Acta Numerica (2013), pp. 1–131
doi:10.1017/S0962492913000032

Mixed-integer nonlinear optimization*†

Pietro Belotti
Department of Mathematical Sciences,
Clemson University,
Clemson, SC 29634, USA

Christian Kirches
Interdisciplinary Center for Scientific Computing,
Heidelberg University,
69120 Heidelberg, Germany
and
Mathematics and Computer Science Division,
Argonne National Laboratory,
Argonne, IL 60439, USA

Sven Leyffer
Mathematics and Computer Science Division,
Argonne National Laboratory,
Argonne, IL 60439, USA

Jeff Linderoth
Department of Industrial and Systems Engineering,
University of Wisconsin–Madison,
Madison, WI 53706, USA

James Luedtke
Department of Industrial and Systems Engineering,
University of Wisconsin–Madison,
Madison, WI 53706, USA

Ashutosh Mahajan
Industrial Engineering and Operations Research,
Indian Institute of Technology Bombay,
Powai, Mumbai, MH 400076, India

* Colour online for monochrome figures available at journals.cambridge.org/anu.

† This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, US Department of Energy, under Contract DE-AC02-06CH11357.

Many optimal decision problems in scientific, engineering, and public sector applications involve both discrete decisions and nonlinear system dynamics that affect the quality of the final design or plan. These decision problems lead to mixed-integer nonlinear programming (MINLP) problems that combine the combinatorial difficulty of optimizing over discrete variable sets with the challenges of handling nonlinear functions. We review models and applications of MINLP, and survey the state of the art in methods for solving this challenging class of problems.

Most solution methods for MINLP apply some form of tree search. We distinguish two broad classes of methods: single-tree and multitree methods. We discuss these two classes of methods first in the case where the underlying problem functions are convex. Classical single-tree methods include nonlinear branch-and-bound and branch-and-cut methods, while classical multitree methods include outer approximation and Benders decomposition. The most efficient class of methods for convex MINLP are hybrid methods that combine the strengths of both classes of classical techniques.

Non-convex MINLPs pose additional challenges, because they contain non-convex functions in the objective function or the constraints; hence even when the integer variables are relaxed to be continuous, the feasible region is generally non-convex, resulting in many local minima. We discuss a range of approaches for tackling this challenging class of problems, including piecewise linear approximations, generic strategies for obtaining convex relaxations for non-convex functions, spatial branch-and-bound methods, and a small sample of techniques that exploit particular types of non-convex structures to obtain improved convex relaxations.

We finish our survey with a brief discussion of three important aspects of MINLP. First, we review heuristic techniques that can obtain good feasible solution in situations where the search-tree has grown too large or we require real-time solutions. Second, we describe an emerging area of mixed-integer optimal control that adds systems of ordinary differential equations to MINLP. Third, we survey the state of the art in software for MINLP.

CONTENTS

1. Introduction

Many optimal decision problems in scientific, engineering, and public sector applications involve both discrete decisions and nonlinear system dynamics that affect the quality of the final design or plan. Mixed-integer nonlinear programming (MINLP)[1] problems combine the combinatorial difficulty of optimizing over discrete variable sets with the challenges of handling nonlinear functions. MINLP is one of the most general modelling paradigms in optimization and includes both nonlinear programming (NLP) and mixed-integer linear programming (MILP) as subproblems. MINLPs are conveniently expressed as

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X, \\ & x_i \in \mathbb{Z},\ \forall i \in I, \end{cases} \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable functions, $X \subset \mathbb{R}^n$ is a bounded polyhedral set, and $I \subseteq \{1, \ldots, n\}$ is the index set of integer variables. We note that we can readily include maximization and more general constraints, such as equality constraints, or lower and upper bounds $l \leq c(x) \leq u$. More general discrete constraints that are not integers can be modelled by using so-called special-ordered sets of type I (Beale and Tomlin 1970, Beale and Forrest 1976).

Problem (1.1) is an NP-hard combinatorial problem, because it includes MILP (Kannan and Monma 1978), and its solution typically requires searching enormous search trees: see Figure 1.1. Worse, non-convex integer optimization problems are in general undecidable (Jeroslow 1973). Jeroslow provides an example of a quadratically constrained integer program and shows that no computing device exists that can compute the optimum for all problems in this class. In the remainder of this paper, we concentrate on the case where (1.1) is decidable, which we can achieve either by ensuring that X is compact or by assuming that the problem functions are convex.

1.1. MINLP notation and basic definitions

Throughout this paper we use $x^{(k)}$ to indicate iterates of x and $f^{(k)} = f(x^{(k)})$ to denote the evaluation of the objective at $x^{(k)}$. Similar conventions apply to constraints, gradients, or Hessians at $x^{(k)}$; for example, $\nabla f^{(k)} = \nabla f(x^{(k)})$. We use subscripts to denote components; for example, x_i is the ith component of vector x. For a set $J \subset \{1, \ldots, n\}$ we let x_J denote the components

[1] Note that we use 'MINLP' to refer to both 'mixed-integer nonlinear programming' and 'mixed-integer nonlinear program'.

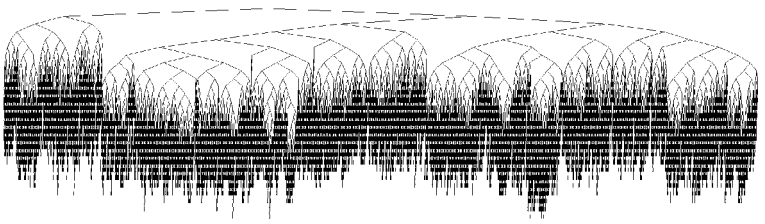


Figure 1.1. A branch-and-bound tree without presolve after 360 s CPU time has more than 10 000 nodes.

of x corresponding to J. In particular, x_I are the integer variables. We also define $C = \{1, \dots, n\} - I$ and let x_C denote the continuous variables. We denote by p the dimension of the integer space, $p = |I|$. We denote the floor and ceiling operator by $\lfloor x_i \rfloor$ and $\lceil x_i \rceil$, which denote the largest integer smaller than or equal to x_i and the smallest integer larger than or equal to x_i, respectively. Given two $n \times n$ matrices Q and X, $Q \bullet X = \sum_{i=1}^{n} \sum_{j=1}^{n} Q_{ij} X_{ij}$ represents their inner product.

In general, the presence of integer variables $x_i \in \mathbb{Z}$ implies that the feasible set of (1.1) is not convex. In a slight abuse of terminology, we distinguish convex from non-convex MINLPs.

Definition 1.1. We say that (1.1) is a *convex MINLP* if the problem functions $f(x)$ and $c(x)$ are convex functions. If either $f(x)$ or any $c_i(x)$ is a non-convex function, then we say that (1.1) is a *non-convex MINLP*.

Throughout this paper, we use the notion of a convex hull of a set S.

Definition 1.2. Given a set S, the *convex hull of* S is denoted by $\text{conv}(S)$ and defined as

$$\text{conv}(S) := \left\{ x : x = \lambda x^{(1)} + (1 - \lambda) x^{(0)},\ \forall 0 \le \lambda \le 1,\ \forall x^{(0)}, x^{(1)} \in S \right\}.$$

If $X = \{x \in \mathbb{Z}^p : l \le x \le u\}$ and $l \in \mathbb{Z}^p$, $u \in \mathbb{Z}^p$, then $\text{conv}(X) = [l, u]$ is simply the hypercube. In general, however, even when X itself is polyhedral, it is not easy to find $\text{conv}(X)$. The convex hull plays an important role in mixed-integer linear programming. Because an LP obtains a solution at a vertex, we can solve an MILP by solving an LP over its convex hull. Unfortunately, finding the convex hull of an MILP is just as hard as solving the MILP.

The same result does not hold for MINLP, as the following example illustrates:

$$\underset{x}{\text{minimize}} \sum_{i=1}^{n} (x_i - \tfrac{1}{2})^2, \quad \text{subject to } x_i \in \{0, 1\}.$$

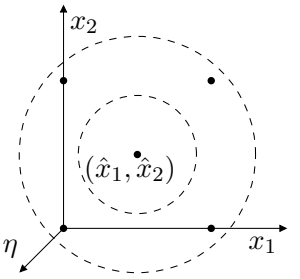


Figure 1.2. Small MINLP to illustrate the need for a linear objective function.

The solution of the continuous relaxation is $x = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$, which is not an extreme point of the feasible set and, in fact, lies in the strict interior of the MINLP: see Figure 1.2. Because the continuous minimizer lies in the interior of the convex hull of the integer feasible set, it cannot be separated from the feasible set. However, we can reformulate (1.1) by introducing an objective variable z and a constraint $z \geq f(x)$. We obtain the following equivalent MINLP:

$$\begin{cases} \underset{z,x}{\text{minimize}} & z, \\ \text{subject to} & f(x) \leq z, \\ & c(x) \leq 0, \\ & x \in X, \\ & x_i \in \mathbb{Z}, \ \forall i \in I. \end{cases} \tag{1.2}$$

The optimal solution of (1.2) always lies on the boundary of the convex hull of the feasible set and therefore allows us to use cutting-plane techniques.

1.2. Preview of key building blocks of MINLP algorithms

A wide variety of methods exists for solving MINLP. Here, we briefly introduce the two fundamental concepts underlying these algorithms: *relaxation* and *constraint enforcement*. A relaxation is used to compute a lower bound on the optimal solution of (1.1). A relaxation is obtained by enlarging the feasible set of the MINLP, for example, by ignoring some constraints of the problem. Typically, we are interested in relaxations that are substantially easier to solve than the MINLP itself. Together with upper bounds, which can be obtained from any feasible point, relaxations allow us to terminate the search for a solution whenever the lower bound is larger than the current upper bound. Constraint enforcement refers to procedures used to exclude solutions that are feasible for the relaxation but not to the original MINLP. Constraint enforcement may be accomplished by refining or tightening the

relaxation, often by adding valid inequalities, or by *branching*, where the relaxation is divided into two or more separate problems.

In general, upper bounds are obtained from any feasible point. Often, we fix the integer variables at an integral value and solve the resulting NLP to obtain an upper bound (which we set to infinity if the NLP is infeasible).

Relaxations. Formally, an optimization problem $\min\{\xi_R(x) : x \in S_R\}$ is a relaxation of a problem $\min\{\xi(x) : x \in S\}$ if (i) $S_R \supseteq S$ and (ii) $\xi_R(x) \leq \xi(x)$ for each $x \in S$. The feasible set $\mathcal{R}$ of a relaxation of a problem with feasible set $\mathcal{F}$ contains all feasible points of $\mathcal{F}$. The main role of the relaxation is to provide a problem that is easier to solve and for which we can obtain globally optimal solutions that allow us to derive a lower bound. Relaxations that fall into this category are convex NLPs, for which nonlinear optimization solvers will converge to the global minimum, and MILPs, which can often be solved efficiently (for practical purposes) by using a branch-and-cut approach.

Several strategies are used to obtain relaxations of MINLPs.

(1) *Relaxing integrality.* Integrality constraints $x_i \in \mathbb{Z}$ can be relaxed to $x_i \in \mathbb{R}$ for all $i \in I$. This procedure yields a *nonlinear relaxation* of MINLP. This type of relaxation is used in branch-and-bound algorithms (Section 3.1) and is given by

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X. \end{cases} \tag{1.3}$$

(2) *Relaxing convex constraints.* Constraints $c(x) \leq 0$ and $f(x) \leq z$ containing convex functions c and f can be relaxed with a set of supporting hyperplanes obtained from first-order Taylor series approximation,

$$z \geq f^{(k)} + \nabla f^{(k)^T}(x - x^{(k)}), \tag{1.4}$$

$$0 \geq c^{(k)} + \nabla c^{(k)^T}(x - x^{(k)}), \tag{1.5}$$

for a set of points $x^{(k)}$, $k = 1, \ldots, K$. When c and f are convex, any collection of such hyperplanes forms a *polyhedral relaxation* of these constraints. This class of relaxations is used in the outer approximation methods discussed in Section 3.2.1.

(3) *Relaxing non-convex constraints.* Constraints $c(x) \leq 0$ and $f(x) \leq z$ containing non-convex functions require more work to be relaxed. One approach is to derive convex underestimators, $\breve{f}(x)$ and $\breve{c}(x)$, which are convex functions that satisfy

$$\breve{f}(x) \leq f(x) \quad \text{and} \quad \breve{c}(x) \leq c(x), \quad \text{for all } x \in \text{conv}(X). \tag{1.6}$$

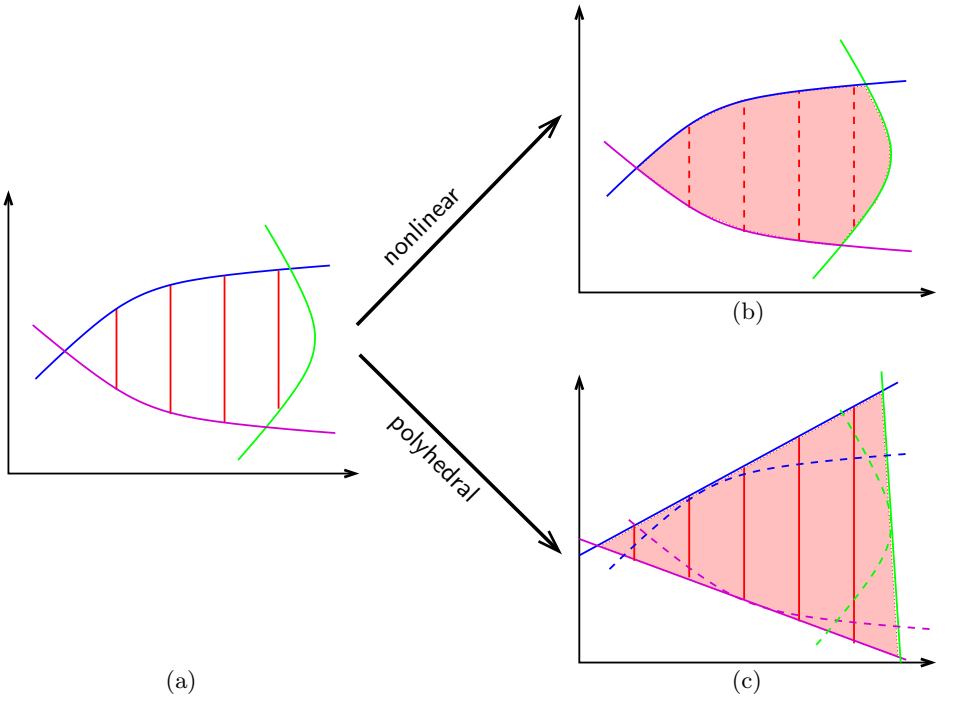


Figure 1.3. Illustration of the two classes of relaxation. (a) The mixed-integer feasible set, (b) the nonlinear relaxation, and (c) the polyhedral relaxation.

Then the constraints $z \geq f(x)$ and $0 \geq c(x)$ are relaxed by replacing them with the constraints

$$z \geq \breve{f}(x) \quad \text{and} \quad 0 \geq \breve{c}(x).$$

In Section 5 we review classes of nonlinear functions for which convex underestimators are known, and we describe a general procedure to derive underestimators for more complex nonlinear functions.

All these relaxations enlarge the feasible set of (1.2), and they can be combined with each other. For example, a convex underestimator of a nonconvex function can be further relaxed by using supporting hyperplanes, yielding a polyhedral relaxation.

Figure 1.3 illustrates the relaxation of integrality constraints and convex nonlinear constraints: (a) the mixed-integer feasible set (the union of the vertical segments), (b) the nonlinear relaxation obtained by relaxing the integrality constraints (the shaded area is the NLP feasible set), and (c) a polyhedral relaxation (the union of the vertical segments) as well as its LP relaxation (the shaded area). We note that an infinite number of

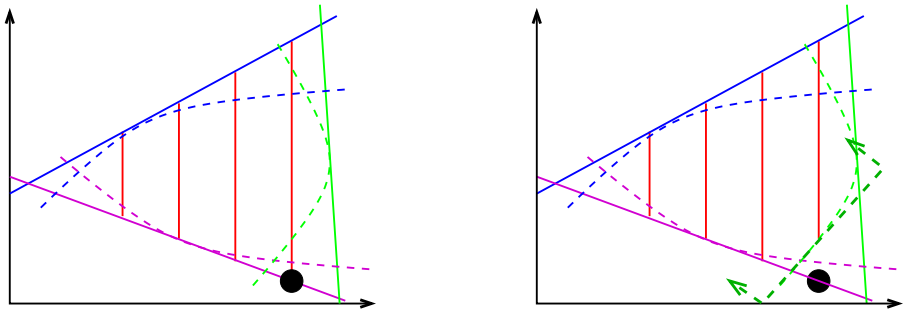


Figure 1.4. Separation of infeasible point (black dot) by adding a separating hyperplane. The dashed line with arrows (b) shows a separating hyperplane with arrows indicating the feasible side.

possible polyhedral relaxations exists, depending on the choice of the points $x^{(k)} \in \text{conv}(X),\ k = 1, \ldots, K$.

If the solution to a relaxation is feasible in (1.2), then it also solves the MINLP. In general, however, the solution is not feasible in (1.2), and we must somehow exclude this solution from the relaxation.

Constraint enforcement. Given a point $\hat{x}$ that is feasible for a relaxation but is not feasible for the MINLP, the goal of constraint enforcement is to exclude this solution, so that the algorithm can eventually converge to a solution that satisfies all the constraints. Two broad classes of constraint enforcement strategies exist: relaxation refinement and branching. Most modern MINLP algorithms use both classes.

The goal of relaxation refinement is to tighten the relaxation in such a way that a solution $\hat{x}$ of the relaxation that is infeasible for the MINLP is removed from the relaxation. Most commonly, this is achieved by adding a new *valid inequality* to the relaxation. A valid inequality is an inequality that is satisfied by all feasible solutions for the MINLP. When a valid inequality successfully excludes a given infeasible solution, it is often called a *cut*. Valid inequalities are usually linear but may be convex. For example, after relaxing a convex constraint with a polyhedral relaxation, a valid inequality can be obtained by linearizing the nonlinear functions about $\hat{x}$. This valid inequality will successfully cut off $\hat{x}$, unless $\hat{x}$ satisfies the nonlinear constraints, $c(\hat{x}) \leq 0$: see Figure 1.4. This class of separation is used in outer approximation, Benders decomposition, and the ECP algorithm discussed in Section 3.2.1. Valid inequalities can also be useful after relaxing integrality. In this case, the goal is to obtain an inequality that is valid because it does not cut off any *integer feasible* solution but will cut off an

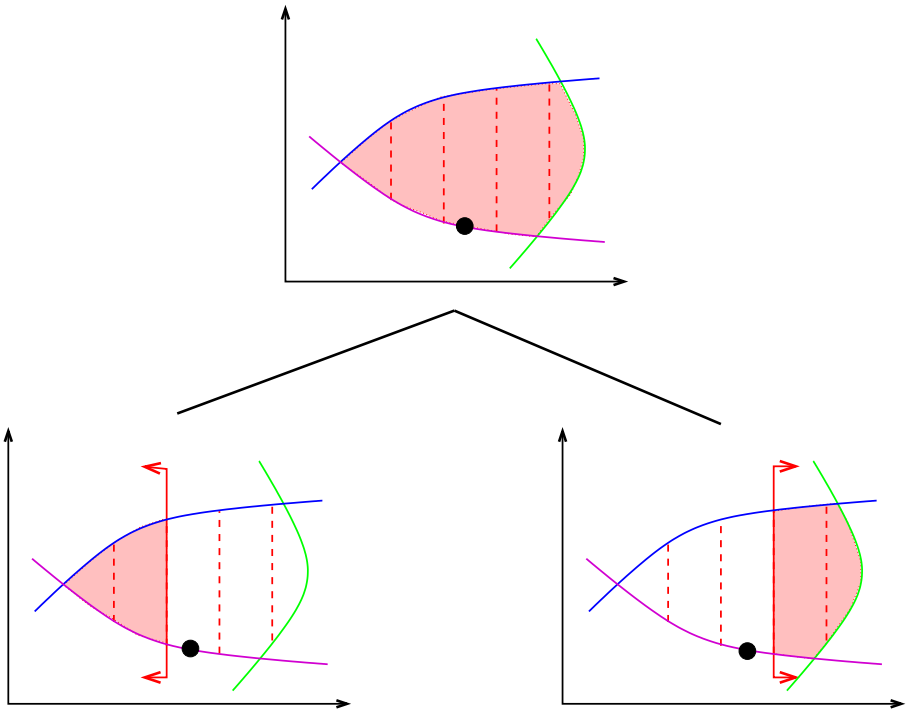


Figure 1.5. Branching on the values of an integer variable creates two new nonlinear subproblems that both exclude the infeasible point, denoted by the black dot.

integer infeasible solution $\hat{x}$. This technique has been critical to the success of algorithms for solving mixed-integer *linear* programs.

