta(x)$ caused by the aforementioned polynomial underestimator can be quite large. Moreover, it is

possible that f is not defined for all $x \in [\underline{x}, \overline{x}]$. In this case, the underestimation can be improved by sampling over a smaller region $Z \subset [\underline{x}, \overline{x}]$ containing the feasible set of the given MINLP. Let $Z = \{x \in \mathbb{R}^n \mid h(x) \leq 0\}$, where $h(x) = b^T x_L + f(x_N) \leq 0$ is an inequality constraint of the given MINLP, and L and N are the index sets of linear and nonlinear variables, respectively.

If a sample point $\hat{x}$ is not in Z, i.e. $b^T \hat{x}_L + f(\hat{x}_N) > 0$, $\hat{x}_L$ is replaced by $\tilde{x}_L = \text{argmin}_{x \in [\underline{x}, \overline{x}]} b^T x_L$. If still $b^T \tilde{x}_L + f(\hat{x}_N) > 0$, $f(x_N)$ is minimized by a projected-gradient algorithm starting from $\hat{x}_N$, to find a point $\tilde{x}_N$ such that $b^T \tilde{x}_L + f(\tilde{x}_N) \leq 0$. The same technique can be used to generate sample points in the set $Z = \{x \in \mathbb{R}^n \mid h(x) = 0\}$, where $h(x) = 0$ is an equality constraint of the MINLP.

Chapter 7

Cuts, Lower Bounds and Box Reduction

In this chapter several cuts for improving polyhedral and nonlinear relaxations are presented. The generation of these cuts is based on an extended reformulation of a given MINLP with linear coupling constraints. Most of the cuts are computed by solving MINLP separation problems. Other cuts are generated by linearizing convex constraints.

On the basis of polyhedral and nonlinear relaxations, NLP, MINLP, LP and dual lower bounding methods are proposed and analyzed. Furthermore, several box reduction procedures are proposed that use the aforementioned lower-bounding methods. Numerical results for MINLPs are presented that show that the proposed box reduction procedures are able to reduce the initial bounding box as well as to fix binary variables.

7.1 Valid cuts

Consider the extended reformulation of a general MINLP with linear coupling constraints defined in Section 2.5:

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in G_k, \quad k = 1, \ldots, p \end{array} \tag{7.1}$$

where

$$G_k = \{x_{J_k} \in [\underline{x}_{J_k}, \overline{x}_{J_k}] \mid g_{i,k}(x_{J_k}) \leq 0,\, i \in M_k,\, x_{J_k \cap B} \text{ binary}\}. \tag{7.2}$$

In the following, we study several *valid cuts* for (7.1), i.e. redundant constraints that do not cut off any parts of the feasible set S of (7.1) containing solution

points. Valid cuts are used to improve a polyhedral or nonlinear relaxation of (7.1). If they cut off all solution points of a relaxation, the improvement is strict.

A valid cut can be computed by solving a *separation problem* that yields an upper bound $\overline{a}$ on the support function $\sigma_{\tilde{S}}(a) = \max\{a^T x \mid x \in \tilde{S}\}$ for a given vector $a \in \mathbb{R}^n$, where $\tilde{S} \supset S$ is a relaxation of the feasible set S of (7.1). Then $a^T x \leq \overline{a}$ is a valid cut.

7.1.1 Linearization cuts

A *linearization cut* is based on linearizing an active constraint of an extended convex underestimating relaxation defined by

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in \breve{G}_k, \quad k = 1, \ldots, p \end{array} \tag{7.3}$$

where

$$\breve{G}_k = \{x_{J_k} \in [\underline{x}_{J_k}, \overline{x}_{J_k}] \mid \breve{g}_{i,k}(x_{J_k}) \leq 0,\, i \in M_k\} \tag{7.4}$$

and $\breve{g}_{i,k}$ are convex underestimators of $g_{i,k}$ over $[\underline{x}_{J_k}, \overline{x}_{J_k}]$. Let $\hat{x} \in [\underline{x}, \overline{x}]$ be a trial point and denote by $\mathcal{A}_k$ the active set of the nonlinear constraints of (7.3) at $\hat{x}$, i.e. $\mathcal{A}_k = \{i \in M_k \mid \breve{g}_{i,k}(\hat{x}_{J_k}) = 0\}$. By linearizing the active constraints of (7.3) at $\hat{x}$, we obtain the following linearization cuts

$$\nabla \breve{g}_{i,k}(\hat{x}_{J_k})^T (x_{J_k} - \hat{x}_{J_k}) \leq 0, \quad i \in \mathcal{A}_k. \tag{7.5}$$

If the trial point $\hat{x}$ is a minimizer of the convex relaxation (7.3), then the optimal value of a polyhedral relaxation obtained from adding the cuts (7.5) is greater than or equal to the optimal value of (7.3). Note that linearization cuts are only valid if $\breve{g}_{i,k}$ is a true convex underestimator of $g_{i,k}$ over the feasible set of (7.1). This cannot be guaranteed for a heuristic convex underestimating-method, such as the method described in Section 6.5.

7.1.2 Knapsack cuts

If a convex underestimator $\breve{g}_{i,k}$ is a bad approximation of a nonconvex function $g_{i,k}$, the constraint $g_{i,k}(x) \leq 0$ might be strongly violated at a trial point $\hat{x}$. In this case, the following nonlinear *Knapsack cut* could be stronger:

$$a^T x_{J_k} \geq c(a) \tag{7.6}$$

where

$$c(a) = \min\{a^T x_{J_k} \mid g_{i,k}(x_{J_k}) \leq 0,\, x_{J_k} \in \hat{G}_k,\, x_{J_k \cap B} \text{ binary}\}, \tag{7.7}$$

$\hat{G}_k \supseteq G_k$ is a polyhedral outer approximation and $a \in \mathbb{R}^{|J_k|}$ is a given direction. In the simplest case, the polyhedral relaxation $\hat{G}_k$ is defined by $\hat{G}_k = [\underline{x}_{J_k}, \overline{x}_{J_k}]$.

However, if $g_{i,k}$ is not defined for all $x \in [\underline{x}_{J_k}, \overline{x}_{J_k}]$, $\hat{G}_k$ has to be defined by a smaller set. Note that the global optimization problem (7.7) can be simplified if $g_{i,k}$ is a separable function of the form

$$g_{i,k}(x) = b_j x_j + \sum_{i \in K} s_i(x_i),$$

where $K \cup \{j\} \subseteq J_k$. Assuming that the constraint $g_{i,k}(x) \leq 0$ is active at a solution point of problem (7.7), we have

$$c(a) = \min\{a_K^T x_K - \frac{a_j}{b_j} \sum_{i \in K} s_i(x_i) \mid x \in \hat{G}_k,\ x_{J_k \cap B} \text{ binary}\}.$$

If $\hat{G}_k = [\underline{x}_{J_k}, \overline{x}_{J_k}]$, this problem is equivalent to

$$c(a) = \sum_{i \in K} \min\{a_i x_i - \frac{a_j}{b_j} s_i(x_i) \mid x_i \in [\underline{x}_i, \overline{x}_i],\ x_{\{i\} \cap B} \text{ binary}\}. \tag{7.8}$$

For minimizing the $|K|$ univariate functions of (7.8) over a box, a specialized algorithm or a general global optimization method, as described in Chapter 10, can be used. If the constraint $g_{i,k}(x) \leq 0$ is not active at a solution point of (7.6), the minimum of (7.7) is attained at a vertex of $[\underline{x}_{J_k}, \overline{x}_{J_k}]$ and can be computed by

$$c(a) = \min\{a^T x_{J_k} \mid x_{J_k} \in [\underline{x}_{J_k}, \overline{x}_{J_k}]\}.$$

Three methods are used for defining the direction a in (7.6):

(i) The direction is defined by $a = \nabla g_{i,k}(\hat{x})$. This definition leads to consistent LP-bounds, which are required for the convergence of branch-and-bound methods (Section 13.3.2).

(ii) The direction is defined by $a = \nabla \breve{g}_{i,k}(\hat{x})$.

(iii) Let v be a minimizer of $g_{i,k}$ over the vertex set, i.e. $v = \text{argmin}\{g_{i,k}(x) \mid x \in \text{vert}([\underline{x}_{J_k}, \overline{x}_{J_k}])\}$. The direction is defined by

$$a_j = (g_{i,k}(v) - g_{i,k}(v + \sigma_j e_j(\overline{x}_j - \underline{x}_j)))/(\overline{x}_j - \underline{x}_j),$$

where $\sigma_j = 1$ if $v_j = (\underline{x}_{J_k})_j$ and $\sigma_j = -1$ if $v_j = (\overline{x}_{J_k})_j$, $j = 1, \ldots, |J_k|$. This definition gives an estimate for the affine convex envelope if $g_{i,k}$ is concave.

7.1.3 Interval-gradient cuts

Let $g(x) \leq 0$ be a nonlinear inequality constraint, where g is a continuously differentiable function over $[\underline{x}, \overline{x}]$. Denote the interval-gradient of g over $[\underline{x}, \overline{x}]$ by $[\underline{d}, \overline{d}]$, i.e. $\nabla g(x) \in [\underline{d}, \overline{d}]$ for all $x \in [\underline{x}, \overline{x}]$. An *interval-gradient cut* to g with respect to a point $\hat{x} \in [\underline{x}, \overline{x}]$ is defined by

$$\underline{g}(x) = g(\hat{x}) + \min_{d \in [\underline{d}, \overline{d}]} d^T(x - \hat{x}) \leq 0.$$

From the intermediate value theorem it follows that the interval-gradient cut is valid. Furthermore, we have $\underline{g}(\hat{x}) = g(\hat{x})$. From this it is shown in Section 8.5 that interval-gradient cuts can be used to verify global optimality of strict local minimizers. An interval-gradient cut can be formulated as the following mixed-integer linear constraint:

$$\begin{aligned} g(\hat{x}) + \sum_{i=1}^{n} \underline{d}_i y_i^+ - \overline{d}_i y_i^- &\leq 0 \\ x - \hat{x} &= y^+ - y^- \\ y_i^+ &\leq z_i(\overline{x}_i - \underline{x}_i), \qquad i = 1, \ldots, n \\ y_i^- &\leq (1 - z_i)(\overline{x}_i - \underline{x}_i), \qquad i = 1, \ldots, n \\ y^+ \geq 0,\ y^- &\geq 0 \\ z_i &\in \{0, 1\}^n. \end{aligned}$$

Note that a slack variable (y_i^+, y_i^-, z_i) has only to be introduced if x_i is a nonlinear variable of g. Assuming that g has the form $g(x) = b^T x_L + s(x_N)$, where s is a nonlinear function and N is the index set of nonlinear variables, an interval-gradient cut consists of $1 + 3|N|$ linear constraints, $2|N|$ simple box-constraints, and $|N|$ binary constraints. Interval-gradient cuts for all-quadratic programs are proposed in Boddy and Johnson, 2003. By relaxing the binary constraints of an interval-gradient cut the following linear cut is defined[1]:

$$\begin{aligned} g(\hat{x}) + \sum_{i=1}^{n} \underline{d}_i y_i^+ - \overline{d}_i y_i^- &\leq 0 \\ x - \hat{x} &= y^+ - y^- \\ y^+ &\leq \overline{x} - \underline{x} \\ y^- &\leq \overline{x} - \underline{x} \\ y^+ \geq 0,\ y^- &\geq 0. \end{aligned}$$

Figure 7.1 shows linearization, Knapsack and interval-gradient cuts.

7.1.4 Lagrangian cuts

Deeper cuts can be generated by solving Lagrangian subproblems. Let $L_k(x_{J_k}; \mu) = a_k(\mu)^T x_{J_k}$ with $a_k(\mu) = c_{J_k} + A_{J_k}^T \mu$ be the k-th partial Lagrangian to (7.1) and $\hat{\mu}$ be a dual point related to the linear coupling constraint $Ax + b \leq 0$. A *Lagrangian cut* is defined by

$$a_k(\hat{\mu})^T x_{J_k} \geq D_k(\hat{\mu}) \tag{7.9}$$

where $D_k(\hat{\mu}) = \min_{x \in G_k} L_k(x; \hat{\mu})$ and G_k is defined as in (7.2). Lagrangian cuts are used in the column generation method Algorithm 4.8 to generate a polyhedral outer approximation of (7.1).

[1] This linear cut is implemented in LaGO

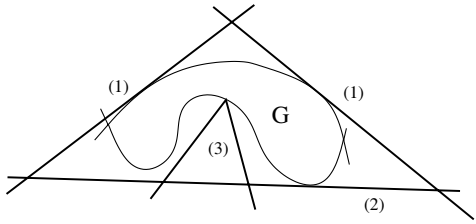

Figure 7.1: Linearization (1), Knapsack(2) and interval-gradient (3) cuts

7.1.5 Level cuts

Let $\overline{v}$ be an upper bound of the optimal value of (7.1). Such a bound can be computed by $\overline{v} = c^T\hat{x} + c_0$ if a feasible point $\hat{x}$ of (7.1) is available. Otherwise, it can be computed by maximizing $c^Tx + c_0$ over a convex relaxation of the feasible set of (7.1). Then the following *linear level cut* is valid:

$$c^Tx + c_0 \leq \overline{v}. \tag{7.10}$$

A *nonlinear level cut* can be formulated by:

$$L(x; \hat{\mu}) \leq \overline{v},$$

where $L(\cdot, \hat{\mu})$ is a convex Lagrangian $L(\cdot, \hat{\mu})$, which can be related to a semidefinite or convex underestimating-relaxation.

7.1.6 Other valid cuts

1. In the presence of a large duality gap, val(7.1) − val(3.13), *deeper cuts* can be computed by solving separation problems involving several variable blocks, defined by:

$$\begin{array}{lll} \min & \displaystyle\sum_{k\in K} L_k(x_{J_k}; \hat{\mu}) & \\ \text{s.t.} & g_{i,k}(x_{J_k}) \leq 0, & i \in M_k,\ k \in K \\ & x_{J_k} \in [\underline{x}_{J_k}, \overline{x}_{J_k}], & k \in K \\ & x_{J_k \cap B} \text{ binary} & \end{array} \tag{7.11}$$

   where $K \subseteq \{1, \ldots, p\}$ is a *super-block* and $\hat{\mu}$ is a given dual point. Let δ be a lower bound on the optimal value of (7.11). Then

$$\sum_{k\in K} L_k(x_{J_k}; \hat{\mu}) \geq \delta$$

   is a valid cut. In order to determine a super-block automatically, a partition

of the graph (V_b, E_b) can be computed that is defined by the vertices $V_b = \{1, \ldots, p\}$ and the edges

$$E_b = \{\{k, l\} \mid J_k \cap V_i \neq \emptyset \text{ and } J_l \cap V_i \neq \emptyset \text{ for some } i \in \{0, \ldots, m\}\},$$

where the index set V_i corresponds to the variables entering the objective or constraint function h_i, $i = 0, \ldots, m$, of the given MINLP. It is possible to strengthen the separation problem (7.11) by adding disjunctive cuts, as proposed in (Vecchietti et al., 2003).

2. The following class of valid MINLP cuts is proposed in (Tawarmalani and Sahinidis, 1999). Let $L(\cdot; \mu) = f(x) + \mu^T g(x)$ be the Lagrangian of $\min\{f(x) \mid g(x) \leq 0\}$, $\underline{f} = \min_{x \in X} L(x; \mu)$ be a lower bound, and $\overline{f} = f(\hat{x})$ be an upper bound, where $\hat{x}$ is a feasible point. From $\underline{f} \leq f(x) + \mu^T g(x) \leq \overline{f} + \mu^T g(x)$ we get the valid cut

$$g_i(x) \geq -\frac{1}{\mu_i}(\overline{f} - \underline{f}).$$

3. Other cuts can be constructed by multiplication of two constraints. Let $g_i(x) \leq 0$ and $g_j(x) \leq 0$ be two given inequality constraints of a MINLP. Then $-g_i(x) \cdot g_j(x) \leq 0$ is a valid cut.

7.2 Initialization of polyhedral relaxations

The presented cuts are used to initialize and update a *polyhedral relaxation* of (7.1) of the form:

$$\begin{array}{lll} \min & c^T x + c_0 & \\ \text{s.t.} & Ax + b \leq 0 & \\ & c^T x + c_0 \leq \overline{v} & \\ & b^T x_{J_k} \leq \overline{b}, & (b, \overline{b}) \in N_k,\ k = 1, \ldots, p \\ & x \in [\underline{x}, \overline{x}] & \end{array} \tag{7.12}$$

where $N_k \subset \mathbb{R}^n \times \mathbb{R}$ is a finite set. Algorithm 7.1 builds (7.12) by, first, constructing an extended reformulation (7.3) of the nonlinear convex underestimating-relaxation (3.4), and, second, by adding cuts. Since the linearized active constraints at a solution point of (3.4) are included in the resulting polyhedral relaxation, (7.12) is as least as strong as (3.4), i.e. val(7.12) $\geq$ val(3.4).

7.3 Lower bounds

Based on convex and Lagrangian relaxations, several lower bounding methods for the MINLP (7.1) are proposed. These bounds are used for reducing the box $[\underline{x}, \overline{x}]$ (see Section 7.4) as well as in branch-and-bound methods (see Chapter 13).

Construct an extended convex relaxation (7.3) by reformulating the convex relaxation (3.4).

Initialize the polyhedral relaxation (7.12) by setting $N_k = \emptyset$, $k = 1, \ldots, p$.

Compute a solution $\hat{x}$ of (7.3).

If a nonconvex constraint $g_{i,k}$ is strongly violated at $\hat{x}$, or an active underestimator $\breve{g}_{i,k}$ is not rigorous, replace $\breve{g}_{i,k}$ by a Knapsack cut (7.6).

Add linearization cuts (7.5) at $\hat{x}$.

Add Lagrangian cuts generated by Algorithm 4.8.

Compute an upper bound $\overline{v}$ of the optimal value of the MINLP (7.1) and add the level cut (7.10) to (7.12).

Algorithm 7.1: Initialization of a polyhedral relaxation

7.3.1 NLP-bounds

A NLP-bound of a MINLP is defined by solving the convex nonlinear underestimating-relaxation (3.4), i.e.

$$\underline{v}_{\text{NLP1}} = \text{val}(3.4). \tag{7.13}$$

A further NLP-bound is based on Lagrangian decomposition of (7.3). Let $L_k(x_{J_k}; \mu) = c_{J_k}^T x_{J_k} + \mu^T A_{J_k} x_{J_k}$ be a partial Lagrangian of both (7.3) and (7.1). Let $\hat{\mu}$ be a dual solution point of the polyhedral outer approximation (7.12) or inner approximation (4.16) related to the linear coupling constraints $Ax + b \leq 0$. Then a lower bound to (7.1) is defined by

$$\underline{v}_{\text{NLP2}} = \breve{D}(\hat{\mu}) = c_0 + \hat{\mu}^T b + \sum_{k=1}^{p} \min_{y \in \breve{G}_k} L_k(y; \hat{\mu}), \tag{7.14}$$

where $\breve{G}_k$ is the feasible set of the k-th Lagrangian subproblem to the extended convex relaxation (7.3) defined in (7.4).

Observation 7.1. *Since* val(3.4) = val(7.3) *and* $\breve{D}(\hat{\mu}) \leq$ val(7.3)*, we have* $\underline{v}_{NLP1} \geq \underline{v}_{\text{NLP2}}$.

7.3.2 MINLP-bounds

Similarly, a MINLP-bound for (7.1) is defined by

$$\underline{v}_{MINLP} = D(\hat{\mu}) = c_0 + \hat{\mu}^T b + \sum_{k=1}^{p} \min_{y \in G_k} L_k(y; \hat{\mu}) \tag{7.15}$$

where G_k defined in (7.2) is the feasible set of the k-th Lagrangian subproblem to (7.1). Again, $\hat{\mu}$ is computed by solving a linear relaxation (7.12) or (4.16), since maximizing the dual function D is in general too difficult.

Observation 7.2. *From* $D(\mu) \geq \breve{D}(\mu)$ *we get* $\underline{v}_{\text{MINLP}} \geq \underline{v}_{\text{NLP2}}$.

The nonconvex Lagrangian subproblems in (7.15) can be solved by any global optimization algorithm (see Chapter 10).

7.3.3 Dual bounds

Stronger bounds are dual bounds defined by

$$\underline{v}_{\text{dual}} = \text{val}(\text{Dual}(7.1)).$$

Since $\underline{v}_{\text{dual}} \geq D(\mu)$ for all $\mu \in \mathbb{R}^m_+$, we have $\underline{v}_{\text{dual}} \geq \underline{v}_{\text{MINLP}}$. Dual bounds can be computed by using the column generation method Algorithm 4.8, or by any other dual method described in Chapter 4.

7.3.4 LP-bounds

A linear programming bound to (7.1) is defined by solving the polyhedral relaxation (7.12), i.e.

$$\underline{v}_{\text{LP}} = \text{val}(7.12). \tag{7.16}$$

In Section 13.3.2 it is shown that LP-bounds are consistent if Knapsack cuts of type (i) are used, which are defined in Section 7.1.2.

Observation 7.3. *Since* $\text{val}(7.12) = \max_{\mu \in \mathbb{R}} \min_{y \in \hat{G}_k} L_k(y; \mu)$, *it follows that*

$$\underline{v}_{\text{LP}} \geq D(\hat{\mu}) = \underline{v}_{\text{MINLP}},$$

if the Lagrangian cuts $L_k(x_{J_k}; \hat{\mu}) \geq D_k(\hat{\mu}) = \min_{y \in G_k} L_k(y; \hat{\mu})$, $k = 1, \ldots, p$, *are added to* (7.12).

Remark 7.4. LP-bounds are in particular attractive, since they can be computed very fast. From Observation 7.3 it follows, that the addition of Lagrangian cuts related to a near optimal dual point gives an LP-bound that is almost as strong as a dual bound $\underline{v}_{\text{dual}}$.

Remark 7.5. If the evaluation of all partial dual functions is expensive, it may be more efficient to solve only some Lagrangian subproblems. Assume that the following Lagrangian cuts are added to the polyhedral outer approximation (7.12):

$$L_k(x_{J_k}; \hat{\mu}) \geq D_k(\hat{\mu}), \quad k \in K,$$

where $K \subset \{1, \ldots, p\}$ and D_k is a partial dual function to (7.1). Similarly to Observation 7.3, it can be shown that

$$\underline{v}_{\mathrm{LP}} \geq \sum_{k \in K} D_k(\hat{\mu}) + \sum_{k \in \{1,\ldots,p\} \setminus K} \min_{y \in \hat{G}_k} L_k(y; \hat{\mu}).$$

The set K can be defined, for example, by those subproblems for which the gap between an inner and outer approximation is large, i.e.

$$K = \{k \in \{1, \ldots, p\} \mid \tilde{R}_k(\hat{\mu}) \geq \frac{\delta}{p}(|\operatorname{val}(7.12)| + 1)\},$$

where $\tilde{R}_k(\hat{\mu}) = \min_{y \in \operatorname{conv}(W_k)} L_k(y; \hat{\mu}) - \min_{y \in \hat{G}_k} L_k(y; \hat{\mu})$ is an estimate for the reduced cost, as proposed in Remark 4.17, $\delta \in (0, 1)$, and $W_k \subset \mathbb{R}^{|J_k|}$ is a set of admissible inner approximation points defined in Section 4.3.

7.4 Box reduction

In practice, the bounding box $[\underline{x}, \overline{x}]$ of a given MINLP may be quite large. In this case, the quality of the convex underestimators and cuts may be bad. This drawback might be prevented if a box reduction procedure is applied. Reducing the box may also diminish the duality gap, since

$$\min_{x \in G \cap X'} L(x; \mu) \geq \min_{x \in G \cap X} L(x; \mu)$$

for two intervals X', X with $X' \subset X$. Box reduction techniques for MINLP were first presented in (Ryoo and Sahinidis, 1996). In the following, box reduction procedures are described based on the lower bounds presented in Section 7.3.

Let S be the feasible set of the given MINLP and $\breve{S} \supset S$ be a convex outer approximation of S. A box reduction procedure is defined by replacing the box $[\underline{x}, \overline{x}]$ by the interval $X = [\underline{x}, \overline{x}] \cap \Box\breve{S}$, where $\Box\breve{S}$ is the *interval hull* of $\breve{S}$, i.e. the smallest box containing $\breve{S}$.

Denote by S_1, S_2, S_3 and S_4 the feasible set of the convex relaxation (3.4), the extended convex relaxation (7.3), the extended MINLP (7.1) and the polyhedral relaxation (7.12), respectively. Consider the optimization problem:

$$(\mathrm{B}_k[a]) \qquad \min \quad \{a^T x \mid x \in S_k\}.$$

Then for the interval hull $\Box S_k = [\underline{x}^*_k, \overline{x}^*_k]$ we have

$$\underline{x}^*_{k,i} = \operatorname{val}(B_k[e_i]) \quad \text{and} \quad \overline{x}^*_{k,i} = -\operatorname{val}(B_k[-e_i]), \quad i = 1, \ldots, n.$$

Define a lower bound $\underline{v}_k(a)$ of $\mathrm{val}(B_k[a])$ for $k \in \{1,2,3,4\}$ as in (7.13), (7.14), (7.15) and (7.16) respectively. Then for $k \in \{1,2,3,4\}$ a box reduction procedure is defined by replacing the box $[\underline{x}, \overline{x}]$ by the interval $X_k = [\underline{x}, \overline{x}] \cap [\underline{v}_k, \overline{v}_k]$, where $v_{k,i} = \underline{v}_k(e_i)$ and $\overline{v}_{k,i} = -\underline{v}_k(-e_i)$, $i = 1, \ldots, n$ (see Figure 7.2).

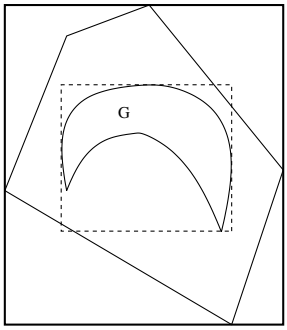

Figure 7.2: Interval hull and reduced box

Note that $\Box S_1 = [\underline{v}_1, \overline{v}_1]$. For $k \in \{2,3,4\}$, the above box reduction procedures can be repeated with respect to the modified box $[\underline{v}_k, \overline{v}_k]$, as long as there is no significant reduction. Since the sequence $\{[\underline{v}_k^j, \overline{v}_k^j]\}$ of reduced boxes is nested, i.e. $[\underline{v}_k^{j+1}, \overline{v}_k^{j+1}] \subseteq [\underline{v}_k^j, \overline{v}_k^j]$, the sequence converges to a fixpoint $[\underline{v}_k^*, \overline{v}_k^*] \supseteq \Box S_k$.

Assuming that an upper bound $\overline{v}$ on the optimal value of (7.1) is available, the level inequality $c^T x + c_0 \leq \overline{v}$ can be included into S_k, in order to further reduce the box $[\underline{v}_k, \overline{v}_k]$, $k \in \{2,3,4\}$. Assume that (7.1) has a unique solution x^* and $\overline{v} = \mathrm{val}(7.1)$. In this case, $\Box S_k$ is a convex relaxation of the singleton $\{x^*\}$, $k \in \{2,3,4\}$.

7.5 Numerical results

In order to compare the lower bounds and the related box reduction operations, numerical experiments were carried out by using instances from the MINLPLib (Bussieck et al., 2003a) described in Appendix B.1. The lower bound and box reduction procedures were implemented as part of the C++ library LaGO (see Chapter 14). The sequential quadratic programming code SNOPT (Gill et al., 1997) is used for finding local solutions of nonlinear optimization problems. CPLEX (ILOG, Inc., 2005) is used for solving linear optimization problems. For computing convex underestimating-relaxations, α-underestimators (see Section 6.3) were used if a function is quadratic, and polynomial underestimators (see Section 6.5) otherwise. The sample set for generating polynomial underestimators was defined by $\max\{2^{|B|}, 100\}$ vertices and 20 random points.

Four different box reduction methods and their corresponding lower bounds were compared. In the first and second experiment, the box reduction methods

that use the lower bounds $\underline{v}_{\mathrm{NLP1}}$ and $\underline{v}_{\mathrm{NLP2}}$ described in Section 7.3.1 were tested. Since for the generation of a convex underestimating-relaxation the lower and upper variable bounds have to be finite, the interval $[\underline{x}, \overline{x}]$ was replaced by the interval hull $[\underline{x}^0, \overline{x}^0] = \Box S_0$ of the following nonlinear convex relaxation:

$$S_0 = \{x \in \mathbb{R}^n \mid h_i(x) \leq 0,\, i \in I_{\mathrm{conv}}\},$$

where I_{conv} is the index set of convex constraints of the MINLP (2.1). The box reduction methods that use NLP1 and NLP2 bounds applied to the interval $[\underline{x}^0, \overline{x}^0]$ are called NLP0/NLP1 and NLP0/NLP2 respectively.

In the third and fourth experiment, the box reduction methods that use the lower bounds $\underline{v}_{\mathrm{NLP2}}$ and $\underline{v}_{MINLP}$ described in Section 7.3.1 and 7.3.2 were tested. Here, the initial interval $[\underline{x}, \overline{x}]$ was replaced by $[\underline{y}^0, \overline{y}^0]$, where $\underline{y}_i^0 = \underline{x}_i^0$ if $\underline{x}_i = -\infty$ and $\underline{y}_i^0 = \underline{x}_i$ else, and $\overline{y}_i^0 = \overline{x}_i^0$ if $\overline{x}_i = \infty$ and $\overline{y}_i^0 = \overline{x}_i$ else. The box reduction methods that use NLP2 and MINLP bounds applied to the interval $[\underline{y}^0, \overline{y}^0]$ are called NLP0′/NLP2 and NLP0′/MINLP respectively.

The code was run on a machine with a 1GHz Pentium III processor and 256 MB RAM. Tables 7.2, 7.3, 7.4 and 7.5 show the results. The columns of these tables are described in Table 7.1. N/A means that no feasible solution was found.

avr red	The average of the relative box reduction over all variables in percentage measured by $\frac{w_i^{\mathrm{new}}}{w_i}$, where $w_i = \overline{x}_i - \underline{x}_i$.
red var	The percentage number of variables where the box is reduced by more than 20%.
bin fix	The percentage number of binary variables that are fixed.
box time	Time in seconds for performing the box reduction.
bnd err	The quality of the lower bound, if available, computed as $\frac{v^* - \underline{v}}{1+\lvert v^*\rvert}$, where v^* is the best known optimal value and $\underline{v}$ is the value of the lower bound.
bnd time	Time in seconds for computing a lower bound.

Table 7.1: Descriptions of the columns of Tables 7.2, 7.3, 7.4 and 7.5.