The second class of constraint enforcement strategy is branching. Branching consists in dividing the feasible region into subsets such that every solution to MINLP is feasible in one of the subsets. When integrality is relaxed, it can be enforced by branching on an integer variable that takes a fractional value $\hat{x}_i$ for some $i \in I$. Branching creates two new separate relaxations: the constraint $x_i \leq \lfloor \hat{x}_i \rfloor$ is added to the first relaxation, and the constraint $x_i \geq \lceil \hat{x}_i \rceil$ is added to the second relaxation (see Figure 1.5). All solutions of the MINLP now lie in one of these two new relaxations. The resulting subproblems are managed in a search tree that keeps track of all subproblems that remain to be solved. This approach is the basis of the branch-and-bound algorithms described in detail in Section 3.1.

Constraint enforcement for relaxed non-convex constraints involves a combination of branching and relaxation refinement. These techniques are discussed in detail in Section 5, but here we outline the general idea using Figure 1.6. Following the solution of the relaxation (given by the dashed objective function in Figure 1.6(a)), we branch on a *continuous* variable

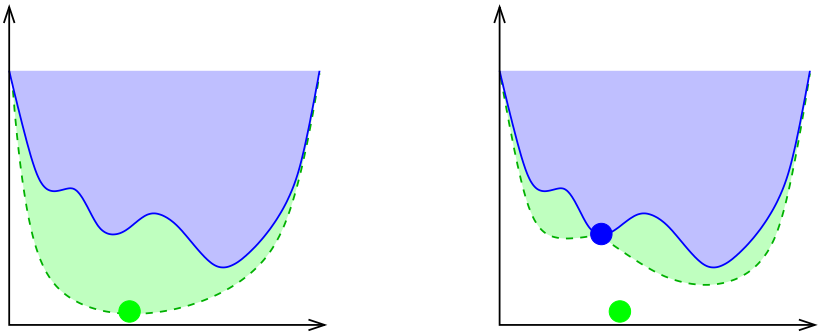


Figure 1.6. Constraint enforcement by using spatial branching for global optimization.

and hence split its domain into two subdomains. We then compute new underestimators for use in (1.6) that are valid on each of the two subdomains (*i.e.*, we refine the relaxation). In the example, these refined underestimators are indicated by the two dashed objective functions in Figure 1.6(b). This approach, which we refer to as *spatial branching*, results in a branch-and-bound algorithm similar to the one for discrete variables. We continue to divide the domain into smaller subdomains until the lower bound on a subdomain is larger than the upper bound, at which point we can exclude this domain from our search. For MINLPs having both integer variables and non-convex constraints, branching may be required on both integer and continuous decision variables.

1.3. Scope and outline

The past 20 years or so have seen a dramatic increase in new mixed-integer nonlinear models and applications, which has motivated the development of a broad range of new techniques to tackle this challenging class of problems. This survey presents a broad overview of *deterministic methodologies for solving mixed-integer nonlinear programs*. In Section 2 we motivate our interest in MINLP methods by presenting some small examples, and we briefly discuss good modelling practices. In Section 3 we present deterministic methods for *convex* MINLPs, including branch-and-bound, outer approximation, and hybrid techniques. We also discuss advanced implementation considerations for these methods. Cutting planes have long played a fundamental role in mixed-integer linear programming, and in Section 4 we discuss their extension to MINLP. We review a range of cutting planes such as conic MIR cuts, disjunctive cuts, and perspective cuts. In Section 5 we outline methods for solving *non-convex* MINLPs. A range of heuristics to obtain good incumbent solutions quickly is discussed in Section 6.

We review two classes of deterministic heuristics: search and improvement heuristics. In Section 7 we discuss an emerging extension of MINLP, the mixed-integer optimal control problem. In Section 8 we review the state of the art in software for MINLP and categorize the different solvers within the context of the previous sections.

Given the wide applicability of MINLP and the ensuing explosion of numerical methods, it would be prohibitive to discuss all methods. Instead, we focus this survey on deterministic methods that tackle both integer variables and nonlinear constraints. In particular, we do not survey the two main subproblems of MINLP, mixed-integer linear programming and nonlinear programming, for which there exist excellent textbooks (Nocedal and Wright 1999, Fletcher 1987, Wolsey 1998, Nemhauser and Wolsey 1988) and recent survey articles (Gould and Leyffer 2003). We also do not cover three other related topics.

(1) *Algorithms that are polynomial when the number of variables is fixed, or which require a polynomial number of calls to an oracle.* Lenstra's algorithm (Lenstra 1983) solves integer linear programming problems in polynomial time when the number of variables in the problem is fixed. Khachiyan and Porkolab (2000) extended this algorithm to integer programs with convex polynomial objective and constraints. Generalizations and improvements in the complexity bound have been made by Heinz (2005), De Loera, Hemmecke, Koppe and Weismantel (2006), Hildebrand and Köppe (2013) and Dadush, Peikert and Vempala (2011*d*). Using ideas from Graver bases, Hemmecke, Onn and Weismantel (2011) derive an algorithm that requires a polynomial number of *augmentation steps* to solve specially structured convex minimization problems over integer sets. Baes *et al.* (2012) investigate the problem of minimizing a strongly convex Lipschitz-continuous function over a set of integer points in a polytope, obtaining an algorithm that provides a solution with a constant approximation factor of the best solution by solving a polynomial number of specially structured quadratic integer programs.

(2) *Properties of closures of convex integer sets.* Recently, Dey and Vielma (2010) and Dadush, Dey and Vielma (2011*a*, 2011*b*, 2011*c*) have been studying *closures* of certain classes of valid inequalities for convex MINLPs. A closure is the set obtained after including all inequalities from a particular class. In particular, for the class of Gomory–Chvátal (G-C) inequalities, inequalities which can be obtained from a simple rounding argument, they have shown that the resulting closure is a rational polyhedron if the original convex set is either strictly convex or compact. This is an interesting result for two reasons: the original convex set may not be described by finitely many linear inequalities,

and also the number of G-C inequalities is not finite. In the same spirit, Dey and Morán (2013) study conditions under which the convex hull of a set of integer points in a convex set is closed, and polyhedral.

(3) *Mixed-integer derivative-free optimization problems.* Such problems arise when the problem functions are not given explicitly and can be evaluated only as the result of a (black-box) simulation $\mathcal{S}(x)$. In this case, derivatives are typically not available, and the simulation is often computationally expensive. One example of this class of problem is the design of landfill soil liners, where integer variables model the choice of material and the permeative and decomposition processes are modelled via simulation (Bartelt-Hunt *et al.* 2006). Other applications include the design of nanophotonic devices (Maria *et al.* 2009, Miller *et al.* 2010) and the determination of loop unroll factors in empirical performance tuning (Balaprakash, Wild and Hovland 2011). Algorithms for derivative-free mixed-integer nonlinear optimization employ either global surrogates (Müller 2012, Müller, Shoemaker and Piché 2013, Davis and Ierapetritou 2009, Hemker 2008, Hemker, Fowler, Farthing and von Stryk 2008, Rashid, Ambani and Cetinkaya 2013), or mesh- and pattern-based techniques (Liuzzi, Lucidi and Rinaldi 2012, Audet and Dennis, Jr 2000, Fowler *et al.* 2008, Abramson, Audet, Chrissis and Walston 2009).

We mention a number of surveys and monographs on MINLP, including a review of MINLP and disjunctive programming (Grossmann 2002), a survey of MINLP algorithms (Grossmann and Kravanja 1997), a survey of algorithms and software for MINLP (Bonami, Kılınç and Linderoth 2012), a comprehensive textbook on MINLP by Floudas (1995), and a collection of articles related to a recent IMA workshop on MINLP (Lee and Leyffer 2012).

2. Nonlinear models with integer variables

In this section we review a small number of models and modelling tricks to motivate the algorithmic developments that are discussed in the subsequent sections. The models are chosen to provide insight into the interactions of the two main modelling paradigms: integer variables and nonlinear equations. We start by presenting a well-known application from chemical engineering that models the design of a multiproduct batch plant (Kocis and Grossmann 1988, Grossmann and Sargent 1979), which can be formulated as a convex MINLP. We then present a non-convex MINLP that arises in the design of water distribution networks (Eiger, Shamir and Ben-Tal 1994, Sherali and Smith 1997, Sherali, Subramanian and Loganathan

2001, Bragalli *et al.* 2006, Bragalli *et al.* 2012). Finally, we present a time- and energy-optimal subway control example (Bock and Longman 1985) that adds time-dependent integer variables and constraints to MINLP.

2.1. Modelling practices for MINLP

Modelling plays a fundamental role in MILP (see the textbook by Williams (1999)) and is arguably more critical in MINLP, because of the additional nonlinear relationships. The nonlinearities often allow for equivalent formulations with more favourable properties. For example, in Section 2.2 we present a model of a multiproduct batch plant, and show that by using a nonlinear transformation we are able to reformulate the non-convex problem as an equivalent convex problem, which is typically easier to solve. Here, we briefly review a number of other modelling tricks that can be useful in formulating easier-to-solve problems.

Convexification of binary quadratic programs. We can always make the quadratic form in a pure binary quadratic program convex, because for any $x_i \in \{0,1\}$ it follows that $x_i^2 = x_i$. For $x \in \{0,1\}^n$ we consider the quadratic form

$$q(x) = x^T Q x + g^T x,$$

and let λ be the smallest eigenvalue of Q. If $\lambda \geq 0$, then $q(x)$ is convex. Otherwise, we define a new Hessian matrix $W := Q - \lambda I$, where I is the identity matrix, and a new gradient $c := g + \lambda e$, where $e = (1, \ldots, 1)$. It follows that

$$q(x) = x^T Q x + g^T x = x^T W x + c^T x,$$

where the second quadratic is now convex.

Exploiting low-rank Hessians. In many applications such as model structure determination and parameter estimation problems (Skrifvars, Leyffer and Westerlund 1998), the Hessian matrix is a large, dense, and low-rank matrix. In this application, the Hessian matrix can be written as $W = Z^T R^{-1} Z$, where $R \in \mathbb{R}^{m \times m}$ and $Z \in \mathbb{R}^{m \times n}$, where $m \ll n$ and Z is sparse. Forming the Hessian matrix and operating with it can be prohibitive. Some nonlinear solvers only require matrix–vector products of the Hessian and can readily exploit this situation. An alternative is to introduce additional variables z and constraints $z = Zx$, and then rewrite $x^T W x = z^T R^{-1} z$, which allows the solver to exploit the sparsity of the constraint matrix Z.

Linearization of constraints. A simple transformation is to rewrite $x_1/x_2 = a$ as $x_1 = a x_2$, where a is a constant. Special-ordered sets provide a systematic way to formulate nonlinear expressions as piecewise linear functions (see Section 5.1), and these can be effective for expressions that involve a small number of variables.

Linearization of x_1x_2, for $x_2 \in \{0,1\}$. We can linearize the expression x_1x_2, when $0 \leq x_1 \leq u$ and $x_2 \in \{0,1\}$, by observing that the product x_1x_2 is either equal to zero (if $x_2 = 0$) or equal to x_1 (if $x_2 = 1$). By introducing the new variable x_{12} and adding the constraints

$$0 \leq x_{12} \leq x_2 u \quad \text{and} \quad -u(1-x_2) \leq x_1 - x_{12} \leq u(1-x_2),$$

we can replace the non-convex term x_1x_2 by x_{12}. This trick readily generalizes to situations where $l \leq x_1 \leq u$.

Avoiding undefined nonlinear expressions. In many practical instances the MINLP solvers fail, because the nonlinear solver generates an iterate at which the nonlinear functions cannot be evaluated. An example is the constraint

$$c(x_1) = -\ln(\sin(x_1)) \leq 0,$$

which cannot be evaluated whenever $\sin(x_1) \leq 0$. Because NLP solvers typically remain feasible with respect to simple bounds, we can reformulate this constraint equivalently as

$$\tilde{c}(x_2) = -\ln(x_2) \leq 0, \quad x_2 = \sin(x_1), \quad \text{and} \quad x_2 \geq 0.$$

Interior-point methods will never evaluate the constraint $\tilde{c}(x_2)$ at a point $x_2 \leq 0$, and even active-set methods can be expected to avoid this region, because the constraint violation becomes unbounded near $x_2 = 0$. An additional benefit of this reformulation is that it reduces the 'degree of nonlinearity' of the resulting constraint set.

A common feature of these reformulation techniques is that convex formulations are typically preferred over non-convex ones and linear formulations over nonlinear formulations. More useful modelling tricks for integer and binary variables are given in the classic textbook by Williams (1999).

2.2. Design of multiproduct batch plants

Multiproduct batch plants produce a number of different products on parallel lines in a multistage batch process. In the following we use subscripts j to index the stages and subscripts i to index the products. A multiproduct batch plant is characterized by the following set of (fixed) parameters:

M the set of batch processing stages,
N the set of different products,
H the time horizon,
Q_i the required quantity of product $i \in N$,
t_{ij} processing time, product i, stage $j \in M$, and
S_{ij} size factor, product $i \in N$, stage $j \in M$.

Our goal is to find the minimum cost design by choosing optimal values for

the design (free) variables of the batch process, which are given by:

- B_i the batch size of product $i \in N$,
- V_j the size of stage $j \in M$, which must satisfy $V_j \geq S_{ij} B_i$ for all i, j,
- N_j the number of machines at stage $j \in M$, and
- C_i the longest stage time for product $i \in N$, which satisfies $C_i \geq t_{ij}/N_j$ for all i, j.

Given a set of cost parameters, $\alpha_j, \beta_j > 0$, we can formulate the minimum cost design of a multiproduct batch plant as a non-convex MINLP:

$$\begin{cases} \underset{V,C,B,N}{\text{minimize}} & \sum_{j \in M} \alpha_j N_j V_j^{\beta_j}, & \\ \text{subject to} & V_j - S_{ij} B_i \geq 0, & \forall i \in N, \forall j \in M, \\ & C_i N_j \geq t_{ij}, & \forall i \in N, \forall j \in M, \\ & \sum_{i \in N} \frac{Q_i}{B_i} C_i \leq H, & \\ & V_j \in [V_l, V_u],\ C_i \in [C_l, C_u],\ B_i \in [B_l, B_u], & \forall i \in N, \forall j \in M, \\ & N_j \in \{1, 2, \ldots, N_u\}, & \forall j \in M. \end{cases}$$

Unfortunately this model is a non-convex MINLP, because the objective function, the horizon-time constraint, and the constraint defining the longest stage time are non-convex functions. Fortunately, we can apply a variable transformation which convexifies the problem. We introduce new log-transformed variables v_j, n_j, b_i, and c_i defined by

$$v_j = \ln(V_j), \quad n_j = \ln(N_j), \quad b_i = \ln(B_i), \quad c_i = \ln C_i.$$

This transformation provides an equivalent convex reformulation of the multiproduct batch plant design problem:

$$\begin{cases} \underset{v,c,b,n}{\text{minimize}} & \sum_{j \in M} \alpha_j \mathrm{e}^{n_j + \beta_j v_j}, & \\ \text{subject to} & v_j - \ln(S_{ij}) b_i \geq 0, & \forall i \in N, \forall j \in M, \\ & c_i + n_j \geq \ln(t_{ij}), & \forall i \in N, \forall j \in M, \\ & \sum_{i \in N} Q_i \mathrm{e}^{c_i - b_i} \leq H, & \\ & v_j \in [v_l, v_u],\ c_i \in [c_l, c_u],\ b_i \in [b_l, b_u], & \forall i \in N, \forall j \in M, \\ & n_j \in \{0, \ln(2), \ldots, \ln(N_u)\}, & \forall j \in M, \end{cases}$$

where v_l, v_u, c_l, c_u, b_l, and b_u are suitably transformed bounds on the variables. The last constraint, $n_j \in \{0, \ln(2), \ldots, \ln(N_u)\}$, is difficult to enforce directly, but it can be modelled using a special-ordered set of type I (SOS-1) (Beale and Tomlin 1970). By introducing binary variables $y_{kj} \in \{0, 1\}$ we

can replace the last constraint in the convex model by

$$\sum_{k=1}^{K} \ln(k) y_{kj} = n_j \quad \text{and} \quad \sum_{k=1}^{K} y_{kj} = 1, \quad \text{for all } j \in M. \tag{2.1}$$

The resulting model is available as batch on MacMINLP (Leyffer 2003). Similar reformulations have been proposed using $\sqrt{x}$-transforms for models involving bilinear terms (Harjunkoski, Westerlund, Pörn and Skrifvars 1998).

2.3. Design of water distribution networks

Many MINLPs model flows on networks. Examples include the optimal design of water networks (Burgschweiger, Gnädig and Steinbach 2008, Bragalli *et al.* 2006, Bragalli *et al.* 2012), the design of gas networks (Wolf and Smeers 2000) and the optimal regulation of ventilation systems for mines (Wu, Topuz and Karfakis 1991, Maonan and Wenjun 1991). All applications share the same structural flow constraints. Here, we concentrate on the design of water networks. The model is defined by a set on nodes, $i \in \mathcal{N}$, and a set of arcs $(i, j) \in \mathcal{A}$ that define the (fixed) topology of the network. The goal is to determine the pipe diameter for all arcs that minimizes the installation cost, meets the demand at all nodes, and satisfies the hydraulic (flow) constraints along the arcs. The pipe diameters must be chosen from a discrete set of available diameters, giving rise to integrality constraints, while the hydraulic constraints involve nonlinear relationships, giving rise to a non-convex MINLP.

We simplify the model by assuming that certain constants are uniform throughout the network. The sets and (fixed) parameters that define the model are:

$\mathcal{N}$	the set of nodes in the network,
$\mathcal{S}$	the set of source nodes in the network, $\mathcal{S} \subset \mathcal{N}$,
$\mathcal{A}$	the set of arcs in the network, $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$,
L_{ij}	the length of pipe $(i, j) \in \mathcal{A}$,
K_{ij}	a physical constant that models the roughness of pipe $(i, j) \in \mathcal{A}$,
D_i	the demand at node $i \in \mathcal{N}$,
$V_{\max}$	the maximum velocity of flow in all pipes,
$Q_{\max}$	the maximum magnitude of flow in all pipes,
P_k	the pipe diameters available to the network, $k = 1, \ldots, r$,
H_s	the hydraulic head at source node $s \in \mathcal{S}$, and
H_l, H_u	the lower and upper bounds on the hydraulic head.

The variables of the model are:

q_{ij} the flow in pipe $(i,j) \in \mathcal{A}$ (from node i to node j),
d_{ij} the diameter of pipe $(i,j) \in \mathcal{A}$, where $d_{ij} \in \{P_1, \ldots, P_r\}$,
h_i the hydraulic head at node $i \in \mathcal{N}$, where $h_s = H_s$ for all $s \in \mathcal{S}$, and $H_l \leq h_i \leq H_u$ for all $i \in \mathcal{N} - \mathcal{S}$,
z_{ij} binary variables that model the direction of flow in pipe $(i,j) \in \mathcal{A}$,
a_{ij} the area of the cross-section of pipe $(i,j) \in \mathcal{A}$, and
y_{ijk} a set of SOS-1 variables (see (2.1)) that models the pipe type on arc $(i,j) \in \mathcal{A}$.