The results show:

- For about 30% of the problems some variables were fixed.
- For more than 60% of the problems the box was reduced significantly.
- The decomposition-based bound $\underline{v}_{\mathrm{NLP2}}$ is not faster than $\underline{v}_{NLP1}$.
- Comparing Table 7.3 and Table 7.4 shows that the initial box reduction $[\underline{y}^0, \overline{y}^0]$ is much faster than $[\underline{x}^0, \overline{x}^0]$ and gives almost the same results.
- Comparing Table 7.2 and Table 7.5 shows that only in few cases the results with $\underline{v}_{\mathrm{MINLP}}$-bounds are better than with $\underline{v}_{\mathrm{NLP1}}$-bounds.

example	n	$\|B\|$	m	c	avg red	red var	bin fix	box time	bnd err	bnd time
alan	9	4	8	y	44.4	55	25	0.01	0	0.00
elf	55	24	39	n	45.5	54	0	2.54	.16	0.03
ex1265	131	100	75	n	91.8	9	8	0.6	.01	0.02
ex1266	181	138	96	n	81.5	23	17	1.37	0	0.02
fac3	67	12	34	y	24.2	80	0	0.13	.30	0.03
fuel	16	3	16	n	45	68	33	0.04	0	0.01
gbd	5	3	5	y	71.7	40	33	0.01	0	0.00
meanvarx	36	14	45	y	36.1	63	14	0.06	0	0.00
nous1	51	2	44	n	92.2	7	0	0.06	2.63	0.29
nous2	51	2	44	n	92.2	7	0	0.06	3.57	0.45
sep1	30	2	32	n	40.8	73	0	0.02	.22	0.02
space25	894	750	236	n	87.2	12	0	52.76	.86	0.55
space25a	384	240	202	n	70.3	29	0	10.03	.86	0.23
spectra2	70	30	73	n	42.9	57	0	15.78	.37	0.16
util	146	28	168	n	19.9	80	14	2	.05	0.08
eniplac	142	24	190	n	25.1	75	4	1.57	N/A	0.23
enpro48	154	92	215	n	96.7	3	0	4.33	N/A	0.14
enpro56	128	73	192	n	91.9	7	4	4.04	N/A	0.02
ex3	33	8	32	n	53.4	48	0	0.26	.76	0.01
fac1	23	6	19	y	59.7	69	0	0.02	0	0.00
gastrans	107	21	150	n	30.1	75	66	1.89	N/A	0.07
gear2	29	24	5	n	100	0	0	0	22.13	0.00
gkocis	12	3	9	n	43.5	58	0	0.01	1.49	0.00
parallel	206	25	116	n	8.73	91	36	1.69	N/A	0.80
procsel	11	3	8	n	45.5	54	0	0.01	1.17	0.00
synthes2	12	5	15	y	82.5	16	0	0.02	.99	0.00
synthes3	18	8	24	y	98.1	5	0	0.05	.76	0.01
waterx	71	14	55	n	42.8	57	0	0.77	.94	0.05

Table 7.2: Box reduction results with NLP0/NLP1-bounds.

example	n	$\|B\|$	m	c	avg red	red var	bin fix	box time	bnd err	bnd time
alan	9	4	8	y	44.4	55	25	0.1	.13	0.00
elf	55	24	39	n	45.5	54	0	6.26	.16	0.01
ex1265	131	100	75	n	91.8	9	8	1.64	.01	0.00
ex1266	181	138	96	n	81.5	23	17	2.67	0	0.01
fac3	67	12	34	y	24.2	80	0	0.52	.99	0.02
fuel	16	3	16	n	45	68	33	0.12	.03	0.01
gbd	5	3	5	y	71.7	40	33	0.05	0	0.00
meanvarx	36	14	45	y	36.1	63	14	0.26	.05	0.01
nous1	51	2	44	n	84.4	21	100	1.67	1.84	0.02
nous2	51	2	44	n	83.5	25	100	1.66	2.33	0.03
sep1	30	2	32	n	41.5	73	0	0.37	.40	0.01
space25	894	750	236	n	87.2	12	0	145.02	.86	0.01
space25a	384	240	202	n	70.3	29	0	48.37	.86	0.01
spectra2	70	30	73	n	42.9	57	0	11.44	.37	0.00
util	146	28	168	n	19.9	80	14	5.46	.12	0.03
eniplac	142	24	190	n	22.5	78	16	5.13	N/A	0.04
enpro48	154	92	215	n	96.7	3	0	11.11	.99	0.00
enpro56	128	73	192	n	90.6	9	4	7.74	N/A	0.00
ex3	33	8	32	n	53.4	48	0	0.43	.85	0.01
fac1	23	6	19	y	59.7	69	0	0.19	.99	0.01
gastrans	107	21	150	n	30.1	75	66	1.94	N/A	0.00
gear2	29	24	5	n	100	0	0	0.1	22.13	0.01
gkocis	12	3	9	n	43.5	58	0	0.05	2.38	0.00
parallel	206	25	116	n	8.73	91	36	6.58	40.14	0.04
procsel	11	3	8	n	45.5	54	0	0.05	1.18	0.01
synthes2	12	5	15	y	82.5	16	0	0.11	1.12	0.01
synthes3	18	8	24	y	98.1	5	0	0.18	.80	0.01
waterx	71	14	55	n	42.8	57	0	2.35	.94	0.02

Table 7.3: Box reduction results with NLP0/NLP2-bounds.

example	n	$\|B\|$	m	c	avg red	red var	bin fix	box time	bnd err	bnd time
alan	9	4	8	y	44.4	55	25	0.06	.31	0.00
elf	55	24	39	n	45.5	54	0	3.99	.16	0.02
ex1265	131	100	75	n	92.3	9	8	0.97	.02	0.01
ex1266	181	138	96	n	82.2	23	17	1.6	.01	0.01
fac3	67	12	34	y	24.2	80	0	0.46	.99	0.01
fuel	16	3	16	n	45	68	33	0.12	.05	0.00
gbd	5	3	5	y	71.7	40	33	0.07	0	0.00
meanvarx	36	14	45	y	36.1	63	14	0.27	.03	0.01
nous1	51	2	44	n	83.7	23	100	1.63	1.84	0.03
nous2	51	2	44	n	83.5	25	100	1.63	2.33	0.03
sep1	30	2	32	n	42.3	73	0	0.24	.41	0.01
space25	894	750	236	n	87.2	12	0	57.88	.86	0.01
space25a	384	240	202	n	70.3	29	0	29.96	.86	0.01
spectra2	70	30	73	n	42.9	57	0	7.76	.93	0.02
util	146	28	168	n	19.9	80	14	2.42	.45	0.01
eniplac	142	24	190	n	23.2	78	12	2.74	.43	0.05
enpro48	154	92	215	n	96.8	3	0	2.98	.99	0.00
enpro56	128	73	192	n	95.3	4	0	1.59	N/A	0.01
ex3	33	8	32	n	53.4	48	0	0.27	.85	0.01
fac1	23	6	19	y	59.7	69	0	0.09	.99	0.00
gastrans	107	21	150	n	27.9	79	76	2.05	0	0.01
gear2	29	24	5	n	100	0	0	0.1	22.13	0.00
gkocis	12	3	9	n	43.5	58	0	0.07	2.38	0.00
parallel	206	25	116	n	8.73	91	36	3.46	50.62	0.06
procsel	11	3	8	n	45.5	54	0	0.04	1.18	0.00
synthes2	12	5	15	y	82.6	16	0	0.1	1.01	0.01
synthes3	18	8	24	y	98.1	5	0	0.14	.80	0.01
waterx	71	14	55	n	42.8	57	0	1.05	.94	0.02

Table 7.4: Box reduction results with NLP0′/NLP2-bounds.

example	n	$\|B\|$	m	c	avg red	red var	bin fix	box time	bnd err	bnd time
alan	9	4	8	y	44.4	55	25	0.06	.31	0.00
elf	55	24	39	n	45.5	54	0	4.24	.16	0.01
ex1265	131	100	75	n	90.3	11	8	1.13	.02	0.00
ex1266	181	138	96	n	80	24	17	2.28	.01	0.01
fac3	67	12	34	y	24.2	80	0	0.38	.99	0.01
fuel	16	3	16	n	45	68	33	0.16	N/A	0.00
gbd	5	3	5	y	71.7	40	33	0.05	0	0.00
meanvarx	36	14	45	y	36.1	63	14	0.2	.03	0.01
nous1	51	2	44	n	89.4	13	0	0.93	.73	0.08
nous2	51	2	44	n	92	7	0	1.15	.67	0.04
sep1	30	2	32	n	42.3	73	0	0.29	.41	0.00
space25	894	750	236	n	87.2	12	0	46.24	.86	0.02
space25a	384	240	202	n	70.2	29	0	22.2	.86	0.01
spectra2	70	30	73	n	42.9	57	0	7.76	.93	0.00
util	146	28	168	n	19.9	80	14	2.51	.30	0.05
eniplac	142	24	190	n	20.4	80	29	2.77	.48	0.05
enpro48	154	92	215	n	82.1	22	8	4.71	.99	0.01
enpro56	128	73	192	n	95.3	4	0	1.63	N/A	0.01
ex3	33	8	32	n	53.5	48	0	0.2	.86	0.01
fac1	23	6	19	y	59.7	69	0	0.12	.99	0.00
gastrans	107	21	150	n	33.4	68	61	2.09	0	0.01
gear2	29	24	5	n	100	0	0	0.07	-.01	0.00
gkocis	12	3	9	n	43.5	58	0	0.07	2.38	0.00
parallel	206	25	116	n	8.73	91	36	3.37	.81	0.41
procsel	11	3	8	n	45.5	54	0	0.04	1.18	0.01
synthes2	12	5	15	y	82.6	16	0	0.11	1.01	0.01
synthes3	18	8	24	y	98.1	5	0	0.13	.80	0.01
waterx	71	14	55	n	42.8	57	0	1.05	N/A	0.03

Table 7.5: Box reduction results with NLP0′/MINLP-bounds.

Chapter 8

Local and Global Optimality Criteria

This chapter presents local and global optimality criteria. After a short overview on first- and second-order necessary and sufficient local optimality criteria, a strong duality result for nonconvex all-quadratic problems with convex constraints is proven (Nowak, 2000). Based on this result, optimality cuts are proposed that split off a local minimizer, and some global optimality criteria are derived. Finally, it is shown that global optimality of general MINLP solutions can be verified via interval-gradient cuts.

8.1 Local optimality conditions

Consider a general nonlinear program of the form

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0 \end{array} \tag{8.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $h : \mathbb{R}^n \mapsto \mathbb{R}^p$ are continuously twice-differentiable nonlinear functions. The feasible set of (8.1) is denoted by S.

A point $x^* \in S$ is a called *local minimizer* of (8.1) if there exists an $\epsilon \in \mathbb{R}_+$ such that $f(x^*) \leq f(x)$ for all $x \in S \cap I\!B^n(x^*, \epsilon)$. A local minimizer x^* is called *strict* if there exists a ball $I\!B^n(x^*, \epsilon)$ containing no other local minimizer $\hat{x} \neq x^*$. It is called a *global minimizer* if $f(x^*) \leq f(x)$ for all $x \in S$. An ϵ-minimizer of (8.1) is a point satisfying $f(x^*) \leq f(x) + \epsilon$ for all $x \in S$. The set of *ϵ-minimizers* of (8.1) is denoted by $\mathrm{sol}_\epsilon(8.1)$. The *Lagrangian* related to (8.1) is the function

$$L(x; \mu) = f(x) + (\mu^g)^T g(x) + (\mu^h)^T h(x),$$

where $\mu = (\mu^g, \mu^h)$ is a *dual point* or *Lagrangian multiplier*, which is in the *Lagrangian multiplier set*,

$$\mathcal{M} = \{(\mu^g, \mu^h) \in \mathbb{R}^m \times \mathbb{R}^p \mid \mu^g \geq 0\}.$$

In the following, some basic local optimality conditions for minimizers of (8.1) are reviewed. These conditions are based on linearizations of the objective and constraint functions and require *regularity* assumptions called *constraint qualifications*. They ensure that the set of feasible directions of the original problem and of the linearized problem are identical. In order to state them, we need the definition of an *active set* related to a point $x^* \in \mathbb{R}^n$,

$$\mathcal{A}(x^*) = \{i \in \{1, \ldots, m\} \mid g_i(x^*) = 0\}.$$

Condition 8.1 (Mangasarin–Fromowitz). *The Mangasarin–Fromowitz constraint qualification (Mangasarin and Fromowitz, 1967) holds at a point $x^* \in \mathbb{R}^n$ if:*

(i) *There exists a vector $z \in \mathbb{R}^n$ such that: $\nabla g_i(x^*)^T z > 0$ for all $i \in \mathcal{A}(x^*)$, $\nabla h_i(x^*)^T z = 0$ for $i = 1, \ldots, p$.*

(ii) *The gradients $\nabla h_i(x^*)$, $i = 1, \ldots, p$, are linearly independent.*

In the case of convex problems, where g is convex and h is affine, the following condition is equivalent to Condition 8.1.

Condition 8.2 (Slater). *The Slater constraint qualification is satisfied if g is convex, h is affine, and there exists a point $\hat{x} \in \mathbb{R}^n$ with $g(\hat{x}) < 0$ and $h(\hat{x}) = 0$.*

The next condition is the strongest of them and implies Conditions 8.1 and 8.2.

Condition 8.3 (linear independence). *The linear independence constraint qualification holds at x^* if the gradients $(\nabla g_{\mathcal{A}(x^*)}(x^*), \nabla h(x^*))$ are linearly independent.*

A deeper discussion of constraint qualifications can be found in (Mangasarin, 1969). If a constraint qualification holds, a local solution of (8.1) can be characterized by the Karush–Kuhn–Tucker (KKT) (Karush, 1939; Kuhn and Tucker, 1951) necessary first-order optimality conditions.

Proposition 8.1 (first-order optimality condition). *Let x^* be a local minimizer of (8.1) and assume that f, g, h are continuously differentiable and that Condition 8.1 holds. Then there exists a vector $\hat{\mu} = (\hat{\mu}^g, \hat{\mu}^h) \in \mathbb{R}^m \times \mathbb{R}^p$ such that the following first-order necessary conditions hold:*

(i) $\nabla_x L(x^*; \hat{\mu}) = 0$ *(stationarity of the Lagrangian)*

(ii) $x^* \in S$ *(primal feasibility)*

(iii) $\hat{\mu} \in \mathcal{M}$ *(dual feasibility)*

(iv) $\hat{\mu}_i^g \cdot g_i(x^*) = 0$ *for $i = 1, \ldots, m$ (complementary slackness).*

A pair (x^*, μ^*) satisfying (i)–(iv) of Proposition 8.1 is called a Karush–Kuhn–Tucker (KKT) pair. Under stronger conditions, the reverse statement of Proposition 8.1 holds (Fletcher, 1987).

Proposition 8.2 (second-order optimality condition). *Assume that the functions f, g and h are continuously twice-differentiable. Let (x^*, μ^*) be a KKT-pair satisfying*

$$z^T \nabla^2 L(x^*; \mu^*) z > 0 \text{ for all } z \in \mathcal{T}$$

where

$$\mathcal{T} = \left\{ z \in \mathbb{R}^n \setminus \{0\} \mid \begin{array}{l} \nabla g_i(x^*)^T z \geq 0 \text{ for } i \in \mathcal{A}(x^*), \\ \nabla g_i(x^*)^T z = 0 \text{ for } i \in \mathcal{A}(x^*) \text{ such that } \mu_i^* > 0 \\ \nabla h_i(x^*)^T z = 0 \text{ for } i \in \{1, \ldots, p\} \end{array} \right\}.$$

Then x^* is a strict local minimizer.

An important case, in which the second-order optimality condition is fulfilled, is if (8.1) is strictly convex. If (8.1) is convex, then every local minimizer is a global minimizer.

8.2 Local strong duality of nonconvex QQPs

The duality gap of a nonconvex all-quadratic program is usually nonzero. It is shown now that in the case of nonconvex all-quadratic programs with convex constraints it is always possible to close the duality gap by shrinking the feasible set in a neighborhood of a local minimizer fulfilling a certain constraint qualification. Consider an all-quadratic program of the form

$$\begin{array}{ll} \min & q_0(x) \\ \text{s.t.} & q_i(x) \leq 0, \quad i = 1, \ldots, m \end{array} \tag{8.2}$$

where $q_i(x) = x^T A_i x + 2a_i^T x + d_i$, $A_i \in \mathbb{R}^{(n,n)}$, $a_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $i = 0, \ldots, m$. For the sake of simplicity, here only inequality constrained problems are considered. However, all results in this section can be generalized without difficulties to problems with equality constraints.

The Lagrangian of (8.2) is the function $L(x; \mu) = q_0(x) + \mu^T q(x)$, where $q = (q_1, \ldots, q_m)^T$. Defining the dual function $D(\mu) = \inf_{x \in \mathbb{R}^n} L(x; \mu)$, the dual of (8.2) is formulated by

$$\max_{\mu \in \mathbb{R}^m_+} D(\mu). \tag{8.3}$$

A simple global optimality criterion for (8.2) is given by:

Lemma 8.3 (strong duality). *Let S be the feasible set of* (8.2)*. The dual problem* (8.3) *has a zero duality gap, i.e.* val(8.2) = val(8.3)*, if and only if there exist $\hat{\mu} \in \mathbb{R}^m_+$ and $\hat{x} \in S$ such that*

$$\left.\begin{array}{lcl} L(\hat{x};\hat{\mu}) & = & q_0(\hat{x}) \\ \nabla L(\hat{x};\hat{\mu}) & = & 0 \\ \nabla^2 L(\hat{x};\hat{\mu}) & \succcurlyeq & 0. \end{array}\right\} \tag{8.4}$$

A point $\hat{x}$ fulfilling the conditions (8.4) *is a global minimizer of problem* (8.2)*.*

Proof. Let μ^* be a solution of (8.3) and let x^* be a global minimizer of problem (8.2). If val(8.2) = val(8.3), it follows that $x^* \in \operatorname{argmin}_{x\in\mathbb{R}^n} L(x;\mu^*)$. Hence, (μ^*, x^*) fulfills condition (8.4).

Now let $(\hat{\mu}, \hat{x})$ be a point satisfying (8.4). Then val(8.3) $\le$ val(8.2) $\le q_0(\hat{x}) = \min_{x\in\mathbb{R}^n} L(\hat{x};\mu) \le$ val(8.3). Hence, val(8.2) = val(8.3). □

The duality gap is usually not zero in nonconvex programming. It is shown now that nevertheless condition (8.4) can be satisfied in a neighborhood of $\hat{x}$, provided that $\hat{x}$ satisfies the following second-order optimality condition.

Condition 8.4 (modified second-order optimality condition). *Let*

$$\mathcal{A}(x^*) = \{i \in \{1,\dots,m\} \mid q_i(x^*) = 0\}$$

be the active set,

$$\mathcal{A}_+(x^*) = \{i \in \mathcal{A}(x^*) \mid \mu_i^* > 0\},$$

be a restricted active set, and

$$\mathcal{T}_+ = \{x \in \mathbb{R}^n \mid \nabla q_i(x^*)^T x = 0 \text{ for } i \in \mathcal{A}_+(x^*)\}$$

be the extended tangent space. A KKT-pair (x^*, μ^*) fulfills the modified second-order optimality condition *if the Hessian $\nabla^2 q_0(\cdot)$ is copositive with respect to $\mathcal{T}_+$, i.e.*

$$y^T \nabla^2 q_0(x) \cdot y \ge 0 \text{ for all } y \in \mathcal{T}_+.$$

We give now several conditions implying Condition 8.4.

Lemma 8.4. *Let x^* be a local minimizer of problem* (8.2) *and $\mu \in \mathbb{R}^m_+$ be a Lagrange multiplier fulfilling the strict complementarity condition:*

$$\mu_i^* > 0 \quad \text{for} \quad i \in \mathcal{A}(x^*). \tag{8.5}$$

The following conditions imply Condition 8.4*:*

(i) *The Hessian $\nabla^2 q_0(x)$ is copositive with respect to the tangent space $\mathcal{T}$ (see Section* 8.1*), i.e. $y^T \nabla^2 q_0(x) y \ge 0$ for all $y \in \mathcal{T}$.*

(ii) *The constraints of problem* (8.2) *are linear and* x^* *fulfills the second-order optimality condition of Proposition* 8.2.

(iii) *The constraints of problem* (8.2) *are linear and* x^* *is a regular point, i.e. the vectors* $\nabla q_i(x^*)$, $i \in \mathcal{A}(x^*)$, *are linearly independent.*

Proof. (i) From the strict complementarity condition (8.5) it follows that $\mathcal{A}(x^*) = \mathcal{A}_+(x^*)$. This implies $\mathcal{T}_+ = \mathcal{T}$, which proves the assertion.
(ii) Since the constraints of problem (8.2) are linear, it holds that $\nabla^2 L(x;\mu) = \nabla^2 q_0(x)$. Therefore, (ii) is equivalent to (i) in this case.
(iii) Since a local minimizer that is a regular point fulfills the second-order optimality condition, (iii) implies (ii). □

Example 8.5. Consider the following example: $\min\{-\|x\|^2 \mid 0 \leq x \leq e\}$, where $x \in \mathbb{R}^n$ and $e \in \mathbb{R}^n$ is the vector of 1s. This problem has a unique global minimizer $x^* = e$ fulfilling the strict complementarity condition (8.5). From Lemma 8.4 (iii) it follows that x^* fulfills Condition 8.4.

In order to prove the main result of this section, the following generalization of Debreu's Lemma (Lemaréchal and Oustry, 1999) is required:

Lemma 8.6. *Let* $A \in \mathbb{R}^{(n,n)}$ *be a symmetric matrix that is copositive with respect to the linear subspace* $\operatorname{span}\{w_1, .., w_p\}^\perp$, *where* $w_i \in \mathbb{R}^n$ *for* $1 \leq i \leq p$. *Then there exists* $\bar{\tau} \in \mathbb{R}^p$ *such that*

$$A + \sum_{i=1}^{p} \tau_i w_i w_i^T \succcurlyeq 0 \text{ for all } \tau \geq \bar{\tau}.$$

Proof. Let $B = \sum_{i=1}^{p} \rho_i w_i w_i^T$, where $\rho_i > 0$ for $1 \leq i \leq p$. Let $V = \operatorname{span}\{w_1, .., w_p\}$, $R = \operatorname{kern}(A)$, $S = V \cap R^\perp$ and $T = V^\perp \cap R^\perp$. Define

$$c_1 = \min_{x \in T \setminus \{0\}} \frac{x^T A x}{\|x\|^2}, \quad c_2 = \min_{x \in V \setminus \{0\}} \frac{x^T B x}{\|x\|^2}, \quad c_3 = \|A\|_2.$$

Since A is copositive with respect to $V^\perp$, it holds that $c_1 > 0$. By using $x^T B x = \sum_{k=1}^{p} \rho_k (w_k^T x)^2 \geq 0$, we infer that the matrix B is positive semidefinite and, hence, copositive with respect to V, thus, implying $c_2 > 0$. Given $x \in \mathbb{R}^n$, there exist $r \in R, s \in S$ and $t \in T$ such that $x = r + s + t$, since $\mathbb{R}^n = R \bigoplus S \bigoplus T$. Therefore,

$$\begin{aligned} x^T(A + \mu B)x &= (s+t)^T A(s+t) + \mu(r+s)^T B(r+s) \\ &\geq c_1|t|^2 - 2c_3 \cdot |s| \cdot |t| - c_3|s|^2 + \mu \cdot c_2 \cdot (|r|^2 + |s|^2) \\ &= (\sqrt{c_1}|t| - c_2/\sqrt{c_1}|s|)^2 + (\mu c_2 - c_3 - c_2^2/c_1)|s|^2. \end{aligned}$$

This implies $A + \mu_0 B \succcurlyeq 0$, where $\mu_0 = (c_3 + c_2^2/c_1)/c_2$. Setting $\bar{\tau} = \mu_0 \cdot \rho$, we obtain $A + \sum_{i=1}^{p} \tau_i w_i w_i^T = A + \mu B + \sum_{i=1}^{p} (\tau_i - \bar{\tau}_i) w_i w_i^T \succcurlyeq 0$. □

Note that c_1 and c_2 can be determined by computing minimum eigenvalues of reduced matrices. The main result of this section is the following global optimality criterion.

Proposition 8.7 (local strong duality). *Let (x^*, μ^*) be an optimal primal-dual pair of problem (8.2) fulfilling Condition 8.4. Assume that the constraint functions q_i are convex for $i \in \mathcal{A}_+(x^*)$. Choose $\hat{\tau}_i \geq 0$, $i \in \mathcal{A}_+(x^*)$, according to Lemma 8.6 such that $A(\hat{\tau}) \succcurlyeq 0$, where*

$$A(\tau) = \nabla^2 q_0(\cdot) + \sum_{i \in \mathcal{A}_+(x^*)} \tau_i \nabla q_i(x^*) \nabla q_i(x^*)^T.$$

Define

$$S_{\hat{\tau}} = \{x \in \mathbb{R}^n \mid 0 \geq \nabla q_i(x^*)^T (x - x^*) \geq -\frac{\mu_i^*}{\hat{\tau}_i}, \quad i \in \mathcal{A}_+(x^*) \text{ and } \hat{\tau}_i > 0\}.$$

Then

$$\min_{x \in S \cap U} q_0(x) = q_0(x^*) \text{ for all } U \subset \mathbb{R}^n \text{ with } S \cap U \subseteq S_{\hat{\tau}},$$

where S is the feasible set of (8.2). In particular, if $S \subseteq S_{\hat{\tau}}$, then x^* is a global minimizer of (8.2).

Proof. Let

$$\delta_i(U) = -\min_{x \in U} \nabla q_i(x^*)^T (x - x^*), \quad i \in \mathcal{A}_+(x^*) \tag{8.6}$$

and define the optimization problem (Q[U]):

$$\begin{array}{lll} \min & q_0(x) & \\ \text{s.t.} & \nabla q_i(x^*)^T (x - x^*) \cdot (\nabla q_i(x^*)^T (x - x^*) + \delta_i(U)) \leq 0, & i \in \mathcal{A}_+(x^*) \quad (I) \\ & \nabla q_i(x^*)^T (x - x^*) \leq 0, & i \in \mathcal{A}_+(x^*) \quad (II). \end{array}$$

Let $\mu = (\mu^{(1)}, \mu^{(2)})$ be a dual point, where $\mu^{(1)}$ and $\mu^{(2)}$ pertain to the constraints (I) and (II) respectively. Let $L(x; \mu)$ be the Lagrangian to (Q[U]). Then it holds that $L(x^*; \mu) = q_0(x^*)$. From the Karush–Kuhn–Tucker condition

$$\nabla q_0(x^*) + \sum_{i \in \mathcal{A}_+(x^*)} \mu_i^* \nabla q_i(x^*) = 0$$

and from

$$\nabla L(x^*; \mu) = \nabla q_0(x^*) + \sum_{i \in \mathcal{A}_+(x^*)} (\mu_i^{(1)} \delta_i(U) + \mu_k^{(2)}) \nabla q_i(x^*)$$

we obtain

$$\nabla L(x^*;\mu) = \sum_{i\in\mathcal{A}_+(x^*)} (\mu_i^{(1)}\delta_i(U) + \mu_i^{(2)} - \mu_i^*)\nabla q_i(x^*).$$

Choosing $\mu^{(1)} = \hat{\tau}$ and $\mu_i^{(2)} = \mu_i^* - \hat{\tau}_i\delta_i(U)$ for $i \in \mathcal{A}_+(x^*)$, it holds that $\nabla L(x^*;\mu) = 0$. If $S \cap U \subseteq S_{\hat{\tau}}$, it holds that $\delta_k(U) \leq \frac{\mu_k^*}{\hat{\tau}_k}$ for $k \in \mathcal{A}_+(x^*)$ and $\hat{\tau}_k > 0$ implying $\mu_k^{(2)} = \mu_k^* - \hat{\tau}_k\delta_k(U) \geq 0$. Hence, (μ, x^*) fulfills (8.4), and by Lemma 8.3 we conclude that the dual of (Q[U]) has a zero duality gap. □

Remark 8.8. From Proposition 8.7 it follows that the dual bound of $(Q[U])$ is equal to $q_0(x^*)$ if the diameter of the set U is small enough. This property ensures finite convergence of branch-and-bound algorithms (see Section 13.2). Lower bounds for integer programs have always this property, whereas most lower bounds for continuous global optimization problems do not have this property. An exception is the linear programming bound of Epperly and Swaney (Epperly and Swaney, 1996).

8.3 Global optimality cuts

Based on Proposition 8.7 a cutting-plane can be constructed that splits off a given local minimizer from the feasible set. Such a cut is called a *global optimality cut*. From Proposition 8.7 follows:

Corollary 8.9 (optimality cut). *Let x^* be a local minimizer of problem* (8.2) *fulfilling Condition* 8.4*, and let $H \subset \mathbb{R}^n$ be a half-space such that $x^* \in \operatorname{int} H$ and $S \cap H \subseteq S_{\hat{\tau}}$. Then*

$$\min\{q_0(x) \mid x \in S \cap H\} = q_0(x^*), \tag{8.7}$$

where $S_{\hat{\tau}}$ is defined as in Proposition 8.7 *and S is the feasible set of* (8.2)*.*

A half-space that meets the conditions of Corollary 8.9 defines an optimality cut with respect to x^*. The following proposition gives a method for constructing such a half-space.

Proposition 8.10 (construction of optimality cuts). *Let x^* be a local minimizer of problem* (8.2) *fulfilling Condition* 8.4*, and let $\hat{\tau}_i \geq 0$, $i \in \mathcal{A}_+(x^*)$, be parameters fulfilling $A(\hat{\tau}) \succeq 0$, where $A(\hat{\tau})$ is defined as in Proposition* 8.7*. Then*

$$H = \{x \in \mathbb{R}^n \mid \eta^T(x - x^*) \leq 1\}$$

defines an optimality cut with respect to x^*, where $\eta = \sum_{i\in\mathcal{A}_+(x^*)} -\frac{\hat{\tau}_i}{\mu_i^*}\nabla q_i(x^*)$, and μ_i^* is an optimal Lagrangian multiplier corresponding to x^*.

Proof. Obviously, it holds $x^* \in \operatorname{int} H$. Let K_{x^*} be the cone defined by

$$K_{x^*} = \{x \in \mathbb{R}^n \mid \nabla q_i(x^*)^T(x - x^*) \leq 0 \text{ for } i \in \mathcal{A}_+(x^*)\}.$$

Let

$$V_j = \{x \in \mathbb{R}^n \quad | \quad \nabla q_j(x^*)^T(x - x^*) = -\delta_j^*,\ \nabla q_i(x^*)^T(x - x^*) = 0, \\ i \in \mathcal{A}_+(x^*) \setminus \{j\}\}$$

for $j \in \mathcal{A}_+(x^*)$ and

$$V_0 = \{x \in \mathbb{R}^n \mid \nabla q_i(x^*)^T(x - x^*) = 0, i \in \mathcal{A}_+(x^*)\}$$

where $\delta_i^* = \frac{\mu_i^*}{\hat{\tau}_i}$ if $\hat{\tau}_i > 0$ and $\delta_i^* = \infty$ else. Then $\eta^T(x - x^*) = 1$ for $x \in V_i$ and $i \in \mathcal{A}_+(x^*)$, and $\eta^T(x - x^*) = 0$ for $x \in V_0$. Hence, $H \cap K_{x^*} = \operatorname{conv}\{V_i \mid i \in \mathcal{A}_+(x^*) \cup \{0\}\}$. Due to $V_i \subset S_{\hat{\tau}}$ for $i \in \mathcal{A}_+(x^*) \cup \{0\}$ we have

$$H \cap S \subset H \cap K_{x^*} \subset S_{\hat{\tau}}.$$

From Proposition 8.7 follows (8.7). This proves the assertion using Corollary 8.9. □

The parameter $\hat{\tau}$ should be computed in a way such that $\operatorname{diam}(S_{\hat{\tau}})$ is as large as possible. Since $\delta_i^*/\|\nabla q_i(x^*)\|$ is an upper bound on the diameter of $S_{\hat{\tau}}$ along the direction w_i, maximizing $\operatorname{diam}(S_{\hat{\tau}})$ is similar to maximizing $\delta_i^*/\|\nabla q_i(x^*)\|$ for all $i \in \mathcal{A}_+(x^*)$ or to minimizing $\displaystyle\sum_{i \in \mathcal{A}_+(x^*)} \frac{1}{\delta_i^*}\|\nabla q_i(x^*)\| = \sum_{i \in \mathcal{A}_+(x^*)} \frac{\hat{\tau}_i}{\mu_i^*}\|\nabla q_i(x^*)\|$. This motivates us to compute $\hat{\tau}$ by solving the following semidefinite program:

$$\begin{array}{ll} \min_\tau & \sum_{i \in \mathcal{A}_+(x^*)} \frac{\tau_i}{\mu_i^*}\|\nabla q_i(x^*)\| \\ \text{s.t.} & A(\tau) \succcurlyeq 0 \\ & \tau \geq 0. \end{array} \tag{8.8}$$

From Proposition 8.7 it follows that $\hat{\tau} \in \operatorname{sol}(8.8)$ is well-defined if x^* fulfills Assumption 8.4. Note that for the construction of an optimality cut, it is sufficient to find a feasible point of (8.8), which is a much simpler problem than solving (8.8).

8.4 Some global optimality criteria for QQPs

For special cases of problem (8.2) it is possible to define an extended quadratic program that includes the constraints (I) and (II) of problem (Q[U]) in Proposition 8.7 with respect to **all** global minimizers. We define such programs for the box-constrained and the standard quadratic program. Using Proposition 8.7 we derive conditions that lead to a zero duality gap of the corresponding dual bound.

Consider a box-constrained quadratic program defined by

$$(\mathrm{B}_1) \qquad \begin{array}{ll} \min & q(x) \\ \text{s.t.} & x \in [\underline{x}, \overline{x}] \end{array}$$

where $q(x) = x^T Ax + 2a^T x + c$ and $\underline{x}, \overline{x} \in \mathbb{R}^n$. An all-quadratic reformulation to (B_1) is given by:

$$(\mathrm{B}_2) \qquad \begin{array}{ll} \min & q(x) \\ \text{s.t.} & x \in [\underline{x}, \overline{x}] \\ & \mathrm{Diag}(x - \underline{x})(x - \overline{x}) \leq 0. \end{array}$$

Obviously, problem (B_2) contains the constraints (I) and (II) of problem (Q[U]) with respect to all global minimizers of problem (B_1). From this it follows that under certain assumptions the dual bound of (B_2), denoted by $\mathrm{val}(\mathrm{Dual}(\mathrm{B}_2))$, coincides with the optimal value of (B_1). More precisely, the following holds.

Lemma 8.11. *Let x^* be a local minimizer of problem (B_1) fulfilling Condition* 8.4. *Define $\mathcal{A}_+(x^*) = \{i \in \{1, .., n\} \mid x_i^* = \overline{x}_i$ or $x_i^* = \underline{x}_i$ and the related optimal dual point is greater than zero $\}$. Let $d_i = \left|\frac{\partial}{\partial x_i} q(x^*)\right|$ for $i \in \mathcal{A}_+(x^*)$ and let $\hat{\tau} \in \mathbb{R}^n$ be a parameter (which exists according to Lemma* 8.6*) such that $\nabla^2 q + \mathrm{Diag}(\hat{\tau}) \succcurlyeq 0$, $\hat{\tau}_i \geq 0$ for $i \in \mathcal{A}_+(x^*)$ and $\hat{\tau}_i = 0$ for $i \in \{1, .., n\} \setminus \mathcal{A}_+(x^*)$. If*

$$d_i \geq (\overline{x}_i - \underline{x}_i)\hat{\tau}_i \quad \text{for} \quad i \in \mathcal{A}_+(x^*) \tag{8.9}$$

then x^* is a global minimizer and $\mathrm{val}(\mathrm{Dual}(B_2)) = q(x^*)$.

Proof. We can assume that $x_i^* = \overline{x}_i$ for all $i \in \mathcal{A}_+(x^*)$. The set $S_{\hat{\tau}}$ reads $S_{\hat{\tau}} = \{x \in \mathbb{R}^n \mid 0 \geq e_i^T(x - x^*) \geq -\frac{\mu_i^*}{\hat{\tau}_i}, \quad i \in \mathcal{A}_+(x^*)$ and $\hat{\tau}_i > 0\}$, where μ^* is the dual point related to the constraint $x - \overline{x} \leq 0$. Since $d_i = \mu_i^*$ and $0 \geq e_i^T(x - x^*) \geq \underline{x}_i - \overline{x}_i$ for all $i \in \mathcal{A}_+(x^*)$, from (8.9) it follows that $[\underline{x}, \bar{x}] \subset S_{\hat{\tau}}$, which proves the statement due to Proposition 8.7. □

From Lemma 8.3 follows:

Corollary 8.12. *Let $X^* = \underset{x \in \mathbb{R}^n}{\mathrm{Argmin}}\, L_2(x; \mu^*)$, where L_2 is the Lagrangian corresponding to (B_2) and μ^* is a dual solution of (B_2). Assume there exists a local minimizer of (B_1) fulfilling the assumption of Lemma* 8.11. *Then there exists a global minimizer of (B_1) in X^*. If (B_1) has a unique solution x^*, then $X^* = \{x^*\}$.*

This shows that all instances of problem (B_1) fulfilling the assumption of Corollary 8.12 can be solved by simply computing $\mathrm{val}(\mathrm{Dual}(\mathrm{B}_2))$. This can be done in polynomial time and it is not necessary to compute a local minimizer. Note that, assuming that Condition 8.4 is fulfilled at a point x^*, condition (8.9) can always be satisfied if $\mathrm{diam}([\underline{x}, \overline{x}])$ is sufficiently small.

Example 8.13. Consider again Example 8.5: $\min\{-\|x\|^2 \mid 0 \leq x \leq e\}$, where $x \in \mathbb{R}^n$ and $e \in \mathbb{R}^n$ is the vector of 1s. The unique global minimizer $x^* = e$ fulfills

Condition 8.4. Since $\mu^* = d = 2e$, $\mathcal{A}_+(x^*) = \{1, .., n\}$ and $\hat{\tau} = 2e$, it follows that x^* fulfills (8.9).

Another important quadratic program is the standard quadratic program (Bomze, 1998) defined by

$$(S_1) \qquad \begin{array}{ll} \min & q(x) \\ \text{s.t.} & 0 \leq x \leq e \\ & e^T x - 1 = 0 \end{array}$$

where $q(x) = x^T A x + 2a^T x + c$ and $e \in \mathbb{R}^n$ is the vector of 1s. Consider the extended quadratic program

$$(S_2) \qquad \begin{array}{lll} \min & q(x) & \\ \text{s.t.} & 0 \leq x \leq e & \\ & x_i(x_i - 1) \leq 0, & 1 \leq i \leq n \\ & e^T x - 1 = 0 & \\ & (e^T x - 1)^2 = 0. & \end{array}$$

Let $E = \{(i, j) \mid 1 \leq i < j \leq n, \partial_{ii} q(x) - 2\partial_{ij} q(x) + \partial_{jj} q(x) > 0\}$, where $\partial_{ij} q(x)$ denotes the second derivative of $q(x)$ with respect to the variables x_i and x_j. A further reformulation of (S_1) is

$$(S_3) \qquad \begin{array}{lll} \min & q(x) & \\ \text{s.t.} & x \geq 0 & \\ & x_i x_j \geq 0, & (i, j) \in E \\ & e^T x - 1 = 0 & \\ & (e^T x - 1)^2 = 0. & \end{array}$$

Denote by Dual(S_2) and by Dual(S_3) the dual problems of (S_2) and (S_3) respectively. Problem (S_2) contains the redundant constraints (I) and (II) in problem (Q[U]) in Proposition 8.7 with respect to all global minimizers. Therefore, we can expect that a similar result as in Lemma 8.11 holds for problem (S_2) and (S_3).

Lemma 8.14. (i) *It holds that* $\text{val}(\text{Dual}(S_2)) \leq \text{val}(S_3)$.