The conservation of flow at every node gives rise to a linear constraint

$$\sum_{(i,j)\in\mathcal{A}} q_{ij} - \sum_{(j,i)\in\mathcal{A}} q_{ji} = D_i, \quad \text{for all } i \in \mathcal{N} - \mathcal{S}.$$

Because our model contains cross-section variables, $a_{ij} = \pi d_{ij}^2/4$, we can model the bounds on the flow along an arc depending on a linear set of constraints

$$-V_{\max} a_{ij} \leq q_{ij} \leq V_{\max} a_{ij}, \quad \text{for all } (i,j) \in \mathcal{A}.$$

The choice of pipe types, $d_{ij} \in \{P_1, \ldots, P_r\}$, is modelled using the SOS-1 variables y_{ijk}, giving rise to the set of constraints

$$y_{ijk} \in \{0,1\}, \quad \text{for all } k = 1, \ldots, r, \quad \text{and}$$
$$\sum_{k=1}^{r} y_{ijk} = 1, \quad \text{and} \quad \sum_{k=1}^{r} P_k y_{ijk} = d_{ij}, \quad \text{for all } (i,j) \in \mathcal{A},$$

The MINLP solvers can now either branch on individual y_{ijk} or on SOS-1 sets $(y_{ij1}, \ldots, y_{ijr})$, which is generally more efficient. See Williams (1999) for a discussion on branching on special-ordered sets. We can use the same SOS-1 set to linearize the nonlinear equation, $a_{ij} = \pi d_{ij}^2/4$, by adding the constraints

$$\sum_{k=1}^{r} (\pi P_k^2/4) y_{ijk} = a_{ij}, \quad \text{for all } (i,j) \in \mathcal{A}.$$

The final set of constraints is an empirical model of the pressure loss along the arc $(i,j) \in \mathcal{A}$,

$$h_i - h_j = \frac{\operatorname{sgn}(q_{ij}) |q_{ij}|^{c_1} c_2 L_{ij} K_{ij}^{-c_1}}{d_{ij}^{c_3}}, \quad \text{for all } (i,j) \in \mathcal{A}, \tag{2.2}$$

where $c_1 = 1.852$, $c_2 = 10.7$, and $c_3 = 4.87$ are constants that depend on the medium of the fluid. This last equation appears to be non-smooth because

it contains terms of the form $h(x) = \operatorname{sgn}(x)|x|^{c_1}$. However, it is easy to verify that $h(x)$ is continuously differentiable, because $c_1 > 1$. To model $h(x)$, we split the flow into its positive and negative parts by adding binary variables $z_{ij} \in \{0, 1\}$ and the following set of constraints,

$$0 \le q^+_{ij} \le Q_{\max} z_{ij}, \quad 0 \le q^-_{ij} \le Q_{\max}(1 - z_{ij}), \quad q_{ij} = q^+_{ij} - q^-_{ij}.$$

An alternative formulation based on complementarity constraints avoids the introduction of the binary variables z_{ij}, and instead models the disjunction as $0 \le q^+_{ij} \perp q^-_{ij} \ge 0$, where $\perp$ means that either $q^+_{ij} = 0$ or $q^-_{ij} = 0$. With these new flow variables, we can now rewrite (2.2) equivalently as

$$h_i - h_j = \frac{\left[(q^+_{ij})^{c_1} - (q^-_{ij})^{c_1}\right] c_2 L_{ij} K_{ij}^{-c_1}}{d_{ij}^{c_3}}, \quad \text{for all } (i, j) \in \mathcal{A}.$$

Finally, we can lower the degree of nonlinearity by substituting d_{ij} in this last constraint by $d_{ij} = \sqrt{4a_{ij}/\pi}$. This substitution generally provides better Taylor-series approximations than the 'more nonlinear' version involving d_{ij}, which ensures better practical convergence behaviour of the underlying NLP solvers.

2.4. A dynamic subway operation problem

As an example of a dynamic nonlinear mixed-integer problem, we present a control problem that goes back to work by Bock and Longman (1985) optimizing the subway of the city of New York. The problem has been treated by Sager (2005), for example. We are interested in minimizing the total energy consumption

$$\underset{x(\cdot),u(\cdot),v(\cdot),T}{\text{minimize}} \int_0^T L(x(t), v(t))\, \mathrm{d}t \tag{2.3}$$

of a subway train, subject to a system of ordinary differential equations that models the dynamic behaviour of a subway train's state $x(t)$ comprising position $x_{\mathrm{s}}(t)$ and velocity $x_{\mathrm{v}}(t)$,

$$\dot{x}_{\mathrm{s}}(t) = x_{\mathrm{v}}(t), \qquad t \in [0, T], \tag{2.4}$$

$$\dot{x}_{\mathrm{v}}(t) = f_{\mathrm{v}}(x(t), u(t), v(t)), \quad t \in [0, T], \tag{2.5}$$

on a horizon $[0, T] \subset \mathbb{R}$ with free end time T. The continuous control $0 \le u(t) \le u_{\max}$ indicates the braking deceleration. The integer control vector $v(t) \in \{1, 2, 3, 4\}$ indicates the operation of the subway's two electric engines in one of four discrete modes that affect the subway train's acceleration and energy consumption, as follows.

(1) *Serial*, $v(t) = 1$. The energy consumption rate is given by

$$L(x(t),1) = \begin{cases} e\, p_1 & \text{if } x_{\rm v}(t) \le v_1, \\ e\, p_2 & \text{if } v_1 < x_{\rm v}(t) \le v_2, \\ e \sum_{i=0}^{5} c_i(1)\big(0.1\gamma x_{\rm v}(t)\big)^{-i} & \text{if } v_2 < x_{\rm v}(t), \end{cases} \tag{2.6}$$

and the subway train's dynamics are described by

$$W_{\rm eff}\, f_{\rm v}(t) = \begin{cases} g\, e\, a_1 & \text{if } x_{\rm v}(t) \le v_1, \\ g\, e\, a_2 & \text{if } v_1 < x_{\rm v}(t) \le v_2, \\ g\, (e\, F(x_{\rm v}(t),1) - R(x_{\rm v}(t))) & \text{if } v_2 < x_{\rm v}(t). \end{cases} \tag{2.7}$$

(2) *Parallel*, $v(t) = 2$. The energy consumption rate is given by

$$L(x(t),2) = \begin{cases} 0 & \text{if } x_{\rm v}(t) \le v_2, \\ e\, p_3 & \text{if } v_2 < x_{\rm v}(t) \le v_3, \\ e \sum_{i=0}^{5} c_i(2)\big(0.1\gamma x_{\rm v}(t) - 1\big)^{-i} & \text{if } v_3 < x_{\rm v}(t), \end{cases} \tag{2.8}$$

and the subway train's dynamics are described by

$$W_{\rm eff}\, f_{\rm v}(t) = \begin{cases} 0 & \text{if } x_{\rm v}(t) \le v_2, \\ g\, e\, a_3 & \text{if } v_2 < x_{\rm v}(t) \le v_3, \\ g\, (e\, F(x_{\rm v}(t),2) - R(x_{\rm v}(t))) & \text{if } v_3 < x_{\rm v}(t). \end{cases} \tag{2.9}$$

(3) *Coasting*, $v(t) = 3$. The energy consumption rate is zero, $L(x(t),3) = 0$, and the subway train's dynamics are described by

$$W_{\rm eff}\, f_{\rm v}(t) = -g\, R(x_{\rm v}(t)) - C\, W_{\rm eff}. \tag{2.10}$$

(4) *Braking*, $v(t) = 4$. The energy consumption rate is zero, $L(x(t),4) = 0$, and the subway train's dynamics are described by

$$f_{\rm v}(t) = -u(t). \tag{2.11}$$

The forces occurring in these dynamics are given by

$$R(x_{\rm v}(t)) = ca\gamma^2 x_{\rm v}(t)^2 + bW\gamma x_{\rm v}(t) + \frac{1.3}{2000}W + 116, \tag{2.12}$$

$$F(x_{\rm v},1) = \sum_{i=0}^{5} b_i(1)\big(0.1\gamma x_{\rm v}(t) - 0.3\big)^{-i}, \tag{2.13}$$

$$F(x_{\rm v},2) = \sum_{i=0}^{5} b_i(2)\big(0.1\gamma x_{\rm v}(t) - 1\big)^{-i}. \tag{2.14}$$

In addition, the system will satisfy certain path and point constraints as follows. Initial and terminal states for the system trajectory are constrained to

$$x(0) = (0,0)^T, \quad x(T) = (S,0)^T. \tag{2.15}$$

A maximum on the allowable driving time to complete the distance S is imposed,

$$T \leq T_{\max}. \tag{2.16}$$

Different scenarios can be defined for this problem by prescribing values for the parameters S and W. In addition, scenarios may include speed limits at certain points or on certain parts of the track. A description of several scenarios, units and numerical values for the model parameters a, a_1, a_2, a_3, b, $b_i(v)$, C, c, $c_i(v)$, e, g, γ, p_1, p_2, p_3, S, $T_{\max}$, $u_{\max}$, v_1, v_2, v_3 , W, W_{eff}, and analytical investigations along with numerical solutions can be found in Bock and Longman (1985) or Sager (2005), for example. This problem shows three challenging features that are typical for real-world dynamic mixed-integer problems: integer variables in time, system-state-dependent switches that need to be modelled appropriately using, for example, additional integer variables, and higher-order polynomial approximations to nonlinear and possibly non-convex system characteristics.

2.5. Summary of MINLP applications

In addition to the models discussed above, MINLPs arise in a broad range of engineering and scientific applications, including chemical, civil, electrical, nuclear, and communication engineering, as well as emerging scientific applications in the design of nanophotonic materials and devices.

Electrical engineering applications of MINLPs include the efficient management of electricity transmission (Bacher 1997, Momoh *et al.* 1997), transmission expansion (Romero, Monticelli, Garcia and Haffner 2002, Garver 1997), transmission switching (Bartholomew, O'Neill and Ferris 2008, Hedman, O'Neill, Fisher and Oren 2008), and contingency analysis, and blackout prevention of electric power systems (Bienstock and Mattia 2007, Donde *et al.* 2005).

MINLPs also arise in many chemical engineering applications, such as the design of water (Bragalli *et al.* 2006, Karuppiah and Grossmann 2006) and gas (Martin, Möller and Moritz 2006) distribution networks, the minimization of the environmental impact of utility plants (Eliceche, Corvalán and Martínez 2007), the integrated design and control of chemical processes (Flores-Tlacuahuac and Biegler 2007), block layout design in the manufacturing and service sectors (Castillo, Westerlund, Emet and Westerlund 2005), and in the optimization of polymerization plants (Prata, Oldenburg, Kroll and Marquardt 2008). A related area is systems biology, where topics

such as circadian rhythms (Shaik, Sager, Slaby and Lebiedz 2008) and protein folding (Klepeis and Floudas 2003) are addressed using mixed-integer optimization.

Applications in communication engineering and computer science include the optimal response to a cyber attack (Goldberg, Leyffer and Safro 2012, Altunay, Leyffer, Linderoth and Xie 2011), wireless bandwidth allocation (Bhatia, Segall and Zussman 2006, Sheikh and Ghafoor 2010, Costa-Montenegro, González-Castaño, Rodriguez-Hernández and Burguillo-Rial 2007), selective filtering (Sinha, Yener and Yates 2002, Soleimanipour, Zhuang and Freeman 2002), optical network performance optimization (Elwalid, Mitra and Wang 2006), network design with queuing delay constraints (Boorstyn and Frank 1977), and network design topology (Bertsekas and Gallager 1987, Chi, Jiang, Horiguchi and Guo 2008), multi-vehicle swarm communication network optimization (Abichandani, Benson and Kam 2008), the design of optimal paths (*i.e.*, minimum time) for robotic arms (Gentilini, Margot and Shimada 2013), the synthesis of periodic waveforms by tripolar pulse codes (Callegari, Bizzarri, Rovatti and Setti 2010), and the solution of MILP under uncertainty of the parameters through robust optimization (Ben-Tal and Nemirovski 1995).

Other engineering applications include the operational reloading of nuclear reactors (Quist *et al.* 1998), the optimal response to catastrophic oil spills such as the recent Deepwater oil spill in the Gulf of Mexico (You and Leyffer 2010, 2011), the design of load-bearing thermal insulation system for the Large Hadron Collider (Abramson 2004, Abhishek, Leyffer and Linderoth 2010), concrete structure design (Guerra, Newman and Leyffer 2011), and the design and operation of drinking water networks (Burgschweiger *et al.* 2008), gas networks (Martin *et al.* 2006), electric distribution networks (Lakhera, Shanbhag and McInerney 2011), and mining networks (Pruitt, Leyffer, Newman and Braun 2012). Applications in traffic modelling and optimization are found in Fügenschuh, Herty, Klar and Martin (2006), and Soler, Olivares, Staffetti and Bonami (2011) consider a flight path optimization problem. An important financial application of MINLP arises in portfolio optimization (Bienstock 1996, Jobst, Horniman, Lucas and Mitra 2001).

MINLP models can also be used for stochastic service system design problems (Elhedhli 2006), since performance metrics are often nonlinear in the decision variables.

Another developing application area of both game theory and optimization is resource allocation for homeland security; see, *e.g.*, Bier (2005), Sandler and Arce M (2003) and Zhuang and Bier (2007*b*). Simple versions of these models (involving, for example, a single target and attacker) have closed-form optimal (equilibrium) solutions; see *e.g.*, Powell (2007), Sandler and Siqueira (2006) and Zhuang and Bier (2007*a*). In more realistic models,

however, the best defensive strategy must be computed. Such models often have important binary choices, including which targets to assign a high priority in defending (Bier, Nagaraj and Abhichandani 2005, Bier, Oliveros and Samuelson 2007) and which strategies to employ in defending a target (*e.g.*, whether to attempt to deceive attackers about the level of resources invested to defend a given target: see Zhuang and Bier 2010, Zhuang 2008). Moreover, the utility functions of the attackers and defenders in these models are highly nonlinear. Powerful MINLP solvers are expected to provide decision support in this area.

An emerging area with challenging MINLP tasks is human decision analysis and support in complex problem solving (Engelhart, Funke and Sager 2013).

A special domain of MINLP is dynamic problems constrained by ordinary differential equations (ODEs) or differential-algebraic equations (DAEs), often called mixed-integer optimal control problems (MIOCPs). One of the earliest practical problems was the optimal switched operation of the New York subway (Bock and Longman 1985). Recent applications include gear shifts in automotive control (Gerdts 2005, Kirches, Sager, Bock and Schlöder 2010), automated cruise controllers (Terwen, Back and Krebs 2004, Hellström, Ivarsson, Aslund and Nielsen 2009, Kirches 2011), superstructure detection in simulated moving bed processes (Sager *et al.* 2007), and the optimization of batch distillation processes (Oldenburg, Marquardt, Heinz and Leineweber 2003).

3. Deterministic methods for convex MINLP

In general, we resolve the integrality constraints using some form of tree search strategy. MINLPs pose the additional challenge of having nonlinear functions. Consequently, two broad classes of methods for solving (1.1) exist: single-tree methods and multitree methods. In this section we concentrate on methods for convex objective functions and convex constraints. See Section 5 for a discussion of methods for non-convex MINLPs. Throughout this section, we make the following assumptions.

Assumption 3.1. Consider problem (1.1) and assume the following.

- **A1** The set X is a bounded polyhedral set.
- **A2** The functions f and c are twice continuously differentiable convex functions.
- **A3** Problem (1.1) satisfies a constraint qualification for every point in the convex hull of the feasible set of (1.1).

The most restrictive assumption is the convexity assumption A2. Assumption A1 is rather mild, and A3 is technical. We do not specify the

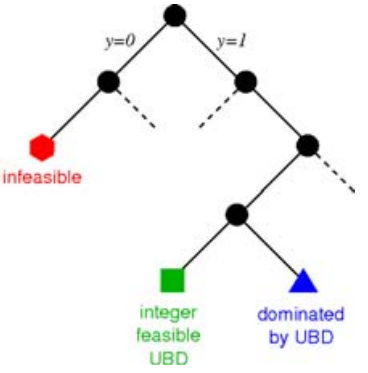


Figure 3.1. Illustration of a nonlinear branch-and-bound algorithm that traverses the tree by solving NLPs at every node of the tree.

particular constraint qualification; it is needed merely to ensure the existence of multipliers and the convergence of the NLP solvers. We note that if we assume the existence of a strictly interior feasible point for (1.1), then A3 follows from A2 as a consequence of Slater's constraint qualification (Griva, Nash and Sofer 2009), which is one of the strongest constraint qualifications.

We start by discussing nonlinear branch-and-bound, whose basic concepts underpin both classes of methods and which is a first example of a single-tree method.

3.1. Nonlinear branch-and-bound

The nonlinear branch-and-bound method for MINLPs dates back to Dakin (1965); see also Gupta and Ravindran (1985). The algorithm starts by solving the NLP relaxation of (1.1), the root node, defined by relaxing the integrality conditions on the integer variables, $x_i, i \in I$. If this relaxation is infeasible, then MINLP is also infeasible. If the solution of the relaxation is integer, then it also solves the MINLP. Otherwise, branch-and-bound searches a tree whose nodes correspond to NLP subproblems, and whose edges correspond to branching decisions. We use both optimality and feasibility of NLP subproblems to prune nodes in the tree. We define a node problem and then define the branching and pruning operations before stating the algorithm, which is also illustrated in Figure 3.1.

A node in the branch-and-bound tree is uniquely defined by a set of bounds, (l, u), on the integer variables and corresponds to the NLP

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X, \\ & l_i \leq x_i \leq u_i, \ \forall i \in I. \end{cases} \qquad (\text{NLP}(l, u))$$

We note that the root node relaxation corresponds to NLP$(-\infty, \infty)$. Next, we describe the branching and pruning rules for branch-and-bound.

Branching. If the solution x' of (NLP(l, u)) is feasible but not integral, then we branch on any non-integral variable, say x'_i. Branching introduces two new NLP nodes, also referred to as child nodes of (NLP(l, u)). In particular, we initialize bounds for two new problems as $(l^-, u^-) := (l, u)$ and $(l^+, u^+) := (l, u)$ and then modify the bound corresponding to the branching variable

$$u_i^- := \lfloor x'_i \rfloor \quad \text{and} \quad l_i^+ := \lceil x'_i \rceil. \tag{3.1}$$

The new NLP problems are then defined as NLP(l^-, u^-) and NLP(l^+, u^+). In practice, the new problems are stored on a heap $\mathcal{H}$, which is updated with these two new problems.

Pruning rules. The pruning rules for NLP branch-and-bound are based on optimality and feasibility of NLP subproblems. We let U be an upper bound on the optimal value of (1.1) (initialized as $U = \infty$).

- *Infeasible nodes.* If any node (NLP(l, u)) is infeasible, then any problem in the subtree rooted at this node is also infeasible. Thus, we can prune infeasible nodes (indicated by a hexagon in Figure 3.1).
- *Integer feasible nodes.* If the solution $x^{(l,u)}$ of (NLP(l, u)) is integral, then we obtain a new incumbent solution if $f(x^{(l,u)}) < U$, and we set $x^* = x^{(l,u)}$ and $U = f(x^{(l,u)})$. Otherwise, we prune the node because its solution is dominated by the upper bound.
- *Upper bounds on NLP nodes.* If the optimal value of (NLP(l, u)), $f(x^{(l,u)})$ (or in fact a lower bound on the optimal value), is dominated by the upper bound, that is, if $f(x^{(l,u)}) \geq U$, then we can prune this node because there cannot be any better integer solution in the subtree rooted at (NLP(l, u)).

The complete nonlinear branch-and-bound algorithm is described in Algorithm 3.1, and Proposition 3.2 establishes its convergence.

Proposition 3.2. Consider solving (1.1) by nonlinear branch-and-bound. Assume that the problem functions f and c are convex and twice continuously differentiable and that X is a bounded polyhedral set. Then it follows that branch-and-bound terminates at an optimal solution after searching a finite number of nodes or with an indication that (1.1) has no solution.