(ii) *Let x^* be a local minimizer of problem (S_1) fulfilling Condition 8.4 and $l \in \{1, .., n\}$ be an index with $x_l^* > 0$. Define $d_i = \frac{\partial}{\partial x_i} q(x^*) - \frac{\partial}{\partial x_l} q(x^*)$ for $i \in \mathcal{A}_+(x^*)$ (where $\mathcal{A}_+(x^*)$ is defined as in Condition 8.4). Let $\hat{\tau} \in \mathbb{R}^n$ and $\mu \in \mathbb{R}$ be parameters (which exist according to Lemma 8.6) such that $\nabla^2 q + \text{Diag}(\hat{\tau}) + \mu J \succcurlyeq 0$, $\hat{\tau}_i \geq 0$ for $i \in \mathcal{A}_+(x^*)$ and $\hat{\tau}_i = 0$ for $i \in \{1, .., n\} \setminus \mathcal{A}_+(x^*)$, where $J \in \mathbb{R}^{(n,n)}$ is the matrix of 1s. If*

$$d_i \geq \hat{\tau}_i \text{ for } i \in \mathcal{A}_+(x^*), \tag{8.10}$$

then x^* is a global minimizer of problem (S_1) and

$$\text{val}(\text{Dual}(S_2)) = \text{val}(\text{Dual}(S_3)) = q(x^*).$$

Proof. (i) Denote by $L_2(x;\mu)$ and $L_3(x;\mu)$ the Lagrange functions to (S_2) and (S_3) respectively and let μ^* be an optimal dual point to (S_2). From Nowak, 1999 it follows that the constraints $x_i x_j \geq 0$ for $(i,j) \in E$ can be replaced by the constraints

$$x_i x_j \geq 0, \quad 1 \leq i,j \leq n.$$

Let $x \in \mathbb{R}^n$ be a point fulfilling $e^T x - 1 = 0$. Then

$$x_i(x_i - 1) = - \sum_{1 \leq k \leq n, k \neq i} x_i x_k, \quad 1 \leq i \leq n+1.$$

This implies that there exists $\hat{\mu}$ such that $L_1(x;\mu^*) = L_2(x;\hat{\mu})$ for all $x \in \mathbb{R}^n$ with $e^T x = 1$. From Lemma 5.3 it follows that

$$\operatorname{val}(\operatorname{Dual}(S_2)) = \min_{e^T x = 1} L_1(x;\mu^*) = \min_{e^T x = 1} L_2(x;\hat{\mu}) \leq \operatorname{val}(S_3).$$

This proves the assertion.

(ii) The set $S_{\hat{\tau}}$ defined in Proposition 8.7 reads $S_{\hat{\tau}} = \{x \in \mathbb{R}^n \mid 0 \geq e_i^T(x - x^*) \geq -\frac{\mu_i^*}{\hat{\tau}_i}, \quad i \in \mathcal{A}_+(x^*) \text{ and } \hat{\tau}_i > 0\}$. From $\nabla q(x^*) + \sum_{i \in \mathcal{A}_+(x^*)} -\mu_i^* e_i + \mu_0^* e = 0$, where μ_i^* and μ_0^* correspond to the constraints $x_i \geq 0$ and $e^T x = 1$ respectively, we have $\mu_0^* = -\frac{\partial}{\partial x_l} q(x^*)$ and $\mu_i^* = \frac{\partial}{\partial x_i} q(x^*) + \mu_0^*$ for $i \in \mathcal{A}_+(x^*)$. Since $d_i = \mu_i^*$ and $0 \geq e_i^T(x - x^*) \geq -1$ for all $i \in \hat{\mathcal{A}}_+(x^*)$, from (8.10) it follows that $[0,e] \subset S_{\hat{\tau}}$. This proves the statement. □

Similarly as in Corollary 8.12, from Lemma 8.3 follows:

Corollary 8.15. *Let* $X_2^* = \operatorname{Argmin}_{x \in \mathbb{R}^n} L_2(x;\mu_2^*)$ *and* $X_3^* = \operatorname{Argmin}_{x \in \mathbb{R}^n} L_3(x;\mu_3^*)$, *where* L_2 *and* L_3 *are the Lagrangian corresponding to* (S_2) *and* (S_3), *respectively, and* μ_2^* *and* μ_3^* *are solutions of Dual*(S_2) *and Dual*(S_3), *respectively. Assume there exists a local minimizer of* (S_1) *fulfilling the assumption of Lemma* 8.14. *Then there exists a global minimizer of* (S_1) *in* X_1^* *and in* X_2^*, *respectively. If* (S_1) *has a unique solution* x^*, *then* $X_2^* = X_3^* = \{x^*\}$.

Remarks.

1. In (Nowak, 1998) the lower bound val(Dual(S_3)) was computed for random examples up to 30 variables. The numerical results showed that very often val(Dual(S_3)) is equal to the optimal value.

2. The redundant constraints of (B_2) and (S_2) are also used in (Sherali and Tuncbilek, 1995) for defining so-called RLT-relaxations of nonconvex quadratic programs.

3. Other strong duality results for MIQQPs are given in (Neumaier, 1992; Beck and Teboulle, 2000). A global optimality result for nonlinear programming

based on strong duality is given in (Neumaier, 1996). Apart from strong duality results, global optimality criteria can be derived by checking monotonicity, convexity or uniqueness of a KKT-solution. For this, interval arithmetic can be used (see Section 6.1 and the following Section).

8.5 Global optimality via interval-gradient cuts

This section describes a method for verifying global optimality of local solutions of general nonlinear programs based on interval-gradient cuts, which were introduced in Section 7.1.3. Consider a nonlinear program of the form

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & g_j(x) \le 0, \quad j = 1, \dots, m \\ & x \in [\underline{x}, \overline{x}] \end{array} \tag{8.11}$$

where the functions g_j, $j = 1, \dots, m$, are continuously differentiable. Let $[\underline{d}^j, \overline{d}^j]$ be an interval-gradient of g_j over $[\underline{x}, \overline{x}]$, i.e. $\nabla g_j(x) \in [\underline{d}^j, \overline{d}^j]$ for all $x \in [\underline{x}, \overline{x}]$. An *interval-gradient cut* to g_i is defined, as in Section 7.1.3, by

$$\underline{g}_j(x) = g_j(\hat{x}) + \min_{d \in [\underline{d}^j, \overline{d}^j]} d^T(x - \hat{x}) \le 0.$$

Using interval-gradient cuts, we define the following nonconvex polyhedral outer approximation of (8.11):

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \underline{g}_j(x) \le 0, \quad j = 1, \dots, m \\ & x \in [\underline{x}, \overline{x}]. \end{array} \tag{8.12}$$

Proposition 8.16. *Let $\hat{x}$ be a local minimizer of* (8.11) *that fulfils the following strict KKT-condition:*

$$c + \sum_{j=1}^{m} \mu_j \nabla g_j(\hat{x}) = 0 \tag{8.13}$$

where $\mu_j > 0$ for $j = 1, \dots, m$, all constraints are active, i.e. $g_j(\hat{x}) = 0$ for $j = 1, \dots, m$, and the gradients $\nabla g_j(\hat{x})$, $j = 1, \dots, m$, are linearly independent. Then there exists $\delta > 0$ such that the optimal value of (8.11) *is equal to the optimal value of* (8.12) *whenever $\|\overline{x} - \underline{x}\|_\infty \le \delta$, which shows that $\hat{x}$ is a global minimizer of* (8.11).

Proof. Define for all $z \in \{0,1\}^n$ the following sub-intervals of X:

$$X_z = \{x \in [\underline{x}, \overline{x}] \mid x_i \ge \hat{x}_i \text{ if } z_i = 1 \text{ and } x_i \le \hat{x}_i \text{ if } z_i = 0\}.$$

Defining $d_i^{j,z} = \underline{d}_i^j$ if $z_i = 1$ and $d_i^{j,z} = \overline{d}_i^j$ if $z_i = 0$, the piecewise linear underestimator $\underline{g}_j$ of g_j can be formulated as

$$\underline{g}_j(x) = g_j(\hat{x}) + (d^{j,z})^T(x - \hat{x}) \tag{8.14}$$

for all $x \in X_z$ and $z \in \{0,1\}^n$. From the strict KKT-condition (8.13) and the continuity of ∇g_j it follows that there exist $\epsilon > 0$ such that for all vectors d^j, $j = 1, \ldots, m$, with $\|d^j - \nabla g_j(\hat{x})\|_\infty \leq \epsilon$ there exist $\mu_j \geq 0$ such that

$$c + \sum_{j=1}^{m} \mu_j d^j = 0.$$

From the continuity of the interval-gradient of g_j with respect to $\underline{x}$ and $\overline{x}$ it follows that there exists $\delta > 0$ such that $\|d^{j,z} - \nabla g_j(\hat{x})\|_\infty \leq \epsilon$ for all $z \in \{0,1\}^n$ whenever $\|\overline{x} - \underline{x}\| \leq \delta$. We assume now that $\|\overline{x} - \underline{x}\| \leq \delta$. The previous considerations and $\underline{g}_j(\hat{x}) = g_j(\hat{x}) = 0$ for $j = 1, \ldots, m$ show that for all $z \in \{0,1\}^n$, $\hat{x}$ is a KKT-point of the linear program (8.12) with the additional constraint $x \in X_z$. Hence, $\hat{x}$ is a global minimizer of (8.12), proving the statement. □

Note that if g_j is a convex function, a linearization cut can be used instead of an interval-gradient cut for defining the polyhedral outer-approximation (8.12).

Chapter 9

Adaptive Discretization of Infinite Dimensional MINLPs

This chapter presents a framework for simultaneously improving relaxations and discretizations of infinite dimensional (or very large) optimization problems in multistage stochastic programming and optimal control. In other words, the mesh or scenario generation is included in the optimization method. The approach is based on a new idea for checking the importance of new discretization points via dual solutions of convex relaxations. The concepts presented here are of preliminary character. Only the basic ideas are described, without numerical results. The use of MINLP in optimal control and stochastic programming is quite new. The following approach may give some new directions for further research in these interesting fields.

Several scenario reduction/generation approaches have been proposed in the literature. For example, in (Casey and Sen, 2003) a scenario generation algorithm for linear stochastic programs is proposed and in (Dupacová et al., 2003) a scenario reduction approach based on Fortet–Morier distances is discussed. An adaptive scenario reduction method based on some ideas of this chapter is proposed in (Vigerske, 2005).

9.1 Aggregated discretizations

9.1.1 Multistage stochastic programs

This section describes discretized multistage stochastic programs. The notation is based mainly on Dentcheva and Römisch, 2002. Let $\{\xi_t \mid t = 1, 2, \dots\}$ be some discrete-time stochastic process defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with values in $\mathbb{R}^{s_t}$. It is assumed that the modeling time horizon includes T time periods, and that sequential decisions $x_t \in \mathbb{R}^{q_t}$ are made at every stage $t = 1, \dots, T$ based

on the information $\zeta_t = (\xi_1, \ldots, \xi_t)$ available at that time. The condition that x_t may depend only on ζ_t is called the *nonanticipativity condition*. This property is equivalent to the measurability of x_t with respect to the σ-algebra $\mathcal{F}_t$ that is generated by ζ_t. We consider a *multistage stochastic program* (MSP) of the form

$$\begin{array}{lll} \min_x & \mathbb{E} \sum_{t=1}^{T} f_t(\zeta_t, x_t) & \\ \text{s.t.} & \sum_{\tau=1}^{t} A_{t,\tau}(\zeta_t) x_t \geq c_t(\zeta_t), & t = 1, \ldots, T, \ \mathbb{P}\text{-a.s.} \\ & x_t \in X_t(\zeta_t), & t = 1, \ldots, T, \mathbb{P}\text{-a.s.} \\ & x_t = \mathbb{E}[x_t \mid \mathcal{F}_t], & t = 1, \ldots, T, \mathbb{P}\text{-a.s.} \end{array} \tag{9.1}$$

A discretization of (9.1) is defined by considering a finite subset of scenarios $\Omega_N = \{\omega_n\}_{n \in N} \subset \Omega$, where $N \subset \mathbb{N}$. Related to Ω_N, we define decision variables $x_{n,t}$, probabilities p_n and process values $\zeta_{n,t}$, $n \in N$. It can be shown (Dentcheva and Römisch, 2002) that for the discretized MSP there exists a finite partition $\mathcal{E}_t$ of Ω_N such that

$$\mathbb{E}[x_t \mid \mathcal{F}_t] = \sum_{C \in \mathcal{E}_t} \frac{1}{\mathbb{P}(C)} \cdot \left(\sum_{\omega_s \in C} p_s x_{s,t} \right) \xi_C.$$

Defining the relative probability $\bar{p}_m = (\sum_{j \in C_{n,t}} p_j)^{-1} p_m$, a discretized MSP takes the form:

$$(\mathrm{P}_s[N]) \qquad \begin{array}{lll} \min & \mathbb{E} \sum_{t=1}^{T} \sum_{n \in N} f_{n,t}(\zeta_{n,t}, x_{n,t}) & \\ \text{s.t.} & \sum_{\tau=1}^{t} A_{t,\tau}(\zeta_{n,t}) x_{n,t} \geq c_t(\zeta_{n,t}), & t = 1, \ldots, T, \ n \in N \\ & x_{n,t} \in X_t(\zeta_{n,t}), & t = 1, \ldots, T, \ n \in N \\ & x_{n,t} = \sum_{m \in C_{n,t}} \bar{p}_m x_{m,t}, & t = 1, \ldots, T, \ n \in N. \end{array}$$

Consider now an aggregated problem to $(\mathrm{P}_s[N])$ with respect to an aggregated node set $N_{\mathrm{agg}} \subset N$. Let $\{N_j\}_{j \in N_{\mathrm{agg}}}$ be a partition of N, i.e. $\bigcup_{j \in N_{\mathrm{agg}}} N_j = N$ and $N_i \cap N_j = \emptyset$ for $i \neq j$. For the aggregated problem, we claim

$$x_{j,t} = x_{m,t}, \quad t = 1, \ldots, T, \ m \in N_j, \ j \in N_{\mathrm{agg}}. \tag{9.2}$$

Then the aggregated problem to $(\mathrm{P}_s[N])$ reads:

$$
(\tilde{P}_s[N]) \qquad \begin{array}{lll}
\min & \mathbb{E}\sum_{t=1}^{T}\sum_{n\in N} f_{n,t}(\zeta_{n,t}, x_{n,t}) & \\
\text{s.t.} & \sum_{\tau=1}^{t} A_{t,\tau}(\zeta_{n,t})x_{n,t} \geq c_t(\zeta_{n,t}), & t=1,\dots,T,\ n\in N \\
& x_{n,t} \in X_t(\zeta_{n,t}), & t=1,\dots,T,\ n\in N \\
& x_{n,t} = \sum_{m\in C_{n,t}} \bar{p}_m x_{m,t}, & t=1,\dots,T,\ n\in N \\
& x_{j,t} = x_{m,t}, & t=1,\dots,T,\ m\in N_j,\ j\in N_{\text{agg}}.
\end{array}
$$

9.1.2 Optimal control problems

Consider the following *mixed integer optimal control problem* (MIOCP) of the form:

$$
\begin{array}{lll}
\min & \int_{\underline{t}}^{\overline{t}} f(x(t),u,y,t)dt & \\
\text{s.t.} & \dot{x}(t) = h(x(t),u(t),y,t), & t\in[\underline{t},\overline{t}] \quad \text{(a.e.)} \\
& (x(t),u(t)) \in G(y,t), & t\in[\underline{t},\overline{t}] \quad \text{(a.e.)} \\
& y\in[\underline{y},\overline{y}], \quad y_B \text{ binary} & \\
& u\in\mathcal{F}_u, \quad x\in\mathcal{F}_x &
\end{array}
\tag{9.3}
$$

where $x:\mathbb{R}\mapsto\mathbb{R}^{n_x}$, $u:\mathbb{R}\mapsto\mathbb{R}^{n_u}$ and $y\in\mathbb{R}^{n_y}$. Such problems arise in hybrid optimal control, such as the motorized traveling salesman problem (von Stryk and Glocker, 2000). A discretization of (9.3) is defined by replacing the infinite dimensional function spaces $\mathcal{F}_u$ and $\mathcal{F}_x$ with the finite dimensional spaces,

$$
\mathcal{F}_u^N = \{u(t) = \sum_{n\in N} u_n\cdot\varphi_n(t) \mid u_n\in\mathbb{R}^{n_u}\}
$$

$$
\mathcal{F}_x^N = \{x(t) = \sum_{n\in N} x_n\cdot\psi_n(t) \mid x_n\in\mathbb{R}^{n_x}\}
$$

respectively, where φ_n and ψ_n are in appropriate function spaces. Let $\underline{t} = t(n_1) < \cdots < t(n_l) = \overline{t}$ be discretization points of the interval $[\underline{t},\overline{t}]$. For the sake of simplicity, it is assumed that the functions φ_n and ψ_n are affine over the interval $[t(n_k), t(n_{k+1})]$. Note that in this way, higher-degree polynomials can also be represented by adding additional linear equations. Then the discretized MIOCP takes the form:

$$
(\mathrm{P}_c[N]) \qquad \begin{array}{lll}
\min & \int_{\underline{t}}^{\overline{t}} f(x(t),u,y,t)dt & \\
\text{s.t.} & \dot{x}(t_n) = h(x(t_n),u(t_n),y,t_n), & n\in N \\
& (x(t_n),u(t_n)) \in G(y,t_n), & n\in N \\
& y\in[\underline{y},\overline{y}], \quad y_B \text{ binary} & \\
& u\in\mathcal{F}_u^N, \quad x\in\mathcal{F}_x^N. &
\end{array}
$$

For an aggregated node set $N_{\text{agg}} \subset N$, consider a cover $\{N_j\}_{j \in N_{\text{agg}}}$ of N, i.e. $\bigcup_{j \in N_{\text{agg}}} N_j \supseteq N$ and $N_i \cap N_j = \emptyset$ for $i \neq j$, such that

$$t_j = \sum_{m \in N_j} \delta_m t_m, \quad j \in N_{\text{agg}}.$$

Then $(\mathrm{P}_c[N_{\text{agg}}])$ is equivalent to the aggregated problem

$$
(\tilde{P}_c[N]) \qquad
\begin{array}{lll}
\min & \displaystyle\int_{\underline{t}}^{\overline{t}} f(x(t), u, y, t)dt & \\
\text{s.t.} & \dot{x}(t_n) = h(x(t_n), u(t_n), y, t_n), & n \in N \\
& (x(t_n), u(t_n)) \in G(y, t_n), & n \in N \\
& y \in [\underline{y}, \overline{y}], \quad y_B \text{ binary} & \\
& u \in \mathcal{F}_u^N, \quad x \in \mathcal{F}_x^N & \\
& u_j = \displaystyle\sum_{m \in N_j} \delta_m u_m, & j \in N_{\text{agg}} \\
& x_j = \displaystyle\sum_{m \in N_j} \delta_m x_m, & j \in N_{\text{agg}}.
\end{array}
$$

9.1.3 Abstract formulation

Both discretizations $(\mathrm{P}_s[N])$ and $(\mathrm{P}_c[N])$ can be written as a MINLP of the form:

$$
(\mathrm{P[N]}) \qquad
\begin{array}{ll}
\min & F_N(x, y) \\
\text{s.t.} & G_N(x, y) \leq 0 \\
& x \in X_N,\ y \in Y.
\end{array}
$$

Let $\{N_j\}_{j \in N_{\text{agg}}}$ be a cover of N. An aggregated problem to $(\mathrm{P[N]})$ is given by:

$$
(\tilde{P}[\mathrm{N}]) \qquad
\begin{array}{lll}
\min & F_N(x, y) & \\
\text{s.t.} & G_N(x, y) \leq 0 & \\
& x \in X_N,\ y \in Y & \\
& W_j x_{N_j} = 0, & j \in N_{\text{agg}},
\end{array}
$$

where the matrices W_j are defined in such a way that $\mathrm{val}(P[N_{\text{agg}}]) = \mathrm{val}(\tilde{P}[N])$.

9.2 Optimal mesh and scenario refinement

Given a large node set $\overline{N}$ and a coarse discretization $(\mathrm{P}[N^0])$, a method for adaptively generating a discretization $(\mathrm{P}[N])$ of $(\mathrm{P}[N^0])$ is now presented that tries to keep the approximation error $|\,\mathrm{val}(P[N]) - \mathrm{val}(P[\overline{N}])|$ for all node sets $N \subset \overline{N}$ with $|N| \leq \overline{n}$ as small as possible (see Algorithm 9.1).

Let N^j be the node set of the j-th iteration. In each iteration of the method, a set $\overline{N}^j$ of disaggregated nodes is computed, where $\mathrm{val}(P[N^j]) = \mathrm{val}(\tilde{P}[\overline{N}^j])$ and

$\overline{N}^j = \bigcup_{i \in N^j} N_i^j$. The disaggregated node set $\overline{N}^j$ is defined by locally refining scenarios or mesh points. The new node set is defined by $N^{j+1} = \overline{N}^j(M^j)$, where

$$\overline{N}^j(M) = (N^j \setminus M) \cup (\bigcup_{i \in M} N_i^j)$$

is a *partially disaggregated node set*, and $M^j \subset N^j$ is a set of disaggregated nodes with $|M^j| \leq \overline{m}$. The set M^j of disaggregated nodes is computed such that

$$|\operatorname{val}(P[\overline{N}^j]) - \operatorname{val}(P[\overline{N}^j(M^j)])| \quad \text{is small,}$$

which is equivalent to

$$|\operatorname{val}(\tilde{P}[\overline{N}^j]) - \operatorname{val}(P[\overline{N}^j(M^j)])| \quad \text{is large,} \tag{9.4}$$

since $\operatorname{val}(P[N^j]) = \operatorname{val}(\tilde{P}[\overline{N}^j])$. A convex relaxation to $(\tilde{P}[\overline{N}^j])$ is defined by

$$(\tilde{C}[\overline{N}^j]) \qquad \begin{array}{lll} \min & \breve{F}_{\overline{N}^j}(x,y) & \\ \text{s.t.} & \breve{G}_{\overline{N}^j}(x,y) \leq 0 & \\ & W_i x_{N_i^j} = 0, & i \in N^j \end{array}$$

where $\breve{F}_N$ and $\breve{G}_N$ are appropriate convex functions. If the gap $\operatorname{val}(\tilde{P}[\overline{N}^j]) - \operatorname{val}(\tilde{C}[\overline{N}^j])$ is not too large, then (9.4) is similar to

$$|\operatorname{val}(\tilde{C}[\overline{N}^j]) - \operatorname{val}(C[\overline{N}^j(M^j)])| \quad \text{is large.} \tag{9.5}$$

Let τ_i be a dual solution point of problem $(\tilde{C}[\overline{N}^j])$ related to the equality constraint $W_i x_{N_i} = 0$, $i \in N^j$. Since $\|\tau_i\|_\infty$ measures the sensitivity of $\operatorname{val}(\tilde{C}[\overline{N}^j])$ with respect to the constraint $W_i x_{N_i} = 0$, (9.5) is similar to

$$M^j = \operatorname*{argmax}_{|M| \leq \overline{m},\, M \subset N^j} \sum_{i \in M} \|\tau_i\|_\infty. \tag{9.6}$$

Algorithm 9.1 shows an adaptive procedure for generating a discretization (P[N]), based on the refinement criterion (9.6).

9.3 Updating and solving relaxations

Assume that the convex relaxation $(\tilde{C}[N])$ of the MINLP $(\tilde{P}[N])$ has the form of the dual-equivalent relaxation (3.13) defined in Section 3.4, i.e. $\operatorname{val}(\tilde{C}[N]) = \operatorname{val}(\operatorname{Dual}(\tilde{P}[N]))$. Since in Algorithm 9.1 a sequence of similar relaxations $(\tilde{C}[N^j])$ has to be solved, it is highly desirable that the solution-information obtained in the j-th iteration can be used in the $(j+1)$-th iteration.

Initialize N^0, where $|N^0|$ is small.

for $j = 0, \ldots, l$

Update and solve the convex relaxation $(\tilde{C}[\overline{N}^j])$
obtaining dual points τ_i, $i \in N^j$.

Compute M^j according to (9.6) and set $N^{j+1} = \overline{N}^j(M^j)$.

if $|N^{j+1}| > \overline{n}$: **stop**.

end for

Algorithm 9.1: Node generation algorithm

Problem $(\tilde{C}[N])$ can be generated by dual (bundle) methods, cutting-plane methods or column generation methods (see Chapter 4). Bundle methods are based on a bundle of subgradients. Updating such a bundle seems difficult, since it is not clear how subgradients of the j-th iteration can be updated to be valid subgradients for the $(j+1)$-th iteration.

Cutting-plane methods could be efficient if it is possible to update cuts efficiently. For the case when nodes of the same type are aggregated, this seems possible. However, in general it is not clear how a cut obtained in the j-th iteration can be updated to be valid for the $(j+1)$-th iteration.

Column generation seems to be better suited for updating relaxations, since the inner approximation points need not necessarily be extreme points, i.e. solutions of Lagrangian subproblems (see end of Section 4.3). In the case of stochastic programs, an inner approximation point $w_{N,t}$ can be updated by

$$w_{j,t} = w_{m,t}, \quad t = 1, \ldots, T,\ m \in N_j,\ j \in N_{\text{agg}}.$$

In the case of optimal control problems, an inner approximation point $w_{N,t}$ can be updated by

$$w_j = \sum_{m \in N_j} \delta_m w_m, \quad j \in N_{\text{agg}}$$

where $w_n = (x_n, u_n)$.

Part II

Algorithms

Chapter 10

Overview of Global Optimization Methods

Over the last decades many approaches for globally solving nonconvex programs have been developed. These methods can be classified into *exact methods* and *heuristics*. A method is called exact (or deterministic) if it guarantees to find and verify global solutions. Otherwise, it is called a heuristic.

Heuristics try to find global solutions without verifying global optimality. Since reliable deterministic solvers for large-scale MINLPs are often not available, heuristics play a fundamental role in large-scale nonconvex optimization. They can be used as stand-alone solvers or as an acceleration tool in deterministic methods. Apart from providing upper bounds on the global optimum, they can also be used to compute relaxations, to generate cuts, and to find good partitions of the feasible set.

Heuristics with a performance guarantee, in the sense that the expected value of the relative error can be estimated, are called *approximation algorithms*. Of special interest are polynomial time approximation algorithms for NP-hard problems. The derivation of a performance guarantee for such an algorithm often requires a deep analysis of the method. For an overview of this field we refer to (Fisher, 1980; Ausiello et al., 1999; Hochbaum, 1999; Vazirani, 2001). Polynomial time approximation algorithms have mainly been developed for special subclasses of MIP.

The MaxCut heuristic of Goemann and Williamson (Goemans and Williamson, 1995) may be the first approximation algorithm for a quadratic binary program. Approximation algorithms for MINLP can be derived from MIP approximations that are based on approximating univariate functions by piecewise linear functions (see Section 2.4). Since the error of approximating a univariate function by a piecewise linear function is predictable, a performance guarantee for the MINLP-method can be derived, provided that a performance guarantee for the

related MIP-method is available.

Exact global optimization methods find and verify global ϵ-solutions in a finite number of steps. If an algorithm finds and verifies a global solution in finitely many steps, it is called *finite*. Enumeration algorithms for solving integer or concave problems with a bounded feasible set are finite. Other finite algorithms are the simplex method for solving linear programs and active set methods for solving convex quadratic programs. Methods for solving general nonlinear convex problems, such as SQP or interior point methods, are usually not finite.

Although the existing global optimization methods are very different, they all generate a *crude model* of the original problem for finding solutions. If an optimization method uses a sample set as a crude model, i.e. a finite set of points, it is called a *sampling heuristic*. If it uses a relaxation as a crude model, i.e. a mathematical program that is easier to solve than the original problem, it is called a *relaxation-based* method.

In sampling heuristics the points of the sample set are distributed over a bounded set. The distribution of points is usually more dense in 'regions of interest'. These methods use random behavior to try to include all possible solutions. Since in continuous spaces the random selection has an infinite number of possibilities, it cannot be guaranteed that the optimization is global. In general, it is only possible to prove convergence with probability arbitrarily close to 1 for such type of methods.

The type of the crude model influences also the problem description. Whereas for sampling heuristics it is advantageous to formulate the problem in an *aggregated* form with few variables and a simple feasible set, for relaxation-based methods it is often better to work with a *disaggregated* model containing objective and constraint functions that can be relaxed easily.

In the sequel, sampling and relaxation-based methods for solving nonconvex MINLPs are reviewed. Relaxation-based methods are divided into three classes. The first class contains *branch-and-bound methods* that subdivide the original problem into subproblems by partitioning the feasible set. The second class contains *successive relaxation methods* that successively improve an initial relaxation without subdividing it into subproblems. The third class contains heuristics that retrieve solution candidates from a given relaxation without modifying the relaxation. These methods are called *relaxation-based heuristics*.

Currently, there is no method that is able to solve reliably large scale nonconvex MINLPs. In a recent comparison between the two sampling codes LGO (Pintér, 2005) and OQNLP (Lasdon, 2003) and the branch-and-bound code BARON (Sahinidis, 2002), none of the solvers was superior to all others (Bussieck et al., 2003b). There is still a huge gap between MIP and MINLP solver technology. Modern MIP solvers are branch-cut-and-price algorithms with clever preprocessing and constraint propagation techniques (Bixby et al., 2000). Current general purpose MINLP solvers are not much developed, and the methods are often implemented in a rudimentary form. As a result, MIP is often used for modeling practically-relevant large-scale problems.

For more detailed information on global optimization and MINLP methods, the reader is referred to (Horst et al., 1995; Horst and Pardalos, 1995; Horst and Tuy, 1990; Forgó, 1988; Pardalos and Rosen, 1987; Pintér, 1996; Neumaier, 2004; Schichl, 2004) and to (Floudas, 2000; Grossmann, 2001; Grossmann and Kravanja, 1997; Floudas, 1995; Floudas, 2000; Tawarmalani and Sahinidis, 2002) respectively. An overview on sampling heuristics can be found in (Törn and Zilinskas, 1989; Boender and Romeijn, 1995; Strongin and Sergeyev, 2000).

10.1 Sampling heuristics

Multistart. An obvious probabilistic global search procedure is to use a local algorithm starting from several points uniformly distributed over the whole optimization region. This global search procedure is named *Multistart* and is certainly one of the earliest global procedures used. It has even been used in local optimization for increasing the confidence in the obtained solution. The starting points can be generated randomly or by a deterministic method, for example, by using *space filling curves* (Strongin and Sergeyev, 2000). One drawback of Multistart is that when many starting points are used the same minimum will eventually be determined several times.

Clustering methods. *Clustering methods* try to avoid the repeated determination of the same local minima. This is realized in three steps which may be iteratively used. The three steps are: (i) Sample points in the region of interest. (ii) Transform the sample to obtain points grouped around the local minima. (iii) Use a clustering technique to recognize these groups (i.e. neighborhoods of the local minima). If the procedure employing these steps is successful then starting a single local optimization from each cluster would determine the local minima and thus also the global minimum. The advantage in using this approach is that the work saved by computing each minimum just once can be spent on computations in (i)–(iii), which will increase the probability that the global minimum will be found (Becker and Lago, 1970; Dixon and Szegö, 1975; Törn and Zilinskas, 1989).

Evolutionary algorithms. *Evolutionary algorithms* (Forrest, 1993) are search methods that take their inspiration from natural selection and survival of the fittest in the biological world. They differ from more traditional optimization techniques in that they involve a search from a "population" of solutions, not from a single point. In each iteration, the algorithm uses three operations to modify the population: *reproduction*, *crossover* and *mutation*. Reproduction copies a solution from the old population to the new population with a probability depending on the fitness of the solution, which is determined by the value of the objective or penalty function. Crossover combines two solutions to two new solutions by swapping binary sections. For example, the crossover of 10|001 and 11|101 may produce the new solutions 10|101 and 11|001. The operation tries to create a new solution

that has the best properties of the old solutions. Mutation produces new solutions by randomly changing a small part of an old solution. This operation allows the algorithm to jump into unexplored regions, which might contain better solutions. Such kind of algorithms may be well-suited if the problem is highly nonlinear and discrete. In the presence of continuous variables the random net could be not tight enough to reach the global minimum. The larger the initial sample set is, the higher is the probability to find the right solution. However, working with a large sample set can be very time-consuming.

Simulated annealing. In *simulated annealing* (Kirkpatrick et al., 1983; Locatelli, 2002) points of a sample set are modified by applying a descent step with a random search direction and an initially large step-size that is gradually decreased during the optimization process. The method generalizes a Monte Carlo method for examining the equations of state and frozen states of n-body systems (Metropolis et al., 1953). It takes the inspiration from the slow cooling of a metal that brings it to a crystalline state where the free energy of bulk matter could take its global minimum. Simulated annealing has similar disadvantages and advantages as evolutionary algorithms.

Tabu search. The *Tabu search* (Glover and Laguna, 1997) begins by marching to a local minimum. To avoid retracing the steps used, the method records recent moves in one or more Tabu lists. The Tabu lists form the Tabu search memory. The role of the memory can change as the algorithm proceeds. At initialization the goal is to make a coarse examination of the solution space, known as 'diversification', but as candidate locations are identified the search is more focused to produce local optimal solutions in a process of 'intensification'. In many cases the differences between the various implementations of the Tabu method have to do with the size, variability, and adaptability of the Tabu memory to a particular problem domain.

Statistical global optimization. *Statistical global optimization* algorithms (Mockus, 1989) employ a statistical model of the objective function to bias the selection of new sample points. These methods are justified with Bayesian arguments which suppose that the particular objective function being optimized comes from a class of functions that is modeled by a particular stochastic function. Information from previous samples of the objective function can be used to estimate parameters of the stochastic function, and this refined model can subsequently be used to bias the selection of points in the search domain.

Greedy randomized adaptive search procedure. A *greedy randomized adaptive search procedure* (GRASP) (Resende and Ribeiro, 2002) is a multistart metaheuristic that applies local search to starting points generated by a greedy randomized construction procedure.

10.2 Branch-and-bound methods

Branch-and-bound. Originally invented for solving combinatorial optimization problems, *branch-and-bound* was generalized to solve continuous problems. A detailed introduction of branch-and-bound for global optimization is given in (Horst and Tuy, 1990).

To apply branch-and-bound, one must have a means of computing a lower bound on an instance of the optimization problem and a means of dividing the feasible region of a problem to create smaller subproblems. There must also be a way to compute an upper bound (feasible solution) for at least some instances.

The method starts by considering the original problem with the complete feasible region, which is called the root problem. The lower-bounding and upper-bounding procedures are applied to the root problem. If the bounds match, then an optimal solution has been found and the procedure terminates. Otherwise, the feasible region is divided into two or more regions. These subproblems become children of the root search node. The algorithm is applied recursively to the subproblems, generating a tree of subproblems. If an optimal solution is found to a subproblem, it can be used to prune the rest of the tree: if the lower bound for a node exceeds the best known feasible solution, no globally optimal solution can exist in the partition subspace of the feasible region represented by the node. Therefore, the node can be removed from consideration. The search proceeds until all nodes have been solved or pruned, or until some specified threshold is met between the best solution found and the lower bounds on all unsolved subproblems.

Some branch-and-bound methods for MINLPs are the *reformulation / spatial branch-and-bound approach* (Smith and Pantelides, 1996; Smith and Pantelides, 1999) and the *interval analysis based approach* (Vaidyanathan and EL-Halwagi, 1996; Ratschek and Rokne, 1995).