Proof. The assumptions in Proposition 3.1 ensure that every NLP node can be solved to global optimality. In addition, the boundedness of X ensures that the search tree is finite. The proof now follows similarly to the proof for MILP branch-and-bound; see, for example, Theorem 24.1 of Schrijver (1986). □

Branch-and-bound for MINLP
Choose a tolerance $\epsilon > 0$, set $U = \infty$, and initialize the heap of open problems $\mathcal{H} = \emptyset$.
Add (NLP$(-\infty, \infty)$) to the heap: $\mathcal{H} = \mathcal{H} \cup \{\text{NLP}(-\infty, \infty)\}$.
while $\mathcal{H} \neq \emptyset$ **do**
  Remove a problem (NLP(l, u)) from the heap:
  $\mathcal{H} = \mathcal{H} - \{\text{NLP}(l, u)\}$.
  Solve (NLP(l, u)) and let its solution be $x^{(l,u)}$.
  if (NLP(l, u)) *is infeasible* **then**
    Node can be pruned because it is infeasible.
  else if $f(x^{(l,u)}) > U$ **then**
    Node can be pruned, because it is dominated by upper bound.
  else if $x_I^{(l,u)}$ *integral* **then**
    Update incumbent solution: $U = f(x^{(l,u)})$, $x^* = x^{(l,u)}$.
  else
    BranchOnVariable$(x_i^{(l,u)}, l, u, \mathcal{H})$; see Algorithm 3.2.

Algorithm 3.1. Branch-and-bound for MINLP.

Subroutine: $\mathcal{S} \leftarrow$ **BranchOnVariable** $(x_i^{(l,u)}, l, u, \mathcal{H})$
// Branch on a fractional $x_i^{(l,u)}$ *for* $i \in I$
Set $u_i^- = \lfloor x_i^{(l,u)} \rfloor$, $l^- = l$ and $l_i^+ = \lceil x_i^{(l,u)} \rceil$, $u^+ = u$.
Add NLP(l^-, u^-) and NLP(l^+, u^+) to the heap:
$\mathcal{H} = \mathcal{H} \cup \{\text{NLP}(l^-, u^-), \text{NLP}(l^+, u^+)\}$.

Algorithm 3.2. Branch on a fractional variable.

We note that the convergence analysis of nonlinear branch-and-bound requires us only to ensure that every node that is pruned is solved to global optimality. The convexity assumptions are one convenient sufficient condition that ensures global optimality, but clearly not the only one!

Our description of branch-and-bound leaves open a number of important questions. In particular, two important strategic decisions concern the selection of branching variable and the next problem to be solved. Details of these important algorithmic decisions can be found in Achterberg, Koch and Martin (2004) for the MILP case, and in Bonami, Lee, Leyffer and Wächter (2011) for the MINLP case. It turns out that most observations generalize from the MILP to the MINLP case. In particular, branching decisions are best based on estimates of the effect of branching on a particular variables.

On the other hand, a depth-first search is typically preferred initially in order to obtain a good incumbent solution, after which most solvers switch to a best-estimate node selection strategy. In the next two sections we discuss the choice of branching variable and search strategy, before discussing other implementation considerations and presenting a generic branch-and-cut approach for convex MINLPs.

3.1.1. Selection of branching variable

Choosing a good branching variable is a crucial component of branch-and-bound. Ideally, we would like to choose the sequence of branching variables that minimizes the size of the tree that we need to search. Doing so, however, is impractical, because the sequence of branching variables is not known *a priori*. A more achievable goal in selecting a branching variable is to choose a variable that maximizes the increase in the lower bound at a node.

We denote by $I_c \subset I$ the set of all candidate branching variables at a particular node in the branch-and-bound tree. For example, I_c could be the index set of all fractional integer variables or a subset chosen according to some user-defined priorities (see, *e.g.*, Williams 1999). A simple branching rule is to select the variable with the largest integer violation for branching, in other words, choose

$$\arg\max_{i \in I_c} \{\min(x_i - \lfloor x_i \rfloor,\ \lceil x_i \rceil - x_i)\},$$

which is known as *maximum fractional branching*. In practice, however, this branching rule is not efficient: it performs about as well as randomly selecting a branching variable (Achterberg *et al.* 2004).

The most successful branching rules estimate the change in the lower bound after branching. Because we prune a node of the branch-and-bound tree whenever the lower bound for the node is above the current upper bound, we want to increase the lower bound as much as possible. For every integer variable $x_i, i \in I$, we define degradation estimates D_i^+ and D_i^- for the increase in the lower bound value after branching up and down on x_i, respectively. A reasonable choice would be to select the variable for which both D_i^+ and D_i^- are large. We combine the up and down degradations D_i^+ and D_i^- to compute a score for each candidate branching variable; the variable of highest score is selected. A common formula for computing this score is

$$s_i := \mu \min(D_i^+, D_i^-) + (1 - \mu) \max(D_i^+, D_i^-),$$

where $\mu \in [0, 1]$ is a prescribed parameter typically close to 1. We then select the branching variable that maximizes s_i:

$$\arg\max_{i \in I_c} \{s_i\}.$$

We next describe two methods for estimating D_i^+ and D_i^- and show how they can be combined.

Strong branching computes the degradations D_i^+ and D_i^- by solving both child nodes for all branching candidates, $x_i, i \in I_c$, which requires the solution of $2 \times |I_c|$ NLPs. To this end, we let the solution to the current node, (NLP(l, u)), be $f^{(l,u)}$. For every branching candidate $x_i, i \in I_c$ we create two temporary branching problems, NLP$_i(l^-, u^-)$ and NLP$_i(l^+, u^+)$, as in (3.1) and solve them. If both NLPs are infeasible for an index i, then we can prune the parent node (NLP(l, u)); if one of them is infeasible, then we can tighten that integer variable in the parent node and re-solve it; otherwise, we let the solution be f_i^+ and f_i^-, respectively. Given the values f_i^+ and f_i^-, we compute D_i^+ and D_i^- as

$$D_i^+ = f_i^+ - f \quad \text{and} \quad D_i^- = f_i^- - f.$$

Strong branching can significantly reduce the number of nodes in a branch-and-bound tree, but it is often slow overall because of the added computational cost of solving two NLP subproblems for each fractional variable. In order to reduce the computational cost of strong branching, it is often efficient to solve the subproblems only approximately. If the relaxation is an LP, as in the case of LP/NLP-BB (see Section 3.3.1), then one can limit the number of pivots. In the NLP case, we can limit the number of iterations, but that does not reduce the solve time sufficiently. Instead, we can use approximations of the NLP, which can be warm-started much faster than NLPs can. One approach that has been shown to be efficient is to use the basis information from a quadratic program solved at the parent node, and perform a limited number of pivots on this quadratic approximation in order to obtain estimates for f_i^+ and f_i^-, respectively (Bonami *et al.* 2011).

Pseudocost branching keeps a history of the results of past branching decisions for every variable and computes the degradations D_i^+ and D_i^- by averaging the increase in the objective value over the history. For every integer variable, we let n_i^+, n_i^- denote the number of times we have solved the up/down node for variable i. We update the per unit change in the objective when branching on x_i by computing the pseudocost, p_i^+, p_i^-, whenever we solve an up or down child node:

$$\begin{aligned} p_i^+ &= \frac{f_i^+ - f}{\lceil x_i \rceil - x_i} + p_i^+, \quad n_i^+ = n_i^+ + 1 \quad \text{or} \\ p_i^- &= \frac{f_i^- - f}{x_i - \lfloor x_i \rfloor} + p_i^-, \quad n_i^- = n_i^- + 1. \end{aligned} \tag{3.2}$$

The pseudocosts are then used to estimate D_i^+ and D_i^- whenever we need to make a branching decision, *i.e.*,

$$D_i^+ = (\lceil x_i \rceil - x_i)\frac{p_i^+}{n_i^+} \quad \text{and} \quad D_i^- = (x_i - \lfloor x_i \rfloor)\frac{p_i^-}{n_i^-}.$$

Pseudocosts are typically initialized by using strong branching. The update of the pseudocosts is relatively cheap compared with that of strong branching, because the solutions of the parent and child node are available. Statistical experience on MILP problems has shown that pseudocosts are reasonable estimates of the degradation in the objective after branching (Linderoth and Savelsbergh 1999). One difficulty that arises with pseudocosts, however, is how to update the pseudocosts if the NLP is infeasible. Typically the update is skipped in this case, but a fruitful alternative motivated by NLP might be to use the value of the ℓ_1 exact penalty functions (Fletcher 1987, Chapter 12.3), which is readily available at the end of the NLP solve.

Reliability branching is an attractive hybrid between strong branching and pseudocost branching. It performs strong branching on a variable and updates the pseudocosts until n_i^+ or n_i^- are greater than a threshold τ (a typical value for τ is 5). This corresponds to using strong branching early during the tree search, when branching decisions are more critical, and then switching to pseudocost branching once reliable estimates are available.

Branching on general disjunctions is an alternative to branching on a variable. We can branch in many ways. One popular way is to branch on special-ordered sets (Beale and Tomlin 1970). A more general approach is to branch on split disjunctions of the form

$$\left(a^T x_I \leq b\right) \vee \left(a^T x_I \leq b + 1\right), \tag{3.3}$$

where $a \in \mathbb{Z}^p$ and $b \in \mathbb{Z}$. Split disjunctions have been shown to produce promising results for MILP (Jobst *et al.* 2001) but have not been used in MINLP.

3.1.2. Node selection strategies

The second important strategic decision is which node should be solved next. The goal of this strategy is to find a good feasible solution quickly in order to reduce the upper bound, and to prove optimality of the current incumbent x^* by increasing the lower bound as quickly as possible. We introduce two popular strategies, *depth-first search* and *best-bound search*, and discuss their strengths and weaknesses. We also present two hybrid schemes that aim to overcome the weaknesses of these two strategies.

Depth-first search selects the deepest node in the tree (or the last node that was added to $\mathcal{H}$). One advantage of this strategy is that it keeps the list of open nodes, $\mathcal{H}$, as small as possible. This property made it a popular strategy for the earliest implementations of branch-and-bound (Dakin 1965). Another advantage is that this strategy minimizes the change to subsequent NLP relaxations (NLP(l, u)) that are solved, because only a single bound is changed. This fact allows us to exploit warm-start techniques that re-use the existing basis factors in MILP problems, or make use of a good starting point in MINLP problems. Though some attempts have been made to use warm-starts in MINLP (see Bonami *et al.* 2011 and Mahajan, Leyffer and Kirches 2012), they have generally not been as successful in MINLP, mainly because the Jacobian and Hessian matrices change from one node to another so that factorizations are always outdated. Unfortunately, depth-first search can exhibit extremely poor performance if no upper bound is found, exploring many nodes with a lower bound that is larger than the solution.

Best-bound search selects the node with the best lower bound. Its advantage is that for a fixed sequence of branching decisions it minimizes the number of nodes that are explored, because all nodes that are explored would have been explored independently of the upper bound. On the other hand, the weaknesses of this strategy are that it may require significantly more memory to store the open problems, the sequence of NLP subproblems does not readily allow for warm-starts, and it usually does not find an integer feasible solution before the end of the search. This last point is particularly relevant for very large problems or if the solution time is limited such as in real-time applications, because best-bound search may fail to produce even a feasible point. Like depth-first search, this strategy has also been used since the very first branch-and-bound algorithms (Land and Doig 1960, Lawler and Woods 1966).

Variants of best-bound search. Two variants of best-bound search have been proposed. Both try to estimate the effect of the branching on the bound by using pseudocosts. We let f_p denote the lower bound or estimate of problem p on the heap.

(1) *Best expected bound* selects the node with the best expected bound after branching, which is estimated as

$$b_p^+ = f_p + (\lceil x_i \rceil - x_i)\frac{p_i^+}{n_i^+} \quad \text{and} \quad b_p^- = f_p + (x_i - \lfloor x_i \rfloor)\frac{p_i^-}{n_i^-}.$$

The next node is selected as $\max_p\{\min(b_p^+, b_p^-)\}$.

(2) *Best estimate* chooses the node that is expected to contain the best expected integer solution within its subtree based on pseudocosts. The

best expected solution within a subtree can be estimated as

$$e_p = f_p + \sum_{i:x_i \text{fractional}} \min\left((\lceil x_i \rceil - x_i)\frac{p_i^+}{n_i^+} \, , \, (x_i - \lfloor x_i \rfloor)\frac{p_i^-}{n_i^-} \right),$$

namely, by adding the pseudocost estimates for *all* non-integral integer variables to the lower bound at that node. The next node to be solved is then chosen as $\max_p\{e_p\}$.

Both strategies have been reasonably successful in the context of MINLP (Goux and Leyffer 2003).

Hybrid search strategies. Good search strategies try to combine depth-first and best-bound search. Two such strategies are the two-phase method and the diving method (Eckstein 1994, Linderoth and Savelsbergh 1999, Achterberg 2005).

(1) *Two-phase methods* start with depth-first search until one or a small number of integer solutions have been found. It then switches to best-bound search in order to prove optimality. If the tree becomes too large during this second phase, then the method switches back to depth-first search in order to keep the number of open nodes manageable.

(2) *Diving methods* are also two-phase methods. The method starts with depth-first search until a leaf node (feasible or infeasible) is found. It then backtracks to the best bound on the tree to start another depth-first dive. Diving methods continue to alternate between this diving step and the backtracking step.

3.1.3. Other implementation considerations

Two other considerations are important when implementing an MINLP branch-and-bound solver: (1) the degree to which inexact subproblem solves can be used during the tree search and (2) the use of heuristics to find good incumbents quickly that will then reduce the size of the search tree.

Inexact NLP subproblem solves. There exists considerable freedom in how exactly NLP subproblems are solved in the tree search. In fact, at any non-leaf node of the tree, we need only to provide a branching variable. We can exploit this freedom to consider inexact solves at NLP subproblems in order to improve the performance of branch-and-bound. The first approach to using inexact NLP solves is due to Borchers and Mitchell (1994). The authors consider a sequential quadratic programming (SQP) approach to solving every NLP subproblem and interrupt SQP after every iteration to check whether SQP appears to converge to a non-integral solution. If convergence to non-integral solution is detected, the authors stop the current NLP solve

and proceed to branching on any non-integral integer variable. Note, however, that the inexact solution of an NLP subproblem does not provide a valid lower bound for the node. Hence, the authors propose occasionally solving an augmented Lagrangian dual to obtain a lower bound. This approach can be improved by combining insights from outer approximation to obtain implicit bounds: see Leyffer (2001). An alternative approach is presented by Mahajan, Leyffer and Kirches (2012). The authors search the branch-and-bound tree using a single quadratic program generated at the root node. The advantage of this approach is that quadratic programs, unlike NLPs, can be warm-started efficiently after branching by re-using factors of the basis of the parent node. The authors show how the pruning rules described above can be adapted to ensure that the algorithm guarantees a global solution to a convex MINLP. Numerical experience shows that this approach can reduce the CPU time of branch-and-bound by several orders of magnitude for some problems.

Heuristics for finding good incumbents. The bounding in Algorithm 3.1 shows that it is important to find good incumbents quickly in order to prune more parts of the tree. A range of heuristics exists that can be used at the root node or at intermediate nodes of the tree; these methods are discussed in Section 6.

3.1.4. Cutting planes for nonlinear branch-and-bound

Branch-and-bound algorithms can be enhanced by adding cutting planes, as described by Stubbs and Mehrotra (1999) for convex mixed 0–1 nonlinear programs, based on Balas, Ceria and Cornuéjols (1996) for MILP. The branch-and-cut algorithm extends the branch-and-bound Algorithm 3.1 by an additional step during which one or more cutting planes may be generated and added to a node (NLP(l, u)) in order to cut off a fractional optimal solution $x^{(l,u)}$. A node is branched on only if the relaxed optimal solution remains fractional even after a prescribed number of rounds of cuts have been added or if no suitable cuts could be generated at all. The hope is that adding cutting planes will lead to a significant reduction of the tree size or will ideally remove the need for branching by producing a locally tight description of the convex hull. An outline of the branch-and-cut algorithm for MINLP is given as Algorithm 3.3.

The most significant difference from the branch-and-bound algorithm for MINLP is the generation of additional cutting planes. We rely on a generic subroutine GenerateCuts that produces a new valid inequality for the current tree node. The high-level description of this step is to solve a separation problem. Given a point $x^{(l,u)}$ with $x_j^{(l,u)} \notin \{0, 1\}$ and a feasible set $\mathcal{F}(l, u)$ associated with the current node NLP(l, u), find a vector $(\pi_0, \pi^T)^T$ such that the inequality $\pi^T x \leq \pi_0$ holds for all $x \in \mathcal{F}(l, u)$ but cuts off $x^{(l,u)}$ by

Branch-and-cut for MINLP
Choose a tolerance $\epsilon > 0$, and set $U = \infty$.
Initialize the heap of open problems $\mathcal{H} = \emptyset$.
Add (NLP$(-\infty, \infty)$) to the heap: $\mathcal{H} = \mathcal{H} \cup \{\text{NLP}(-\infty, \infty)\}$.
while $\mathcal{H} \neq \emptyset$ **do**
  Remove a problem (NLP(l, u)) from the heap:
  $\mathcal{H} = \mathcal{H} - \{\text{NLP}(l, u)\}$.
  repeat
    Solve (NLP(l, u)) and let its solution be $x^{(l,u)}$.
    if (NLP(l, u)) *is infeasible* **then**
      Node can be pruned because it is infeasible.
    else if $f(x^{(l,u)}) > U$ **then**
      Node can be pruned, because it is dominated by upper bound.
    else if $x_I^{(l,u)}$ *integral* **then**
      Update incumbent solution: $U = f(x^{(l,u)})$, $x^* = x^{(l,u)}$.
    else if *more cuts shall be generated* **then**
      GenerateCuts$(x^{(l,u)}, j)$; see Algorithm 3.4.
  until *no new cuts generated*
  if (NLP(l, u)) *not pruned and not incumbent* **then**
    BranchOnVariable$(x_j^{(l,u)}, l, u, \mathcal{H})$; see Algorithm 3.2.

Algorithm 3.3. Branch-and-cut for MINLP.

Subroutine: GenerateCuts $(x^{(l,u)}, j)$
// Generate a valid inequality that cuts off $x_j^{(l,u)} \notin \{0, 1\}$
Solve a separation problem in $x^{(l,u)}$ to obtain an inequality that cuts off $x_j^{(l,u)} \notin \{0, 1\}$ from the feasible set of (NLP(l, u)). Add this inequality to (NLP(l, u)).

Algorithm 3.4. Generate a subgradient cut by solving a separation problem.

satisfying $\pi^T x^{(l,u)} > \pi_0$. Various approaches to this problem are discussed in Section 4.

Like branch-and-bound, a number of implementation issues are open. In their original work Stubbs and Mehrotra (1999) generate cuts only at the root node. Such cuts will then be valid for the entire branch-and-bound tree and are hoped to reduce it in size. In general, cuts generated in tree nodes will be valid only for that particular subtree. Lifting procedures can

sometimes be obtained in order to make locally generated cuts also globally valid for the entire branch-and-bound tree. Implementations of branch-and-cut solvers may choose to maintain both a global and a local pool of cuts.

Depending on the type of cut generated, the separation problems may themselves be difficult to solve numerically. The performance of a branch-and-cut scheme may depend crucially on the ability to reliably and precisely solve the arising separation problems. Solver failures may reduce the number of valid cuts obtained. Care must be taken not to cut off the optimal solution because of numerical difficulties.

3.2. Multitree methods for MINLP

One drawback of nonlinear branch-and-bound is that it requires the solution of a large number of NLPs that cannot be easily hot-started (unlike MILP where we can re-use the LP basis factors after branching). This observation led to the development of another class of methods that we term multitree methods because they decompose the MINLP into an alternating sequence of NLP subproblems and MILP relaxations. Here, we review three such methods: outer approximation (Duran and Grossmann 1986, Fletcher and Leyffer 1994, Bonami *et al.* 2008), generalized Benders decomposition (Geoffrion 1972, Van Roy 1983), and the extended cutting-plane method (Westerlund and Pettersson 1995).

3.2.1. Outer approximation

We start by defining the NLP subproblem obtained by fixing the integer variables to $x_I^{(j)}$ in (1.1),

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X \quad \text{and} \quad x_I = x_I^{(j)}, \end{cases} \tag{NLP($x_I^{(j)}$)}$$

and we let its solution be $x^{(j)}$. If (NLP($x_I^{(j)}$)) is infeasible, then most NLP solvers will return a solution to a feasibility problem of the form

$$\begin{cases} \underset{x}{\text{minimize}} & \displaystyle\sum_{i \in J^{\perp}} w_i c_i^{+}(x), \\ \text{subject to} & c_i(x) \leq 0, \ i \in J, \\ & x \in X \quad \text{and} \quad x_I = x_I^{(j)}, \end{cases} \tag{F($x_I^{(j)}$)}$$

where $w_i > 0$ is a set of weights that can be chosen to reduce to ℓ_1 or ℓ_∞ norm minimization, J is a set of constraints that can be satisfied, and its complement $J^{\perp}$ is the set of infeasible constraints; see, for example, Fletcher and Leyffer (1994, 2003) and Gould, Leyffer and Toint (2004). An

optimal solution of $(\mathrm{F}(x_I^{(j)}))$ with a positive objective is a certificate that the corresponding NLP $(\mathrm{NLP}(x_I^{(j)}))$ is infeasible.

Next we consider (1.2), and observe that the convexity of f and c implies that the linearization about the solution $x^{(j)}$ of $(\mathrm{NLP}(x_I^{(j)}))$ or $(\mathrm{F}(x_I^{(j)}))$, given by

$$\eta \geq f^{(j)} + \nabla f^{(j)^T}(x - x^{(j)}) \quad \text{and} \quad 0 \geq c^{(j)} + \nabla c^{(j)^T}(x - x^{(j)}), \tag{3.4}$$

is an outer approximation of the feasible set of (1.2). We can show that if $(\mathrm{NLP}(x_I^{(j)}))$ is infeasible, then the corresponding set of outer approximations ensures that $x_I = x_I^{(j)}$ violates (3.4).

Lemma 3.3 (Fletcher and Leyffer 1994, Lemma 1). If $(\mathrm{NLP}(x_I^{(j)}))$ is infeasible and $x^{(j)}$ is an optimal solution of $(\mathrm{F}(x_I^{(j)}))$, then $x_I = x_I^{(j)}$ violates (3.4) for all $x \in X$.

Next, we define an index set of all possible feasible integer assignments:

$$\mathcal{X} := \left\{x^{(j)} \in X : x^{(j)} \text{ solves } (\mathrm{NLP}(x_I^{(j)})) \text{ or } (\mathrm{F}(x_I^{(j)}))\right\}. \tag{3.5}$$

Because X is bounded, there exists only a finite (albeit exponentially large) number of different integer points $x_I^{(j)}$, and hence $\mathcal{X}$ is a finite set. Now we can construct an MILP that is equivalent to (1.2):

$$\begin{cases} \underset{\eta,x}{\text{minimize}} & \eta, \\ \text{subject to} & \eta \geq f^{(j)} + \nabla f^{(j)^T}(x - x^{(j)}),\ \forall x^{(j)} \in \mathcal{X}, \\ & 0 \geq c^{(j)} + \nabla c^{(j)^T}(x - x^{(j)}),\ \forall x^{(j)} \in \mathcal{X}, \\ & x \in X, \\ & x_i \in \mathbb{Z},\ \forall i \in I. \end{cases} \tag{3.6}$$

We can show that (3.6) and (1.2) have the same optimal value and that any solution of (1.2) is an optimal solution of (3.6). However, the converse is not true, as the following example from Bonami *et al.* (2008) shows:

$$\underset{x}{\text{minimize}}\ x_3 \quad \text{subject to } (x_1 - \tfrac{1}{2})^2 + x_2^2 + x_3^3 \leq 1, \quad x_1 \in \mathbb{Z} \cap [-1, 2].$$

The MILP created from outer approximations contains no coefficient for x_2, because $x_2 = 0$ is optimal in all NLP subproblems. Hence, any value of x_2 is optimal in the MILP.

Theorem 3.4. Assume that the assumptions in Proposition 3.1 hold, and let x^* solve (1.1). Then it follows that x^* also solves (3.6). Conversely, if (η^*, x^*) solves (3.6), then it follows that the optimal value of (1.1) is η^* and x_I^* is an optimal solution in both problems.

Proof. The proof follows from the results by Bonami *et al.* (2008) and Fletcher and Leyffer (1994). □

Of course, it is not practical to solve (3.6) because by the time it is set up, we already know the solution of (1.1). Instead, Duran and Grossmann (1986) propose a relaxation algorithm that solves an alternating sequence of MILP problems and NLP subproblems. Initially, we solve (NLP($x_I^{(j)}$)) for a given initial point $x^{(j)} = x^{(0)}$ and set up a relaxation of (3.6) in which we replace $\mathcal{X}$ by a subset $\mathcal{X}^k \subset \mathcal{X}$, with $\mathcal{X}^k = \{0\}$. We also add an upper bound on η corresponding to the best solution found so far:

$$\eta < U^k := \min_{j \le k} \{ f^{(j)} \mid (\text{NLP}(x_I^{(j)})) \text{ is feasible} \}.$$

We note, however, that this latter constraint is not enforceable in practice, and is typically replaced by $\eta \le U^k - \epsilon$, where $\epsilon > 0$ is a small tolerance. This upper bound ensures that once we have solved (NLP(l, u)) and added its outer approximations (3.4) to ($M(\mathcal{X}^k)$), $x_i^{(j)}$ is not feasible in ($M(\mathcal{X}^k)$) for $k \ge j$, ensuring that outer approximation terminates finitely. Thus, the MILP master problem solved at iteration k is given by

$$\left\{ \begin{array}{ll} \underset{\eta, x}{\text{minimize}} & \eta, \\ \text{subject to} & \eta \le U^k - \epsilon, \\ & \eta \ge f^{(j)} + \nabla f^{(j)^T}(x - x^{(j)}),\ \forall x^{(j)} \in \mathcal{X}^k, \\ & 0 \ge c^{(j)} + \nabla c^{(j)^T}(x - x^{(j)}),\ \forall x^{(j)} \in \mathcal{X}^k, \\ & x \in X, \\ & x_i \in \mathbb{Z},\ \forall i \in I. \end{array} \right. \qquad (M(\mathcal{X}^k))$$

The outer approximation algorithm is described in Algorithm 3.5, and its convergence result is stated in Theorem 3.5.

The algorithm also detects whether (1.2) is infeasible. If $U^k = \infty$ on exit, then all integer assignments visited by the algorithm are infeasible, and hence (1.2) is infeasible. The use of upper bounds on η and the definition of the set and $\mathcal{X}^k$ ensure that no $x_I^{(j)}$ is replicated by the algorithm. Thus, one can prove that the algorithm terminates after a finite number of steps, provided that there is only a finite number of integer assignments.

Theorem 3.5. If Assumptions 3.1 hold and if the number of integer points in X is finite, then Algorithm 3.5 terminates in a finite number of steps at an optimal solution of (1.1) or with an indication that (1.1) is infeasible.

A proof of Theorem 3.5 was given by Fletcher and Leyffer (1994). The main argument of the proof is that the optimality of $x^{(j)}$ in (NLP($x_I^{(j)}$)) implies that $\eta \ge f^{(j)}$ for any feasible point in ($M(\mathcal{X}^k)$). The upper bound

Outer approximation
Given $x^{(0)}$, choose a tolerance $\epsilon > 0$, set $U^{-1} = \infty$, set $k = 0$, and initialize $\mathcal{X}^{-1} = \emptyset$.
repeat
  Solve (NLP($x_I^{(j)}$)) or (F($x_I^{(j)}$)) and let the solution be $x^{(j)}$.
  if (NLP($x_I^{(j)}$)) *is feasible and* $f^{(j)} < U^{k-1}$ **then**
    Update current best point: $x^* = x^{(j)}$ and $U^k = f^{(j)}$.
  else
    Set $U^k = U^{k-1}$.
  Linearize objective and constraint f and c about $x^{(j)}$ and set $\mathcal{X}^k = \mathcal{X}^{k-1} \cup \{j\}$.
  Solve ($M(\mathcal{X}^k)$), let the solution be $x^{(k+1)}$ and set $k = k + 1$.
until MILP ($M(\mathcal{X}^k)$) *is infeasible.*

Algorithm 3.5. Outer approximation.

$\eta \leq f^{(j)} - \epsilon$ therefore ensures that the choice $x_I = x_I^{(j)}$ in ($M(\mathcal{X}^k)$) is not feasible. Hence, the algorithm is finite. The optimality of the algorithm follows from the convexity of f and c, which ensures that the linearizations are supporting hyperplanes.

Worst-case complexity of outer approximation. In practice, outer approximation often works efficiently. However, Hijazi, Bonami and Ouorou (2010) provide an example where outer approximation takes an exponential number of iterations. In particular, they consider the following MINLP:

$$\underset{x}{\text{minimize}}\ 0 \quad \text{subject to} \quad \sum_{i=1}^{n}\left(x_i - \frac{1}{2}\right)^2 \leq \frac{n-1}{4}\ x \in \{0,1\}^n. \tag{3.7}$$

Geometrically, the problem corresponds to a feasibility problem that seeks a point that lies in the intersection of the n-dimensional ℓ_2 ball with radius $\frac{n-1}{4}$, centred at $\hat{x} = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right)$, with the unit hypercube $\{0,1\}^n$. Since the distance from $\hat{x}$ to any vertex of $\{0,1\}^n$ is $\frac{\sqrt{n}}{2} > \frac{n-1}{4}$, the problem is infeasible. Figure 3.2 shows a 3D plot of this example. The black lines illustrate the unit hypercube, and the spherical surface is the boundary of the nonlinear constraint in (3.7). The authors show that no outer approximation cut can cut more than one vertex. Thus, outer approximation must visit all 2^n vertices of the hypercube. The example generalizes to Benders decomposition and extended cutting-plane methods, because these methods are relaxations of outer approximation.

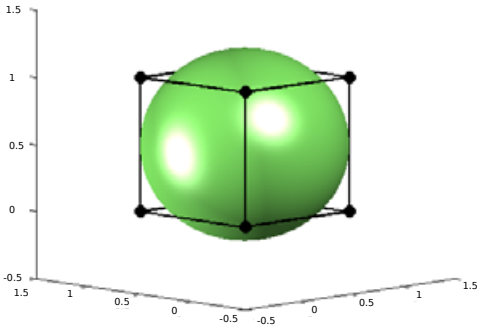


Figure 3.2. 3D plot of worst-case example of outer approximation (3.7).

3.2.2. Generalized Benders decomposition

Generalized Benders decomposition was developed before outer approximation; see Geoffrion (1972). Given outer approximation, however, it is straightforward to develop the Benders cuts and present Benders decomposition. We start by considering the outer approximations (3.4) and assume that the constraints X are inactive. Summing up the outer approximations (3.4) weighted with $(1, \lambda^{(j)})$, where $\lambda^{(j)} \geq 0$ are the multipliers of (NLP($x_I^{(j)}$)), we obtain the following valid inequality:

$$\eta \geq \left(f^{(j)} + \lambda^{(j)^T} c^{(j)}\right) + \left(\nabla f^{(j)} + \sum_{i=1}^{m} \lambda_i^{(j)} \nabla c_i^{(j)}\right)^T (x - x^{(j)}). \tag{3.8}$$

We observe that $\lambda^{(j)^T} c^{(j)} = 0$ as a result of complementary slackness and that the continuous variable component of the gradient

$$\nabla_C f^{(j)} + \sum_{i=1}^{m} \lambda_i^{(j)} \nabla_C c_i^{(j)} = 0$$

as a result of the optimality of $x^{(j)}$. Thus, we can rewrite the cut (3.8) in the integer variables only as

$$\eta \geq f^{(j)} + \left(\nabla_I f^{(j)} + \sum_{i=1}^{m} \lambda_i^{(j)} \nabla_I c_i^{(j)}\right)^T (x_I - x_I^{(j)}), \tag{3.9}$$

which is the Benders cut for feasible subproblems. We also observe that the optimality of (NLP($x_I^{(j)}$)) implies the existence of multipliers $\mu_I^{(j)}$ of the bounds $x_I = x_I^{(j)}$ and that their value is equal to the gradient in the Benders cut. Thus, we can write the Benders cut compactly as

$$\eta \geq f^{(j)} + \mu_I^{(j)^T} (x_I - x_I^{(j)}). \tag{3.10}$$

With $(\mathrm{F}(x_I^{(j)}))$, a similar derivation shows that the Benders cut for infeasible subproblems is

$$0 \geq \sum_{i \in J^{\perp}} w_i c_i^{+}(x) + \mu_I^{(j)^T}(x_I - x_I^{(j)}), \tag{3.11}$$

where $\mu_I^{(j)}$ are the multipliers of $x_I = x_I^{(j)}$ in $(\mathrm{F}(x_I^{(j)}))$.

The advantage of the Benders cuts is that they involve only the integer variables and one objective variable. A disadvantage is that the Benders cuts are almost always dense. Moreover, the Benders cuts are weaker than the outer approximation cuts from which they are derived.

3.2.3. Extended cutting-plane method

The extended cutting-plane method (Westerlund and Lundqvist 2005, Westerlund and Pettersson 1995) can be viewed as a variant of outer approximation that does not solve NLP subproblems such as $(\mathrm{NLP}(x_I^{(j)}))$ or $(\mathrm{F}(x_I^{(j)}))$. Instead, the extended cutting-plane method linearizes all functions at the solution of the MILP master problem, $x^{(k)}$. If $x^{(k)}$ satisfies all linearizations, then we have solved the MINLP. Otherwise, we choose one (or a small number of) the most violated linearizations and add it to the MILP master problem. The method alternates between the solution of the master problem and the generation of linearizations that are underestimators if the MINLP problem is convex.

Convergence of the extended cutting-plane method follows similarly to that of outer approximation. The convexity of f and c ensures that the linearizations are a separating hyperplane, and the convergence in the continuous space follows from the convergence of Kelley's cutting-plane method (Kelley 1960).

One weakness of the extended cutting-plane method is that it can produce the same integer assignment multiple times, because the cuts are not generated from solutions of $(\mathrm{NLP}(x_I^{(j)}))$ or $(\mathrm{F}(x_I^{(j)}))$. The rate of convergence of Kelley's cutting-plane method is in general linear, and hence it may require a larger number of iterations. In practice, however, the extended cutting-plane method is competitive with outer approximations, and the cutting planes it creates have been used to accelerate outer-approximation-based schemes, such as the LP/NLP-based branch-and-bound method discussed in Section 3.3.1; see for example Abhishek *et al.* (2010).

3.3. Single-tree methods for MINLP

One single-tree method was already described above, namely branch-and-bound. This section shows how we can develop hybrid or integrated approaches that use outer approximation properties but require only a single MILP tree to be searched. The advantages of these approaches are twofold. First, we avoid the need to re-solve related MILP master problems; second,

we search a tree whose nodes can be effectively warm-started by re-using basis information from parent nodes.

An alternative motivation for these hybrid techniques is to interpret them as branch-and-cut algorithms for solving the large MILP, (3.6), with the full set of linearizations $\mathcal{X}$ as in (3.5). This problem is clearly intractable, so instead we apply a delayed constraint generation technique of the 'formulation constraints' $\mathcal{X}^k \subset \mathcal{X}$. At integer solutions we can separate cuts by solving the NLP (NLP($x_I^{(j)}$)) or (F($x_I^{(j)}$)) for fixed integer variables.

3.3.1. LP/NLP-based branch-and-bound

Introduced by Quesada and Grossmann (1992), LP/NLP-based branch-and-bound (LP/NLP-BB) methods have emerged as a powerful class of algorithms for solving convex MINLPs. LP/NLP-BB improves outer approximation by avoiding the solution of multiple MILP master problems, which take an increasing amount of computing time. Moreover, since these MILP relaxations are strongly related to one another a considerable amount of information is regenerated each time a relaxation is solved.

Instead, the LP/NLP-BB algorithm solves the continuous relaxation of $(M(\mathcal{X}^k))$ and enforces integrality of the x_I variables by branching. Whenever a new integer solution is found, the tree search is interrupted to solve (NLP($x_I^{(j)}$)), and the master MILP is updated with new outer approximations generated from the solution of the subproblem. Finally the node corresponding to the integer point is re-solved, and the tree search continues.

The previous integer feasible node must be re-solved, because unlike ordinary branch-and-bound, a node cannot be pruned if it produces an integer feasible solution, since the previous solution at this node is cut out by the linearizations added to the master program. Thus, only infeasible nodes can be pruned.

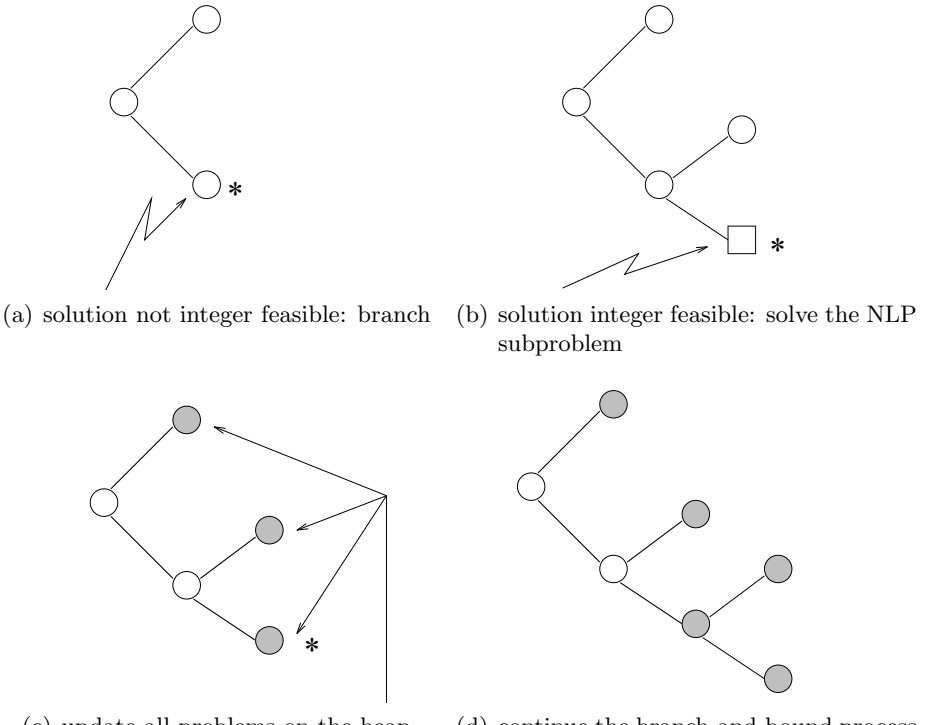
We now formally define the LP/NLP-based branch-and-bound algorithm. It can be viewed as a hybrid algorithm between nonlinear branch-and-bound (Algorithm 3.1) and outer approximation (Algorithm 3.5); see also Bonami *et al.* (2008). We denote by $(\mathrm{LP}(\mathcal{X}^k, l^i, u^i))$ the LP node (relaxation) of the MILP master problem $(M(\mathcal{X}^k))$ with bounds $l^i \leq x_I \leq u^i$. In particular, $(\mathrm{LP}(\mathcal{X}^k, -\infty, \infty))$ is the LP root node relaxation of $(M(\mathcal{X}^k))$. The algorithm uses an initial integer point $x^{(0)}$ to set up the initial master problem, but it could also solve the NLP relaxation in order to derive the initial outer approximations.

As in the outer approximation algorithms, the use of an upper bound implies that no integer assignment is generated twice during the tree search. Since both the tree and the set of integer variables are finite, the algorithm eventually encounters only infeasible problems, and the heap is thus emptied so that the procedure stops. This provides a proof of the following corollary to Theorem 3.5.