Branch-and-cut. The integration of a cut generating procedure into branch-and-bound is called *branch-and-cut*. The roots of this approach go back to (Padberg and Rinaldi, 1991). In branch-and-cut, cutting-planes are added iteratively until either a feasible solution is found or it becomes impossible or too expensive to find another cutting-plane. In the latter case, a traditional branching operation is performed and the search for cutting-planes continues on the subproblems.

Branch-and-reduce. *Branch-and-reduce* is branch-and-bound combined with box reduction for tightening lower bounds (Sahinidis, 1996).

Branch-and-price. *Branch-and-price* is essentially branch-and-bound combined with column generation. This method is used mainly to solve integer programs where there are too many variables to represent the problem explicitly. Thus, only the active set of variables are maintained and columns are generated as needed

during the solution of the linear master program. Column generation techniques are problem specific and can interact with branching decisions.

Branch-cut-and-price. The integration of both cutting-planes and column generation into branch-and-bound is called *branch-cut-and-price* (BCP). BCP is used mainly in MIP. In Chapter 13 a BCP algorithm for general MINLPs is presented.

Branch-and-infer. *Branch-and-infer* is the combination of branch-and-bound with *constraint propagation* (CP). This method uses tests of infeasibility and global optimality to prune the search tree (Van Hentenryck et al., 1997). In the last years the integration of concepts from operations research and CP has been studied. A recent overview on this integration is provided in (Bliek et al., 2001; Hooker, 2000). A CP-based algorithm for finding solutions of large systems of quadratic constraints is proposed in (Boddy and Johnson, 2003). A combination of CP and Lagrangian relaxation is presented in (Sehlmann and Fahle, 2003).

10.3 Successive approximation methods

Successive approximation algorithms start with an initial relaxation that is successively improved without subdividing the given optimization problem into subproblems, such as in branch-and-bound methods. During the iteration, lower and upper bounds of the optimal value are generated that converge towards the optimal value.

Extended cutting-plane method. The *extended cutting-plane method* (ECP) solves a (quasi) convex MINLP using a LP master program (Westerlund et al., 1994; Westerlund and Petterson, 1995; Westerlund et al., 1998). In each iteration, the method generates cuts by solving MIP-subproblems obtained from linearizations of nonlinear objective and constraint functions at trial points.

Generalized Benders decomposition. In *general Benders decomposition* (GBD) (Geoffrion, 1972; G. E. Paules and Floudas, 1989; Floudas et al., 1989) the MIP master program is defined by fixing variables and adding cuts obtained from the solution of the NLP subproblems via duality. The minimization of the MIP master problem gives a lower bound, and the solution of the NLP subproblems give an upper bound on the optimal value. The method solves alternatively the MIP master problem and the NLP subproblems until the difference between the upper and lower bound is smaller than a given error tolerance.

Outer approximation. The *outer approximation* (OA) method is a cutting-plane method that uses a MIP master program (Duran and Grossmann, 1986). The cuts are generated by minimizing the NLP subproblems obtained from fixing the integer

variables and by linearizing the nonlinear objective and constraint functions. In (Fletcher and Leyffer, 1994) an OA method that uses a MIQP master problem is proposed.

In general, the OA method requires fewer iterations and thus the solution of fewer NLP subproblems than GBD, but the MIP problems require more computation. Similar to the ECP and GBD method, the OA method requires that the MINLP is convex. Attempts to generalize OA to solve nonconvex problems are proposed in (Kocis and Grossmann, 1987; Viswanathan and Grossmann, 1990; Kesavan et al., 2001). A *hybrid branch-and-bound and outer approximation approach* that updates a branch-and-bound method for solving the MIP master problem is described in (Zamora and Grossmann, 1998b; Zamora and Grossmann, 1998a).

Logic-based approach. The *logic-based approach* is a cutting-plane method for solving convex MINLPs that uses an MIP master problem. In each iteration, cuts are generated by solving a separation problem that is defined by disjunctive constraints (Turkay and Grossmann, 1996; Vecchietti and Grossmann, 1999).

Generalized cross decomposition. *Generalized cross decomposition* is the integration of Benders decomposition and Lagrangian decomposition (Holmberg, 1990).

Successive semidefinite relaxation. The *successive semidefinite relaxation* method solves general polynomial programs by iteratively improving semidefinite relaxations (Henrion and Lasserre, 2002). The method is based on the results of (Lasserre, 2001) that show that general nonconvex polynomial programs can be approximated by semidefinite relaxations with arbitrary precision. A general framework for successive convex relaxation of polynomial programs is proposed in (Kojima et al., 2003).

Lagrangian and domain cut method. In this recent approach, a MINLP is solved by successively refining a Lagrangian relaxation via nonconvex rectangular subdivisions of the domain (Li et al., 2002).

10.4 Relaxation-based heuristics

Relaxation-based heuristics generate solution candidates by using a given relaxation without improving the relaxation. In contrast to exact methods, the relaxation can be any problem that is easier to solve than the original problem and that need not be rigorous.

Rounding heuristics. *Rounding heuristics* in MIP are based on rounding fractional solutions of LP relaxations. Several MIP rounding heuristics are proposed and compared in (Burkard et al., 1997). A rounding heuristic for obtaining solutions of convex MINLPs is proposed in (Mawengkang and Murtagh, 1986). Here, a

relaxed NLP solution is rounded to an integral solution with the best local degradation by successively forcing the superbasic variables to become nonbasic based on the reduced cost information. Rounding heuristics for nonconvex quadratic (integer) programs based on semidefinite relaxations are described in Goemans and Williamson, 1995 and Zwick, 1999. Chapter 12 presents rounding heuristics for general nonconvex MINLPs.

Lagrangian heuristics. *Lagrangian heuristics* generate solution candidates by making solutions of Lagrangian relaxations feasible with respect to coupling constraints. In order to facilitate this task, 'user knowledge' or problem specific rules can be used (see for example Holmberg and Ling, 1997; Nowak and Römisch, 2000). A Lagrangian heuristic for MINLP is proposed in Section 12.4.

Deformation heuristics. *Deformation heuristics* are based on gradually deforming an initial relaxation that has few local solutions into the original problem. During the deformation, trial points are used to generate new solution candidates via neighborhood techniques (Moré and Wu, 1997; Alperin and Nowak, 2002). Scheltstraete et al. (Schelstraete et al., 1998) provide an overview on this kind of heuristics. Chapter 11 presents deformation heuristics for MaxCut and MINLP.

MIP approximation. Since there exist powerful codes for solving MIPs, MINLPs are often solved in practice by *MIP approximation* (Neumaier, 2004). To this end, the problem is reformulated as a separable program (see Section 2.4) and univariate nonlinear functions are approximated by piecewise linear functions. The approach is only efficient if the number of additional constraints and logical variables is not too large.

Successive linear programming. In *successive linear programming*, solution candidates are computed by alternatively solving MIP approximations generated by linearizing nonlinear functions at trial points and NLP subproblems with fixed integer variables. If the objective function or some constraints of the MINLP are nonconvex, it cannot be guaranteed that a MIP linearization is feasible. The approach is local in the sense that the solution depends on the starting point of the iteration.

Chapter 11

Deformation Heuristics

Deformation heuristics are based on a smoothing transformation that changes a difficult optimization problem into a relaxed problem that is easier to solve. They solve a sequence of relaxed problems converging towards the original problem. Since the approach is generic, it can theoretically be applied to any optimization problem. The best-known deformation methods may be interior point methods for solving convex optimization problems, where the smoothing transformation is defined by a barrier or potential function.

A deformation heuristic for distance geometry problems that is based on smoothing the objective function by using the so-called Gaussian transformation is proposed in (Moré and Wu, 1997). A deformation heuristic for a combinatorial optimization problem is presented in Chapter 6 of (Warners, 1999), where the smoothing operator is defined via a potential function. Scheltstraete et al. (Schelstraete et al., 1998) provide an overview of deformation heuristics for solving nonconvex energy minimization problems.

The deformation heuristics for MaxCut (Alperin and Nowak, 2002) and general MINLPs presented in this chapter are based on smoothing the objective function by combining it with a convex underestimator. Numerical results for both MaxCut and MINLP instances are reported.

11.1 The algorithm of Moré and Wu

Moré and Wu presented in (Moré and Wu, 1997) a deformation heuristic for solving distance geometry problems of the form:

$$\text{(P)} \qquad \min f(x) = \sum_{i,j \in I} p_{i,j}(x_i - x_j)$$

where $p_{i,j} : \mathbb{R}^n \mapsto \mathbb{R}$ is a pair-wise potential function and $I \subset \mathbb{N}$ is an index set. This problem is known to have a large number of local minimizers. In order to

relax the problem, the *Gaussian smoothing transformation*:

$$G(x;t) = \frac{1}{\pi^{n/2}t^n} \int_{\mathbb{R}^n} f(y) \exp\left(-\frac{\|y-x\|^2}{t^2}\right) dy$$

is used. The parameter t controls the degree of smoothing. The original function is obtained if $t \to 0$, while smoother functions are obtained as t increases. This transformation reduces the number of local minimizers, while the overall structure is maintained. The solution approach of Moré and Wu is based on solving the parametric optimization problem:

$$(\mathrm{P}_t) \qquad \min_x G(x;t)$$

by using the method described in Algorithm 11.1.

Input: a sequence of continuation points $t^0 > \cdots > t^l = 0$

Choose a random vector $x^0 \in \mathbb{R}^n$.

for $j = 0, \ldots, l$

Determine a local minimizer x^{j+1} of (P_{t^j}) starting from x^j.

end for

Algorithm 11.1: Deformation heuristic of Moré and Wu

The computational experiments in (Moré and Wu, 1997) show that Algorithm 11.1 with an iteration number $l > 0$ requires less than twice the effort (measured in terms of function and gradient evaluations) than $l = 0$, although Algorithm 11.1 has to solve $l + 1$ optimization problems. Motivated by these results, deformation heuristics for MaxCut and MINLP are proposed in the following that use a convex relaxation instead of Gaussian smoothing. The smoothing of a nonconvex function by using a convex underestimator is shown in Figure 11.1.

11.2 A MaxCut deformation heuristic

This section describes a deformation heuristic for MaxCut (Alperin and Nowak, 2002) that uses a smoothing transformation defined by a convex relaxation.

11.2.1 Problem formulation

Let $G = (V, E)$ be an undirected weighted graph consisting of the set of nodes $V = \{1, \ldots, n\}$ and the set of edges E. Let a_{ij} be the cost of edge (i, j), and

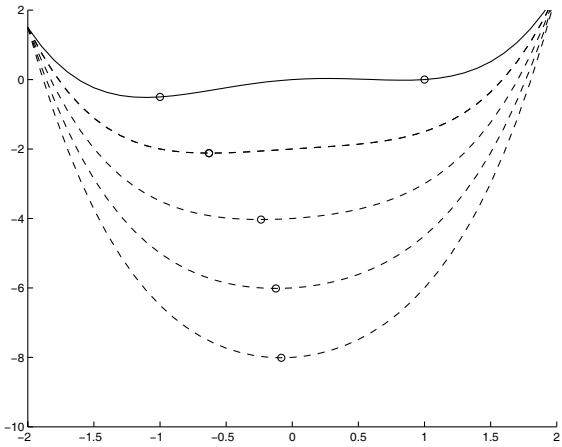

Figure 11.1: Deformation of a convex underestimator into the original function

assume that G is complete, otherwise set $a_{ij} = 0$ for every edge (i, j) not in E. The *Maximum Cut problem* (MaxCut) consists of finding a subset of nodes $S \subset N$ that maximizes the cut function:

$$\text{cut}(S) = \sum_{(i,j)\in\delta(S)} a_{ij},$$

where the incidence function $\delta(S) = \{(i, j) \in E \mid i \in S \text{ and } j \notin S\}$ is defined to be the set of arcs that cross the boundary of S.

It is well known that MaxCut can be formulated as the following nonconvex quadratic problem:

$$\text{(MC)} \qquad \begin{array}{ll} \min & x^T Ax \\ \text{s.t.} & x \in \{-1, 1\}^n. \end{array}$$

Since

$$x^T Ax = \sum_{\substack{i,j=1 \\ x_i x_j > 0}}^{n} a_{ij} - \sum_{\substack{i,j=1 \\ x_i x_j < 0}}^{n} a_{ij} = \sum_{i,j=1}^{n} a_{ij} - 2 \sum_{\substack{i,j=1 \\ x_i x_j < 0}}^{n} a_{ij} = e^T Ae - 4 \sum_{ij\in\delta(S)} a_{ij}$$

if $x \in \{-1, 1\}^n$, the maximum $\text{cut}(S)$ can be produced by minimizing $x^T Ax$, then adding the constant $e^T Ae$, and finally dividing by 4. From Section 5.2 it follows that (MC) is equivalent to the *unconstrained quadratic binary problem*:

$$\text{(QBP)} \qquad \begin{array}{ll} \min & x^T Ax + 2b^T x + c \\ \text{s.t.} & x \in \{0, 1\}^n, \end{array}$$

where $A \in \mathbb{R}^{(n,n)}$ is symmetric, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. The equivalence between (MC) and (QBP) is also shown in Helmberg, 2000.

Although problem (MC) was proven to be NP-hard (Garey and Johnson, 1979), some interesting heuristics to obtain good solutions have been proposed. Following, the most well-known and recent ones are presented. An excellent overview of solution approaches including applications of MaxCut is given in (Helmberg, 2000).

Goemans and Williamson (Goemans and Williamson, 1995) used the solution of the semidefinite program:

$$\text{(SDP)} \qquad \begin{array}{lll} \min & \langle A, X\rangle & \\ \text{s.t.} & \operatorname{diag}(X) & = e \\ & X & \succcurlyeq 0 \end{array}$$

to generate solution candidates. Assuming that X^* is an (SDP) optimal solution, which is not necessarily rank-1, their strategy consists of finding a factorization $X^* = V^T V$, where V can be the Cholesky factorization. A feasible solution $\hat{x} = \operatorname{sign}(V^T u)$ can be produced using the random vector $u \sim U[B(0,1)]$, uniformly distributed over the zero-centered n-dimensional ball of radius 1. For the case of non-negative edge weights, $a_{ij} \geq 0$, they proved a bound on the expected value of the randomly generated solutions that is

$$E(\hat{x}^T A \hat{x}) \geq .878 \text{ val(MC)},$$

where val(MC) is the optimal value of problem (MC), and $E(\hat{x}^T A \hat{x})$ is the expected value of the objective function of (MC) using the randomly generated feasible point $\hat{x}$. A similar procedure for rounding solutions of SDP relaxations was proposed by Zwick (Zwick, 1999).

Burer et. al. (Burer et al., 2001) have devised a rank-2 relaxation of problem (MC). In their heuristic, they relax the binary vector into a vector of angles and work with an angular representation of the cut. They maximize an unconstrained sigmoidal function to obtain heuristic points that later are perturbed to improve the results of the algorithm. Their approach is similar to the Lorena (Berloni et al., 1998) algorithm. Metaheuristics for solving (QBP) are proposed and studied for cases containing up to 2500 variables in (Beasley, 1998).

11.2.2 A MaxCut algorithm

Consider a box-constrained quadratic parametric reformulation of MaxCut defined by

$$(\mathrm{P}_t) \qquad \begin{array}{ll} \min & H(x;t) \\ \text{s.t.} & x \in [-e, e], \end{array}$$

where the function

$$H(x;t) = tP(x;t) + (1-t)L(x;\mu)$$

is a convex combination of the Lagrangian

$$L(x;\mu) = x^T A x + \sum_{i=1}^{n} \mu_i (x_i^2 - 1)$$

and the penalty function

$$P(x;t) = x^T Ax + \frac{1}{1-t}(n - \|x\|^2).$$

Lemma 11.1. *There exists a value* $t_{\min} \in (0,1)$ *such that* $\mathrm{val}(MC) = \mathrm{val}(P_t)$ *for all* $t \in [t_{\min}, 1)$.

Proof. Since

$$\nabla^2 H(x;t) = 2\left(tA - \frac{t}{1-t}I + (1-t)\,\mathrm{Diag}(\mu)\right),$$

there exists $t_{\min}$ such that $H(\cdot;t)$ is concave for all $t \in [t_{\min}, 1)$. Furthermore, $H(x;t) = x^T Ax$ for all $x \in \{-1,1\}^n$, $\mu \in \mathbb{R}^n$, and $t \in [0,1)$. This proves the statement. □

Remark 11.2. Note that it may occur that the path $x(t)$ of the parametric optimization problem (P_t) related to a solution x^* of (MC) is discontinuous (Guddat et al., 1990).

Remark 11.3. Dentcheva et al. (Dentcheva et al., 1995) and Guddat et al. (Guddat et al., 1998) pointed out general disadvantages of the formulation (P_t), namely the one-parameter optimization is not defined for $t = 1$ and the objective function can be only once continuously differentiable. However, for MaxCut the penalty objective function used is quadratic, thus infinitely many times differentiable. On the other side, from Lemma 11.1, the path need not be traced until $t = 1$.

Assuming that the dual point $\mu \in \mathbb{R}^n_+$ is large enough, the function $H(\cdot;0) = L(\cdot;\mu)$ is a convex underestimator of $x^T Ax$ over $[-e,e]$ and (P_0) is a convex relaxation of (MC). From Lemma 11.1 it follows that the solution set of (P_t) approaches the solution set of (MC) when t tends to 1. Similar to the previously described deformation heuristic of Moré and Wu, Algorithm 11.2 solves MaxCut by successively computing near-optimal solutions of (P_t) with a projected gradient algorithm. Here, $\Pi_{[-e,e]}(x)$ denotes the projection of x onto the interval $[-e,e]$, i.e.

$$\Pi_{[-e,e]}(x)_i = \begin{cases} -1 & \text{if} \quad x_i < -1 \\ x_i & \text{if} \quad -1 \le x_i \le 1 \\ 1 & \text{if} \quad x_i > 1. \end{cases}$$

The parameters β^i in Algorithm 11.2 determine the step-length of the projected gradient algorithm. It is possible to auto-tune β^i to guarantee descent steps using the bisection rule, or to use a fixed value $\beta^i = \beta$.

The continuation points, $t^1 < \cdots < t^j < \cdots < t^l$, determine the values at which the function $H(x;t^j)$ is optimized. It is possible to generate t^j, using a *geometric* sequence, i.e. $t^j = 1 - \rho^j$ with $\rho \in (0,1)$, or using a *uniform* sequence, i.e. $t^j = j/(l+1)$.

Input: a sequence of continuation points $0 < t^0 < \cdots < t^l < 1$

Choose a random vector $x^0 \in [\underline{x}, \overline{x}]$ and a dual point μ defining a convex Lagrangian $L(\cdot; \mu)$.

for $j = 0, \ldots, l$

$y^0 = x^j$.

for $i = 0, \ldots, m-1$

$$y^{i+1} = \Pi_{[-e,e]} \left(y^i - \beta^i \frac{\nabla H(y^i; t^j)}{\|\nabla H(y^i; t^j)\|} \right).$$

end for

$x^j = y^m$.

end for

Algorithm 11.2: MaxCut deformation heuristic

Remark 11.4. From Lemma 11.1 it follows that $H(\cdot; t^j)$ is concave if $t^j \geq t_{\min}$. Assuming that m is large enough and $t^j \geq t_{\min}$, the projected gradient algorithm converges to a vertex in finitely many steps, and Algorithm 11.2 can be stopped without changing the final result.

11.2.3 Sampling

Algorithm 11.2 depends highly on the initial primal points and on the dual point that defines the convex relaxation. Several techniques for generating sample points in the primal and dual space are used.

Random primal. The vector $\mu = -\lambda_1(A)e$ is used as a dual point and an initial primal point is chosen with uniform distribution over the ball $I\!B(n) = I\!B^n(0, n^{1/2})$, i.e. the sampling set is defined by

$$S_{\mathrm{RP}} = \{(x^i, \mu) \mid i = 1, \ldots, p,\ x^i \sim U[I\!B(n)],\ \mu = -\lambda_1(A)e\},$$

where $x^i \sim U[S]$, means that the sample points x^i are independently drawn from a uniform distribution over the set S.

Eigenspace sampling. If the duality gap is zero, an optimal primal solution lies in the eigenspace of the minimum eigenvalue. Motivated by this fact, random points in the space spanned by the eigenvectors that correspond to a certain number of smallest eigenvalues are generated. In particular, we define

$$S_{\mathrm{E}} = \{(x^i, \mu) \mid i = 1, \ldots, p,\ x^i = n^{1/2} y^i / \|y^i\|,\ \mu = -\lambda_1(A)e\},$$

where

$$y^i = \sum_{k=1}^{r} \alpha_k v_k(A + \mu I),$$

$\alpha_k \sim N(0,1)$ are independent normally distributed, and $v_i(\cdot)$ is the eigenvector corresponding to the i-th lowest eigenvalue. The resulting random linear combination of eigenvectors, y^i is projected onto $\mathbb{B}(n)$, the ball that contains the $[-e, e]$ box.

Eigenspace after dual termination. In this last sampling, primal starting points are produced in the eigenspace of a near optimal dual point. The dual point is generated by optimizing the dual problem until a convergence criterion is fulfilled, $\|\mu^k - \mu^*\| < \epsilon$, using the bundle method (Kiwiel, 1994b). The sample is defined as

$$S_{\text{ED}} = \{(x^i, \mu) \mid i = 1, \ldots, p,\ x^i = \rho^i y^i,\ \mu = \mu^* - \lambda_1(A + \mu^* I)e\},$$

where $\rho^i = n^{1/2}/\|y^i\|$, $y^i = \sum_{k=1}^{r} \alpha_k v_k(A + \mu^* I)$, and μ^* a near optimal dual solution. The primal points are sampled from the space generated by the eigenvectors corresponding to the smallest eigenvectors of $A + \mu^* I$.

11.2.4 Numerical results

Algorithm 11.2 was coded in `C++`. Supergradients for the dual function were computed according to Lemma 5.14. The Lanczos method ARPACK++ (Gomes and Sorensen, 1997) was used for the computation of the minimum eigenvalue and a corresponding eigenvector.

Kiwiel's proximal bundle algorithm NOA 3.0 (Kiwiel, 1990; Kiwiel, 1994b) was used for solving the dual problem.

The algorithm was tested using a set of examples from the 7th DIMACS Implementation Challenge (Pataki and Schmieta, 2000), and using several instances created with RUDY, a machine independent graph generator written by G. Rinaldi, which is standard for MaxCut (Helmberg and Rendl, 2000).

The tests were run on a machine that has two 700MHz Pentium III processors and 1Gb RAM. The sample size for all the sample sets was set to 10, and the best result over each sample type was reported.

Table 11.1 shows the results for the different sampling techniques. The computing time and the value in percentage referred to the most elaborated sample S_{ED}, eigenspace after dual termination, is reported. For the reported runs, a fixed number of major iterations l, a fixed step-length β, and a uniform sequence t^j is used.

Previous evidence with other Lagrangian heuristics for the unit commitment problem suggests that higher dual objective accuracy need not necessarily imply better quality of the heuristic primal solution Feltenmark and Kiwiel, 2000. To evaluate the importance of the information provided by the dual for the heuristic,

example	size		time			sol. quality			dual
name	n	m	S_{RP}	S_{E}	S_{ED}	S_{RP}	S_{E}	S_{ED}	bound
g3	800	19176	15	17	32	99	99	11608	12084
g6	800	19176	14	16	1:13	99	100	2135	2656
g13	800	1600	4	6	12:46	100	100	568	647
g14	800	4694	5	6	2:30	99	99	3024	3192
g19	800	4661	5	6	1:11	98	98	868	1082
g23	2000	19990	24	29	2:50	99	99	13234	14146
g31	2000	19990	22	28	11:40	100	100	3170	4117
g33	2000	4000	15	20	3:27:25	99	100	1342	1544
g38	2000	11779	17	21	9:02	99	99	7512	8015
g39	2000	11778	18	20	5:13	98	98	2258	2877
g44	1000	9990	10	13	1:04	99	99	6601	7028
g50	3000	6000	25	36	55	98	99	5830	5988
g52	1000	5916	7	8	3:14	100	100	3779	4009

Table 11.1: **Comparison of computing time and solution quality.** Samples: S_{RP} random primal, S_{E} eigenspace, S_{ED} eigenspace after dual stop. The columns report the computing time in hh:mm:ss, hh hours, mm minutes, and ss seconds. mm reported when total seconds were more than 60, and similarly with hh. The first two columns show the best case in percentage of the best value from the last sampling technique S_{ED}, in the last column whose result is reported in absolute value. The last column provides information about the dual bound. The run was performed with fixed step-length $\beta = 5$, minor iterations $m = 10$, major iteration horizon $l = 20$ and uniform update of t, i.e. t values: $1/(l+1), \ldots, l/(l+1)$.

we plot comparatively the dual sequence and its corresponding heuristic primal solution sequence for some graph examples in Figure 11.2.

It was observed that meanwhile the dual improves its value, reducing the duality gap, the heuristic primal sequence is not monotonically decreasing. This means that dual points closer to the optimum do not necessarily provide better heuristic primal points.

Table 11.2 shows a comparison with rank-2 and GRASP algorithm (Festa et al., 2002). The numbers of rank-2 were generated using a sample of size 1, and without the use of the random perturbation, since the information regarding the random perturbation parameters was not available. In the paper (Burer et al., 2001) better results are reported.

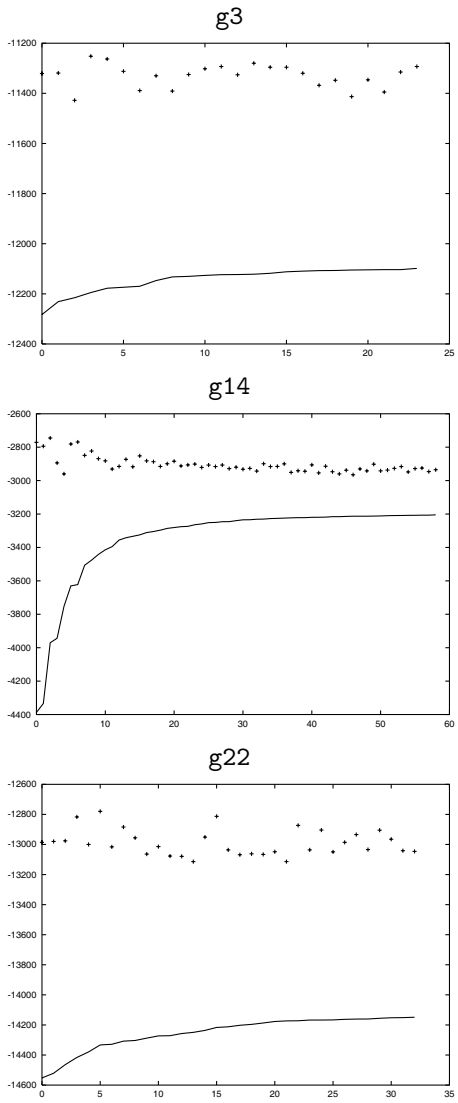

Figure 11.2: Plots of sequences of dual points and their correspondent primal heuristic solution produced by Algorithm 11.2 from `rudy` graphs g3, g14, and g22.

example	size		S_{RP}		GRASP-VNS		rank2	
name	n	m	ss	result	$ss.dd$	result	$ss.dd$	result
g11	800	1600	4	550	10.00	552	3.88	554
g12	800	1600	3	542	9.36	532	3.76	552
g13	800	1600	4	570	12.41	564	3.45	572
g14	800	4694	4	3006	12.89	3040	5.53	3053
g15	800	4661	5	3002	18.09	3017	5.91	3039
g20	800	4672	4	920	–	N/A	5.56	939
g22	2000	19990	21	13193	56.98	13087	22.31	13331
g24	2000	19990	25	13165	192.81	13209	27.30	13287
g31	2000	19990	20	3193	–	N/A	19.61	3255
g32	2000	4000	14	1346	99.91	1368	13.09	1380
g34	2000	4000	14	1334	55.22	1340	9.82	1358

Table 11.2: **Comparison with other methods.** Comparison of time and result among (i) random primal sampling S_{RP} with sample size = 10 and Algorithm `pathfollowing` with steplength $\beta = 5$, fixed minor iterations $m = 10$, uniform update on t with major iteration horizon $\mathrm{l} = 20$, t values: $1, (l-1)/l, (l-2)/l, ..., 1/l, 0$; (ii) `GRASP-VNS` method with 1 iteration and (iii) results of the rank-2 heuristic with parameters N=10 and M=8 obtained on a SGI Origin2000 machine with a 300MHZ R12000 processor (Burer et al., 2001). The time is presented in $ss.dd$ seconds.fraction expressed in decimal format. The results were rounded to the closest integer for ease of reading.

11.3 Generalization to MINLP

11.3.1 Parametric problem formulation

Consider a general MINLP of the form:

$$\begin{array}{lll} \min & h_0(x) & \\ \text{s.t.} & h_i(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary}\,. \end{array} \tag{11.1}$$

In order to solve (11.1) by a deformation heuristic, it is reformulated as the following parametric box-constrained optimization problem

$$(\mathrm{P}_t) \qquad \begin{array}{ll} \min & H(x;t) \\ \text{s.t.} & x \in [\underline{x}, \overline{x}] \end{array}$$

where the smoothing function

$$H(x;t) = tP(x;t) + (1-t)\breve{L}(x;\hat{\mu})$$

is a convex combination of an exact penalty function and a convex Lagrangian. The penalty function is defined by

$$P(x;t) = h_0(x) + \frac{1}{1-t}\left(\sum_{i=1}^{m} \delta_i \max\{0, h_i(x)\}^2 - \gamma e^T r_B(x)\right)$$

where $r(x) = \mathrm{Diag}(x - \underline{x})(x - \overline{x})$, $\delta \in \mathbb{R}^m_+$ and $\gamma \in \mathbb{R}_+$. The convex Lagrangian, defined by $\breve{L}(x;\hat{\mu}) = \breve{h}_0(x) + \sum_{i=1}^m \hat{\mu}_i \breve{h}(x)$, is related to a convex underestimating-relaxation to (11.1) of the form:

$$\begin{array}{ll} \min & \breve{h}_0(x) \\ \text{s.t.} & \breve{h}_i(x) \le 0, \quad i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}] \end{array} \tag{11.2}$$

where $\breve{h}_i$ is a convex underestimator of h_i over $[\underline{x}, \overline{x}]$. Moreover, the dual point $\hat{\mu} \in \mathbb{R}^m_+$ is a near optimal dual point, i.e.

$$\mathrm{val}(11.2) \simeq \min_{x \in [\underline{x}, \overline{x}]} \breve{L}(x;\hat{\mu}).$$

Lemma 11.5. *The optimal value of (P_t) converges to the optimal value of* (11.1) *for $t \to 1$, provided that the penalty parameters $\delta \in \mathbb{R}^m_+$ and $\gamma \in \mathbb{R}_+$ are large enough.*

Proof. Since $P(x;t)$ is an exact penalty function, the optimal value of the box-constrained parametric program

$$\min\{P(x;t) \mid x \in [\underline{x}, \overline{x}]\}$$

converges to the optimal value of (11.1) for $t \to 1$ if δ and γ are large enough. Since $|H(x;t) - P(x;t)| \to 0$ for $t \to 1$, the assertion follows. □

11.3.2 A MINLP deformation algorithm

Algorithm 11.3 shows a deformation heuristic for computing a set S^* of solution candidates for problem (11.1). Instead of calling the deformation heuristic in serial for each starting point, the method starts in the beginning with a set of starting points. At certain branching values of t it modifies this set by adding new points through a neighborhood search and by deleting points which tend to cluster. The sample set is also pruned from points that are not very likely to converge to the global optimum.

The local optimization of (11.1) is performed by first rounding the binary variables of the starting point and then optimizing the NLP-subproblem with fixed binary variables.

Input: a sequence of continuation points $t^0 > \cdots > t^l = 0$ and branching points $B \subset \{1, \ldots, l\}$

Choose an initial sampling set $S^0 \subset [\underline{x}, \overline{x}]$.

for $j = 0, \ldots, l$

Determine the set S^{j+1} of local minimizers of (P_{t^j}) starting from points $x \in S^j$.

if $k \in B$:

Prune S^{j+1} by removing nonpromising and clustering points.

Enlarge S^{j+1} by adding new points by using neighborhood search.

end if

end for

Determine a set S^* of local minimizers of (11.1) starting from points $x \in S^{l+1}$.

Algorithm 11.3: MINLP deformation heuristic

11.3.3 Numerical results

The performance of Algorithm 11.3 was tested for a set of instances from the MINLPLib (Bussieck et al., 2003a) described in Appendix B.1. Convex relaxations were computed as in Section 7.5. Three experiments were made. In the first experiment pure multistart was used for optimization with the following parameters of Algorithm 11.3: $l = 0$ and $B = \emptyset$. In the second experiment, Algorithm 11.3 was used with the parameters $l = 5$ and $B = \emptyset$. In the third experiment, the parameters of Algorithm 11.3 were set to $l = 5$ and $B = \{3\}$. In all experiments the initial sample set S^0 was defined by 40 uniformly distributed sample points.

The code was run on a machine with a 1GHz Pentium III processor and 256 MB RAM. Table 11.4 shows the result. The columns of this table are described in Table 11.3. The last line of the table shows the number of solved problems. N/A means that no feasible solution was computed. The results show:

- The solution quality of the deformation heuristic is better than of the multistart method.
- The computation time of the deformation heuristics compared to the multistart heuristic is not much larger.
- The inclusion of a branching point, $B = \{3\}$, further improves the results.

It would be interesting to analyze the performance of the proposed deformation method with respect to the Lagrangian smoothing operator $H(x;t) = tP(x;t) + (1-t)\breve{L}(x;\hat{\mu})$. If the positive curvature of $\breve{L}(x;\hat{\mu})$ is strong enough to cancel the negative curvature of $P(x;t)$, the function $H(x;t)$ is almost convex if t is small. In this case, the parametric optimization problem (P_t) has few minimizers and there is a high probability to find a good minimizer with a deformation heuristic that uses neighborhood search.

example	The name of the problem
n	The number of variables
$\|B\|$	The number of binary variables
m	The number of constraints
err.	The relative error of the solution value computed as $\frac{\bar{v}-v^*}{1+\|\bar{v}\|}$, where v^* is the best known optimal value.
time	Time in seconds spent by Algorithm 11.3

Table 11.3: Descriptions of the columns of Table 11.4.

				multistart		no branch point		one branch point	
example	n	$\|B\|$	m	err.	time	err.	time	err.	time
alan	9	4	8	0	0.44	0	0.63	0	0.122
elf	55	24	39	1.31	0.39	1.31	6.76	1.31	12.09
ex1223a	8	4	10	.12	0.41	.12	0.90	0	0.156
ex4	37	25	31	.16	0.93	.16	5.91	.16	9.55
feedtray2	88	36	284	N/A	0.56	0	23.44	0	42.40
fuel	16	3	16	0	0.51	0	1.79	0	3.11
gbd	5	3	5	0	0.59	0	0.45	0	0.70
meanvarx	36	14	45	.19	0.48	.19	3.103	.04	6.143
nous1	51	2	44	.03	0.79	.03	4.89	0	8.48
nous2	51	2	44	0	0.75	0	5.13	0	8.64
sep1	30	2	32	0	0.47	0	3.33	0	6.14
spectra2	70	30	73	1.28	1.37	1.28	25.51	1.01	34.125
batch	47	24	74	.24	0.43	.24	7.94	.14	14.45
batchdes	20	9	20	0	0.38	0	1.69	0	3.03
ex1221	6	3	6	0	0.34	0	0.75	0	0.134
ex1222	4	1	4	0	0.39	0	0.50	0	0.65
ex1223b	8	4	10	.12	0.47	.12	0.87	0	0.161
ex1224	12	8	8	.01	0.41	.01	0.85	.01	1.49
ex1225	9	6	11	.18	0.31	.18	0.68	.18	0.141
ex1226	6	3	6	0	0.38	0	0.82	0	0.125
ex1252	40	15	44	.19	0.73	.19	0.75	.13	7.69
ex3	33	8	32	0	0.54	0	4.40	0	7.82
gkocis	12	3	9	0	0.42	0	0.112	0	1.110
oaer	10	3	8	0	0.39	0	1.44	0	1.111
procsel	11	3	8	0	0.48	0	0.84	0	1.70
synheat	57	12	65	.18	0.63	.18	7.99	.11	14.54
synthes1	7	3	7	0	0.32	0	0.76	0	0.130
synthes2	12	5	15	.01	0.38	.01	0.106	0	1.116
synthes3	18	8	24	.07	0.42	.07	1.89	.07	2.116
29				14		16		20	

Table 11.4: Performance of the MINLP deformation heuristic

Chapter 12

Rounding, Partitioning and Lagrangian Heuristics

This chapter presents two heuristic methods for solving MINLPs. The first method is a rounding heuristic based on rounding fractional solutions of convex relaxations and computing solution candidates of continuous subproblems with fixed binary variables via a partitioning heuristic by using so-called central splitting-cuts (Nowak et al., 2003).