LP/NLP-based branch-and-bound
Given $x^{(0)}$, choose a tolerance $\epsilon > 0$, set $U^{-1} = \infty$, set $k = 0$, and initialize $\mathcal{X}^{-1} = \emptyset$.
Initialize MILP:
Solve (NLP($x_I^{(j)}$)) or (F($x_I^{(j)}$)) and let the solution be $x^{(j)}$.
Linearize objective and constraint f and c about $x^{(j)}$ and set $\mathcal{X}^k = \mathcal{X}^{k-1} \cup \{j\}$.
if (NLP($x_I^{(j)}$)) *is feasible* **then**
  Update current best point: $x^* = x^{(j)}$ and $U^k = f^{(j)}$.
Initialize MILP Search Tree:
Initialize the heap of open problems $\mathcal{H} = \emptyset$.
Add (LP($\mathcal{X}^k, -\infty, \infty$)) to the heap: $\mathcal{H} = \mathcal{H} \cup \{\text{LP}(\mathcal{X}^k, -\infty, \infty)\}$.
while $\mathcal{H} \neq \emptyset$ **do**
  Remove an LP problem from the heap: $\mathcal{H} = \mathcal{H} - \{\text{LP}(\mathcal{X}^k, l, u)\}$.
  Solve (LP($\mathcal{X}^k, l, u$)) and let its solution be $x^{(l,u)}$.
  if (LP($\mathcal{X}^k, l, u$)) *is infeasible* **then**
    Node can be pruned because (LP($\mathcal{X}^k, l, u$)) and hence (NLP($x_I^{(j)}$)) are infeasible.
  else if $x_I^{(l,u)}$ *integral* **then**
    Set $x_I^{(j)} = x_I^{(l,u)}$ and solve (NLP($x_I^{(j)}$)) or (F($x_I^{(j)}$)) and let the solution be $x^{(j)}$.
    Linearize objective and constraint f and c about $x^{(j)}$ and set $\mathcal{X}^k = \mathcal{X}^{k-1} \cup \{j\}$.
    if (NLP($x_I^{(j)}$)) *is feasible & $f^{(j)} < U^k$* **then**
      Update current best point: $x^* = x^{(j)}$ and $U^k = f^{(j)}$.
    else
      Set $U^k = U^{k-1}$.
    Add the LP back to the heap: $\mathcal{H} = \mathcal{H} \cup \{\text{LP}(\mathcal{X}^k, l, u)\}$.
    Set $k = k + 1$.
  else
    BranchOnVariable($x_i^{(l,u)}, l, u, \mathcal{H}$).

Algorithm 3.6. LP/NLP-based branch-and-bound.

Theorem 3.6. If Assumptions 3.1 hold, and if the number of integer points in X is finite, then Algorithm 3.6 terminates in a finite number of steps at a solution of (1.1) or with an indication that (1.1) is infeasible.



(a) solution not integer feasible: branch

(b) solution integer feasible: solve the NLP subproblem

(c) update all problems on the heap

(d) continue the branch-and-bound process

Figure 3.3. Progress of LP/NLP-based branch-and-bound.

Figure 3.3 illustrates the progress of Algorithm 3.6. In Figure 3.3(a) the LP relaxation of the initial MILP has been solved, and two branches have been added to the tree. The LP that is solved next (indicated by *) does not give an integer feasible solution, and two new branches are introduced. The next LP in Figure 3.3(b) produces an integer feasible solution indicated by a box. The corresponding NLP subproblem is solved, and in Figure 3.3(c) all nodes on the heap are updated (indicated by the shaded circles) by adding the linearizations from the NLP subproblem, including the upper bound U^k that cuts out the current assignment x_I. Then, the branch-and-bound process continues on the updated tree by solving the LP marked by *.

Algorithmic refinements of LP/NLP-BB. Abhishek *et al.* (2010) have shown that the branch-and-cut LP/NLP-BB algorithm can be improved significantly by implementing it within a modern MILP solver. Advanced MILP search and cut-management techniques dramatically improve the performance of LP/NLP-BB. It is important to generate cuts at nodes that are not integer feasible, in which case it is advantageous to generate outer

approximations around the solution of the LP relaxation, rather than solving an NLP (we term such cuts ECP cuts). We have shown that LP/NLP-BB can be improved dramatically by exploiting MILP domain knowledge, such as strong branching, adaptive node selection, and, most important, cut management. We have also observed that weaker cuts, such as linearizations generated at LP solutions, improve the performance of LP/NLP-BB.

3.3.2. Other single-tree approaches

It is straightforward to develop a single-tree version of generalized Benders decomposition and the extended cutting-plane method. In the case of Benders decomposition, we need only replace the outer approximation cuts (3.4) by the Benders cut (3.10). In fact, the Benders cuts can already be used to condense old outer approximation cuts in order to reduce the size of the LP relaxation. The extended cutting-plane method can be similarly generalized (see Still and Westerlund 2006) by replacing the NLP solver with a simple function and gradient evaluation in order to generate the outer approximations.

The convergence results are readily extended. In the case of Benders decomposition, convergence follows from the convexity and the finiteness of the set of feasible integer variables, because every integer assignment is visited at most once. In the case of the extended cutting-plane method, convergence follows from the finiteness of the integer set and the finite ϵ-convergence of the cutting-plane method.

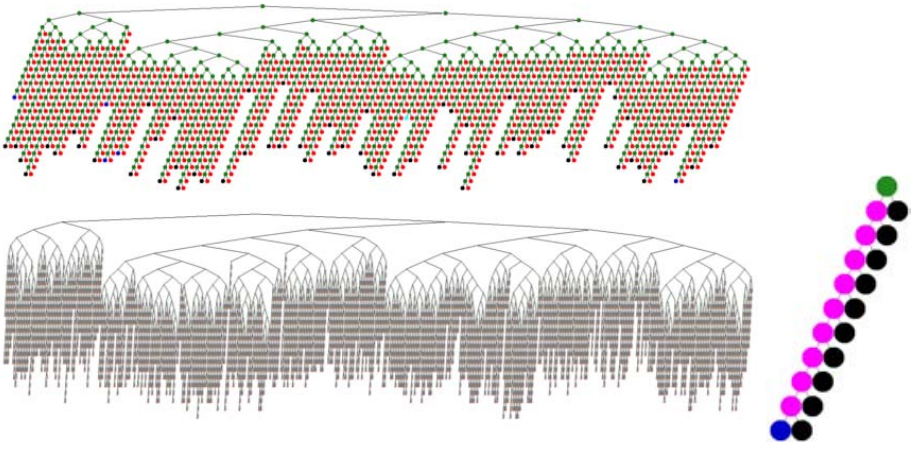
3.4. Presolve techniques for MINLP

A key component in successful MILP software is an efficient presolve. These techniques were popularized by Savelsbergh (1994), and some of these techniques can be readily extended to MINLP. Here, we briefly review these techniques and demonstrate their importance with two examples: coefficient tightening and constraint disaggregation. The goal of the presolve is to create an equivalent but tighter LP (or NLP) relaxation that will likely result in a significantly smaller search tree.

Presolve techniques fall into two broad categories: basic functions for housekeeping and advanced functions for reformulation. Housekeeping includes checking for duplicate rows (or constraints), tightening of bounds on the variables and constraints, fixing and removing variables, and identifying redundant constraints. Reformulations include improvement of coefficients, disaggregation of constraints, and derivation of implications or conflicts.

3.4.1. Coefficient tightening for MINLP

We observed very large search trees when we tried to solve certain chemical engineering synthesis problems with a range of branch-and-bound solvers.



(a) (b)

Figure 3.4. (a) Branch-and-bound tree without presolve after 75 and 200 seconds of CPU time. (b) Complete tree after application of presolve and coefficient tightening.

For example, the search tree generated by MINOTAUR (Mahajan *et al.* 2011) for problem Syn20M04M grows rapidly, as shown in Figure 3.4. Figure 3.4(a) shows the search tree after 75 and 200 seconds of CPU time (the tree after 360 seconds is shown in Figure 1.1). The problem is not solved within 2 hours of CPU time, at which point MINOTAUR has visited 264 000 nodes. Other solvers behave similarly: BONMIN-BB and MINLPBB have searched about 150 000 nodes after 2 hours of CPU time without finding the solution. On the other hand, this problem is solved in 9 seconds by using a hybrid outer approximation branch-and-bound approach (Bonami *et al.* 2008). In this section we show that we can improve the performance of branch-and-bound methods dramatically by extending coefficient tightening to MINLP.

We start by describing the principle of coefficient tightening on a simple MILP, whose feasible set is given by

$$x_1 + 21x_2 \leq 30, \quad 0 \leq x_1 \leq 14, \quad x_2 \in \{0, 1\}. \tag{3.12}$$

The feasible set is the union of the two bold horizontal segments in Figure 3.5. If $x_2 = 1$ then the constraint $x_1 \leq 30 - 21x_2 = 9$ is tight; but if $x_2 = 0$ then $x_1 \leq 30 - 21x_2 = 30$ is loose. The shaded area illustrates the feasible set of the LP relaxation, which is much larger than the convex hull of the integer feasible set. We can tighten the formulation by changing the coefficient of the binary variable x_2. It is easy to see that

$$x_1 + 5x_2 \leq 14, \quad 0 \leq x_1 \leq 14, \quad x_2 \in \{0, 1\}$$

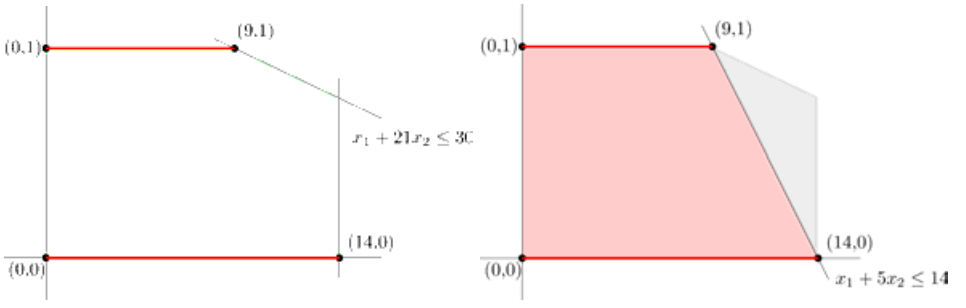


Figure 3.5. (a) Feasible set of MILP example (3.12) and (b) feasible set after coefficient tightening.

is equivalent to (3.12), and corresponds to the convex hull of the two bold horizontal segments (see Figure 3.5(b)).

We can extend this idea to MINLP problems, where we often encounter constraints of the form

$$c(x_1, \ldots, x_k) \leq b + M(1 - y), \quad y \in \{0, 1\}, \quad \text{and}$$
$$l_i \leq x_i \leq u_i, \quad \text{for all } i = 1, \ldots, k,$$

where $M > 0$ is a large constant. This form of constraints corresponds to on/off decisions that arise, for example, in process synthesis problems (Türkay and Grossmann 1996).

If the constraint $c(x_1, \ldots, x_k) \leq b + M(1 - y)$ is loose for $y = 0$, then we can reduce the value of M by an amount determined by the following optimization problem:

$$\begin{cases} \underset{x}{\text{maximize}} & c(x_1, \ldots, x_k), \\ \text{subject to} & l_i \leq x_i \leq u_i, \ \forall i = 1, \ldots, k. \end{cases} \tag{3.13}$$

Let us denote the optimal value by $c^u := c(x_1^*, \ldots, x_k^*)$. If we have $c^u < b + M$, then we can tighten the coefficient M and arrive at the equivalent formulation

$$c(x_1, \ldots, x_k) + (c^u - b)y \leq c^u, \quad y \in \{0, 1\}, \quad \text{and}$$
$$l_i \leq x_i \leq u_i, \quad \text{for all } i = 1, \ldots, k.$$

Unfortunately, this approach requires solving an NLP for every set of constraints for which we wish to tighten the formulation. Moreover, the optimization problem (3.13) is non-convex, because we maximize a convex function. Thus, we would have to apply global optimization techniques to derive the bound c^u, which would be prohibitive.

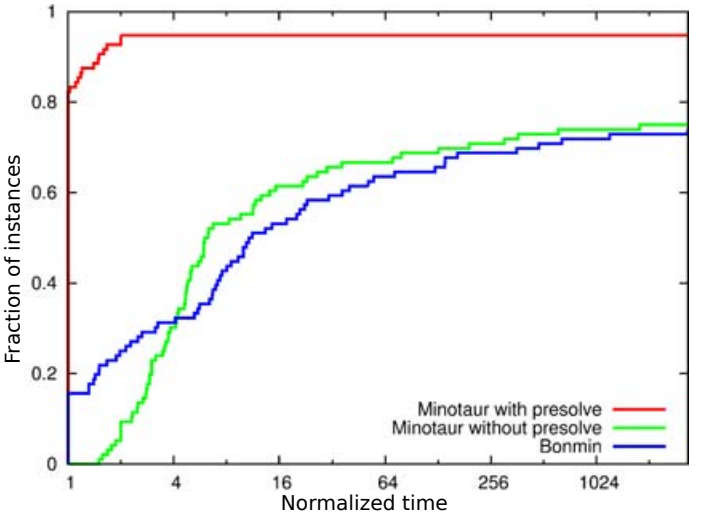


Figure 3.6. Performance profile comparing the effect of presolve on MINLP solver MINOTAUR for Syn* and Rsyn* instances.

We can avoid the solution of this non-convex NLP if the binary variable also appears as an upper bound on the variables. This is indeed the case in many applications, such as the synthesis design problem, where we find the following constraint structure:

$$\begin{gathered} c(x_1, \ldots, x_k) \leq b + M(1-y), \quad y \in \{0,1\}, \quad \text{and} \\ 0 \leq x_i \leq u_i y, \quad \text{for all } i = 1, \ldots, k, \end{gathered} \tag{3.14}$$

where $y = 0$ now switches the constraints and variables off. In this case, we can simply evaluate the constraint $c^u := c(0, \ldots, 0)$ and then derive the following tightened set of constraints (provided that $c^u < b + M$):

$$\begin{gathered} c(x_1, \ldots, x_k) + (c^u - b)y \leq c^u, \quad y \in \{0,1\}, \quad \text{and} \\ 0 \leq x_i \leq u_i y, \quad \text{for all } i = 1, \ldots, k. \end{gathered} \tag{3.15}$$

The effect of this reformulation is dramatic, as can be seen from Figure 3.4. The tree in Figure 3.4(b) shows the complete search tree from MINOTAUR with presolve, and the MINLP is now solved in 2.3 seconds. Similar improvements are obtained for other synthesis problems. The performance profile (Dolan and Moré 2002) in Figure 3.6 compares the performance of MINOTAUR's presolve on the Syn* and Rsyn* instances from the IBM/CMU library. Clearly, in almost all instances the presolve helps to improve its performance. In addition, the presolve enables MINOTAUR to solve 20% more instances than the version without presolve.

3.4.2. Constraint disaggregation for MINLP

The preceding section provides an example of how problem formulation can have a significant impact on our ability to solve problems. Here we present another idea, known as disaggregation of constraints. A well-known example from MILP is the uncapacitated facility location problem (see, *e.g.*, Wolsey 1998), which can be described as follows. Given a set of customers, $i = 1, \dots, m$, and a set of facilities, $j = 1, \dots, n$, which facilities should we open ($x_j \in \{0, 1\}$, $j = 1, \dots, n$) at cost f_j to serve the customers? The decision that facility j serves customer i is modelled with the binary variable $y_{ij} \in \{0, 1\}$. The constraints that every customer is served by exactly one facility and that only facilities that have customers assigned are open can be modelled as

$$\begin{aligned} &\sum_{j=1}^{n} y_{ij} = 1, \quad \text{for all } i = 1, \dots, m, \quad \text{and} \\ &\sum_{i=1}^{m} y_{ij} \leq n x_j, \quad \text{for all } j = 1, \dots, n, \end{aligned} \tag{3.16}$$

respectively. A tighter formulation (in the sense that its LP relaxation is closer to the convex hull of the feasible set) is given by the disaggregated form of the second constraints as

$$\begin{aligned} &\sum_{j=1}^{n} y_{ij} = 1, \quad \text{for all } i = 1, \dots, m, \quad \text{and} \\ &y_{ij} \leq x_j, \quad \text{for all } i = 1, \dots, m, \ j = 1, \dots, n. \end{aligned} \tag{3.17}$$

As with (3.15), the difference in solution time can be dramatic. For a small random example with $n = m = 40$, the CPU time is 53 000 seconds for (3.16) versus 2 seconds for (3.17).

Similar disaggregation tricks have been applied to nonlinear functions. Tawarmalani and Sahinidis (2005) consider constraint sets of the form

$$S := \big\{ x \in \mathbb{R}^n : c(x) = h(g(x)) \leq 0 \big\}, \tag{3.18}$$

where $g : \mathbb{R}^n \to \mathbb{R}^p$ is a smooth convex function and $h : \mathbb{R}^p \to \mathbb{R}$ is a smooth, convex, and non-decreasing function. These two conditions imply that $c : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. We note that this functional form is related to the concept of group partial separability, which is frequently used to compute gradients and Hessians efficiently in NLP (Griewank and Toint 1984, Gay 1991, Bongartz, Conn, Gould and Toint 1995).

We can derive a disaggregated version of the constraint set in (3.18) by introducing new variables $y = g(x) \in \mathbb{R}^p$, which leads to the following convex set:

$$S_d := \big\{ (x, y) \in \mathbb{R}^n \times \mathbb{R}^p : h(y) \leq 0, \ y \geq g(x) \big\}. \tag{3.19}$$

One can show that S is the projection of S_d onto x. This reformulation is of interest because any outer approximation of S_d is stronger than the same outer approximation of S, and hence the formulation (3.19) is preferred in any outer-approximation-based algorithm. In particular, given a set of points $\mathcal{X}^k := \{x^{(1)}, \ldots, x^{(k)}\}$, we construct the outer approximations of S and S_d as

$$S^{oa} := \left\{x : c^{(l)} + \nabla c^{(l)^T}(x - x^{(l)}) \leq 0,\ \forall x^{(l)} \in \mathcal{X}^k\right\} \tag{3.20}$$

and

$$\begin{aligned} S_d^{oa} := \big\{(x, y) :& h^{(l)} + \nabla h^{(l)^T}(y - g(x^{(l)})) \leq 0, \\ & y \geq g^{(l)} + \nabla g^{(l)^T}(x - x^{(l)}),\ \forall x^{(l)} \in \mathcal{X}^k\big\}, \end{aligned} \tag{3.21}$$

respectively. It can be shown that the projection of S_d^{oa} onto x is contained in S^{oa}. Tawarmalani and Sahinidis (2005) give an example that shows that the outer approximation (3.21) is tighter than (3.20). Moreover, for the two outer approximations to be equivalent, we may require an exponentially larger number of linearization points $x^{(l)}$ in (3.20) compared with (3.21).

Hijazi *et al.* (2010) study a similar structure, namely, the partially separable constraint set:

$$\left\{x : c(x) := \sum_{j=1}^{q} h_j(a_j^T x + b_j) \leq 0\right\}, \tag{3.22}$$

where $h_j : \mathbb{R} \to \mathbb{R}$ are smooth and convex functions, which ensures that this set is convex. We can again derive a disaggregated form of this set by introducing new variables $y \in \mathbb{R}^q$,

$$\left\{(x, y) : \sum_{j=1}^{q} y_j \leq 0, \quad \text{and} \quad y_j \geq h_j(a_j^T x + b_j)\right\}. \tag{3.23}$$

Again, one can show that the outer approximation of (3.23) is tighter than the outer approximation of (3.22). We can apply this technique to the worst-case example, (3.7), choosing two linearization points as $x^{(1)} \in \{0, 1\}^n$ and its complement $x^{(2)} := e - x^{(1)}$, where $e = (1, \ldots, 1)$ is the vector of all ones. The combined outer approximation of (3.23) is then given by

$$\sum_{i=1}^{n} y_i, \quad \text{and} \quad x_i - \tfrac{3}{4} \leq y_i, \quad \text{and} \quad \tfrac{1}{4} - x_i \leq y_i,$$

which together with $x_i \in \{0, 1\}$ implies that $z_i \geq 0$, which leads to $\sum z_i \geq \frac{n}{4} > \frac{n-1}{4}$, which shows that any master problem that includes the linearizations from $x^{(1)}$ and $x^{(2)}$ is infeasible. Hijazi *et al.* (2010) suggest two additional improvements for outer approximation for partially separable constraints. The first improvement is to add additional linearization points for

the univariate functions $h_j(t)$ in order to obtain a better description of the nonlinear feasible set. The second improvement is to employ an inner approximation of $h_j(t)$ in order to generate initial feasible points in the case that $a_j^T x + b_j = x_j$.

4. Cutting planes for convex MINLPs

In this section we review different cutting planes for use in a branch-and-cut algorithm solving convex MINLPs. We then review generalized disjunctive cuts and perspective cuts for MINLP. We also cover details of the practical realization of disjunctive cuts in a branch-and-cut framework. The final part of this section addresses the closely related problem class of mixed-integer second-order cone programs (MISOCPs).