The second method is a Lagrangian heuristic that combines inner approximation points generated by a column generation algorithm in such a way that the violation of the linear coupling constraints is as small as possible. The resulting points are used as starting points for a local optimization. Numerical results for MINLPs are presented.

12.1 A rounding heuristic

Consider a general MINLP given in the form:

$$\min\{f(x) \mid x \in S,\, x_B \text{ binary}\} \tag{12.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $S \subset \mathbb{R}^n$ is bounded. Formulation (12.1) includes both the block-separable binary program (2.1) and the reformulation (2.4) with linear coupling constraints. A relaxation to (12.1) is defined by:

$$\min\{f(x) \mid x \in \breve{S}\} \tag{12.2}$$

where $\breve{S} \supseteq S$ is a convex outer approximation of S. A subproblem to (12.1) with partially fixed binary variables is defined by

(P[y,K$_{\text{fix}}$]) $$\min\{f(x) \mid x \in S[y, K_{\text{fix}}],\, x_B \text{ binary}\},$$

where $S[y, K_{\text{fix}}] = \{x \in S \mid x_i = y_i,\, i \in K_{\text{fix}}\}$, $y \in [\underline{x}, \overline{x}]$ and $K_{\text{fix}} \subseteq B$. Similarly, a subproblem to (12.2) is defined by

(R[y,K$_{\text{fix}}$]) $$\min\{f(x) \mid x \in \breve{S}[y, K_{\text{fix}}]\}.$$

where $S[y, K_{\text{fix}}] = \{x \in \breve{S} \mid x_i = y_i,\, i \in K_{\text{fix}}\}$. Furthermore, we define a *measure of binary infeasibility* by:

$$d_{\text{bin}}(K_{\text{fix}}, x) = \max_{i \in B \setminus K_{\text{fix}}} \max\{\overline{x}_i - x_i, x_i - \underline{x}_i\}$$

and

$$\gamma(K_{\text{fix}}, x) = \operatorname*{argmax}_{i \in B \setminus K_{\text{fix}}} \max\{\overline{x}_i - x_i, x_i - \underline{x}_i\}.$$

Algorithm 12.1 shows a rounding heuristic for computing solution candidates for the MINLP (12.1). The heuristic works by subsequently computing trial points $\hat{x} \in \breve{S}[K_{\text{fix}}, x]$ and rounding some binary components of $\hat{x}$. A trial point can be computed by solving the convex relaxation (R[y,K$_{\text{fix}}$]) or by computing a center of $\breve{S}[K_{\text{fix}}, x]$, as described in the next section. If all binary components are fixed, i.e. K$_{\text{fix}} = B$, a heuristic is started to solve the continuous subproblem (P[y,B]). Here, we use a partitioning heuristic that is presented in the next section. The values of the binary variables are recursively switched. The whole process is repeated as long as either all combinations of binary variables are searched, or the number of solution candidates exceeds a given number.

Compute a trial point $\hat{x} \in \breve{S}$ and set $y = \text{round}(\hat{x}, B)$.

Compute solution candidates for (P[y,B]) and update (R[y,B]).

Set $K_{\text{fix}} = \emptyset$ and L=$\{(K_{\text{fix}}, \hat{x})\}$.

repeat

Take (K_{fix}, x) from L with $d_{\text{bin}}(K_{\text{fix}}, x)$ maximum.

Set $K'_{\text{fix}} = K_{\text{fix}} \cup \{j\}$ with $j = \gamma(K_{\text{fix}}, x)$ and round x_j.

if $\breve{S}[K'_{\text{fix}}, x] \neq \emptyset$: Compute a trial point $\hat{x} \in \breve{S}[K'_{\text{fix}}, x]$ and put $(K'_{\text{fix}}, \hat{x})$ into L.

Set $x_j = \underline{x}_j + \overline{x}_j - x_j$.

if $\breve{S}[K'_{\text{fix}}, x] \neq \emptyset$:

Compute a trial point $\hat{x} \in \breve{S}[K'_{\text{fix}}, x]$, $y = \text{round}(\hat{x}, K'_{\text{fix}})$.

Compute solution candidates for (P[y,B]), update (R[y,B]) and put $(K'_{\text{fix}}, \hat{x})$ into L.

end if

until iteration limit is exceeded or $L = \emptyset$.

Algorithm 12.1: Rounding heuristic for solving a MINLP by subsequently rounding binary variables of solutions of convex relaxations

12.2 A partitioning heuristic that uses central cuts

In this section a partitioning heuristic for solving continuous NLPs including the subproblem (P[y,B]) is proposed. Consider a nonconvex NLP problem:

$$\min\{f(x) \mid x \in S\} \tag{12.3}$$

and a related polyhedral relaxation defined by:

$$\min\{f(x) \mid x \in \hat{S}\} \tag{12.4}$$

where $\hat{S}$ is a polyhedral outer approximation of S given in the form:

$$\hat{S} = \{x \in \mathbb{R}^n \mid h(x) = 0, \quad g_i(x) \geq 0, \quad i = 1, \ldots, m\}$$

and h and g_i are affine functions. The *analytic center* of $\hat{S}$, called *convexification center*, is defined by

$$x^c = \min\{-\sum_{i=1}^{m} \ln g_i(x) \mid h(x) = 0\}.$$

Assuming that the polyhedron $\hat{S}$ is a (good) approximation of the convex hull $\text{conv}(X_\epsilon^*)$ of an ϵ-solution set of (12.3), a central point x^c in $\hat{S}$ is also a central point in $\text{conv}(X_\epsilon^*)$. If the number of ϵ-minimizers is greater than 1, there exists a hyperplane through x^c separating one or several ϵ-minimizers. This motivates the definition of the *central splitting-cut*:

$$g_{\text{split}}(x) = (x^c - \hat{x})^T((1-t)x^c + t\hat{x} - x) \leq 0, \tag{12.5}$$

where $t \in (0,1)$, which splits off a solution candidate $\hat{x}$. Algorithm 12.2 describes a heuristic for globally solving (12.3) based on subsequently generating solution candidates $\hat{x}$ and splitting off $\hat{x}$ by adding a central splitting-cut (12.5) to (12.4). If the optimal value was improved, the polyhedral relaxation $\hat{S}$ is improved by adding the level-cut

$$f(x) \leq f(\hat{x}). \tag{12.6}$$

The procedure is repeated as long as no new local optimizer was found.

for $j = 0, \ldots, l$

Set x^c=center($\hat{S}$) and compute a local minimizer $\hat{x}$ of (12.3) starting from x^c.

if $\hat{x}$ is not new: **stop**.

if $\hat{x}$ is not feasible: Compute a local minimizer $\hat{x}$ of (12.3) starting from a local minimizer $\tilde{x}$ of the relaxation (12.4).

if $\hat{x}$ is not feasible: Set $\hat{x} = \tilde{x}$.

if the optimal value was improved: Add the level cut (12.6).

Add the cut (12.5) to (12.4).

end for

Algorithm 12.2: Partitioning heuristic for solving a NLP by subsequently splitting off solution candidates

Remark 12.1. The convexification center x^c can also be used to construct two further central cuts that define branching rules for branch-and-bound algorithms.

The first cut, called *central binary cut*, is defined by splitting a domain according to the most violated binary constraint defined by

$$j = \operatorname*{argmin}_{i \in B} |x_i^c - 0.5(\underline{x}_i + \overline{x}_i)|.$$

The new subproblems are defined by the constraints $x_j = \underline{x}_j$ and $x_j = \overline{x}_j$ respectively.

The second cut, called *central diameter cut*, subdivides the region at the hyperplane which goes through x^c and is parallel to the face of $\hat{S}$ that has the largest distance to x^c, i.e.

$$g_{\text{diam}}(x) = a_j^T x + b_j,$$

where $|a_j^T x^c + b_j| = \max_{i=1,\ldots,m} |a_i^T x^c + b_i|$, $\|a_i\| = 1$ and $g_i(x) = a_i^T x + b_i$. Figure 12.1 illustrates the three central cuts. The central binary cut splits the polyhedral set into s_1 and s_2, the central splitting-cut subdivides it at g_2, and the central diameter cut subdivides it at g_1.

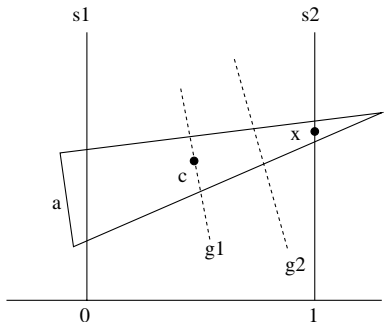

Figure 12.1: Central binary, central splitting and central diameter cut

12.3 Numerical results

Algorithm 12.1 together with Algorithm 12.2 for solving the continuous subproblem were coded as part of the C++ library LaGO. In order to test the performance of the algorithm, numerical experiments with linear relaxations and four different cuts described in Section 7.1 were made by using a set of instances from the MINLPLib Bussieck et al., 2003a described in Appendix B.1. The maximum iteration numbers of Algorithms 12.1 and 12.2 was set to 1000 and 5 respectively. Convex relaxations were computed as in Section 7.5.

In the first experiment, linearization and level cuts were used (see Table 12.2). In the second experiment, the bounding box was reduced and level cuts were used (see Table 12.3). In the third experiment, the bounding box was reduced and level and linearization cuts were used (see Table 12.4). Finally, in the fourth experiment, the bounding box was reduced and level, linearization and Knapsack cuts were used (see Table 12.5).

The columns of these tables are described in Table 12.1. The last line of the tables shows the number of solved problems. N/A means that no feasible solution was computed. The code was run on a machine with a 1GHz Pentium III processor and 256 MB RAM. The results show:

- The results are quite similar.
- The best results in the sense of solved problems were obtained in the last experiment shown in Table 12.5.
- The influence of central splitting cuts is marginal, but they helped to solve nous2 in Table 12.5.

example	The name of the problem
n	The number of variables
$\|B\|$	The number of binary variables
m	The number of constraints
rel err	The relative error of the solution value computed as $\frac{\bar{v}-v^*}{1+\|\bar{v}\|}$, where v^* is the best known optimal value.
iter/$2^{\|B\|}$	The percentage number of iterations, computed by 100 times the number of iterations and divided by $2^{\|B\|}$.
last impr	The percentage number of iterations, till the upper bound was improved the last time.
cuts	The number of linearization cuts which were added.

Table 12.1: Descriptions of the columns of Tables 12.2, 12.3, 12.4 and 12.5.

example	n	$\|B\|$	m	rel err	heu time	iter/ $2^{\|B\|}$	last impr	cuts
alan	9	4	8	0	0.09	62%	60%	5
elf	55	24	39	0	58.13	0.006%	39%	555
ex1223a	8	4	10	0	0.04	25%	75%	32
ex4	37	25	31	0	15.67	0.0005%	59%	774
fac3	67	12	34	0	0.47	1%	32%	15
fuel	16	3	16	0	0.04	62%	60%	33
gbd	5	3	5	0	0.02	25%	100%	3
meanvarx	36	14	45	0	0.11	0.04%	57%	5
nous2	51	2	44	N/A	17.06	100%		0
sep1	30	2	32	0	0.09	100%	100%	37
spectra2	70	30	73	0	22.80	3e-05%	1%	264
batch	47	24	74	0	18.52	0.0003%	95%	55
batchdes	20	9	20	0	0.07	1%	80%	21
ex1221	6	3	6	0	0.07	100%	37%	30
ex1222	4	1	4	0	0.00	100%	50%	4
ex1223b	8	4	10	0	0.17	56%	88%	95
ex1224	12	8	8	0	1.54	45%	4%	130
ex1225	9	6	11	0	0.14	17%	54%	0
ex1226	6	3	6	0	0.11	62%	100%	3
ex1252	40	15	44	.80	10:39.57	3%	0%	18
ex3	33	8	32	0	0.35	6%	87%	69
gkocis	12	3	9	0	0.06	75%	66%	14
procsel	11	3	8	0	0.10	62%	80%	20
synheat	57	12	65	0	1:14.19	18%	59%	720
synthes1	7	3	7	0	0.05	50%	25%	11
synthes2	12	5	15	0	0.13	28%	77%	28
synthes3	18	8	24	0	0.18	5%	84%	56
27				25				

Table 12.2: Rounding heuristic: linearization cuts and no box reduction

example	n	$\|B\|$	m	rel err	heu time	iter/ $2^{\|B\|}$	last impr	cuts
alan	9	4	8	0	0.08	75%	66%	4
elf	55	24	39	0	1:24.88	0.005%	43%	680
ex1223a	8	4	10	0	0.04	25%	75%	34
ex4	37	25	31	0	9.09	0.3%	40%	903
fac3	67	12	34	0	0.42	1%	32%	15
fuel	16	3	16	0	0.03	75%	33%	15
gbd	5	3	5	0	0.03	50%	100%	3
meanvarx	36	14	45	0	0.17	0.2%	71%	5
nous2	51	2	44	N/A	57.05	100%		19
sep1	30	2	32	0	0.13	100%	75%	42
spectra2	70	30	73	0	22.59	2e-05%	14%	456
batch	47	24	74	0	0.23	0.001%	20%	33
batchdes	20	9	20	0	0.03	1%	50%	12
ex1221	6	3	6	0	0.04	75%	16%	10
ex1222	4	1	4	0	0.02	100%	100%	4
ex1223b	8	4	10	0	0.03	37%	16%	27
ex1224	12	8	8	0	1.59	45%	4%	130
ex1225	9	6	11	0	0.19	17%	54%	0
ex1226	6	3	6	0	0.11	62%	100%	3
ex1252	40	15	44	.80	5:43.60	3%	0%	18
ex3	33	8	32	0	0.28	5%	92%	60
gkocis	12	3	9	0	0.03	75%	66%	14
procsel	11	3	8	0	0.08	62%	80%	20
synheat	57	12	65	0	1:07.73	17%	60%	720
synthes1	7	3	7	0	0.06	50%	25%	11
synthes2	12	5	15	0	0.08	28%	77%	28
synthes3	18	8	24	0	0.22	5%	80%	56
27				25				

Table 12.3: Rounding heuristic: linearization cuts and box reduction

example	n	$\|B\|$	m	rel err	heu time	iter/ $2^{\|B\|}$	last impr	cuts
alan	9	4	8	0	0.15	100%	50%	1
elf	55	24	39	1.96	35.81	0.006%	67%	15
ex1223a	8	4	10	0	0.07	31%	60%	7
ex4	37	25	31	.15	1:04.21	6%	30%	0
fac3	67	12	34	0	1.74	4%	14%	0
fuel	16	3	16	0	0.03	75%	33%	6
gbd	5	3	5	0	0.02	75%	66%	1
meanvarx	36	14	45	0	0.40	0.9%	60%	1
nous2	51	2	44	N/A	58.32	100%		19
sep1	30	2	32	0	0.09	100%	75%	6
spectra2	70	30	73	0	28.127	9e-05%	1%	24
batch	47	24	74	0	1.10	0.005%	3%	11
batchdes	20	9	20	0	0.08	6%	12%	5
ex1221	6	3	6	0	0.03	75%	16%	2
ex1222	4	1	4	0	0.02	100%	100%	2
ex1223b	8	4	10	0	0.03	43%	14%	10
ex1224	12	8	8	0	1.67	66%	2%	4
ex1225	9	6	11	0	0.23	17%	54%	0
ex1226	6	3	6	0	0.10	62%	100%	0
ex1252	40	15	44	0	11:15.49	3%	94%	0
ex3	33	8	32	0	1.15	30%	18%	7
gkocis	12	3	9	0	0.10	75%	66%	2
procsel	11	3	8	0	0.06	62%	80%	0
synheat	57	12	65	0	19.85	14%	58%	0
synthes1	7	3	7	0	0.08	87%	14%	2
synthes2	12	5	15	0	0.23	65%	38%	5
synthes3	18	8	24	0	0.70	23%	98%	7
27				24				

Table 12.4: Rounding heuristic: level cuts and box reduction

example	n	$\|B\|$	m	rel err	heu time	iter/ $2^{\|B\|}$	last impr	cuts
alan	9	4	8	0	0.07	75%	66%	4
elf	55	24	39	0	1:24.26	0.005%	43%	680
ex1223a	8	4	10	0	0.05	25%	75%	34
ex4	37	25	31	0	8.91	0.3%	40%	903
fac3	67	12	34	0	0.45	1%	32%	15
fuel	16	3	16	0	0.04	75%	33%	15
gbd	5	3	5	0	0.02	50%	100%	3
meanvarx	36	14	45	0	0.16	0.2%	71%	5
nous2	51	2	44	0	42.32	100%	50%	181
sep1	30	2	32	0	0.10	100%	75%	48
spectra2	70	30	73	0	22.51	2e-05%	14%	456
batch	47	24	74	0	0.27	0.001%	20%	33
batchdes	20	9	20	0	0.02	1%	50%	12
ex1221	6	3	6	0	0.05	75%	16%	12
ex1222	4	1	4	0	0.00	100%	100%	5
ex1223b	8	4	10	0	0.04	37%	16%	27
ex1224	12	8	8	0	1.63	45%	4%	130
ex1225	9	6	11	0	0.21	21%	35%	1
ex1226	6	3	6	0	0.11	62%	100%	4
ex1252	40	15	44	.80	5:46.66	3%	0%	18
ex3	33	8	32	0	0.26	5%	92%	60
gkocis	12	3	9	0	0.07	75%	66%	14
procsel	11	3	8	0	0.07	62%	80%	20
synheat	57	12	65	0	1:09.32	17%	60%	720
synthes1	7	3	7	0	0.06	50%	25%	11
synthes2	12	5	15	0	0.08	28%	77%	28
synthes3	18	8	24	0	0.20	5%	80%	56
27				26				

Table 12.5: Rounding heuristic: all cuts

12.4 A Lagrangian heuristic

This section describes a simple Lagrangian heuristic, shown in Algorithm 12.3, for solving a MINLP of the form:

$$\begin{array}{ll} \min & c^T x + c_0 \\ & Ax + b \leq 0 \\ & x_{J_k} \in G_k, \quad k = 1, \ldots, p. \end{array} \tag{12.7}$$

Algorithm 12.3 is a three-step method that generates solution candidates by combining inner approximation points computed by a column generation algorithm (see Section 4.3). In the first step, a random near-optimal solution $\hat{x}$ of the following problem is computed:

$$\begin{array}{lll} \min & c^T x + c_0 + \delta \|Ax + b\|_{1,+} & \\ \text{s.t.} & x_{J_k} \in \operatorname{conv}(W_k), & k = 1, \ldots, p, \end{array} \tag{12.8}$$

where $\delta > 0$ is a penalty parameter and $W_k \subset \operatorname{conv}(G_k)$ are inner approximation points. In the second step, the point $\hat{x}$ is projected onto the polyhedron $\{x \in \mathbb{R}^n \mid Ax + b \leq 0\}$ by solving the problem:

$$\begin{array}{ll} \min & c^T x + c_0 + \delta \|x - \hat{x}\|_1 \\ \text{s.t.} & Ax + b \leq 0 \end{array} \tag{12.9}$$

which is equivalent to the LP:

$$\begin{array}{ll} \min & c^T x + c_0 + \delta e^T t \\ \text{s.t.} & Ax + b \leq 0 \\ & -t \leq x - \hat{x} \leq t \end{array} \tag{12.10}$$

where $\delta > 0$ is a penalty parameter. Finally, a solution x of (12.9) is rounded and a local search for (12.7) is started from x. These steps are repeated as long as the maximum iteration number is exceeded.

We explain now the first step of the proposed Lagrangian heuristic in more detail. Note that problem (12.8) is equivalent to

$$\begin{array}{lll} \min & c^T AW \bullet z + c_0 + \delta \|AW \bullet z + b\|_{1,+} & \\ \text{s.t.} & e^T z_{I_k} = 1,\ z_{I_k} \geq 0, & k = 1, \ldots, p, \end{array} \tag{12.11}$$

which can be also written as the following LP:

$$\begin{array}{ll} \min & c^T AW \bullet z + c_0 + \delta e^T t \\ \text{s.t.} & AW \bullet z + b \leq t \\ & e^T z_{I_k} = 1,\ z_{I_k} \geq 0, \qquad k = 1, \ldots, p \\ & t \geq 0 \end{array} \tag{12.12}$$

where $W = (W_1, \ldots, W_p)$. Let x^* be a solution of (12.8) and define the points $x^{k,w}$ by $x^{k,w}_{J_l} = x^*_{J_l}$ for $l \neq k$ and $x^{k,w}_{J_k} = w$ else, $w \in W_k$, $k = 1, \ldots, p$. The value of the objective of (12.8) at $x^{k,w}$ is denoted by $v_{k,w} = c^T x^{k,w} + c_0 + \delta \|Ax^{k,w} + b\|_{1,+}$. A random near optimal point $\hat{x}$ of (12.8) is computed by setting $\hat{x}_{J_k} = w$, where $w \in W_k$ is randomly chosen according to a probability that is related to

$$p_{k,w} = ((v_{k,w} - \underline{v}_k)/(\overline{v}_k - \underline{v}_k) + 0.1)^{-1} \tag{12.13}$$

with $\underline{v}_k = \min_{w \in W_k} v_{k,w}$ and $\overline{v}_k = \max_{w \in W_k} v_{k,w}$.

Numerical results for solving MaxCut and MINLP problems with a branch-cut-and-price algorithm that uses Agorithm 12.3 for computing upper bounds are presented in Section 13.5.

Compute a solution x^* of (12.8).

for $j = 1, \ldots, s_{\max}$:

  for $k = 1, \ldots, p$:

    Choose a random point $w \in W_k$ according to a probability that is related to $p_{k,w}$, defined in (12.13), and set $\hat{x}_{J_k} = w$.

  Compute a solution x of (12.9).

  Round x_B and start a local search from x for solving the continuous subproblem of (12.7) with fixed binary variables.

Algorithm 12.3: Lagrangian heuristic that solves a MINLP by combining inner approximation points

Chapter 13

Branch-Cut-and-Price Algorithms

This chapter proposes branch-and-bound algorithms for MINLP. In contrast to the previously presented heuristics, these methods are able to search systematically for a global solution and to prove global optimality. In particular, a branch-cut-and-price (BCP) algorithm for nonconvex MINLPs is presented. To the best of our knowledge, this is the first time that BCP is used for solving general MINLPs. The convergence of the algorithms is analyzed and some algorithmic details are discussed. Moreover, preliminary numerical results for network MaxCut and MINLP instances are reported. Finally, the use of nonconvex polyhedral approximations is discussed.

13.1 Branch-and-bound algorithms

13.1.1 Preliminaries

Consider a MINLP of the form:

$$\min\{f(x) \mid x \in S\} \tag{13.1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $S \subset \mathbb{R}^n$ is bounded. This formulation includes the block-separable binary program (2.1) or the extended reformulation (2.4). Let $L = \{U_1, \ldots, U_l\}$ be a list of *partition elements* covering the feasible S of (13.1), i.e. $\bigcup_{U \in L} U \supseteq S$. Related to a partition element $U \in L$, a *node-subproblem*

$$\text{(P[U])} \qquad \min\{f(x) \mid x \in S \cap U\}$$

and a *node-relaxation*

$$\text{(R[U])} \qquad \min\{\breve{f}_U(x) \mid x \in \breve{S}_U \cap U\}$$

is defined, where $\breve{f}_U(x) \leq f_U(x)$ for all $x \in S \cap U$ and $\breve{S}_U \supseteq S_U$. A lower bound on the optimal value of the node-subproblem (P[U]) is defined by $\underline{v}(U) = \mathrm{val}(R[U])$, i.e. $\underline{v}(U) \leq \mathrm{val}(P[U])$. A *root-problem* and *root-relaxation* is defined by (P$[[\underline{x}, \overline{x}]]$) and (R$[[\underline{x}, \overline{x}]]$) respectively. Furthermore, we denote by $\overline{v}$ an upper bound of the optimal value of (13.1), and by X_{cand} a set of solution candidates for the root-problem (13.1), respectively.

13.1.2 A generic branch-and-bound algorithm

Algorithm 13.1 shows a generic branch-and-bound method for solving (13.1). The method starts with computing a root relaxation (R$[[\underline{x}, \overline{x}]]$) and initializing $\overline{v}$ and X_{cand} by using a heuristic. In the main loop, the original problem is split recursively into subproblems. The loop starts with the selection of a subproblem. Here, the subproblem U with the smallest lower bound $\underline{v}(U)$ is selected. For the selected subproblem U, the lower bound $\underline{v}(U)$ is improved, for example by adding cuts or by applying a box reduction procedure. If $U \cap X_{\mathrm{cand}} = \emptyset$, solution candidates are searched in $U \cap S$ by using a heuristic. If the lower bound $\underline{v}(U)$ was not improved significantly, a *branching* operation subdivides the subproblem U into subproblems U_i, $i = 1, \ldots, l$ (see Figure 13.1). For each new subproblem U_i, a lower bound $\underline{v}(U_i)$ is computed. If the lower bound of a subproblem is greater than or equal to an upper bound $\overline{v}$ of the optimal value, the subproblem is eliminated. The difference between an upper and a global lower bound of the optimal value serves as a quality criterion for the current best solution. If it is smaller than a given tolerance $\epsilon > 0$, the algorithm stops.

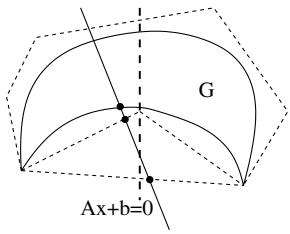

Figure 13.1: Partition and refinement of a polyhedral outer approximation

13.2 Convergence and finiteness

13.2.1 Convergence

The following assumption is required for the convergence of Algorithm 13.1.

Compute a root relaxation (R$[[\underline{x}, \overline{x}]]$) and set $L = \{[\underline{x}, \overline{x}]\}$.

Apply a heuristic to initialize $\overline{v}$ and X_{cand}.

repeat

Take a partition element U from L.

Improve the lower bound $\underline{v}(U)$.

if $U \cap X_{\text{cand}} = \emptyset$:
Search solution candidates in U and update X_{cand} and $\overline{v}$.

if $\underline{v}(U)$ was not improved significantly:
Subdivide U into U_i, $i = 1, \ldots, l$.

Compute $\underline{v}(U_i)$ and put U_i into L for $i \in \{1, \ldots, l\}$.

Prune L by deleting $U \in L$ with $\underline{v}(U) \geq \overline{v}$.

until $L = \emptyset$ or $\overline{v} - \min_{U \in L} \underline{v}(U) < \epsilon$.

Algorithm 13.1: Branch-and-bound algorithm

Assumption 13.1. (i) *An exhaustive partitioning method: for every nested sequence of partitions, the feasible region reduces to a point, i.e.* $U^j \supset U^{j+1}$ *for all* $j \in \mathbb{N}$ *and* $U^\infty = \bigcap_{j=1}^{\infty} U^j = \{x\}$.

(ii) *Consistent bounding operation: every infinite nested sequence* $\{U^j\}_{j \in \mathbb{N}}$ *of successively refined partition sets, i.e.* $U^{j+1} \subset U^j$, *satisfies*

$$\lim_{j \to \infty} \underline{v}(U^j) = \min_{x \in U^\infty} f(x), \tag{13.2}$$

if $U^\infty \subset S$ *and* $\underline{v}(U^j) \to \infty$ *if* $U^\infty \cap S = \emptyset$.

(iii) *Bound improving node selection: after every finite number of steps a node with the least lower bound is selected.*

The following result is proven in (Horst and Tuy, 1990).

Proposition 13.1. *If Assumption* 13.1 *is fulfilled, Algorithm* 13.1 *terminates after finitely many steps for all* $\epsilon > 0$.

13.2.2 Finiteness

Algorithm 13.1 is called finite if it converges after finitely many steps for $\epsilon = 0$. If the MINLP (13.1) is convex and the feasible set S is compact, Algorithm 13.1 with

NLP-bounds is finite and defines an implicit enumeration of all possible solutions. In order to prove finiteness of Algorithm 13.1 for general nonconvex MINLPs, the *region of attraction* is defined,

$$\operatorname{attr}(x^*) = \{x \in \mathbb{R}^n \mid \operatorname{locmin}(x) = x^*\},$$

where x^* is a local minimizer, and $\operatorname{locmin}(x)$ is the solution point obtained by a local search method starting from x.

Assumption 13.2. *The solution set of* (13.1) *is finite. The heuristic used in Algorithm* 13.1 *is based on a local search procedure with starting points* $x \in [\underline{x}, \overline{x}] \supset S$. *For all* $x^* \in \operatorname{sol}(13.1)$ *the set* $attr(x^*)$ *has a nonempty interior.*

Assumption 13.2 is satisfied, for example, if the global minimizers of (13.1) fulfill a certain constraint qualification (see (Spellucci, 1993, Satz 3.6.5.)). The following result is proven in (Nowak, 2000).

Proposition 13.2. *If Assumptions* 13.1 *and* 13.2 *are fulfilled, Algorithm* 13.1 *finds the solution set of* (13.1) *in a finite number of iterations.*

Proof. Assume that Algorithm 13.1 does not compute a global solution in finite time. Then there exists a nested subsequence of partition elements $\{U^j\}$ generated by the algorithm such that $\underline{v}(U^j)$ is the global lower bound of the related partition, i.e. $\underline{v}(U^j) = \min_{U \in L} \underline{v}(U)$, implying $\underline{v}(U^j) \leq \operatorname{val}(13.1)$. Since the partition method is exhaustive, it holds that $\bigcap_{j=1}^{\infty} U^j = \{\hat{x}\}$. We show that the sequence $\{U^j\}$ is finite, which proves the assertion. If $\hat{x}$ is a global minimizer of (13.1), there exists $j \in \mathbb{N}$ such that $U^j \subset \operatorname{attr}(\hat{x})$ due to Assumption 13.2, implying that the heuristic computes $\hat{x}$ after a finite number of iterations. If $\hat{x}$ is not a global minimizer, then either $\hat{x} \notin S$, implying $\underline{v}(U^j) \to \infty$, or $\hat{x} \in S$ and $f(\hat{x}) > \operatorname{val}(13.1)$. Hence, $\underline{v}(U^j) \to f(\hat{x})$ since $\underline{v}(U^j)$ is consistent. In both cases, it follows $\underline{v}(U^j) > \operatorname{val}(13.1)$ if j is sufficiently large. This contradicts $\underline{v}(U^j) \leq \operatorname{val}(13.1)$. □

Proposition 13.2 does not show that Algorithm 13.1 is finite. Finiteness can be ensured by using *optimality cuts* as introduced in Section 8.3 and Section 8.5. In (Nowak, 2000) it is shown:

Corollary 13.3. *If Assumptions* 13.1 *and* 13.2 *are fulfilled, and an optimality cut is added whenever the heuristic finds a local solution, Algorithm* 13.1 *terminates in finitely many iterations.*

Proof. Define the sequence $\{U^j\}$ as in the proof of Proposition 13.2. Then any $\{U^j\}$ converges to a global minimizer $\hat{x}$. In this case, the algorithm makes an optimality cut with respect to $\hat{x}$. Hence, $\{U^j\}$ is finite due to the consistency of the bounding method. If $\{U^j\}$ does not converge to a global minimizer, it is shown in Proposition 13.2 that $\underline{v}(U^j) > \operatorname{val}(13.1)$ if j is sufficiently large. This proves the finiteness of Algorithm 13.1. □

Corollary 13.3 can only be applied to optimization problems for which optimality cuts can be constructed with respect to all global minimizers. Finiteness of branch-and-bound algorithms is also discussed in (Tawarmalani and Sahinidis, 2002).

13.3 Consistent bounding operations

In the following, three lower bounding methods are discussed. In particular, NLP-bounds (see Section 7.3.1), LP-bounds (see Section 7.3.4) and dual bounds (see Section 3.3 and Chapter 4) are analyzed. It is shown that all bounding methods are consistent and ensure convergence of Algorithm 13.1 according to Proposition 13.1.

13.3.1 NLP-bounds

Consider a subproblem of the form:

$$\text{(P[U])} \qquad \begin{array}{lll} \min & h_0(x) & \\ \text{s.t.} & h_i(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}] \cap U, & x_B \text{ binary}. \end{array}$$

A convex nonlinear relaxation to (P[U]) is given by:

$$\text{(R[U])} \qquad \begin{array}{lll} \min & \breve{h}_{0,U}(x) & \\ \text{s.t.} & \breve{h}_{i,U}(x) \leq 0, & i = 1, \ldots, m \\ & x \in [\underline{x}, \overline{x}] \cap U & \end{array}$$

where $\breve{h}_{i,U}$ is a convex underestimator of h_i over $[\underline{x}, \overline{x}] \cap U$. A *NLP-bound* to (P[U]) is defined by

$$\underline{v}_{\text{NLP}}(U) = \begin{cases} \text{val}(R[U]) & \text{if (R[U]) is feasible} \\ \infty & \text{else.} \end{cases} \tag{13.3}$$

A convex underestimator $\breve{h}_{i,U}$ is called *consistent* if for any nested sequence $\{U^j\}$ of partition elements converging to a point $\hat{x}$ it holds that

$$\lim_{j \to \infty} \max_{x \in U^j} |h_i(x) - \breve{h}_{i,U^j}(x)| = 0. \tag{13.4}$$

Remark 13.4. It is well known that interval underestimators (Section 6.1), α-underestimators (Section 6.3) and Bézier-underestimators (Section 6.2) are consistent.

Lemma 13.5. *If the convex underestimators $\breve{h}_{i,U}$, $i = 0, \ldots, m$, are consistent, then $\underline{v}_{\text{NLP}}(U)$ is a consistent lower bounding method.*

Proof. Let $\{U^j\}$ be a nested sequence of partition elements converging to a point $\hat{x} \in [\underline{x}, \overline{x}]$. Assume $\hat{x} \in S$, where S is the feasible set of (P[$[\underline{x}, \overline{x}]$]). Let $\breve{S}_U$ be the feasible set of (R[U]). Since $U \supseteq \breve{S}_U \supseteq S \cap U$, it follows that $\{\breve{S}_{U^j}\}$ converges to $\hat{x}$. Hence, $\underline{v}_{\mathrm{NLP}}(U^j)$ converges to $h_0(\hat{x})$.