4.1. Mixed-integer rounding cuts

The LP/NLP-based branch-and-bound Algorithm 3.1 for MINLP solves MILP relaxations, which we intend to strengthen by iteratively adding cuts to remove fractional solutions from these relaxations as outlined in Algorithm 3.3. We start by considering mixed-integer rounding cuts. These are best introduced for the two-variable set

$$S := \{(x_1, x_2) \in \mathbb{R} \times \mathbb{Z} \mid x_2 \leq b + x_1,\ x_1 \geq 0\}, \tag{4.1}$$

where $C = \{1\}$, $I = \{2\}$. Let $f_0 = b - \lfloor b \rfloor$, and observe that the inequality

$$x_2 \leq \lfloor b \rfloor + \frac{x_1}{1 - f_0} \tag{4.2}$$

is valid for X by verifying it for the two cases $x_2 \leq \lfloor b \rfloor$ and $x_2 \geq \lfloor b \rfloor + 1$. The situation is depicted in Figure 4.1 for the set $S = \{(x_1, x_2) \in \mathbb{R} \times \{0, 1\} \mid x_2 \leq \frac{1}{2} + x_1,\ 0 \leq x_1 \leq 2\}$.

For the general MILP case, it is sufficient to consider the set

$$X := \left\{(x_C^+, x_C^-, x_I) \in \mathbb{R}^2 \times \mathbb{Z}^p \mid a_I^T x_I + x_C^+ \leq b + x_C^-,\ x_C^+ \geq 0, x_C^- \geq 0,\ x_I \geq 0\right\}. \tag{4.3}$$

This set describes a selected constraint row of an MILP, or a *one-row relaxation* of a subset of constraints aggregated in the vector $a \in \mathbb{R}^n$ and scalar b. Real variables are aggregated in x_C^+ and x_C^- depending on the sign of their coefficient in a_C. The extension from (4.1) is now straightforward by observing the following inequality is valid for X:

$$\sum_{i \in I} \left(\lfloor a_i \rfloor + \frac{\max\{f_i - f_0, 0\}}{1 - f_0} \right) x_i \leq \lfloor b \rfloor + \frac{x_C^-}{1 - f_0}, \tag{4.4}$$

where $f_i = a_i - \lfloor a_i \rfloor$ for $i \in I$ and $f_0 = b - \lfloor b \rfloor$ are the fractional parts of a and b.

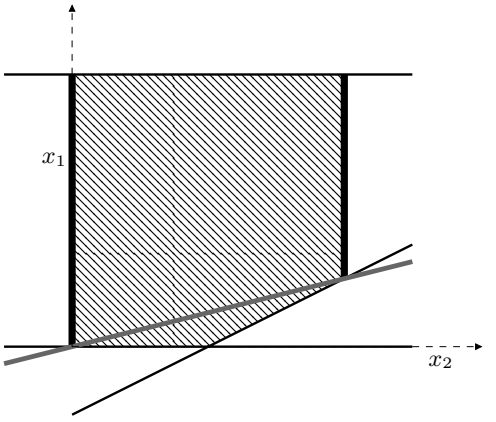


Figure 4.1. Mixed-integer rounding (MIR) cut. Feasible set of the LP relaxation (hatched), integer feasible set (bold black lines), and MIR cut (grey) $x_2 \leq 2x_1$ derived from the inequality $x_2 \leq \frac{1}{2} + x_1$.

Gomory cuts were originally derived by Gomory (1958), Gomory (1960) and Chvátal (1973) for integer linear programs. In the mixed-integer case, a Gomory cut is given by the inequality

$$\sum_{i \in I_1} f_i x_i + \sum_{i \in I_2} \frac{f_0(1-f_i)}{f_i} x_i + x_C^+ + \frac{f_0}{1-f_0} x_C^- \geq f_0, \tag{4.5}$$

where $I_1 = \{i \in I \mid f_i \leq f_0\}$ and $I_2 = I \setminus I_1$. It can be seen to be an instance of a MIR cut by considering the set

$$X = \big\{(x_C, x_0, x_I) \in \mathbb{R}^2 \times \mathbb{Z} \times \mathbb{Z}^p \mid x_0 + a_I^T x_I + x_C^+ - x_C^- = b,\ x_C \geq 0, x_I \geq 0\big\}, \tag{4.6}$$

generating a MIR inequality from it, and eliminating the variable x_0.

To apply MIR cuts in MINLP, we can generate cuts from linearized inequality constraints and the linearized objective constraint of the MINLP reformulation (1.2). Hence, it is sufficient to consider cuts for MILPs. Akrotirianakis, Maros and Rustem (2001) report modest performance improvements using this approach for MINLP.

4.2. Perspective cuts for MINLP

Many MINLP problems employ binary variables to indicate the non-positivity of continuous variables. If $x_i, i \in I$ is a binary variable and $x_j, j \in C$ is the associated continuous variable, the relationship can be modelled with what is known as a *variable upper bound* constraint,

$$x_j \leq u_j x_i. \tag{4.7}$$

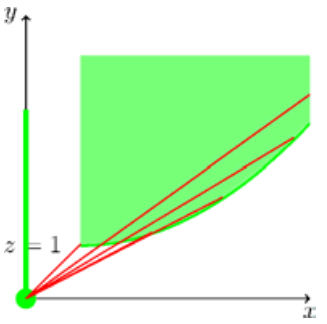


Figure 4.2. The set S.

Note that, since x_i is a $\{0,1\}$ variable, if $x_j > 0$ then $x_i = 1$. If, in addition, the continuous variable x_j appears in a convex, nonlinear constraint, then a reformulation technique called the *perspective reformulation* can lead to significantly improved computational performance. Frangioni and Gentile (2006) pioneered the use of this strengthened relaxation, using cutting planes known as *perspective cuts*. The technique is perhaps best introduced with a simple example. Consider the following mixed-integer set with three variables:

$$S = \big\{(x_1, x_2, x_3) \in \mathbb{R}^2 \times \{0,1\} : x_2 \geq x_1^2, \quad ux_3 \geq x \geq 0\big\}.$$

Figure 4.2 depicts the set S, which is the union of two convex sets $S = S^0 \cup S^1$, where

$$S^0 = \big\{(0, x_2, 0) \in \mathbb{R}^3 : x_2 \geq 0\big\},$$
$$S^1 = \big\{(x_1, x_2, 1) \in \mathbb{R}^3 : x_2 \geq x_1^2,\ u \geq x_1 \geq 0\big\}.$$

One can observe from the geometry that the convex hull of S requires the surface defined by the family of line segments connecting the origin in the $x_3 = 0$ plane to the graph of the parabola $x_2 = x_1^2$ in the $x_3 = 1$ plane. Using this geometric intuition, we can define the convex hull of S as

$$\begin{aligned}\operatorname{conv}(S) = \\ \big\{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_2 x_3 \geq x_1^2,\ ux_3 \geq x_1 \geq 0,\ 1 \geq x_3 \geq 0,\ x_2 \geq 0\big\}.\end{aligned} \tag{4.8}$$

The expression $x_2 x_3 \geq x_1^2$ in (4.8) can be explained in terms of the *perspective function* of the left-hand side of the inequality $x_1^2 - x_2 \leq 0$. For a convex function $f : \mathbb{R}^n \to \mathbb{R}$, the *perspective function* $\mathcal{P} : \mathbb{R}^{n+1} \to \mathbb{R}$ of f is

$$\mathcal{P}(x, z) := \begin{cases} 0 & \text{if } z = 0, \\ zf(x/z) & \text{if } z > 0. \end{cases} \tag{4.9}$$

The epigraph of $\mathcal{P}(x, z)$ is a cone pointed at the origin whose lower shape is $f(x)$. If z_i is an indicator binary variable that forces some variables x to

be 0, or else the convex nonlinear constraint $f(x) \leq 0$ must hold, then by replacing the constraint $f(x) \leq 0$ with

$$z_i f(x/z_i) \leq 0 \tag{4.10}$$

results in a convex inequality that describes a significantly tighter relaxation of the feasible region. Günlük and Linderoth (2012) (Lemma 3.2) slightly generalize this construction to the case where the S_0 side of the disjunction is an unbounded ray (as in Figure 4.2).

The general construction is as follows. We consider the problem

$$\min_{(x,z)\in\mathbb{R}^n\times\{0,1\}} \{f(x) + cz \mid Ax \leq bz\},$$

where (i) $X = \{x \mid Ax \leq b\}$ is bounded (also implying $\{x \mid Ax \leq 0\} = \{0\}$), (ii) $f(x)$ is a convex function that is finite on X, and (iii) $f(0) = 0$. Under these assumptions, for any $\bar{x} \in X$ and subgradient $s \in \partial f(\bar{x})$, the inequality

$$v \geq f(\bar{x}) + c + s^T(x - \bar{x}) + (c + f(\bar{x}) - s^T\bar{x}))(z - 1) \tag{4.11}$$

is valid for the equivalent mixed-integer program

$$\min_{(x,z,v)\in\mathbb{R}^n\times\{0,1\}\times\mathbb{R}} \{v \mid v \geq f(x) + cz, Ax \leq bz\}.$$

Inequality (4.11), called the *perspective cut*, was introduced by Frangioni and Gentile (2006) and used dynamically to build a tight formulation. Günlük and Linderoth (2010) show that perspective cuts are indeed outer approximation cuts for the perspective reformulation for this MINLP. Therefore, adding all (infinitely many) perspective cuts has the same strength as the perspective reformulation. It may be computationally more efficient to use linear outer approximation inequalities of the form (1.5) instead of using the nonlinear form (4.10).

4.3. Disjunctive cutting planes for MINLP

Disjunctive cuts for use in a branch-and-cut procedure were first discussed by Stubbs and Mehrotra (1999) for convex mixed 0–1 nonlinear programs, based on Balas *et al.* (1996) for MILP. Simultaneously, Ceria and Soares (1999) derived the disjunctive programming arguments to be presented in a more general setting. We consider the convex mixed 0–1 nonlinear program

$$\begin{cases} \underset{x,\eta}{\text{minimize}} & \eta, \\ \text{subject to} & f(x) \leq \eta, \\ & c(x) \leq 0, \\ & x \in X, \\ & x_i \in \{0,1\}, \ \forall i \in I. \end{cases} \tag{4.12}$$

For the continuous relaxation of this problem, optimal solutions are guaranteed to lie on the boundary of the continuous relaxation $\mathcal{C} = \{x \in X \mid f(x) \leq \eta,\ c(x) \leq 0,\ 0 \leq x_I \leq 1\}$ of the feasible set. We consider a node in a branch-and-bound tree, with optimal solution x', where x'_j is fractional for some $j \in I$. We denote by $I_0 \subseteq I$, $I_1 \subseteq I$ the index sets of integer variables fixed to zero or one, respectively, by the previous branching decisions that led to the tree node under consideration. We denote by F the index set of free real and integer variables.

Instead of separating the fractional solution x' by simply branching to $x'_j = 0$ and $x'_j = 1$, we are interested in adding a valid inequality to the relaxation that cuts off x'. To this end, we consider the disjoint feasible sets obtained by fixing x_j to either choice,

$$\mathcal{C}_j^0 = \{x \in \mathcal{C} \mid x_j = 0,\ 0 \leq x_i \leq 1,\ \forall i \in F, i \neq j\}, \tag{4.13}$$

$$\mathcal{C}_j^1 = \{x \in \mathcal{C} \mid x_j = 1,\ 0 \leq x_i \leq 1,\ \forall i \in F, i \neq j\}. \tag{4.14}$$

We are interested in a description of $\tilde{M}_j(\mathcal{C}) = \text{conv}(\mathcal{C}_j^0 \cup \mathcal{C}_j^1)$, the convex hull of the continuous relaxation $\mathcal{C}$ of the feasible set with either binary restriction on a single selected x_j. For the set $\tilde{M}_j(\mathcal{C})$, Stubbs and Mehrotra (1999) give the following description:

$$\tilde{M}_j(\mathcal{C}) = \tag{4.15}$$
$$\left\{ (x_F, v_0, v_1, \lambda_0, \lambda_1) \;\middle|\; \begin{array}{ll} v_0 + v_1 = x_F, & v_{0j} = 0,\ v_{1j} = \lambda_1, \\ \lambda_0 + \lambda_1 = 1, & \lambda_0, \lambda_1 \geq 0, \\ \mathcal{P}_{c_i}(v_0, \lambda_0) \leq 0, & \mathcal{P}_{c_i}(v_1, \lambda_1) \leq 0,\ 1 \leq i \leq n_c, \end{array} \right\}$$

where $\mathcal{P}_f(v, \lambda)$ is the perspective function (4.9) for the function f. This procedure can be generalized by repetition to describe the lifted convex hull of the disjoint feasible sets obtained for multiple fractional x_j, $j \in F$ all at once. With that description of the convex hull in hand, we can set up and solve a separation NLP to find a point $\hat{x}$ closest to the fractional one x' and in the convex hull:

$$\left\{ \begin{array}{ll} \underset{x, v_0, v_1, \lambda_0, \lambda_1}{\text{minimize}} & \|x - x'\|, \\ \text{subject to} & (x, v_0, v_1, \lambda_0, \lambda_1) \in \tilde{M}_j(\mathcal{C}), \\ & x_i = 0,\ \forall i \in I_0, \\ & x_i = 1,\ \forall i \in I_1. \end{array} \right. \tag{BC-SEP(x', j)}$$

Let $\hat{x}$ be an optimal solution of (BC-SEP(x', j)), and denote by π_F the Lagrange multipliers for the equality constraint $v_0 + v_1 = x_F$ in (4.15). Then, an inequality that is valid for the current node and cuts off $\bar{x}$ from the feasible set is given by

$$\pi_F^T x_F \leq \pi_F^T \hat{x}_F. \tag{4.16}$$

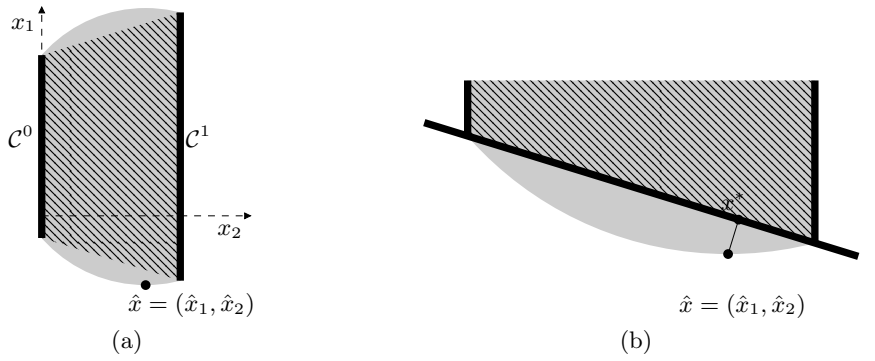


Figure 4.3. (a) NLP relaxation $\mathcal{C}$ (grey), integer feasible convex hull (hatched), and disjoint convex hulls $\mathcal{C}^0$ and $\mathcal{C}^1$ (bold black lines) for the MINLP example (4.17). (b) Solution to the minimum norm problem, and resulting disjunctive cut for the MINLP example (4.17). The next NLP solution including the disjunctive cut will produce the MINLP solution.

As an example of deriving a disjunctive cut, consider the MINLP

$$\begin{cases} \underset{x_1,x_2}{\text{minimize}} & x_1, \\ \text{subject to} & \left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{3}{4}\right)^2 \leq 1, \\ & -2 \leq x_1 \leq 2, \\ & x_2 \in \{0, 1\}, \end{cases} \tag{4.17}$$

with the relaxed optimal solution $x' = (x'_1, x'_2) = \left(-\frac{1}{2}, \frac{3}{4}\right)$ shown in Figure 4.3(a). To construct the separation problem for identifying a disjunctive cut that removed this solution from the relaxation, we consider the individual convex hulls $\mathcal{C}^0$ and $\mathcal{C}^1$ for $x_2 = 0$ and $x_2 = 1$, where the limits are found by solving the constraint $\left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{3}{4}\right)^2 \leq 1$ for x_1 given a fixed x_2,

$$\begin{aligned} \mathcal{C}^0 &= \left\{(x_1, 0) \in \mathbb{R} \times \{0, 1\} \mid 2 - \sqrt{7} \leq 4x_1 \leq 2 + \sqrt{7}\right\}, \\ \mathcal{C}^1 &= \left\{(x_1, 1) \in \mathbb{R} \times \{0, 1\} \mid 2 - \sqrt{15} \leq 4x_1 \leq 2 + \sqrt{15}\right\}. \end{aligned} \tag{4.18}$$

With the Euclidean norm $\|\cdot\|_2$, the minimum norm separation problem for x' yields (approximately) the solution $\hat{x} = (\hat{x}_1, \hat{x}_2) = (-0.40, 0.78)$ with steepest descent direction $\pi = (-0.1, -0.03)$ for the norm objective. This identifies the separating hyperplane

$$\begin{pmatrix} -0.1 & -0.03 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} -0.1 & -0.03 \end{pmatrix} \begin{pmatrix} -0.40 \\ 0.78 \end{pmatrix} \implies x_1 + 0.3x_2 \geq -0.166, \tag{4.19}$$

shown in Figure 4.3(b).

We are free to choose the norm in (BC-SEP(x', j)), but different choices lead to different reformulations, for example for the 1-norm or the ∞-norm.

Stubbs and Mehrotra (1999) observed the most favourable performance of the generated cuts when using the ∞-norm.

The cut (4.16) is in general valid for the local subtree of the branch-and-bound tree only. Stubbs and Mehrotra (1999) show that a *lifting* to a globally valid cut

$$\pi^T x \leq \pi^T \hat{x} \tag{4.20}$$

may be obtained by assigning

$$\pi_i = \min\{e_i^T H_0^T \mu_0, e_i^T H_1^T \mu_1\}, \quad i \notin F, \tag{4.21}$$

where e_i denotes the ith unit vector; $\mu_0 = (\mu_{0F}, 0)$, $\mu_1 = (\mu_{1F}, 0)$, where μ_{0F}, μ_{1F} are the Lagrange multipliers of the perspective inequalities from (4.15) active in $\hat{x}$; and H_0, H_1 are matrices formed from subgradient rows $\partial_v \mathcal{P}_{c_i}(v_0, \lambda_0)^T$, $\partial_v \mathcal{P}_{c_i}(v_1, \lambda_1)^T$ of those inequalities.

Disjunctive cuts are linear in the lifted variable space. In Stubbs and Mehrotra (2002), a step toward nonlinear cutting planes is taken and convex quadratic inequalities are derived as cuts for convex mixed 0–1 integer programs. Computational evidence so far suggests that such cuts do not benefit branch-and-cut procedures in general, although nonlinear cuts for the mixed-integer conic case show otherwise: see the end of Section 4.5.

4.4. Implementation of disjunctive cuts

The nonlinear separation problem (BC-SEP(x', j)) has twice the number of unknowns as the original problem and is not differentiable everywhere because of the perspective constraints. These drawbacks have hindered the use of disjunctive cutting planes for convex 0–1 MINLP, and Stubbs and Mehrotra (1999) have reported computational results for only four instances with no more than 30 unknowns. Addressing this issue, we mention an LP-based iterative separation procedure by Kılınç, Linderoth and Luedtke (2010) and Kılınç (2011) that replaces (BC-SEP(x', j)), and a nonlinear separation approach by Bonami (2011) that circumvents the difficulties inherent in (BC-SEP(x', j)).