Assume now $\hat{x} \notin S$. Then there exists an index $i \in \{1, \ldots, m\}$ and a number $j_1 \in \mathbb{N}$ such that $h_i(x^j) \geq \epsilon > 0$ for all $j \geq j_1$. Since $\breve{h}_{i,U}$ is consistent, there exists a number $j_2 \geq j_1$ such that $|h_i(x^j) - \breve{h}_{i,U}(x^j)| \leq \frac{1}{2}\epsilon$ for all $j \geq j_2$. This proves $\underline{v}_{\mathrm{NLP}}(U^j) = \infty$ for all $j \geq j_2$. □

13.3.2 LP-bounds

Consider a subproblem of the form:

$$\text{(P[U])} \qquad \begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & x \in Y \cap U \end{array}$$

where $Y = \{x \in [\underline{x}, \overline{x}] \mid x_B \text{ binary}\}$ and $g_i : \mathbb{R}^n \mapsto \mathbb{R}$. Let S be the feasible set of (P[U]). Related to (P[U]) the following linear relaxation is defined:

$$\text{(R[U])} \qquad \begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & \overline{g}_{i,U}(x) \leq 0, \quad i = 1, \ldots, m \\ & x \in \hat{G}_U \end{array}$$

where $\hat{G}_U \supseteq S \cap U$ is a polyhedron and $\overline{g}_{i,U}$ is an *affine underestimator* of g_i over $Y \cap U$, i.e. $\overline{g}_{i,U}$ is affine and $\overline{g}_{i,U}(x) \leq g_i(x)$ for all $x \in Y \cap U$. An *LP-bound* to (P[U]) is defined by

$$\underline{v}_{LP}(U) = \begin{cases} \mathrm{val}([R[U]) & \text{if (R[U]) is feasible} \\ \infty & \text{else.} \end{cases}$$

From Lemma 13.5 it follows:

Corollary 13.6. *If $\overline{g}_{i,U}$ is consistent for $i = 1, \ldots, m$, i.e. (13.4) holds, then $\underline{v}_{\mathrm{LP}}(U)$ is consistent.*

The following lemma shows how consistent affine underestimators can be constructed.

Lemma 13.7. *Consider a Knapsack cut $\overline{g}_U(x) \leq 0$ defined by $\overline{g}_U(x) = a_U^T x - \overline{a}_U$ and*

$$\overline{a}_{i,U} = \max\{a_{i,U}^T x \mid x \in Y \cap U,\, g_i(x) \leq 0\}. \tag{13.5}$$

If g is twice-differentiable over U and $a_U = \nabla g_U(\hat{x}_U)$ for some $\hat{x}_U \in U$, then $\overline{g}_U$ is a consistent affine underestimator of g.

Proof. There exists a point $\tilde{x}_U \in U$ such that $g(\tilde{x}_U) = \overline{g}_U(\tilde{x}_U)$. From the Taylor expansions of $g(x)$ at $\hat{x}_U$ and $\tilde{x}_U$ we get $g(x) = g(\hat{x}_U) + a_U^T(x - \hat{x}_U) + O(\mathrm{diam}(U))$

and $g(x) = g(\tilde{x}_U) + O(\mathrm{diam}(U))$ for all $x \in U$. Hence, $g(x) = g(\tilde{x}_U) + a_U^T(x - \tilde{x}_U) + O(\mathrm{diam}(U))$ for all $x \in U$. Since $\overline{g}_U(x) = g(\tilde{x}_U) + a_U^T(x - \tilde{x}_U)$, it follows that $g(x) - \overline{g}_U(x) = O(\mathrm{diam}(U))$ for all $x \in U$. This proves the statement. □

13.3.3 Dual bounds

Consider a subproblem of the form:

$$\text{(P[U])} \qquad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & g(x) \leq 0, \\ & x \in G \cap U \end{array}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $G, U \subset \mathbb{R}^n$. Define a Lagrangian $L(x; \mu) = f(x) + \mu^T g(x)$ and a dual function $D_U(\mu) = \inf_{x \in G \cap U} L(x; \mu)$. A dual bound for (P[U]) is given by

$$\underline{v}_{\mathrm{dual}}(U) = \sup_{\mu \in \mathbb{R}^m_+} D_U(\mu).$$

Clearly, $\underline{v}_{\mathrm{dual}}(U) \leq \mathrm{val}(P[U])$ because of weak duality. In Chapter 4 several methods for computing $\underline{v}_{\mathrm{dual}}(U)$ are discussed. In (Dür, 2001) it is proven:

Lemma 13.8. *Let S be the feasible set of* (13.1)*. Assume that S is nonempty and compact. Let $f : S \mapsto \mathbb{R}$ be l.s.c. and let $\{U^j\}_{j \in \mathbb{N}}$ be a nested sequence of nonempty, compact sets converging to $\emptyset \neq U^\infty \subset S$. Then*

$$\lim_{j \to \infty} (\mathrm{val}(P[U^j]) - \underline{v}_{\mathrm{dual}}(U^j)) = 0.$$

Hence, the dual bound $\underline{v}_{\mathrm{dual}}$ is consistent. It is also shown in (Dür, 2001), that dual bounds can be used to detect infeasible subproblems, which should be deleted in a branch-and-bound process.

Lemma 13.9. *Let $Y_U = \{y \in \mathbb{R}^m \mid y_i = g_i(x)$ for some $i \in \{1, \ldots, m\}$, $x \in U \cap S\}$. Then infeasibility of (P[U]) is equivalent to $Y_U \cap (\mathbb{R}^m_-) = \emptyset$. Assume that f and g are continuous and $S \cap U$ is compact. Then*

$$\underline{v}_{\mathrm{dual}}(U) = +\infty \quad \Leftrightarrow \quad \mathrm{conv}(Y_U) \cap (\mathbb{R}^m_-) = \emptyset.$$

Remark 13.10. The consistency result of Lemma 13.8 can only be applied if the Lagrangian does not depend on the partition set U. The Lagrangian that is used in the semidefinite relaxation defined in Section 5.1 depends on the box $[\underline{x}, \overline{x}]$. Thus, if the subdivision method changes $[\underline{x}, \overline{x}]$, Lemma 13.8 cannot be applied. In (Nowak, 2000) it is shown that nevertheless consistency can be proven in this case.

13.4 Branching

13.4.1 Rectangular subdivision rules

A rectangular subdivision procedure subdivides a rectangle $U = [\underline{x}, \overline{x}]$ into two intervals U_1 and U_2. In particular, let i^* be the index of the branching variable and $\underline{t}, \overline{t}$ be lower and upper cut values, where $\underline{x}_{i^*} \leq \underline{t} \leq \overline{t} \leq \overline{x}_{i^*}$. Then

$$U_1 = \{x \in U \mid x_{i^*} \in [\underline{x}_{i^*}, \underline{t}]\} \quad \text{and} \quad U_2 = \{x \in U \mid x_{i^*} \in [\overline{t}, \overline{x}_{i^*}]\}.$$

Let $\rho(U) = \min\{\overline{x}_{i^*} - \overline{t}, \underline{t} - \underline{x}_{i^*}\}/(\overline{x}_{i^*} - \underline{x}_{i^*})$. Note that a rectangular subdivision method is exhaustive if $\rho(U) \geq \underline{\rho} > 0$ for all partition elements U, and after every finite number of steps each variable is selected as a branching variable. In the following, some rectangular branching rules for MINLP are described, which can be used in Algorithm 13.1. Other branching strategies are proposed in (Horst and Tuy, 1990; Tawarmalani and Sahinidis, 2002).

Bisection. The bisection rule subdivides a rectangle at the midpoint of the largest edge, i.e. the index of the branching variable is defined by $i^* = \operatorname{argmax}_{1 \leq i \leq n} \{\overline{x}_i - \underline{x}_i\}$ and the lower und upper cut value is $\underline{t} = \overline{t} = \frac{1}{2}(\underline{x}_{i^*} + \overline{x}_{i^*})$. It is well known that this branching strategy is exhaustive.

Binary subdivision. This branching procedure is based on measuring the maximum binary constraint violation defined by

$$\delta_i(\hat{x}) = \min\{\overline{x}_i - \hat{x}_i, \hat{x}_i - \underline{x}_i\},$$

where $\hat{x}$ is the solution point of a convex nonlinear or polyhedral relaxation. The index of the branching variable is defined by $i^* = \operatorname{argmax}\{\delta_i(\hat{x}) \mid i \in B\}$ and the lower und upper cut values are $\underline{t} = \underline{x}_{i^*}$ and $\overline{t} = \overline{x}_{i^*}$.

Subdivision based on constraint violation. Here, the branching variable is selected according to the most violated constraint. Let $\hat{x}$ be the solution point of a convex nonlinear relaxation (7.3) or of a polyhedral outer relaxation (7.12) or of a polyhedral inner relaxation (4.16). Related to the constraints $g_j(x) \leq 0$, $j = 1, \ldots, m$, of the given MINLP, we define the sets $N_j = \{i \in \{1, \ldots, n\} \mid \partial_i g_j(\hat{x}) \neq 0\}$ and $M_j = \{i \in N_j \mid \delta_i(\hat{x}) > 0.2\}$, where $\delta_i(\hat{x})$ is defined as above. Let $j^* = \operatorname{argmax}_{j=1,\ldots,m} g_j(\hat{x})/\|\nabla g_j(\hat{x})\|$. If $M_{j^*} \neq \emptyset$ then $i^* = \operatorname{argmin}_{i \in M_{j^*}} |\partial_i g_j(\hat{x})|$. Otherwise, $i^* = \operatorname{argmax}_{i \in N_{j^*}} \delta_i(\hat{x})$. The lower and upper cut values are defined by $\underline{t} = \max\{\hat{x}_{i^*}, \underline{x}_{i^*} + \underline{\rho}(\overline{x}_{i^*} - \underline{x}_{i^*})\}$ and $\overline{t} = \min\{\hat{x}_{i^*}, \overline{x}_{i^*} - \underline{\rho}(\overline{x}_{i^*} - \underline{x}_{i^*})\}$ respectively.

Subdivision based on pseudo costs. In this strategy, the effect of branching is measured by using *pseudo costs*. Let $\hat{x}$ be the solution point of a relaxation (7.3), (7.12) or (4.16), and let $\tilde{x}$ be an estimate for the projection of $\hat{x}$ onto the feasible

set S of the given MINLP (2.4). Let $L(x;\mu) = a(\mu)^T x + b(\mu)$ be the Lagrangian to (2.4). Then

$$p(\hat{x}, \tilde{x}, \mu) = L(\tilde{x};\mu) - L(\hat{x};\mu) = a(\mu)^T(\tilde{x} - \hat{x})$$

is called *pseudo cost*. A branching variable is selected according to $i^* = \text{argmax}\ \{a_i(\mu)(\tilde{x}_i - \hat{x}_i) \mid i \in \{1,\dots,m\}\}$. The lower and upper cut values are defined as in the previous subdivision strategy.

Two procedures are used for computing an estimate $\tilde{x}$ for the projection of $\hat{x}$ onto S. In the first procedure, a local minimizer of the k-th Lagrange problem is computed by first rounding $\hat{x}_{J_k}$ and then starting a local search from $\hat{x}_{J_k}$ for solving the continuous subproblem with fixed binary variables.

The second procedure is based on a simple Newton estimate. Let $g_i(x) \le 0$, $i = 1,\dots,m$, be the constraints of the given MINLP, and $I_{\text{viol}} \subseteq \{1,\dots,m\}$ be the constraints that are violated at $\hat{x}$. The point $x^i = \hat{x} + t_i \nabla g_i(\hat{x})$ with $t_i = -g_i(\hat{x})/\|\nabla g_i(\hat{x})\|^2$ is a Newton estimate for a solution of the equation $g_i(x) = 0$. Then $\tilde{x} = \frac{1}{|I_{\text{viol}}|}\sum_{i\in I_{\text{viol}}} x^i$.

13.4.2 Updating lower bounds

After subdividing a partition element, the related lower bounds have to be updated. We denote by U a new partition element and by $\underline{x}(U)$ and $\overline{x}(U)$ its lower and upper variable bound, respectively.

NLP-bounds. NLP-bounds (see Section 7.3.1) that use α-underestimators are updated by computing an underestimator according to $\breve{f}(x) = f(x) + \alpha^T r(x)$ with $r(x) = \text{Diag}(\overline{x}(U) - \underline{x}(U))(\overline{x}(U) - \underline{x}(U))$.

LP-bounds. LP-bounds (see Section 7.3.4) are updated by adding cuts to a polyhedral relaxation. Lagrangian cuts are updated by generating new inner approximation points $W_k \subset \mathbb{R}^{|J_k|}$, $k = 1,\dots,p$, related to a new partition element U. To this end, the inner approximation points of the mother node are projected onto $[\underline{x}_{J_k}(U), \overline{x}_{J_k}(U)]$, and then the column generation method, Algorithm 4.7, is called.

13.5 Numerical results

The BCP Algorithm 13.1 with several bounding and branching strategies was implemented as part of the C++ library LaGO (see Chapter 14). In order to test the performance of the algorithm, numerical experiments with MIQQPs resulting from MaxCut splitting-schemes and with general MINLPs were carried out.

In all experiments the relative gap tolerance was set to 0.01. The same parameters were chosen for solving the Lagrangian subproblems. The maximum number of linearization cuts was set to 5000, since it is currently not possible to prune

cuts. Lagrangian cuts were produced by using the column generation method, Algorithm 4.8, with a maximum number of five iterations. Furthermore, the LP-estimate $\tilde{R}_k(\mu)$ for the reduced cost described in Remark 4.17 was used.

13.5.1 Network MaxCut experiments

The BCP method was tested, first, by using a block-separable splitting-scheme of a random network MaxCut example as described in Section 5.6.2:

$$\begin{array}{lll} \min & \sum_{k=1}^{p} t_k & \\ \text{s.t.} & x_i - x_j = 0, & (i,j) \in I_{\text{copy}} \\ & q_k(x_{J_k}) \leq t_k, & k = 1, \ldots, p \\ & x \in \{0,1\}^{n+p\cdot f} & \\ & t \in [\underline{t}, \overline{t}] & \end{array} \tag{13.6}$$

where q_k is a quadratic form. Problem (13.6) is generated with respect to the parameters (n, b, p, f), where n is the dimension of the original MaxCut problem, b is the block-size, p is the number of blocks and f is the flow size of the network, i.e. $f = |I_{\text{copy}}|/p$.

The results were obtained on a machine with a 3GHz Pentium III processor and 1G RAM. The maximum iteration limit of the BCP method was set to 1000. In order to find solution candidates of (13.6), Algorithm 13.1 with LP-bounds and the binary subdivision method described in Section 13.4.1 was used. A linearization cut was added whenever a new feasible point was found. Lagrangian subproblems were solved by using the branch-and-bound Algorithm 13.1 with NLP-bounds based on α-underestimators. Three experiments were made:

1. In the first experiment (see Table 13.2) only linearization cuts were used, and upper bounds were computed by rounding the solution of the RMP (4.16).
2. In the second experiment (see Table 13.3) linearization cuts and Lagrangian cuts were used, and upper bounds were computed with the Lagrangian heuristic described in Section 12.4.
3. In the third experiment (see Table 13.4) the same method as in the second experiment was used, but the addition of Lagrangian cuts was stopped if the lower bound was not improved significantly in the last five BCP iterations, or 80 % of the time limit passed.

Table 13.1 describes the columns of these tables.

The experiments show that most of the time is spent for computing lower bounds by solving Lagrangian subproblems. The use of the Lagrangian heuristic greatly improves the upper bounds (compare Table 13.2 with Table 13.3). The performance of the algorithm depends strongly on the computation of the Lagrangian cuts. Generating Lagrangian cuts only in the beginning results in more BCP iterations and reduces sometimes the BCP gap (compare Table 13.3 with Table 13.4).

Figures 13.2, 13.3, 13.4 and 13.5 show the process of the upper and lower bounds for solving two MaxCut examples with the three previously described BCP methods. The first method, used in Table 13.2, is denoted by 'no lagheu', the second method, used in Table 13.3, is denoted by 'lagheu', and the third method, using Table 13.4, is denoted by 'few lagcuts'. It can be seen that the use of the Lagrangian heuristic strongly improves the upper bound, and that for computing lower bounds no method is superior.

$n + n_c$	The dimension of the original problem plus the number of copy variables.
b	The block size.
p	The number of blocks.
f	The flow size
$\overline{v} - D$ gap	The relative error of the upper bound $\overline{v}$ with respect to the semidefinite programming bound val(D) defined in Section 5.1, computed as $100 \cdot \frac{\overline{v} - \mathrm{val}(D)}{1+\lvert\underline{v}\rvert}$
BCP gap	The final gap, computed as $100 \cdot \frac{\overline{v} - \underline{v}}{1+\lvert\underline{v}\rvert}$
BCP iter	The number of iterations of the BCP method
BCP time	Time in 'minutes : seconds' spent by the BCP method
$\underline{v}$ time	The relative time for computing lower bounds
$\overline{v}$ time	The relative time for computing upper bounds
lag sol	The number of solved Lagrangian subproblems

Table 13.1: Descriptions of the columns of Tables 13.2, 13.3 and 13.4

n	b	p	f	$\overline{v} - D$ gap	BCP gap	BCP iter	BCP time	$\underline{v}$ time	$\bar{v}$ time	lag sol
50+10	5	10	1	6%	$< 1\%$	1	0.93	100%	0%	53
50+10	10	5	2	10%	$< 1\%$	9	10.24	99.4%	0%	118
100+20	5	20	1	8%	$< 1\%$	1	1.41	98.5%	0%	71
100+20	10	10	2	12%	$< 1\%$	9	30.78	99.7%	0%	317
150+30	15	10	3	77%	73.8%	13	15:03.14	99.9%	0%	525
300+60	10	30	2	76%	72.8%	122	15:01.19	99.2%	0.4%	5918
300+60	15	20	3	92%	91.9%	9	15:04.07	99.9%	0%	604
500+100	10	50	2	85%	83.4%	74	15:02.80	98.9%	0.6%	4810
900+180	15	60	3	67%	64.5%	3	15:09.80	99.8%	0%	572
1000+200	10	100	2	50%	44.6%	25	15:02.94	96.2%	2.3%	3227

Table 13.2: Solving (13.6) with a BCP method using a rounding heuristic

n	b	p	f	$\overline{v}-D$ gap	BCP gap	BCP iter	BCP time	$\underline{v}$ time	$\bar{v}$ time	lag sol
50+10	5	10	1	6%	$< 1\%$	1	0.94	96.8%	2.1%	53
50+10	10	5	2	10%	$< 1\%$	9	10.00	98.5%	1.3%	118
100+20	5	20	1	8%	$< 1\%$	1	1.42	96.5%	3.4%	71
100+20	10	10	2	12%	$< 1\%$	10	32.34	98.5%	1.3%	326
150+30	15	10	3	18%	5.8%	13	15:02.14	99.8%	0.1%	496
300+60	10	30	2	17%	6.2%	119	15:00.21	96.3%	3%	5725
300+60	15	20	3	29%	20.5%	9	15:02.38	99.6%	0.2%	604
500+100	10	50	2	21%	10.9%	73	15:00.42	94.1%	5%	4613
900+180	15	60	3	27%	20.6%	3	15:09.66	97.4%	2.4%	573
1000+200	10	100	2	23%	13.8%	24	15:09.85	90.8%	5.6%	3165

Table 13.3: Solving (13.6) with a BCP method using a Lagrangian heuristic

n	b	p	f	$\overline{v}-D$ gap	BCP gap	BCP iter	BCP time	$\underline{v}$ time	$\bar{v}$ time	lag sol
50+10	5	10	1	6%	$< 1\%$	1	0.94	97.8%	2.1%	53
50+10	10	5	2	10%	$< 1\%$	9	10.16	98.4%	1.3%	118
100+20	5	20	1	8%	$< 1\%$	1	1.45	95.2%	4.1%	71
100+20	10	10	2	12%	$< 1\%$	10	31.92	98.4%	1.3%	326
150+30	15	10	3	18%	5.6%	505	15:00.00	97.1%	1.6%	617
300+60	10	30	2	16%	5%	775	15:00.42	69.8%	26.7%	2801
300+60	15	20	3	19%	9.5%	116	15:00.15	98%	1.5%	669
500+100	10	50	2	18%	8.1%	384	15:00.79	80.9%	14.6%	2393
900+180	15	60	3	26%	21.9%	36	15:00.13	94.7%	4.5%	409
1000+200	10	100	2	23%	13.9%	56	15:03.13	77.1%	15.9%	1965

Table 13.4: Solving (13.6) with a BCP method using a Lagrangian heuristic, where Lagrangian cuts are generated only in the beginning

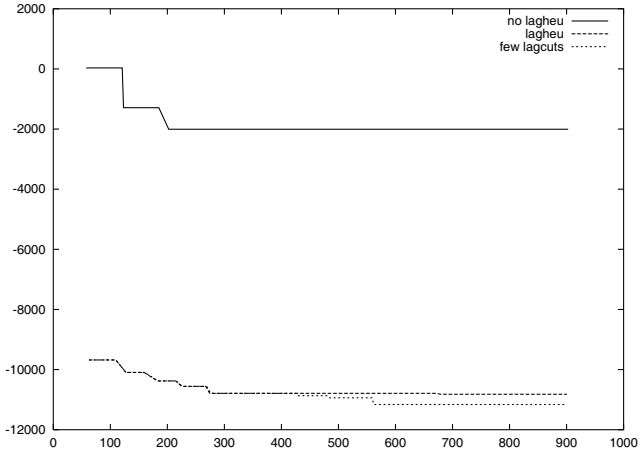

Figure 13.2: Process of upper bounds per seconds for solving a network MaxCut problem, defined by $b = 10$, $p = 50$ and $f = 2$, with three BCP methods

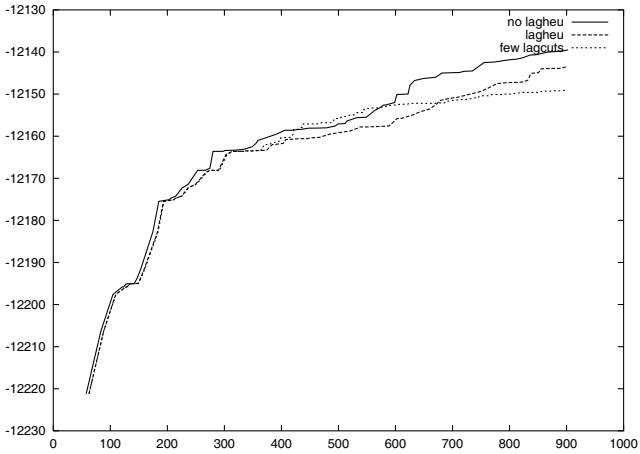

Figure 13.3: Process of lower bounds per seconds corresponding to Figure 13.2

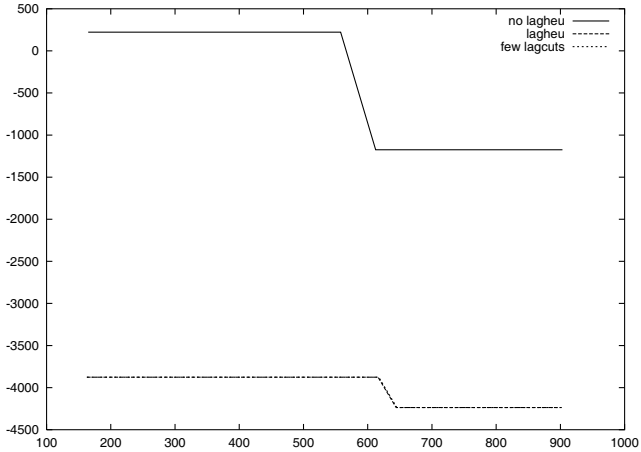

Figure 13.4: Process of upper bounds per seconds for solving a network MaxCut problem, defined by $b = 15$, $p = 10$ and $f = 3$, with three BCP methods

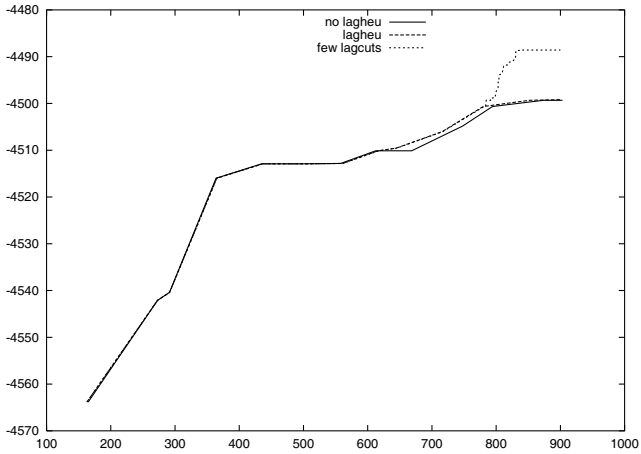

Figure 13.5: Process of lower bounds per seconds corresponding to Figure 13.4

13.5.2 MINLP experiments

The BCP method was also tested using a set of instances from the MINLPLib (Bussieck et al., 2003a) described in Appendix B.1. Here, the initial nonlinear convex relaxation that is computed by using convexified polynomial underestimators, as described in Section 7.5, was not updated. Hence, the resulting NLP-bounds are not consistent. Updating of nonlinear convex relaxations after branching operations is currently not implemented in LaGO, and will be added in the future. Since the used bounding method is not consistent, only results with the binary subdivision method are reported. The results were obtained on a machine with a 1.8 GHz AMD Athlon 64 2800+ processor and 1 GB RAM. A time limit of 20 minutes was set. The maximum iteration limit of the BCP method was set to 10000. CPLEX (ILOG, Inc., 2005) was used for solving LPs, and CONOPT (Drud, 1996) was used for solving NLPs. Four experiments were made:

1. In the first experiment (see Table 13.6) NLP-bounds based on α-underestimators and binary subdivision were used. Upper bounds were computed by rounding the solution of the convex relaxation and starting a local search.
2. In the second experiment (see Table 13.7) LP-bounds were used, and upper bounds were computed by rounding the solution of the outer approximation and starting a local search.
3. In the third experiment (see Table 13.8) the column generation Algorithm 4.8 was used for computing lower bounds. Upper bounds were computed by rounding the solution of the RMP (4.16) and starting a local search. Lagrangian subproblems were solved with a branch-and-bound algorithm that uses NLP-bounds based on α-underestimators and a binary subdivision.
4. In the last experiment (see Table 13.9) the same method as in the third experiment was used, but upper bounds were computed by the Lagrangian heuristic described in Section 12.4.

The columns of the tables are described in Table 13.5. The last line of the tables shows the number of solved problems. N/A means that no feasible solution was computed.

The results show that the BCP algorithm with LP-bounds and without Lagrangian cuts performs best in terms of solution quality and time (see Table 13.7). Comparing Table 13.7 and Table 13.6 shows that the BCP algorithm with NLP-bounds gives sometimes better bounds and solves different problems to global optimality (problem ex1263 and ex1266), but performs in general worse than the BCP algorithm with LP-bounds.

The experiments show also that Lagrangian cuts computed by the column generation Algorithm 4.8 do not improve the performance in most cases (see Table 13.8 and Table 13.9). An exception is problem feedtray, which is solved to global optimality using Lagrangian cuts. Comparing Table 13.8 with Table 13.9 shows that the Lagrangian heuristic improves the final gap in most cases. It can be seen

from these tables that most of the time is spent for computing Lagrangian cuts. In order to make these cuts more efficient, the Lagrangian sub-solvers have to be improved (see Appendix A).

By taking the best method for each problem, 35 of 40 problems can be solved to global optimality, and for the remaining instances, ex1264, nous2, enpro48, enpro56 and stockcycle, a local solution can be computed with a relative error ranging from 0.02 to 0.6. Note that most solutions are computed in only a few seconds.

example	The name of the problem
n	The number of variables
$\|B\|$	The number of binary variables
m	The number of constraints
rel. err.	The relative error of the solution value, computed as $\frac{\bar{v}-v^*}{1+\|\bar{v}\|}$, where v^* is the best known optimal value.
BCP gap	The final gap, computed as $100 \cdot \frac{\bar{v}-\underline{v}}{1+\|\underline{v}\|}$
BCP iter	The number of iterations of the BCP method
BCP time	Time in 'minutes : seconds' spent by the BCP method
$\underline{v}$ time	The relative time for computing lower bounds
lag sol	The number of solved Lagrangian subproblems

Table 13.5: Descriptions of the columns of Tables 13.6, 13.7, 13.8, and 13.9

example	n	$\|B\|$	m	c	rel. err.	gap	BCP iter	BCP time	$\underline{v}$ time	lag sol
alan	9	4	8	y	0	$< 1\%$	3	0.02	23.8%	0
elf	55	24	39	n	0	$< 1\%$	4170	2:07.17	84.8%	0
ex1223a	8	4	10	y	0	$< 1\%$	2	0.01	30%	0
ex1263	93	72	56	n	0	1.5%	7644	1:44.91	34.1%	0
ex1264	89	68	56	n	N/A		10000	2:11.32	30.2%	0
ex1266	181	138	96	n	0	$< 1\%$	1553	35.30	47%	0
ex4	37	25	31	n	0	37.9%	4433	3:47.83	85.3%	0
fac3	67	12	34	y	0	589.8%	17	0.88	82.4%	0
fuel	16	3	16	n	0	61.4%	5	0.03	49.9%	0
meanvarx	36	14	45	y	0	$< 1\%$	4	0.03	31%	0
nous2	51	2	44	n	.46	106.2%	7	1.17	96.8%	0
sep1	30	2	32	n	0	26.6%	6	0.04	35.8%	0
spectra2	70	30	73	n	1.03	223.9%	10000	2:35.88	57.5%	0
util	146	28	168	n	0	2.3%	5486	3:29.16	17.9%	0
batch	47	24	74	y	0	$< 1\%$	5	0.61	93.5%	0
batchdes	20	9	20	n	0	2.9%	8	0.22	79.7%	0
eniplac	142	24	190	n	0	34%	10000	6:38.98	64.5%	0
enpro48	154	92	215	n	.20	120.5%	6163	20:00.06	92.4%	0
enpro56	128	73	192	n	.19	87.4%	10000	9:35.90	74.5%	0
ex1233	53	12	65	n	0	39.7%	173	4.31	78.5%	0
ex1243	69	16	97	n	0	45.6%	84	4.56	88.2%	0
ex1244	96	23	130	n	0	20.3%	1742	1:34.66	86%	0
ex1252	40	15	44	n	N/A		9437	1:21.26	19.2%	0
ex3	33	8	32	n	0	8.1%	8	0.11	47.1%	0
fac1	23	6	19	y	N/A			0.00		
feedtray	98	7	92	n	N/A		13	0.84	68.3%	0
gastrans	107	21	150	n	0	$< 1\%$	22	0.67	29.2%	0
gear2	29	24	5	n	0	$< 1\%$	227	1.14	11.9%	0
gkocis	12	3	9	n	0	15.3%	6	0.08	73.7%	0
johnall	195	190	193	n	0	$< 1\%$	2	2.11	92.4%	0
minlphix	85	20	93	n	0	103.8%	44	0.95	63.8%	0
ortez	88	18	75	n	0	2.5%	50	6.42	96.8%	0
procsel	11	3	8	n	0	$< 1\%$	4	0.03	56.6%	0
ravem	113	54	187	n	.20	192%	3561	20:00.04	97.1%	0
stockcycle	481	432	98	y	1.27	2672%	2411	20:00.03	86.1%	0
synheat	57	12	65	n	0	41.6%	221	3.83	66.1%	0
synthes1	7	3	7	y	0	$< 1\%$	3	0.02	36.3%	0
synthes2	12	5	15	y	0	$< 1\%$	6	0.05	25.5%	0
synthes3	18	8	24	y	0	$< 1\%$	5	0.06	41.3%	0
waterx	71	14	55	n	.02	1924%	10000	6:01.67	15.2%	0
40					29					

Table 13.6: Solving MINLPs with a branch-and-bound algorithm using NLP-bounds and binary subdivision

example	n	$\|B\|$	m	c	rel. err.	gap	BCP iter	BCP time	$\underline{v}$ time	lag sol
alan	9	4	8	y	0	< 1%	9	0.05	2.2%	0
elf	55	24	39	n	0	< 1%	870	1:40.37	56.3%	0
ex1223a	8	4	10	y	0	< 1%	3	0.01	0%	0
ex1263	93	72	56	n	.13	13.1%	10000	1:38.17	9.6%	0
ex1264	89	68	56	n	.17	21.5%	10000	1:36.90	9.3%	0
ex1266	181	138	96	n	N/A		10000	2:15.87	12.6%	0
ex4	37	25	31	n	0	31.8%	4859	2:01.14	38.4%	0
fac3	67	12	34	y	0	3.198e+09%	16	0.25	9.8%	0
fuel	16	3	16	n	0	61.4%	5	0.03	21.7%	0
meanvarx	36	14	45	y	0	< 1%	3	0.02	22.2%	0
nous2	51	2	44	n	.46	106.2%	7	0.06	27.7%	0
sep1	30	2	32	n	0	26.6%	6	0.05	27.2%	0
spectra2	70	30	73	n	.07	71.5%	10000	12:53.89	24.8%	0
util	146	28	168	n	0	2.3%	4834	6:15.02	7.1%	0
batch	47	24	74	y	0	< 1%	7	0.10	21.6%	0
batchdes	20	9	20	n	0	4.1%	6	0.05	22.4%	0
eniplac	142	24	190	n	0	36.7%	6967	20:00.25	16.7%	0
enpro48	154	92	215	n	.23	125.8%	9858	20:00.23	20.7%	0
enpro56	128	73	192	n	.02	59.4%	9402	20:00.16	19.5%	0
ex1233	53	12	65	n	0	42.5%	228	5.27	46.6%	0
ex1243	69	16	97	n	0	47.7%	92	1.55	32.9%	0
ex1244	96	23	130	n	0	12%	1987	1:10.32	36%	0
ex1252	40	15	44	n	0	1.289e+07%	5460	1:08.49	13.1%	0
ex3	33	8	32	n	0	< 1%	6	0.05	27.4%	0
fac1	23	6	19	y	0	8.042e+06%	7	0.06	1.6%	0
feedtray	98	7	92	n	N/A		13	0.94	8.3%	0
gastrans	107	21	150	n	0	< 1%	13	0.33	13.9%	0
gear2	29	24	5	n	0	< 1%	52	0.38	7.3%	0
gkocis	12	3	9	n	0	15.7%	5	0.03	15.3%	0
johnall	195	190	193	n	0	< 1%	22	4.29	35.2%	0
minlphix	85	20	93	n	0	3.167e+04%	44	0.73	23.4%	0
ortez	88	18	75	n	0	2.3%	47	0.74	46.7%	0
procsel	11	3	8	n	0	8%	5	0.03	12%	0
ravem	113	54	187	n	0	27.5%	3782	20:11.68	52.6%	0
stockcycle	481	432	98	y	.60	115.7%	7376	20:00.12	4.8%	0
synheat	57	12	65	n	0	39.3%	185	4.32	45.6%	0
synthes1	7	3	7	y	0	< 1%	4	0.02	7.1%	0
synthes2	12	5	15	y	0	22.9%	9	0.04	17.5%	0
synthes3	18	8	24	y	0	< 1%	10	0.06	16.9%	0
waterx	71	14	55	n	.15	2163%	1034	20:02.52	85.8%	0
40					30					

Table 13.7: Solving MINLPs with a BCP algorithm using LP-bounds and binary subdivision

example	n	$\|B\|$	m	c	rel. err.	gap	BCP iter	BCP time	$\underline{v}$ time	lag sol
alan	9	4	8	y	0	< 1%	1	0.04	7.3%	6
elf	55	24	39	n	.05	25.8%	138	20:02.07	99.5%	38844
ex1223a	8	4	10	y	0	< 1%	2	0.04	63.4%	3
ex1263	93	72	56	n	.17	19.7%	452	20:09.37	99.1%	41193
ex1264	89	68	56	n	.57	62.4%	175	20:04.23	99.7%	53806
ex1266	181	138	96	n	N/A		62	20:11.70	99.8%	31573
ex4	37	25	31	n	.24	48.6%	138	20:28.07	99.8%	4429
fac3	67	12	34	y	.08	656.4%	10	5:40.94	99.8%	2018
fuel	16	3	16	n	0	61.4%	5	0.33	17.1%	16
meanvarx	36	14	45	y	0	< 1%	6	0.93	93%	181
nous2	51	2	44	n	.46	107.3%	7	8.69	54.8%	79
sep1	30	2	32	n	0	26.6%	6	1.85	94.1%	185
spectra2	70	30	73	n	0	58.9%	10000	13:24.85	30.7%	5084
util	146	28	168	n	0	2.3%	2032	20:00.22	49.7%	27745
batch	47	24	74	y	0	< 1%	11	44.86	99.4%	2538
batchdes	20	9	20	n	0	2.6%	6	1.45	33.8%	40
eniplac	142	24	190	n	0	34.7%	1235	20:01.25	71.9%	52263
enpro48	154	92	215	n	.82	233.8%	1618	20:00.73	89.9%	22649
enpro56	128	73	192	n	N/A					
ex1233	53	12	65	n	0	36.6%	170	20:18.84	99.8%	64014
ex1243	69	16	97	n	0	50.5%	98	11:13.30	99.7%	43168
ex1244	96	23	130	n	0	41.5%	233	20:04.21	99.1%	49176
ex1252	40	15	44	n	.02	1.316e+07%	5376	20:00.03	92.1%	31432
ex3	33	8	32	n	0	< 1%	6	6.76	99.1%	993
fac1	23	6	19	y	N/A			0.25		
feedtray	98	7	92	n	0	81%	13	9:28.46	93.7%	1799
gastrans	107	21	150	n	0	< 1%	9	13.81	88.7%	1103
gear2	29	24	5	n	0	< 1%	24	1.51	85.9%	245
gkocis	12	3	9	n	0	26.9%	7	2.45	97.8%	467
johnall	195	190	193	n	.10	10.4%	1	21:00.95	89.4%	497
minlphix	85	20	93	n	N/A		7	20:02.11	99.8%	362
ortez	88	18	75	n	0	2%	58	2:41.90	99.1%	6369
procsel	11	3	8	n	0	< 1%	4	0.17	87.1%	42
ravem	113	54	187	n	0	33.3%	122	20:12.33	99.6%	68115
stockcycle	481	432	98	y	.91	522.4%	835	20:02.66	71.3%	67894
synheat	57	12	65	n	0	38.7%	247	20:10.22	99.5%	57109
synthes1	7	3	7	y	0	< 1%	3	0.10	22.6%	6
synthes2	12	5	15	y	0	29.3%	11	0.48	87.1%	54
synthes3	18	8	24	y	0	24.3%	14	7.16	98%	753
waterx	71	14	55	n	N/A		2	1:57.30	81.2%	66
40					25					

Table 13.8: Solving MINLPs with a BCP algorithm using LP-bounds, binary subdivision, and Lagrangian cuts that are computed with a branch-and-bound algorithm

example	n	$\|B\|$	m	c	rel. err.	gap	BCP iter	BCP time	$\underline{v}$ time	lag sol
alan	9	4	8	y	0	< 1%	1	0.05	5.8%	6
elf	55	24	39	n	.01	7.9%	538	20:02.62	94.3%	44226
ex1223a	8	4	10	y	0	< 1%	2	0.05	67.4%	3
ex1263	93	72	56	n	.22	24.6%	170	20:05.67	69.6%	15517
ex1264	89	68	56	n	.17	21.5%	198	20:02.11	85.7%	30163
ex1266	181	138	96	n	N/A		6	20:17.93	0.1%	119
ex4	37	25	31	n	.17	43.9%	208	20:05.06	99.2%	4439
fac3	67	12	34	y	.08	< 1%	3	22.10	97.8%	192
fuel	16	3	16	n	0	61.4%	5	0.36	16.6%	16
meanvarx	36	14	45	y	0	< 1%	3	0.18	47.4%	38
nous2	51	2	44	n	.46	107.3%	7	8.80	53.6%	79
sep1	30	2	32	n	0	26.6%	6	1.03	42.9%	87
spectra2	70	30	73	n	.21	92.7%	4371	20:00.33	74.8%	4099
util	146	28	168	n	0	2.3%	662	20:01.57	10%	8360
batch	47	24	74	y	0	< 1%	7	12.48	97.2%	1234
batchdes	20	9	20	n	0	4.1%	6	1.68	34.8%	56
eniplac	142	24	190	n	0	36%	503	20:04.45	19.5%	18241
enpro48	154	92	215	n	.69	209.2%	205	20:06.19	12.9%	3449
enpro56	128	73	192	n	.34	110%	291	20:10.76	16.2%	11283
ex1233	53	12	65	n	0	38.4%	191	1:59.76	87.7%	1543
ex1243	69	16	97	n	0	48.5%	94	23.26	80.1%	1386
ex1244	96	23	130	n	0	22.2%	1412	20:00.11	72.9%	35001
ex1252	40	15	44	n	.02	1.316e+07%	4407	20:00.08	88.1%	28688
ex3	33	8	32	n	0	7%	12	6.67	87.2%	823
fac1	23	6	19	y	N/A			0.26		
feedtray	98	7	92	n	0	81%	8	20:15.64	95%	2263
gastrans	107	21	150	n	0	< 1%	1	1.74	10.3%	58
gear2	29	24	5	n	0	< 1%	27	1.29	64.6%	247
gkocis	12	3	9	n	0	26.9%	7	2.48	97.2%	467
johnall	195	190	193	n	0	< 1%	1	2:53.03	2.9%	104
minlphix	85	20	93	n	0	3.167e+04%	1	20:08.59	99.8%	155
ortez	88	18	75	n	0	2.5%	50	30.93	94.7%	828
procsel	11	3	8	n	0	< 1%	4	0.18	79.1%	42
ravem	113	54	187	n	0	20.9%	104	20:04.70	94.1%	58606
stockcycle	481	432	98	y	.86	355.6%	584	20:02.30	19.8%	30617
synheat	57	12	65	n	0	37.3%	209	3:05.59	51.4%	4849
synthes1	7	3	7	y	0	< 1%	3	0.11	21.9%	6
synthes2	12	5	15	y	0	6.3%	10	0.31	51.7%	32
synthes3	18	8	24	y	0	< 1%	9	3.59	93.4%	397
waterx	71	14	55	n	N/A		2	29.80	18.8%	114
40					26					

Table 13.9: Solving MINLPs with a BCP algorithm using LP-bounds, binary subdivision, a Lagrangian heuristic, and Lagrangian cuts that are computed with a branch-and-bound algorithm

13.5.3 Cost-efficient design of energy conversion systems

In a joint research project of mathematicians and engineers funded by the German Science Foundation (Ahadi-Oskui et al., 2001), a new approach has been developed to optimize nonconvex MINLPs resulting from a superstructure of a complex energy conversion system. The goal of the optimization is to design an energy conversion system for a paper factory with minimum total levelized costs per time unit.

Parameter and structural changes are considered for the optimization simultaneously. A complex superstructure of a combined cycle power plant was developed as the basis for the optimization.

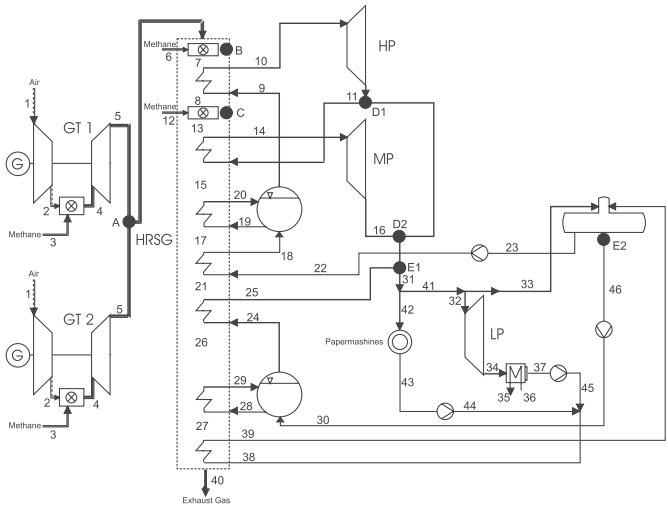

Figure 13.6: Part of the superstructure of the cogeneration plant

Figure 13.6 shows a part of the superstructure consisting of a gas turbine as topping cycle and a subsequent heat recovery steam generator (HRSG) that supplies a steam turbine as bottoming cycle. The process steam is extracted before it enters the low-pressure steam turbine. The required demand of 90 MW electric power and 99.5 t/h process steam at 4.5 bar refers to a real paper factory (Ahadi-Oskui, 2001). The cogeneration plant has to fulfil primarily the needs for thermal energy of the paper machines. If more electricity is produced than required, the excess is sold on the market; in the opposite case, the deficit is bought from the network.

There is a total of 28 binary and 48 continuous structural variables in the superstructure. The optimization problem can be described verbally by:

min	Total levelized costs per time unit
s.t.	Constraints referring to:
	plant components, material properties, investment,
	operating and maintenance cost and economic analysis

All in all the model contains 1308 variables and 1640 constraints. The program is block-separable with a maximum block size of 47. Figure 13.7 shows a part of the related sparsity structure.

The program was coded in `GAMS`, and solved with the BCP Algorithm 13.1. A nonlinear relaxation was generated by using polynomial underestimators (Section 6.5). Since some functions have singularities in $[\underline{x}, \overline{x}]$, the constrained sampling technique described in Section 6.5.2 was used. In order to make the local optimization more robust, the continuous optimization problem was first solved with respect to the thermo-dynamic variables, and afterwards with respect to all variables including the remaining thermo-economic variables. CPLEX was used for solving LPs, and CONOPT was used for solving NLPs. Several bounding and branching strategies of the BCP algorithm were tried.

Similar to the MINLP-results of Section 13.5.2, the BCP Algorithm with LP-bounds and without Lagrangian cuts performed best. It solved the problem in 10 hours and 50 minutes on a Linux PC with a Pentium 4 processor with 3 GHz and 1 GB RAM. For comparison, the problem was also solved with a customized genetic algorithm, which produced a slightly worse solution after 23 hours. More details about these results can be found in (Ahadi-Oskui et al., 2003) and in the Ph.D. thesis of T. Ahadi-Oskui, which is currently under preparation.

13.6 Nonconvex polyhedral inner and outer approximations

BCP methods that use convex polyhedral approximations have three main disadvantages. First, it is not easy to perform warm starts for solving similar (Lagrangian) subproblems. Second, many rectangular subdivisions may be necessary to close a large duality gap. Finally, it can be time-consuming to generate a partition such that a local minimizer that originally was located in the interior of the bounding box is a vertex with respect to all partition elements. In the worst case, 2^n rectangular subdivisions are required.

These disadvantages can be diminished if *nonconvex* polyhedral inner and outer approximations are used. Consider a MINLP given in the form

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in G_k, \quad k = 1, \ldots, p. \end{array} \tag{13.7}$$

A *nonconvex polyhedral inner approximation* of (13.7) is defined by replacing the

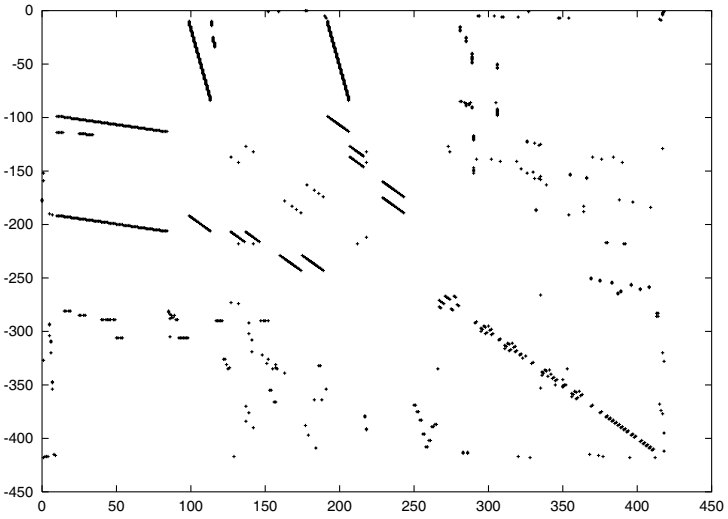

Figure 13.7: Part of the sparsity structure of the analyzed energy conversion system.

sets G_k by

$$\check{G}_k = \bigcup_{i=1}^{l_k} \operatorname{conv}(W_{k,i}), \quad k = 1, \ldots, p,$$

where $W_{k,i} \subset \operatorname{conv}(G_k)$ is a finite set of inner approximation points. The resulting problem is an *MIP restricted master problem* of the form:

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in \check{G}_k, \quad k = 1, \ldots, p. \end{array} \tag{13.8}$$

Problem (13.8) is a disjunctive linear program, which can be formulated as the following MIP:

$$\begin{array}{lll} \min & c^T (W \bullet z) + c_0 & \\ \text{s.t.} & A(W \bullet z) + b \leq 0 & \\ & \sum_{w \in W_{k,i}} z_w = y_{k,i}, & i = 1, \ldots, l_k,\ k = 1, \ldots, p \\ & \sum_{i=1}^{l_k} y_{k,i} = 1, & k = 1, \ldots, p \\ & y_{k,i} \in \{0, 1\}, & i = 1, \ldots, l_k,\ k = 1, \ldots, p \\ & z \geq 0 & \end{array} \tag{13.9}$$

where $W = (W_1, \ldots, W_p)$ with $W_k = (W_{k,1}, \ldots, W_{k,l_k})$.

Similarly, a *nonconvex polyhedral outer approximation* of (13.7) is defined by replacing the sets G_k by

$$\hat{G}_k = \bigcup_{i=1}^{m_k} \hat{G}_{k,i}, \quad k = 1, \ldots, p$$

where $\hat{G}_{k,i}$ is a convex polyhedron and $\hat{G}_k \supset G_k$. This results in the following program

$$\begin{array}{ll} \min & c^T x + c_0 \\ \text{s.t.} & Ax + b \leq 0 \\ & x_{J_k} \in \hat{G}_k, \quad k = 1, \ldots, p \end{array} \tag{13.10}$$

which can be written as an MIP (see Section 2.2.2).

A nonconvex polyhedral outer approximation of the set G_k can be generated by solving a subproblem of the form

$$\min\{a^T x \mid x \in G_k\} \tag{13.11}$$

by an LP-based branch-and-bound method that prunes only infeasible subproblems.

Denote by $\hat{G}_{k,i}, i \in I_{\text{nodes}}$, the polyhedral outer approximations corresponding to the nodes generated by a branch-and-bound algorithm. Then $\hat{G}_k = \bigcup_{i \in I_{\text{nodes}}} \hat{G}_{k,i}$ is a nonconvex polyhedral outer approximation of G_k and

$$\text{val}(13.11) = \min_{i \in I_{\text{nodes}}} \min\{a^T x \mid x \in \hat{G}_{k,i}\}.$$

If a solution candidate x^* of (13.7) is available, a polyhedral outer approximation $\hat{G}_k$ can be refined by adding *interval-gradient cuts*, as defined in Section 7.1.3. Moreover, a polyhedral outer-approximation can be refined by setting $\hat{G}_k^{\text{new}} = \hat{G}_k \setminus \operatorname{int} \hat{T}_k$, where $\hat{T}_k$ is a polyhedral cone pointed at $x^*_{J_k}$. Here, it is assumed that either

$$\operatorname{int} \hat{T}_k \cap G_k = \emptyset, \tag{13.12}$$

or that the subproblem related to $\hat{T}_k$ can be easily fathomed. Condition 13.12 is fulfilled, for example, if G_k is defined by concave inequalities.

Similarly, polyhedral inner approximations can be refined if a solution candidate x^* of (13.7) is available. Define by $w^k = x^*_{J_k}$ a new inner approximation point. Then a refinement of a polyhedral inner approximation $\text{conv}(W_k)$ is given by

$$\bigcup_{w \in W_k} \text{conv}(\{w^k\} \cup W_k \setminus \{w\}).$$

Instead of the convex polyhedral inner and outer approximations (4.16) and (7.12), the nonconvex polyhedral inner and outer approximations (13.9) and

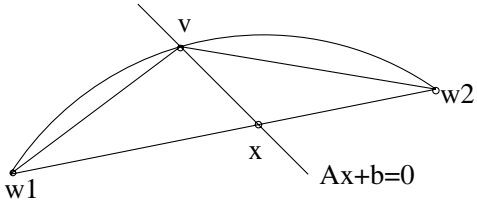

Figure 13.8: Nonconvex polyhedral inner approximation: Refinement of $\text{conv}(\{w1, w2\})$ by $\text{conv}(\{w1, v\}) \cup \text{conv}(\{v, w2\})$, where x is the solution of a convex relaxation.

(13.10) can be used in the BCP Algorithm 13.1 and in the column generation Algorithm 4.8. The latter method is similar to the *Lagrangian and domain cut method* proposed in (Li et al., 2002).

Chapter 14

LaGO — An Object-Oriented Library for Solving MINLPs

LaGO (**La**grangian **G**lobal **O**ptimizer) is an object-oriented library for solving nonconvex MINLPs that contains most of the algorithms described in this work (Nowak et al., 2003). The source code of this software package contains currently more than 33000 lines written in C++. It was developed over the last four years. In the following, a short overview of the software is given. Detailed information about the available classes and methods of LaGO can be found in the online documentation:
`http://www.mathematik.hu-berlin.de/~eopt/LaGO/documentation/` .

14.1 Design philosophy

LaGO was designed with four goals in mind — *general purpose*, *efficiency*, *generic framework* and *ease of use*. With respect to general purpose, it was aimed at solving general structured (sparse and block-separable) nonconvex MINLPs. With respect to efficiency, it was aimed at exploiting problem structure, such as block-separable functions and sparse and low-rank Hessians. With respect to generic framework, it was aimed at using generic objects, such as linear-algebra subroutines, convex underestimators, cuts and (local and global) solvers, which can be replaced by user supplied problem-specific implementations for the use of special structure. With respect to ease of use, it was aimed at using the software as a black-box, whereby the user defines the problem in a modeling language. LaGO is currently linked to the algebraic modeling systems AMPL (Fourer et al., 1993) and GAMS (GAMS, 2003).

14.2 Related work

Here, some of the currently available software packages for solving MINLPs are listed. We do not mention software packages for continuous global optimization, which are described for example in (Neumaier, 2004). Software packages for solving nonconvex MINLPs include:

(i) BARON. A general purpose branch-and-bound solver based on polyhedral relaxation and box reduction (Sahinidis, 1996; Sahinidis, 2002).

(ii) αBB. A general purpose branch-and-bound solver based on nonlinear convex relaxation (Adjiman et al., 2002).

(iii) OQNLP. A combination of tabu, scatter search, simulated annealing and evolutionary algorithms (Lasdon, 2003).

(iv) XPRESS-SLP. A successive linear programming MINLP heuristic (Dash Optimization, 2003).

(v) SCICONIC. [1] A MINLP heuristic based on MIP approximation (SCICON Ltd., 1989).

Software packages for solving convex MINLPs include:

(i) DICOPT. An outer approximation method (Grossmann, 2002).

(ii) AIMMS-OA. An open outer approximation method (Bisschop and Roelofs, 2002).

(iii) MINOPT. Benders and cross decomposition and outer approximation methods (Schweiger and Floudas, 2002).

(iv) MINLPBB. A branch-and-bound method (Fletcher and Leyffer, 2002).

(v) SBB. A branch-and-bound method (Bussiek and Drud, 2002).

(vi) LogMIP. Disjunctive programming solver (Vecchietti and Grossmann, 1999).

(vii) Alpha-ECP. Extended cutting-plane method (Westerlund and Lundquist, 2003).

So far generic BCP frameworks have been developed only for MIP. Among them are:

(i) SYMPHONY. (Ralphs, 2000)

(ii) COIN/BCP. (IBM, 2003)

(iii) ABACUS. (OREAS GmbH, 1999).

[1] This was probably the first commercial MINLP code developed in the mid 1970's (Bussieck and Pruessner, 2003)

14.3 Structure

The three basic modules of LaGO are: *reformulation*, *relaxation* and *solvers*. The reformulation module provides methods for building block-separable reformulations of a given MINLP. The relaxation module contains several methods for generating and improving relaxations. In the solver module several algorithms for computing solution candidates of a general MINLP are implemented.

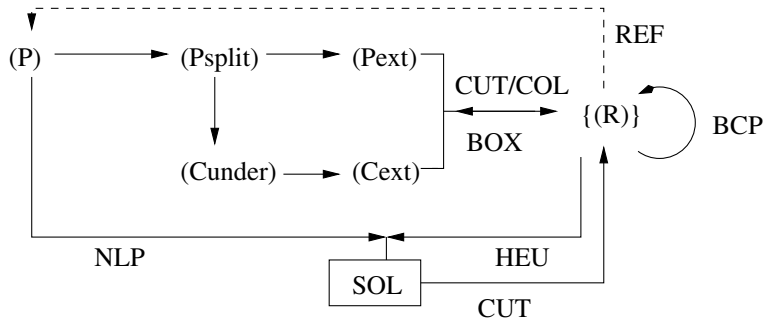

Figure 14.1: Basic components of LaGO

Figure 14.1 illustrates the basic structure of LaGO. Starting from a given MINLP, called (P), LaGO constructs a block-separable problem (P_{split}), a convex underestimating-relaxation (C_{under}), an extended block-separable reformulation (P_{ext}), and an extended convex relaxation (C_{ext}). A polyhedral relaxation (R) is initialized from (P_{ext}) and (C_{ext}) by using the cut generator `CUT` and a box-reduction procedure `BOX` . A branch-cut-and-price algorithm `BCP` splits the root-relaxation (R) into several node-relaxations $\{(R)\}$. A relaxation is improved by adding cuts and columns via the cut generator `CUT` and the column generator `COL`. From a node-relaxation (R), solution candidates are retrieved by using a heuristic `HEU` and a nonlinear solver `NLP`. If a good solution is found, it is added to the solution set `SOL`, and a level or optimality cut is added to (R) through the cut generator `CUT`. In the future, it is planned to update a discretized stochastic optimization or optimal control problem (P) from solutions of relaxations (R) by using a mesh and/or scenario refinement procedure `REF` (see Chapter 9). Figure 14.2 shows the basic objects of LaGO .

14.4 The modules

14.4.1 Reformulation

The reformulation module of LaGO is currently linked to the modeling systems GAMS (GAMS, 2003) and AMPL (Fourer et al., 1993). In both systems a MINLP

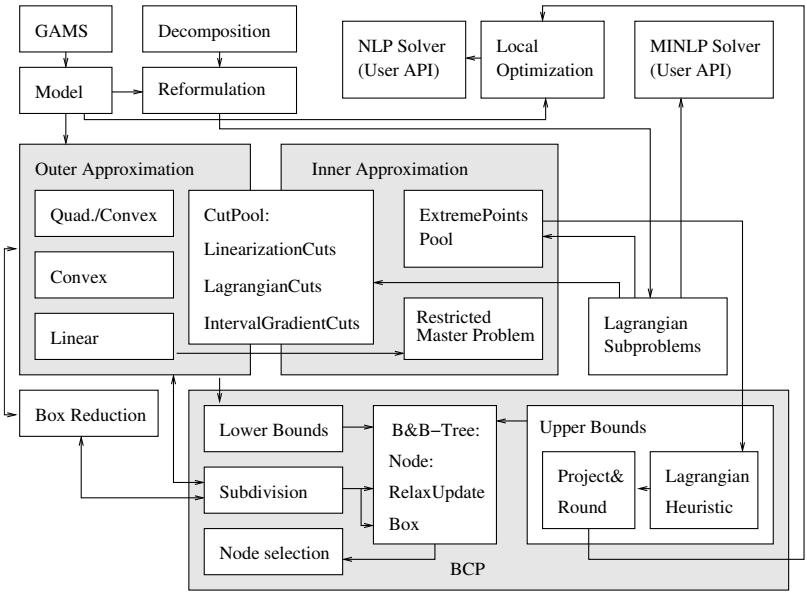

Figure 14.2: Object diagram of LaGO

is represented as a *black-box model* of the form:

$$\begin{array}{lll} \min & h_0(x) & \\ \text{s.t.} & h_i(x) \leq 0 & i \in I \\ & h_i(x) = 0, & i \in E \\ & x \in [\underline{x}, \overline{x}] & \\ & x_B \subset \mathbb{Z}^{|B|} & \end{array} \tag{14.1}$$

where I and E form a partition of $\{1, \ldots, m\}$, and h_i are black-box functions for which there are procedures for evaluating function values, gradients and Hessians. Algorithm 14.1 describes a preprocessing procedure for reformulating a black-box model (14.1) as a block-separable binary program of the form (2.4), where all functions have the form

$$h(x) = c + \sum_{k=1}^{p} b_k^T x_{L_k} + x_{Q_k}^T A_k x_{Q_k} + h_k(x_{N_k}) \tag{14.2}$$

and the index sets L_k, Q_k and N_k of the linear, quadratic and non-quadratic variables are subsets of a block $J_k \subseteq \{1, \ldots, n\}$. The matrix A_k can be dense, sparse or user-defined. The functions $h_k(x_{N_k})$ are evaluated as in Remark 2.2.

Since not all MINLP instances have finite variable bounds, as for example in some instances from the MINLPLib (Bussieck et al., 2003a), the following method

for generating missing bounds is used. First, the type (linear, convex, concave) of all functions is determined by evaluating the minimum and maximum eigenvalue of each Hessian at sample points. Then, all nonconvex constraints are removed from the original problem. Finally, the function x_i is minimized or maximized with respect to the remaining convex constraints, thus defining a lower and an upper bound for x_i.

Transform integer constraints into binary constraints.

Compute non-existing variable bounds and set $E_{\text{sparse}} = \emptyset$.

Replace equality constraints by two inequality constraints.

for $i = 0, \ldots, m$:

Generate a sample set S_i, set $V_i = \emptyset$ and set $N_i = \emptyset$.

for $j = 1, \ldots, n$:
if $\frac{\partial}{\partial x_j} h_i(x) \neq 0$ for some $x \in S_i$: Put j into V_i.

Compute the Hessian $H(x) = \nabla^2 h_i(x)$ for $x \in S_H$.

if $H_{kl}(x) \neq 0$ for some $x \in S_H$: Put (k, l) into E_{sparse}, and k, l into N_i.

if $H(x) = H(y)$ for all $x, y \in S_H$: The function h_i is considered to be quadratic.

end for

end for

Algorithm 14.1: Procedure for representing all functions in form (14.2) and computing the sparsity graph

14.4.2 Relaxation

Nonlinear convex relaxations are constructed by using α-underestimators, CGU-underestimators or convexified polynomial underestimators. Convex relaxations of MIQQPs can be alternatively computed by using the semidefinite relaxation module. A box-reduction procedure is implemented for tightening a root-relaxation. It is planned to use this procedure also for tightening a node-relaxation. For the construction of a polyhedral relaxation the following cuts are implemented: linearization cuts, Knapsack cuts, level cuts and Lagrangian cuts.

14.4.3 Solvers

There are four basic MINLP-solvers: a deformation heuristic, a rounding-and-partitioning heuristic, a Lagrangian heuristic and a branch-cut-and-price algorithm using NLP, LP or dual bounds. The default NLP-solver is SNOPT (Gill et al., 1997). In addition, CONOPT (Drud, 1996) can be used to compute local minimizers of the continuous subproblem with fixed binary variables. The default solver for linear programs is CPLEX (ILOG, Inc., 2005). Moreover, SNOPT can be used to solve LPs. For maximizing a non-differentiable dual function, a simple subgradient algorithm (default) or the proximal bundle code NOA (Kiwiel, 1994b) is used. The parameters of the solvers can be modified via a resource file without recompiling the code.

Appendix

Appendix A

Future Perspectives

We think that deterministic algorithms for global optimization of nonconvex mixed-integer nonlinear programs will become an increasingly important research area in the future. This view is also supported by other authors interested in the future perspectives of optimization (Grossmann and Biegler, 2002).

The concept of the BCP framework presented here for general MINLPs is quite similar to modern BCP methods for MIPs. However, our current BCP-solver for MINLP is still in its infancy, and there is still much room for improvement in order to make it more efficient. The following list includes a number of things that would facilitate the development of a reliable large-scale general purpose MINLP solver:

1. Nonconvex polyhedral outer and inner approximations and an MIP master problem can be used, as described in Section 13.6.
2. Faster solution of (Lagrangian) subproblems: Specialized sub-solvers can be used for solving particular subproblems, such as separable MINLP, convex MINLP, concave NLP or MIP. In particular, LP-based branch-and-bound methods seem to be quite efficient. Since similar subproblems have to be solved, a sub-solver should be able to perform a warm-start.
3. The column generation based *fixing heuristic* of (Borndörfer et al., 2001) can be used to simultaneously generate columns and to fix binary variables.
4. Generation of block-separable reformulations: Instead of black-box representations of MINLPs, *expression trees* or *directed acyclic graph* (DAG) representations Schichl and Neumaier, 2004 can be used to generate splitting schemes and subproblems in a flexible way.
5. Rigorous bounds: Rigorous underestimators can be computed using interval methods, as discussed in Section 6.5.1. Moreover, predefined convex underestimators for special functions, such as Bézier-underestimators defined in Section 6.2, can be used.

6. Box reduction: The described box-reduction methods can be applied to each node produced by the BCP method. Furthermore, constraint propagation tools, such as used in the *Constrained Envelope Scheduling* approach (Boddy and Johnson, 2003), can be included in the MINLP-BCP framework.
7. Parallelization: A parallel MINLP-BCP framework, based on the structure of COIN-BCP (IBM, 2003), can be developed.
8. Support of user-knowledge: Similar to the open outer approximation MINLP solver AIMMS-OA (Bisschop and Roelofs, 2002), an *open BCP algorithm* can be developed, which allows users to tune solution strategies for specific problems.
9. Support of discretized optimization problems: Based on the ideas of Chapter 9, a tool for simultaneously solving and updating discretized stochastic programs and optimal control problems can be implemented.

Appendix B

MINLP Problems

B.1 Instances from the MINLPLib

The MINLPLib is a recently created library of MINLP instances (Bussieck et al., 2003a). These problems come from a very wide variety of applications. Table B.2 shows 49 instances of the MINLPLib that were used in our numerical experiments. The corresponding columns are described in Table B.1. Almost all problems are block-separable and have a small maximum block-size. Moreover, 14 problems are convex, and 19 problems are quadratic.

name	The name of the problem
n	The number of variables
$\|B\|$	The number of binary variables
m	The number of constraints
box diam	The diameter of $[\underline{x}, \overline{x}]$
avg. block size	The average block size
max. block size	The maximum block size
p	The number of blocks
max nr.var	The maximum number of nonlinear variables of the objective or a constraint function
conv	Indicates if the problem is a convex MINLP or not
probl type	The type of the problem: 'Q' means MIQQP and 'N' means MINLP

Table B.1: Descriptions of the columns of Table B.2.

name	n	$\|B\|$	m	box diam.	block size avg	block size max	p	max nl.var	conv	probl type
alan	9	4	7	∞	1.3	3	7	3	yes	Q
batch	47	24	73	1567.64	1.3	2	36	2	yes	N
batchdes	20	9	20	∞	1.3	2	15	2	no	N
elf	55	24	38	∞	1.1	2	52	2	no	Q
eniplac	142	24	189	∞	1.2	2	118	2	no	N
enpro48	154	92	215	∞	1.1	3	138	3	no	N
enpro56	128	73	192	∞	1.1	3	116	3	no	N
ex1221	6	3	5	14.2478	1	1	6	1	no	N
ex1222	4	1	3	1.77255	1	1	4	1	no	N
ex1223	12	4	13	17.5499	1	1	12	1	yes	N
ex1223a	8	4	9	17.4356	1	1	8	1	yes	Q
ex1223b	8	4	9	17.4356	1	1	8	1	yes	N
ex1224	12	8	7	3.31491	1.2	3	10	3	no	N
ex1225	9	6	10	6.16441	1.1	2	8	2	no	N
ex1226	6	3	5	10.4403	1.2	2	5	2	no	N
ex1252	40	15	43	5192.7	1.8	7	22	4	no	N
ex1263	93	72	55	63.8122	1.2	5	77	2	no	Q
ex1264	89	68	55	30.8869	1.2	5	73	2	no	Q
ex1265	131	100	74	35.3129	1.2	6	106	2	no	Q
ex1266	181	138	95	39.2428	1.2	7	145	2	no	Q
ex3	33	8	31	∞	1	1	33	1	no	N
ex4	37	25	31	∞	1	1	37	1	no	Q
fac1	23	6	18	1200	2.6	8	9	8	yes	N
fac3	67	12	33	7348.47	4.2	18	16	18	yes	Q
feedtray2	88	36	284	∞	3.4	63	26	17	no	Q
fuel	16	3	15	∞	1	1	16	1	no	Q
gastrans	107	21	149	∞	1.2	2	86	2	no	N
gbd	5	3	4	1.90788	1	1	5	1	yes	Q
gear2	29	24	4	96.1249	1.1	4	26	4	no	N
gkocis	12	3	8	∞	1	1	12	1	no	N
johnall	195	190	192	13.9284	1	3	193	3	no	N
meanvarx	36	14	44	∞	1.2	7	30	7	yes	Q
nous2	51	2	43	∞	3.4	8	15	5	no	Q
oaer	10	3	7	∞	1	1	10	1	no	N
ortez	88	18	74	∞	1.4	9	61	2	no	N
parallel	206	25	115	∞	3.7	151	56	131	no	N
procsel	11	3	7	∞	1	1	11	1	no	N
ravem	113	54	186	∞	1.1	3	101	3	yes	N
sep1	30	2	31	237.181	1.2	5	26	2	no	Q
space25	894	750	235	∞	1	43	852	7	no	Q
space25a	384	240	201	∞	1.1	43	342	7	no	Q
spectra2	70	30	73	∞	1.6	10	43	10	no	Q
stockcycle	481	432	97	1060.22	1	1	481	1	yes	N
synheat	57	12	64	∞	1.5	5	37	5	no	N
synthes1	7	3	6	3.4641	1.2	2	6	2	yes	N

synthes2	12	5	14	∞	1.1	2	11	2	yes	N
synthes3	18	8	23	6.55744	1.1	2	17	2	yes	N
util	146	28	168	∞	1	5	141	2	no	Q
waterx	71	14	54	∞	1.7	3	41	3	no	N

Table B.2: Instances from the MINLPLib

B.2 Random MIQQP problems

Algorithm B.1 shows a procedure for generating a random MIQQP of the form

$$\begin{array}{lll} \min & q_0(x) & \\ \text{s.t.} & q_i(x) \leq 0, & i = 1, \ldots, m/2 \\ & q_i(x) = 0, & i = m/2+1, \ldots, m \\ & x \in [\underline{x}, \overline{x}], & x_B \text{ binary} \end{array} \tag{B.1}$$

where $q_i(x) = x^T A_i x + 2a_i^T x + d_i$, $A_i \in \mathbb{R}^{(n,n)}$ is symmetric, $a_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$, $i = 0, \ldots, m$. The functions q_i are block-separable with respect to the blocks $J_k = \{(k-1)l + 1, \ldots, kl\}$, $k = 1, \ldots, p$. Since $c_i = 0$ for $i = 0, \ldots, m$, the point $x = 0$ is feasible for (B.1).

Input: (n,m,l)

Set $p = n/l$ (number of blocks).

Set $B = \{1, \ldots, n/2\}$, $\underline{x} = -e$ and $\overline{x} = e$.

for $i = 0, \ldots, m$

Compute symmetric dense matrices $A_{i,k} \in \mathbb{R}^{(l,l)}$ with uniformly distributed random components in $[-10, 10]$ for $k = 1, \ldots, p$.

Compute vectors $b_i \in \mathbb{R}^n$ with uniformly distributed random components in $[-10, 10]$, and set $c_i = 0$.

end for

Algorithm B.1: Procedure for generating random MIQQPs

Bibliography

Adjiman, C. S., Androulakis, I. P., Maranas, C. D., and Floudas, C. A. (2002). *αBB*.
`http://titan.princeton.edu/soft.html\#abb`.

Adjiman, C. S., Dallwig, S., Floudas, C. A., and Neumaier, A. (1998). A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances. *Comp. Chem. Eng.*, pages 1137–1158.

Adjiman, C. S. and Floudas, C. A. (1997). Rigorous convex underestimators for general twice-differentiable problems. *J. Global Opt.*, 9:23–40.

Ahadi-Oskui, T. (2001). Optimierung einer industriellen Kraft-Wärme-Kopplungsanlage mit Hilfe von evolutionären Algorithmen. *Master thesis, Technical University Berlin*.

Ahadi-Oskui, T., Alperin, H., Cziesla, F., Nowak, I., and Tsatsaronis, G. (2001). *Optimization of complex energy conversion systems*.
`http://www.math.hu-berlin.de/~eopt/engl\_index.html`.

Ahadi-Oskui, T., Alperin, H., Cziesla, F., Nowak, I., and Tsatsaronis, G. (2003). A relaxation-based heuristic for the design of cost-effective energy conversion systems. *accepted for publication in ENERGY*.

Al-Khayyal, F. A., Larsen, C., and van Voorhis, T. (1995). A relaxation method for nonconvex quadratically constrained quadratic programs. *J. Glob. Opt*, 6:215–230.

Al-Khayyal, F. A. and van Voorhis, T. (1996). Accelerating convergence of branch-and-bound algorithms for quadratically constrained optimization problems. In Floudas, C. A., editor, *State of the Art in Global Optimization: Computational Methods and Applications*. Kluwer Academic Publishers, Dordrecht.

Alperin, H. and Nowak, I. (2002). Lagrangian Smoothing Heuristics for MaxCut. Technical report, Humboldt-Universität zu Berlin NR–2002–6.

Anstreicher, K. and Wolkowicz, H. (1998). On Lagrangian relaxation of quadratic matrix constraints. *Research Report CORR 98-24, Department of Combinatorics and Optimization,University of Waterloo*.

Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (1999). *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*. Springer, Berlin.

Beasley, J. E. (1998). Heuristic Algorithms for the Unconstrained Binary Quadratic Programming Problem. Technical report, The Management School, Imperial College, London SW7 2AZ, England.
`http://mscmga.ms.ic.ac.uk/jeb/jeb.html`.

Beck, A. and Teboulle, M. (2000). Global optimality conditions for quadratic optimization problems with binary constraints. *SIAM J. Opt.*, pages 179–188.

Becker, R. and Lago, G. (1970). A global optimization algorithm. In *Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*, pages 3–12.

Benson, S. J., Ye, Y., and Zhang, X. (2000). Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Optim.*, 10(2):443–461.

Berloni, A., Campadelli, P., and Grossi, G. (1998). An approximation algorithm for the maximum cut problem and its experimental analysis. *Proceedings of "Algorithms and Experiments"*, pages 137–143.

Bertsekas, D. P. (1995). *Non-linear programming*. Athena Scientific, Belmont, MA.

Bisschop, J. and Roelofs, M. (2002). *AIMMS - The User's Guide*. Paragon Decision Technology B.V., Haarlem, The Netherlands,
`http://www.aimms.com`.

Bixby, R. (2001). Solving Real-World Linear Programs: A Decade and More of Progress. *To appear in Operations Research*.

Bixby, R., Fenelon, M., Gu, Z., Rothberg, E., and Wunderling, R. (2000). MIP: theory and practice - closing the gap. In Powell, M. and Scholtes, S., editors, *System Modelling and Optimization: Methods, Theory and Applications*, pages 19–49. Kluwer Academic Publishers, Dordrecht.

Bliek, C., Spellucci, P., Vicente, L., Neumaier, A., Granvilliers, L., Monfroy, E., Benhamou, F., Huens, E., Hentenryck, P. V., Sam-Haroud, D., and Faltings, B. (2001). *COCONUT Deliverable D1, Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of The Art*.
`http://www.mat.univie.ac.at/~neum/glopt/coconut/StArt.html`.

Boddy, M. and Johnson, D. (2003). A new method for the global solution of large systems of continuous constraints. In Bliek, C., Jermann, C., and Neumaier, A., editors, *Global Optimization and Constraint Satisfaction*, pages 142–156. Springer, Berlin.

Boender, C. G. E. and Romeijn, H. E. (1995). Stochastic methods. In Horst, R. and Pardalos, P., editors, *Handbook of Global Optimization*, pages 829–869. Kluwer Academic Publishers, Dordrecht.

Bomze, I. (1998). On standard quadratic optimization problems. *J. Global Opt.*, 13:369–387.

Borndörfer, R., Grötschel, M., and Löbel, A. (2001). Scheduling duties by adaptive column generation. Technical report, ZIB-Report 01-02, Konrad-Zuse-Zentrum für Informationstechnik Berlin.

Burer, S. and Monteiro, R. D. C. (2001). A Nonlinear Programming Algorithm for Solving Semidefinite Programs via Low-rank Factorization. Technical report, School of ISyE, Georgia Tech, Atlanta.

Burer, S., Monteiro, R. D. C., and Zhang, Y. (2001). Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM J.Opt.*, 12:503–521.

Burkard, R., Kocher, M., and Rudolf, R. (1997). Rounding strategies for mixed integer programs arising from chemical production planning. Technical report, Report No.119 from the Optimization Group at the TU Graz.

Bussieck, M., Drud, A., and Meeraus, A. (2003a). MINLPLib - A Collection of Test Models for Mixed-Integer Nonlinear Programming. *INFORMS J. Comput.*, 15(1).

Bussieck, M. and Pruessner, A. (2003). Mixed-integer nonlinear programming. *SIAG/OPT Newsletter: Views & News.*

Bussieck, M. R., Lasdon, L. S., Pintér, J. D., and Sahinidis, N. V. (2003b). *Global Optimization with GAMS - Applications and Performance.* `http://www.gams.com/presentations/present_BLOglobal.pdf`.

Bussiek, M. and Drud, A. (2002). *SBB User Manual.* `http://www.gams.com/solvers/sbb.pdf`.

Casey, M. and Sen, S. (2003). *The Scenario generation algorithm for multistage stochastic linear programming.* `http://www.math.ups/~mcasey`.

Cohen, E. and Schumaker, L. (1985). Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2:229–235.

Conway, J. H. and Sloane, N. J. A. (1993). *Sphere Packings, Lattices and Groups.* 2nd edn, Springer, New York.

Dahmen, W. (1986). Subdivision algorithms converge quadratically. *J. of Computational and Applied Mathematics*, 16:145–158.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8:101–111.

Dantzig, G. B. and Wolfe, P. (1961). The decomposition algorithm for linear programs. *Econometrica*, 29:767–778.

Dash Optimization (2003). *XPRESS*. `http://www.dashopt.com`.

Demands, E. V. and Tang, C. S. (1992). Linear control of a Markov production system. *Operations Research*, 40:259–278.

Dentcheva, D., Guddat, J., and Rückmann, J.-J. (1995). Pathfollowing methods in nonlinear optimization III: multiplier embedding. *ZOR - Math. Methods of OR*, 41:127–152.

Dentcheva, D. and Römisch, W. (2002). Duality gaps in nonconvex stochastic optimization. Technical report, Preprint 02-5, Institut für Mathematik, Humboldt-Universität zu Berlin, *accepted for publication in Math. Progr.*

Dixon, L. and Szegö, G. (1975). *Towards global optimization*. North-Holland, Amsterdam.

Douglas, J. and Rachford, H. (1956). On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.*, 82:421–439.

Drud, A. (1996). *CONOPT: A System for Large-Scale Nonlinear Optimimization, Refeference Manual for CONOPT Subroutine Library*. ARKI Consulting and Development A/S, Bagsvaerd, Denmark.

Dupacová, J., Gröwe-Kuska, N., and Römisch, W. (2003). Scenario reduction in stochastic programming: An approach using probability metrics. *to appear in Math. Progr.*

Dür, M. (2001). Dual bounding procedures lead to convergent Branch-and-Bound algorithms. *Math. Progr.*, 91:117–125.

Duran, M. A. and Grossmann, I. E. (1986). An outer approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Progr.*, 36:307.

Elzinga, J. and Moore, T. (1975). A central cutting plane algorithm for the convex programming problem. *Math. Progr.*, 8:134–145.

Epperly, T. G. W. and Swaney, R. E. (1996). *Branch and Bound for Global NLP: New Bounding LP*, pages 1–35. Kluwer Academic Publishers, Dordrecht.

Everett, H. (1963). Generalized Lagrange Multiplier Method for Solving Problems of Optimal Allocation of Resources. *Operations Research*, 11:399–417.

Farin, G. (1986). Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3:83–127.

Feltenmark, S. and Kiwiel, K. C. (2000). Dual applications of proximal bundle methods including lagrangian relaxation of nonconvex problems. *SIAM J. Optim.*, 10(3):697–721.

Ferris, M. C. and Horn, J. D. (1998). Partitioning mathematical programs for parallel solution. *Math. Progr.*, 80:35–61.

Festa, P., Pardalos, P. M., Resende, M. G. C., and Ribeiro, C. C. (2002). Randomized heuristics for the max-cut problem. *Optimization Methods and Software*, 7:1033–1058.

Filar, J. A. and Schultz, T. A. (1999). Bilinear programming and structured stochastic games. *J. Opt. Theor. Appl.*, 53:85–104.

Fisher, M. L. (1980). Worst-case analysis of heuristic algorithms. *Management Science*, 26(1):1–17.

Fletcher, R. (1987). *Pratical Methods of Optimization*. John Wiley & Sons, New York, second edition.

Fletcher, R. and Leyffer, S. (1994). Solving Mixed Integer Nonlinear Programs by Outer Approximation. *Math. Progr.*, 66:327.

Fletcher, R. and Leyffer, S. (2002). *MINLP*. `http://www.maths.dundee.ac.uk/~sleyffer/MINLP.html`.

Flippo, O. E. and Kan, A. H. G. R. (1993). Decomposition in general mathematical programming. *Math. Progr.*, 60:361–382.

Floudas, C. A. (1995). *Nonlinear and Mixed Integer Optimization: Fundamentals and Applications*. Oxford University Press, New York.

Floudas, C. A. (2000). *Deterministic Global Optimization: Theory, Algorithms and Applications*. Kluwer Academic Publishers, Dordrecht.

Floudas, C. A., Aggarwal, A., and Ciric, A. R. (1989). Global optimum search for nonconvex NLP and MINLP problems. *Computers and Chemical Engineering*, 13(10):1117–1132.

Forgó, F. (1988). *Nonconvex Programming*. Akadémiai Kiadó, Budapest.

Forrest, S. (1993). Genetic algorithms: principles of natural selection applied to computation. *Science*, pages 872–878.

Fourer, R., Gay, D. M., and Kernighan, B. W. (1993). *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Brooks/Cole Publishing Company, Belmont, CA.

G. E. Paules, I. and Floudas, C. A. (1989). APROS: Algorithmic development methodology for discrete-continuous optimization problems. *Oper. Res.*, 37(6):902.

GAMS (2003). *GAMS - The Solver Manuals*. GAMS Development Corporation, Washington DC.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York.

Garloff, J., Jansson, C., and Smith, A. P. (2002). Lower bound functions for polynomials. *to appear in Journal of Computational and Applied Mathematics*.

Garloff, J. and Smith, A. (2003). An improved method for the computation of affine lower bound functions for polynomials. In Floudas, C. and Pardalos, P., editors, *Frontiers in Global Optimization*, pages 1–10. Kluwer Academic Publishers, Dordrecht.

Geoffrion, A. M. (1972). General Benders decomposition. *J. Opt. Theor. Appl.*, 10 (4):237–260.

Geoffrion, A. M. (1974). Lagrangian Relaxation for Integer Programming. *Math. Progr. Study*, 2:82–114.

Gill, P. E., Murray, W., and Saunders, M. A. (1997). SNOPT 5.3 user's guide. Technical report, University of California, San Diego, Mathematics Department Report NA 97-4.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Boston.

Gomes, F. and Sorensen, D. (1997). *ARPACK++: a C++ Implementation of ARPACK eigenvalue package*.
`http://www.crpc.rice.edu/software/ARPACK/`.

Goemans, M. X. and Williamson, D. P. (1995). Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42:1115–1145.

Goffin, J. L. (1977). On convergence rate of subgradient optimization methods. *Math. Progr.*, 13:329–347.

Goffin, J.-L. and Vial, J.-P. (1999). Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method. Technical report, University of Geneva, 102 Bd Carl-Vogt, CH-1211,HEC/Logilab.

Grossmann, I. (2002). *DICOPT*.
`http://www.gams.com/solvers/dicopt/main.htm`.

Grossmann, I. E. (2001). *Review of Non-Linear Mixed Integer and Disjunctive Programming Techiques for Proces Systems Engineering*.
`http://egon.cheme.cmu.edu/Group/Papers`.

Grossmann, I. E. and Biegler, L. (2002). *Part II: Future Perspective on Optimization*.
`http://egon.cheme.cmu.edu/Group/Papers`.

Grossmann, I. E. and Kravanja, Z. (1997). Mixed-integer nonlinear programming: A survey of algorithms and applications. In Conn, A., Biegler, L., Coleman, T., and Santosa, F., editors, *Large-Scale Optimization with Applicationa, Part II:Optimal Design and Control*. Springer, New York.

Grossmann, I. E. and Sahinidis, N. (2002a). Special Issue on Mixed-Integer Programming and it's Application to Engineering, Part I. *Optim. Eng.*, 3(4), Kluwer Academic Publishers, Dordrecht.

Grossmann, I. E. and Sahinidis, N. (2002b). Special Issue on Mixed-Integer Programming and it's Application to Engineering, Part II. *Optim. Eng.*, 4(1), Kluwer Academic Publishers, Dordrecht.

Grothey, A. (2001). *Decomposition Methods for Nonlinear Nonconvex Optimization Problems*. PhD thesis, The University of Edinburgh, `http://www.maths.ed.ac.uk/~agr/thesis.ps.gz`.

Guddat, J., Guerra, F., and Nowack, D. (1998). On the role of the Mangasarian-Fromovitz constraints qualification for penalty-, exact penalty- and Lagrange multiplier methods. In Fiacco, A. V., editor, *Mathematical Programming with Data Perturbations*, pages 159–183. Marcel Deckker, Inc., New York.

Guddat, J., Vazquez, F. G., and Jongen, H. T. (1990). *Parametric Optimization: Singularities, Pathfollowing and Jumps*. John Wiley and Sons, New York.

Guignard, M. and Kim, S. (1987). Lagrangian decomposition: a model yielding stronger Lagrangean bounds. *Math. Progr.*, 39(2):215–228.

Helmberg, C. (2000). Semidefinite Programming for Combinatorial Optimization. Technical report, ZIB–Report 00–34.

Helmberg, C. and Kiwiel, K. C. (2002). A spectral bundle method with bounds. *Math. Progr.*, 93(2):173–194.

Helmberg, C. and Rendl, F. (2000). A spectral bundle method for semidefinite programming. *SIAM J. Opt.*, 10(3):673–695.

Henrion, D. and Lasserre, J. B. (2002). Solving Global Optimization Problems over Polynomials with GloptiPoly 2.1. Technical report, LAAS-CNRS Research Report No. 02289, Toulouse, France.

Hiriart-Urruty, J. B. and Lemaréchal, C. (1993). *Convex Analysis and Minimization Algorithms I and II*. Springer, Berlin.

Hochbaum, D. (1999). *Randomization, Approximation, and Combinatorial Optimization*. Springer, Berlin.

Holmberg, K. (1990). On the convergence of the cross decomposition. *Math. Progr.*, 47:269.

Holmberg, K. and Ling, J. (1997). A lagrangian heuristic for the facility location problem with staircase costs. *Eur. J. Oper. Res.*, 97:63–74.

Hooker, J. (2000). *Logic-based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York.

Horst, R. and Pardalos, P. (1995). *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht.

Horst, R., Pardalos, P., and Thoai, N. (1995). *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht.

Horst, R. and Tuy, H. (1990). *Global Optimization (Deterministic Approaches)*. Springer, Berlin.

IBM (2003). *COIN-OR: Common Optimization Interface for Operations Research*.
`http://www.coin.org`.

ILOG, Inc. (2005). *CPLEX*.
`http://www.ilog.com/products/cplex/`.

Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. *Master's thesis*.

Kearfott, R. (1996). *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht.

Kesavan, P., Allgor, R. J., Gatzke, E. P., and Barton, P. I. (2001). Outer Approximation Algorithms for Separable Nonconvex Mixed-Integer Nonlinear Programs. *submitted to Mathematical Programming*.

Kirkpatrick, S., Geddat Jr., C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Kiwiel, K. C. (1990). Proximity control in bundle methods for convex nondifferentiable minimization. *Math. Progr.*, 46:105–122.

Kiwiel, K. C. (1993/1994b). *User's Guide for* N*OA 2.0/3.0: A* F*ORTRAN Package for Convex Nondifferentiable Optimization*. Polish Academy of Science, System Research Institute, Warsaw.

Kiwiel, K. C. (1994a). A cholesky dual method for proximal piecewise linear programming. *Numerische Mathematik*, 68.

Kocis, G. R. and Grossmann, I. E. (1987). Relaxation strategy for the structural optimization of nonconvex MINLP problems in process flow sheets. *Ind. Eng. Chem. Res.*, 26(9):1869.

Kojima, M., Kim, S., and Waki, H. (2003). A General Framework for Convex Relaxation of Polynomial Optimization Problems over Cones. *Journal of Operations Research Society of Japan*, 46(2):125–144.

Kojima, M., Matsumoto, T., and Shida, M. (1999). Moderate nonconvexity = convexity + quadratic concavity. Technical report, Research Reports on Mathematical and Comuting Sciences, Series B: Operations Research, Tokyo Institute of Technology, B-348.

Krawczyk, R. (1969). Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing*, 4:187–2001.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. In Neyman, J., editor, *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–493. University Press, Berkeley, California.

Lagrange, J. (1797). *Théorie des fonctions analytiques*. Impr. de la République, Paris.

Lasdon (2003). *OQNLP*. `http://www.gams.com/solvers/oqnlp.pdf`.

Lasserre, J. B. (2001). Global optimization with polynomials and the problem of moments. *SIAM J. Opt.*, 11(3):796–817.

Lemaréchal, C. (2001). Lagrangian relaxation. *Optimization Online*. `http://www.optimization-online.org/DB_HTML/2001/03/298.html`.

Lemaréchal, C. and Oustry, F. (1999). Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization. *Technical Report 3710, INRIA Rhône-Alpes*.

Lemaréchal, C. and Renaud, A. (2001). A geometric study of duality gaps, with applications. *Math. Progr.*, 90:399–427.

Li, D., Sun, X. L., Wang, J., and McKinnon, K. (2002). A convergent lagrangian and domain cut method for nonlinear knapsack problems. Technical report, SEEM2002-10, Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong.

Locatelli, M. (2002). Simulated annealing algorithms for continuous global optimization. In Pardalos, P. and Romeijn, H., editors, *Handbook of Global Optimization, Volume 2*, pages 179–229. Kluwer Academic Publishers, Dordrecht.

Mangasarin, O. L. (1969). *Nonlinear Programming*. McGraw-Hill, New York.

Mangasarin, O. L. and Fromowitz, S. (1967). The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. *J. Math. Anal. Appl.*, 17:37–37.

Martin, A. (1999). Integer programs with block structure. Technical report, ZIB–Report 99–03, Habilitationsschrift.

Mawengkang, H. and Murtagh, B. (1986). Solving nonlinear integer programs with large scale optimization software. *Ann. O. R.*, 5:425.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equations of state calculations by fast computing machine. *J. Chem. Phys.*, 21:1097.

Mockus, J. (1989). *Bayesian Approach to Global Optimization*. Kluwer Academic Publishers, Dordrecht.

Moore, R. (1979). *Methods and Applications of Interval Analysis*. SIAM, Philadelphia.

Moré, J. and Wu, Z. (1997). Global continuation for distance geometry problems. *SIAM J. Optim.*, 7:814–836.

Murty, K. G. (1987). Some NP-complete problems in quadratic and nonlinear programming. *Math. Progr.*, 39:117–129.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. Wiley-Interscience, New York.

NETLIB (1972-1973). *EISPACK*. `http://www.netlib.org/eispack/`.

Neumaier, A. (1990). *Interval Methods for Systems of Equations*. Cambridge Univ. Press, Cambridge, UK.

Neumaier, A. (1992). An optimality criterion for global quadratic optimization. *J. Global Opt.*, 2:201–208.

Neumaier, A. (1996). Second-order sufficient optimality conditions for local and global nonlinear programming. *J. Global Opt.*, 9:141–151.

Neumaier, A. (2001). Constrained Global Optimization. In *COCONUT Deliverable D1, Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of The Art*, pages 55–111. `http://www.mat.univie.ac.at/~neum/glopt/coconut/StArt.html`.

Neumaier, A. (2004). Complete search in continiuous global optimization and constraint satisfaction. In Iserles, A., editor, *Acta Numerica*, pages 271–369. Cambridge University Press, Cambridge, UK.

Nowak, I. (1996). A Branch-and-Bound algorithm for computing the global minimum of polynomials by Bernstein-Bézier patches on simplices. Technical report, Preprint M-09, BTU Cottbus.

Nowak, I. (1998). A global optimality criterion for nonconvex quadratic programming over a simplex. Technical report, Preprint 98-18, Humboldt-Universität zu Berlin.

Nowak, I. (1999). A new semidefinite programming bound for indefinite quadratic forms over a simplex. *J. Glob. Opt.*, 14:357–364.

Nowak, I. (2000). Dual bounds and optimality cuts for all-quadratic programs with convex constraints. *J. Glob. Opt.*, 18:337–356.

Nowak, I. (2004). Lagrangian Decomposition of Block-Separable Mixed-Integer All-Quadratic Programs. *Math. Progr.*, Online First:1–18.

Nowak, I., Alperin, H., and Vigerske, S. (2003). LaGO - An object oriented library for solving MINLPs. In Bliek, C., Jermann, C., and Neumaier, A., editors, *Global Optimization and Constraint Satisfaction*, pages 32–42. Springer, Berlin.
`http://www.mathematik.hu-berlin.de/~eopt/papers/LaGO.pdf`.

Nowak, M. P. and Römisch, W. (2000). Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. *An. Oper. Res.*, 100:251–272.

OREAS GmbH (1999). *ABACUS, A Branch and CUt System, Version 2.3, User's Guide and Reference Manual*.
`http://www.oreas.de`.

Padberg, M. and Rinaldi, G. (1991). A Branch-and-Cut algorithm for the resolution of large scale traveling salesman problems. *SIAM Review*, 33:60–10.

Pardalos, P. M. and Rosen, J. B. (1987). *Constrained Global Optimization: Algorithms and Applications*, volume 268. Springer Lecture Notes Comp. Sci., Berlin.

Pataki, G. and Schmieta, S. H. (2000). The DIMACS Library of Mixed Semidefinite Quadratic Linear Programs.
`http://dimacs.rutgers.edu/Challenges/Seventh/Instances/`.

Phan-huy-Hao, E. (1982). Quadratically constrained quadratic programming: Some applications and a method for solution. *ZOR*, 26:105–119.

Phillips, A., Rosen, J., and Walke, V. (1995). Molecular structure determination by global optimization. In *Dimacs Series in Discrete Mathematics and Theoretical Computer Science*, volume 23, pages 181–198.

Phillips, A. T. and Rosen, J. B. (1994). A Quadratic Assignment Formulation of the Molecular Conformation Problem. *J. Glob. Opt.*, 4:229–241.

Phing, T. Q., Tao, P. D., and An, L. T. H. (1994). A method for solving D.C. programming problems, Application to fuel mixture nonconvex optimization problems. *J. Global Opt.*, 6:87–105.

Pintér, J. (2005). *LGO*.
`http://www.pinterconsulting.com`.

Pintér, J. D. (1996). *Global Optimization in Action*. Kluwer Academic Publishers, Dordrecht.

Poljak, S., Rendl, F., and Wolkowicz, H. (1995). A recipe for semidefinite relaxation for (0,1)-quadratic programming. *J. Global Opt.*, 7(1):51–73.

Polyak, B. T. (1987). *Introduction to optimization*. Optimization Software, Inc., Publications division.

Polyak, B. T. (1993). A general method for solving extremum problems. *Soviet Mathematics*, 8:593–597.

Quesada, L. and Grossmann, I. E. (1992). An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Comp. Chem. Eng.*, 16:937–947.

Ralphs, T. (2000). *SYMPHONY Version 2.8 User's Guide*. `www.branchandcut.org/SYMPHONY`.

Ratschek, H. and Rokne, J. (1995). Interval Methods. In Horst, R. and Pardalos, P., editors, *Handbook of Global Optimization*, pages 751–828. Kluwer Academic Publishers, Dordrecht.

Reemtsen, R. (1994). Some outer approximation methods for semi-infinite programming problems. *J. Comp. Appl. Math.*, 53:87–108.

Rendl, F. and Wolkowicz, H. (1997). A semidefinite framework for trust region subproblems with applications to large scale minimization. *Math. Progr.*, 77 (2):273–299.

Resende, M. G. C. and Ribeiro, C. C. (2002). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G., editors, *State-of-the-Art Handbook in Metaheuristics*. Kluwer Academic Publishers, Dordrecht.

Rockafellar, R. T. (1974). Augmented Lagrangian multiplier functions and duality in nonconvex programming. *SIAM J. Control*, 12(2):268–285.

Rockafellar, R. T. and Wets, R. J. (1997). *Variational Analysis*. Springer, New-York.

Ruszczyński, A. (1997). Decomposition methods in stochastic programming. *Math. Progr.*, 79:333–353.

Rutenberg, D. P. and Shaftel, T. L. (1971). Product design: Sub-assemblies for multiple markets. *Management Science*, 18:B220–B231.

Ryoo, H. S. and Sahinidis, N. (1996). A Branch-and-Reduce Approach to Global Optimization. *J. Global Opt.*, 8:107–138.

Sahinidis, N. V. (1996). BARON: A general purpose global optimization software package. *J. Global Opt.*, 8(2):201–205.

Sahinidis, N. V. (2002). *BARON, Branch And Reduce Navigator, User's Manual*. `http://archimedes.scs.uiuc.edu/baron/baron.html`.

Schelstraete, S., Schepens, W., and Verschelde, H. (1998). Energy minimization by smoothing techniques: a survey. In Balbuena, E. P. and Seminario, J., editors, *From Classical to Quantum Methods*, pages 129–185. Elsevier.

Schichl, H. (2004). *Mathematical Modeling and Global Optimization*. Habilitation Thesis, draft of a book, Cambridge Univ. Press, to appear.

Schichl, H. and Neumaier, A. (2004). Interval analysis on directed acyclic graphs for global optimization. *J. Global Optimization, to appear.*

Schrijver, A. (1986). *Theory of Linear and Integer Programming.* John Wiley & Sons, Chichester.

Schweiger, C. and Floudas, C. (2002). *MINOPT.* `http://titan.princeton.edu/MINOPT/`.

SCICON Ltd. (1989). *SCICONIC User Guide Version 1.40.* Scicon Ltd., Milton Keynes, UK.

Sehlmann, M. and Fahle, T. (2003). Constraint programming based lagrangian relaxation for the automatic recording problem. *An. Oper. Res.*, 118:17–33.

Sherali, H. and Adams, W. (1999). *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems.* Kluwer Academic Publishers, Dordrecht.

Sherali, H. D. and Tuncbilek, C. H. (1995). A reformulation-convexification approach for solving nonconvex quadratic programming problems. *J. Global Opt.*, 7:1–31.

Shor, N. Z. (1987). Quadratic optimization problems. *Soviet J. Circ. Syst. Sci.*, 25, (6):1–11.

Shor, N. Z. (1992). Dual estimates in multiextremal problems. *J. Global Opt.*, 2:411–418.

Shor, N. Z. (1998). *Nondifferentiable Optimization and Polynomial Problems.* Kluwer Academic Publishers, Dordrecht.

Smith, E. M. B. and Pantelides, C. C. (1996). Global optimization of general process models. In Grossmann, I. E., editor, *Global Optimization in Engineering Design*, pages 355–368. Kluwer Academic Publishers, Dordrecht.

Smith, E. M. B. and Pantelides, C. C. (1999). A Symbolic Reformulation/Spatial Branch and Bound Algorithm for the Global Optimization of nonconvex MINLPs. *Comp. Chem. Eng.*, 23:457–478.

Spellucci, P. (1993). *Numerische Verfahren der nichtlinearen Optimierung.* Birkhäuser Verlag, ISNM, Darmstadt.

Stern, R. and Wolkowicz, H. (1995). Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. *SIAM J. Opt.*, 5(2):286–313.

Strongin, R. and Sergeyev, Y. (2000). *Global optimization with non-convex constraints: Sequential and parallel algorithms.* Kluwer Academic Publishers, Dordrecht.

Takriti, S., Birge, J. R., and Long, E. (1996). A stochastic model for unit commitment. *IEEE Transactions on Power Systems*, 11 (3):1497–1508.

Tawarmalani, M. and Sahinidis, N. (1999). Global Optimization of Mixed Integer Nonlinear Programs: A Theoretical and Computational Study. *to appear in Math. Progr.*, pages 1–70.

Tawarmalani, M. and Sahinidis, N. (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Dordrecht.

Thoai, N. (1997). A decomposition method using duality bounds for nonconvex optimization. Technical report, Research Report 97-24, University of Trier.

Törn, A. and Zilinskas, A. (1989). *Global optimization*. Lecture Notes in Computer Science 350, Springer-Verlag, Berlin.

Turkay, M. and Grossmann, I. E. (1996). Logic-based MINLP algorithms for Optimal Synthesis of Process Networks. *Comp. Chem. Eng.*, 20(8):959–978.

Vaidyanathan, R. and EL-Halwagi, M. (1996). Global optimization of nonconvex MINLP's by interval analysis. In Grossmann, I. E., editor, *Global Optimization in Engineering Design*, pages 175–193. Kluwer Academic Publishers, Dordrecht.

Van Hentenryck, P. (1997). Numerica: a modeling language for global optimization. *Proceedings of IJCAI'97*.

Van Hentenryck, P., Michel, L., and Deville, Y. (1997). *Numerica. A Modeling Language for Global Optimization*. MIT Press, Cambridge, MA.

Vandenbussche, D. (2003). *Polyhedral Approaches to Solving Nonconvex Quadratic Programs*. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology.

Vavasis, S. A. (1995). Complexity issues in global optimization: A survey. In Horst, R. and Pardalos, P., editors, *Handbook of Global Optimization*, pages 27–41. Kluwer Academic Publishers, Dordrecht.

Vazirani, V. (2001). *Approximation Algorithms*. Springer, New York.

Vecchietti, A. and Grossmann, I. E. (1999). LOGMIP: A Disjunctive 0-1 Nonlinear Optimizer for Process System Models. *Comp. Chem. Eng.*, 23:555–565.

Vecchietti, A., Lee, S., and Grossmann, I. (2003). Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. *Comp. Chem. Eng.*, pages 433–448.

Vigerske, S. (2005). *Adaptive discretization of stochastic programs*. Draft of a master thesis, Humboldt-Universität zu Berlin.

Viswanathan, J. and Grossmann, I. E. (1990). A Combined Penalty Function and Outer-Approximation Method for MINLP Optimization. *Comp. Chem. Eng.*, 14:769.

Visweswaran, V. and Floudas, C. A. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs : II. Application of theory and test problems. *Comp. Chem. Eng.*

von Stryk, O. and Glocker, M. (2000). Decomposition of mixed-integer optimal control problems using branch and bound and sparse direct collocation. In Engell, S., Kowalewski, S., and Zaytoon, J., editors, *Proc. ADPM 2000 - The 4th International Conference on Automation of Mixed Processes: Hybrid Dynamic Systems*, pages 99–104. Shaker-Verlag, Dortmund.

Warners, J. P. (1999). *Nonlinear Approaches to Satisfiability Problems.* PhD thesis, Eindhoven University of Technology.

Weintraub, A. and Vera, J. (1991). A cutting plane approach for chance-constrained linear programs. *Operations Research*, 39:776–785.

Westerlund, T. and Lundquist, K. (2003). *Alpha-ECP, Version 5.04. An Interactive MINLP-Solver Based on the Extended Cutting Plane Method.* Report 01-178-A, Process Design Laboratory, Abo Akademi University, Abo, Finland, `http://www.abo.fi./~twesterl/A-ECPManual.pdf`.

Westerlund, T. and Petterson, F. (1995). An extended cutting plane method for solving convex MINLP problems. *Comp. Chem. Eng.*, 21:131–136.

Westerlund, T., Petterson, F., and Grossmann, I. E. (1994). Optimization of pump configuration problems as a MINLP problem. *Comp. Chem. Eng.*, 18(9):845–858.

Westerlund, T., Skrifvars, H., Harjunkoski, I., and Pörn, R. (1998). An extended cutting plane method for a class of non-convex MINLP problems. *Comp. Chem. Eng.*, 22(3):357–365.

Wolkowicz, H., Saigal, R., and Vandenberghe, L. (2000). *Handbook of Semidefinite Programming.* Kluwer Academic Publishers, Dordrecht.

Zamora, J. M. and Grossmann, I. E. (1998a). Continuous Global Optimization of Structured Process Systems Models. *Comp. Chem. Eng.*, 22(12):1749–1770.

Zamora, J. M. and Grossmann, I. E. (1998b). A Global MINLP Optimization Algorithm for the Synthesis of Heat Exchanger Networks with no Stream Splits. *Comp. Chem. Eng.*, 22(3):367–384.

Zhao, X. and Luh, B. P. (2002). New bundle methods for solving lagrangian relaxation dual problems. *Journal of Optimization Theory and Applications*, 113:373–397.

Zwick, U. (1999). Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to max cut and other problems. In *Proc. of 31th STOC*, pages 679–687.

Index