Kılınç *et al.* (2010) and Kılınç (2011) propose an iterative procedure to replace (BC-SEP(x', j)) by a sequence of cut generation LPs. They report both an increased number of solvable problems and a significant reduction in run time for a set of 207 instances. To describe this method, we let $\mathcal{B} \supset \mathcal{C}$ be a relaxation of the original MINLP relaxation, and we consider the sets

$$\mathcal{B}_j^0 = \{x \in \mathcal{B}^0 \mid x_j = 0\}, \quad \mathcal{B}_j^1 = \{x \in \mathcal{B}^0 \mid x_j = 1\}. \tag{4.22}$$

Valid inequalities for $\text{conv}(\mathcal{B}_j^0 \cup \mathcal{B}_j^1)$ are also valid for $\text{conv}(\mathcal{C}_j^0 \cup \mathcal{C}_j^1)$. The separation program becomes a linear one if $\mathcal{B}$ is restricted to be a polyhedron and if an appropriate norm is chosen in (BC-SEP(x', j)). Kılınç *et al.* (2010) and Kılınç (2011) iteratively tighten polyhedral outer approximations of $\mathcal{C}_j^0$

and $\mathcal{C}_j^1$ using inequalities generated as disjunctive cuts. To this end, in iteration t two sets $\mathcal{K}_0^t$, $\mathcal{K}_1^t$ of linearization points are maintained, resulting in the polyhedral approximations

$$\mathcal{F}_t^0 = \left\{ x \in \mathbb{R}^n \mid x_i = 0,\ g(x') + \frac{\partial g(x')}{\partial x}(x - x') \leq 0,\ \forall x \in \mathcal{K}_t^0 \right\}, \quad (4.23)$$

$$\mathcal{F}_t^1 = \left\{ x \in \mathbb{R}^n \mid x_i = 1,\ g(x') + \frac{\partial g(x')}{\partial x}(x - x') \leq 0,\ \forall x \in \mathcal{K}_t^1 \right\}. \quad (4.24)$$

Since the sets $\mathcal{F}_t^0$, $\mathcal{F}_t^1$ are polyhedral relaxations of $\mathcal{C}^0$ and $\mathcal{C}^1$, valid disjunctive cuts can be generated. Initially empty, the sets $\mathcal{K}_t^0$, $\mathcal{K}_t^1$ are augmented by the solutions x_t' of the linear separation problems and with two so-called friendly points y_t and z_t, respectively, that satisfy $x_t' = \lambda y_t + (1 - \lambda) z_t$ for some $\lambda \in [0, 1]$. Kılınç *et al.* (2010) prove this approach to be sufficient to ensure that in the limit the obtained inequality is of the same strength as if the nonlinear separation problem (BC-SEP(x', j)) were solved. The process can be terminated early if it is observed that the cuts are not effective at reducing the solution integrality gap. The procedure is applicable to general convex MINLPs with $x_i \in \mathbb{Z}$ as well.

A similar procedure based on the outer approximating set

$$\mathcal{B} = \left\{ x \in \mathbb{R}^n \mid g(x') + \frac{\partial g}{\partial x}(x - x') \leq 0 \right\}$$

was proposed earlier by Zhu and Kuno (2006), but does not necessarily converge (Kılınç *et al.* 2010, Kılınç 2011).

Bonami (2011) addresses the difficulty of solving (BC-SEP(x', j)) in the special situation when x' with $k < x_j < k + 1$ fractional for some $j \in I$ is to be separated from a relaxation that was obtained by a simple disjunction created from a split relaxation. In this case, the separation problem allows an algebraic reduction to

$$\begin{cases} \underset{v_1}{\text{maximize}} & v_{1,j}, \\ \text{subject to} & c_i\left(\frac{v_1}{f_0}\right) \leq f_0 k - v_{1,j}, \\ & c_i\left(\frac{x' - v_1}{1 - f_0}\right) \leq f_0 k - v_{1,j}, \\ & v_1 \in \mathbb{R}^n. \end{cases} \quad (4.25)$$

where $f_0 = x_j' - k > 0$ is the fractional part of x_j. Again, the approach is applicable to general convex mixed-integer problems. In (4.25), the perspective is no longer required, the problem size does not grow compared to the MINLP under consideration, and differentiability is maintained. One can show that the optimal objective value is smaller than $f_0 k$ if and only if $x' \notin \mathcal{C}_j^k$. The separating inequality can be computed from an LP model as

described for the approach by Kılınç *et al.* (2010) and the particular choice

$$\mathcal{K}_t^0 = \left\{ \frac{x' - v_1^*}{1 - f_0} \right\}, \quad \mathcal{K}_t^1 = \left\{ \frac{v_1^*}{f_0} \right\}, \tag{4.26}$$

if v_1^* denotes the optimal solution of (4.25).

4.5. Mixed-integer second-order cone programs

Both mixed-integer rounding cuts and disjunctive cuts can be generalized from the LP cone $\mathbb{R}^n_+$ to other cones such as second-order cones defined by the set $\{(x_0, x) \in \mathbb{R} \times \mathbb{R}^n \mid x_0 \geq \|x\|_2\}$. We consider the class of mixed-integer linear programs with a second-order cone constraint

$$\begin{cases} \underset{x}{\text{minimize}} & c^T x, \\ \text{subject to} & x \in \mathcal{K}, \\ & x \in X, \\ & x_i \in \mathbb{Z}, \ \forall i \in I. \end{cases} \tag{MISOCP}$$

The conic constraint $x \in \mathcal{K}$ represents the product of $k \geq 1$ smaller cones $\mathcal{K} := \mathcal{K}_1 \times \cdots \times \mathcal{K}_k$, defined by

$$\mathcal{K}_j := \left\{ x_j = (x_{j0}, x_{j1}^T)^T \in \mathbb{R} \times \mathbb{R}^{n_j - 1} : \|x_{j1}\|_2 \leq x_{j0} \right\}, \quad 1 \leq j \leq k, \tag{4.27}$$

and $x = (x_1^T, \ldots, x_k^T)^T$. Convex MINLP solvers are usually not directly applicable to this problem class because the conic constraint is not continuously differentiable. Drewes (2009) and Drewes and Ulbrich (2012) propose a variant of the LP/NLP-based branch-and-bound, Algorithm 3.6, that solves continuous relaxations of (MISOCP) instead of NLPs for a fixed integer assignment x_I^k:

$$\begin{cases} \underset{x}{\text{minimize}} & c^T x, \\ \text{subject to} & x \in \mathcal{K}, \\ & x \in X, \\ & x_I = x_I^k. \end{cases} \tag{SOCP(x_I^k)}$$

The algorithm builds MIP outer approximations of (MISOCP) from subgradient information as follows. Denote by (s, y) the dual variables associated with the conic and linear constraints, and define index sets $J_a(\bar{x})$ and $J_{00}(\bar{x})$, $J_{0+}(\bar{x})$ of active conic constraints differentiable and subdifferentiable, respectively, at $\bar{x}$ by

$$J_a(\bar{x}) := \{j : g_j(\bar{x}) = 0, \ \bar{x} \neq 0\}, \tag{4.28}$$

$$J_{0+}(\bar{x}, \bar{s}) := \{j : \bar{x}_j = 0, \ \bar{s}_{j0} > 0\}, \tag{4.29}$$

$$J_{00}(\bar{x}, \bar{s}) := \{j : \bar{x}_j = 0, \ \bar{s}_{j0} = 0\}, \tag{4.30}$$

where g_i is the conic constraint function. Let $S \subset \mathbb{R}^n$ denote the set of previous primal–dual solutions $(\bar{x}, \bar{s})$ to (SOCP(x_I^k)). Then the linear outer approximation MIP for problem (MISOCP) is

$$\begin{cases} \underset{x}{\text{minimize}} & c^T x, & & \\ \text{subject to} & x \in X, & & \\ & c^T x \le c^T \bar{x}, & & \bar{x} \in X,\ \bar{x}_I \in \mathbb{Z}^p, \\ & 0 \ge -\|\bar{x}_{j1}\| x_{j0} + \bar{x}_{j1}^T x_{j1}, & \forall j \in J_a(\bar{x}), & \bar{x} \in X, \\ & 0 \ge -x_{j0} - \frac{1}{\bar{s}_{j0}} \bar{s}_{j1}^T x_{j1}, & \forall j \in J_{0+}(\bar{x}, \bar{s}), & \bar{x} \in X, \\ & 0 \ge -x_{j0}, & \forall j \in J_{00}(\bar{x}, \bar{s}), & \bar{x} \in X, \\ & x_i \in \mathbb{Z}, & \forall i \in I. & \end{cases} \tag{MIP(X))}$$

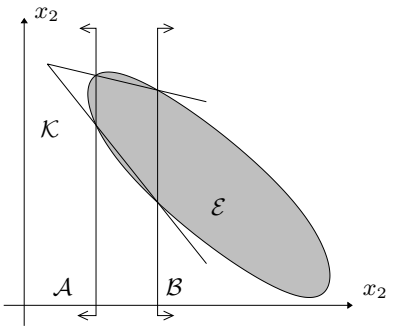
For infeasible SOC subproblems, solutions from feasibility problems can be incorporated into (MIP(X)) in a similar way. Convergence under a Slater constraint qualification, or alternatively by using an additional SOCP branching step if this CQ is violated, is shown by Drewes and Ulbrich (2012).

Polyhedral approximations of the second-order cone constraint are discussed for use in an outer approximation algorithm by Vielma, Ahmed and Nemhauser (2008), who use the polynomial-size relaxation introduced by Ben-Tal and Nemirovski (2001), while Krokhmal and Soberanis (2010) generalize this to p-order cones, that is, sets of the form $\{x \in \mathbb{R}^{n+1} : x_{n+1} \ge \|x\|_p\}$. The drawback here is that one has to choose the size of the approximation *a priori*. Consequently, the LP becomes large when the approximation has to be strengthened. An iterative scheme such as the SOCP-based branch-and-cut procedure for strengthening is hence preferable. The use of a polyhedral second-order conic constraint has been generalized by Masihabadi, Sanjeevi and Kianfar (2011).

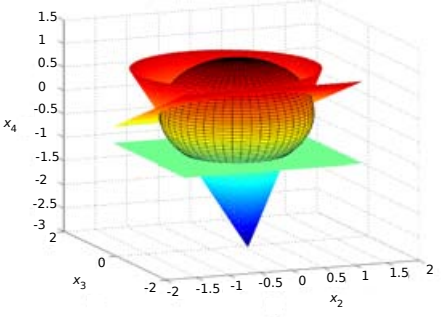
Two efforts in this rapidly evolving area are the work by Dadush, Dey and Vielma (2011*a*), who prove that conic quadratic inequalities are necessary to represent the split closure of an ellipsoid, and the work by Belotti *et al.* (2012), who study the convex hull of the intersection of a disjunction $\mathcal{A} \cup \mathcal{B}$ and a generic convex set $\mathcal{E}$. In general,

$$\mathcal{A} = \{x \in \mathcal{R}^n : a^T x \le \alpha\}, \quad \mathcal{B} = \{x \in \mathcal{R}^n : b^T x \le \beta\};$$

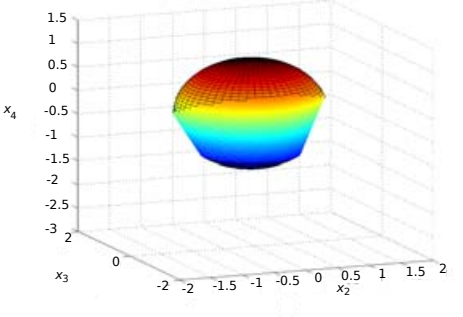
this is therefore a non-parallel disjunction, and a more general disjunction than discussed in Section 3.1.1 and unlike the previous examples. The authors prove that the convex hull is given by intersecting $\mathcal{E}$ with a cone $\mathcal{K}$ such that $\mathcal{K} \cap \partial\mathcal{A} = \mathcal{E} \cap \partial\mathcal{A}$ and $\mathcal{K} \cap \partial\mathcal{B} = \mathcal{E} \cap \partial\mathcal{B}$, where $\partial\mathcal{S}$ is the frontier of set $\mathcal{S}$, if one such cone exists. The authors then provide an algorithm to find this cone for MISOCP, and they prove that it is a second-order cone. In general, this conic cut, which is shown in two and three dimensions in Figure 4.4, is proved to be more effective than the conic MIR cut presented by Atamtürk and Narayanan (2010), and is discussed below.



(a)



(b)



(c)

Figure 4.4. Disjunctive conic cuts as generated by Belotti *et al.* (2012). (a) $\mathcal{K}$ is the disjunctive cone generated when intersecting the ellipsoid $\mathcal{E}$ with the intersection $\mathcal{A} \cup \mathcal{B}$. (b) A disjunction described by two half-spaces delimited by non-parallel hyperplanes is intersected with an ellipsoid $\mathcal{E}$, and the intersection with the arising disjunctive cone, also shown in (b), returns a tighter feasible set depicted in (c).

4.5.1. Gomory cuts for MISOCP

Drewes (2009) describes Gomory cuts for (MISOCP) based on the work by Çezik and Iyengar (2005) for pure integer conic programs. We assume here that the bounded polyhedral set X is described by

$$X = \{x \in \mathbb{R}^n \mid Ax = b,\ l \leq x \leq u\}, \tag{4.31}$$

with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. The additional non-negativity requirement $l_i \geq 0$ holds for all $i \in I$. Then, the following theorem describes a Gomory cut for (MISOCP).

Theorem 4.1 (Drewes 2009, Theorem 2.2.6). Assume that the continuous relaxation (SOCP(x_I^k)) and its dual have feasible interior points. Let $\bar{x}$ with $\bar{x}_I \notin \mathbb{Z}^p$ be a solution of (SOCP(x_I^k)), and let $(\bar{s}, \bar{y})$ be the

corresponding dual solution. Then the following cut is a valid inequality for (MISOCP),

$$\lceil (A_I^T(\bar{y} - \Delta y)\bar{s}_I \rceil^T s_I \geq \lceil (\bar{y} - \Delta y)^T b \rceil, \tag{4.32}$$

where Δy solves

$$\begin{pmatrix} -A_C \\ A_I \end{pmatrix} \Delta y = \begin{pmatrix} c_C \\ 0 \end{pmatrix}. \tag{4.33}$$

Furthermore, if $(\bar{y} - \Delta y)^T b \notin \mathbb{Z}$, then (4.32) cuts off $\bar{x}$ from the integer feasible set.

Gomory cuts are of restricted applicability because the requirement that $l_I \geq 0$, which turns out to be violated frequently by MISOCP instances of practical interest. Consequently, Gomory cuts were found to be largely ineffective for those MISOCP instances evaluated in the computational studies presented by Drewes (2009).

As an example due to Drewes (2009), we consider the MISOCP

$$\begin{cases} \underset{x}{\text{minimize}} & -x_2, \\ \text{subject to} & -3x_2 + x_3 \leq 0, \\ & 2x_2 + x_3 \leq 3, \\ & 0 \leq x_1,\ x_2 \leq 3, \\ & x_1 \geq \|(x_2, x_3)^T\|_2, \\ & x_1,\ x_2 \in \mathbb{Z}, \end{cases} \tag{4.34}$$

whose SOCP relaxation has the optimal solution $\left(3, \frac{12}{5}, -\frac{9}{5}\right)$. The Gomory cut $x_2 \leq 2$ that can be deduced from this point is shown in Figure 4.5 and cuts off the relaxed solution.

4.5.2. Lift and project cuts for MISOCP

Next, we consider the restricted class of mixed 0–1 second-order cone programs,

$$\begin{cases} \underset{x}{\text{minimize}} & c^T x, \\ \text{subject to} & x \in \mathcal{K}, \\ & Ax = b, \\ & l \leq x \leq u, \\ & x_i \in \{0, 1\},\ \forall i \in I, \end{cases} \tag{4.35}$$

and follow the notation of the lift and project procedure described for disjunctive cuts in Section 4.3. We are again interested in a finite conic linear description of the convex hull of the feasible set of (4.35), and we investigate the convex hull of the union of the disjoint sets $\mathcal{C}_j^0$ and $\mathcal{C}_j^1$. For MISOCP,

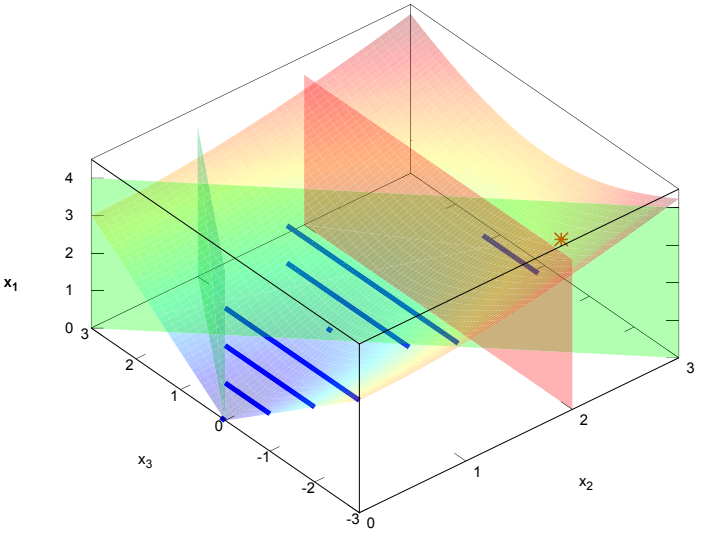


Figure 4.5. MISOCP example (4.34). Feasible cone, linear constraints, integer feasible set (solid lines), relaxed optimal solution (asterisk), and Gomory cut cutting off the relaxed optimal solution (plane at $x_2 = 2$).

the convex hull is described by the set $\tilde{M}_j(\mathcal{C})$:

$$\tilde{M}_j(\mathcal{C}) := \tag{4.36}$$
$$\left\{ (x, v^0, v^1, \lambda^0, \lambda^1) \;\middle|\; \begin{array}{ll} v^0 + v^1 = x, & \\ \lambda^0 + \lambda^1 = 1, & \lambda^0, \lambda^1 \geq 0, \\ Av^0 - \lambda^0 b = 0, & Av^1 - \lambda^1 b = 0, \\ v^0 \in \mathcal{K}, & v^1 \in \mathcal{K}, \\ v^0_j = 0, & v^1_j = \lambda^1, \\ 0 \leq v^0_k \leq \lambda^0, & 0 \leq v^1_k \leq \lambda^1, \;\; k \in J, k \neq j. \end{array} \right\}$$

If we fix a set $B \subseteq J$, the definition of the convex hull $\tilde{M}_B(\mathcal{C})$ is straightforward by repetition of the lifting process. We are now prepared to state the subgradient cut theorem from Stubbs and Mehrotra (1999) for MISOCP.

Theorem 4.2 (Drewes 2009, Proposition 2.1.6). Fix a set $B \subseteq I$, let $\bar{x} \notin \tilde{M}_B(\mathcal{C})$, and let x^* be the solution of

$$\min_{(x,v^0,v^1,\lambda^0,\lambda^1) \in \tilde{M}_B(\mathcal{C})} \|x - \bar{x}\|_2. \tag{4.37}$$

Then the subgradient cut

$$(x^* - \bar{x})^T x \geq x^{*T}(x^* - \bar{x}) \tag{SGC}$$

is a valid linear inequality for all $x \in \tilde{M}_B(\mathcal{C})$ that cuts off $\bar{x}$.

Drewes (2009) describes further lift-and-project cuts for MISOCP, based on work by Çezik and Iyengar (2005) and Stubbs and Mehrotra (1999) for integer SOCPs and MILPs. Similar to the procedure described for MINLP disjunctive cuts above, these cuts can be constructed by solving linear, quadratic, and conic auxiliary programs, and have been found to be considerably more efficient than Gomory cuts when used in a MISOCP branch-and-cut procedure.

4.5.3. Mixed-integer rounding cuts for MISOCP

Atamtürk and Narayanan (2010) describe mixed-integer rounding cuts for MISOCPs. To this end, we consider the following formulation of problem (MISOCP):

$$\begin{cases} \underset{x}{\text{minimize}} & c^T x, \\ \text{subject to} & \|A_j x - b_j\|_2 \le d_j^T x - h_{j0},\ 1 \le j \le k, \\ & x \ge 0, \\ & x_i \in \mathbb{Z},\ \forall i \in I, \end{cases} \tag{4.38}$$

where $A_j \in \mathbb{R}^{n_j \times n}$ are matrices, $b_j \in \mathbb{R}^{n_j}$, $d_j \in \mathbb{R}^n$ are column vectors, and $h_{j0} \in \mathbb{R}$ are scalars. Atamtürk and Narayanan (2010) introduce the following *polyhedral second-order conic constraint formulation* that allows the exploitation of polyhedral information on conic constraints of this shape. For simplicity of exposition, we consider the case of $k = 1$ conic constraint of dimension n_1 only. We introduce $t = (t_0, t_1, \ldots, t_{n_1}) \in \mathbb{R}^{1+n_1}$, and we denote by $a_l^T \in \mathbb{R}^n$ the lth row of matrix $A \in \mathbb{R}^{n_1 \times n}$:

$$\begin{cases} \underset{x,t}{\text{minimize}} & c^T x, \\ \text{subject to} & t_l \ge |a_l^T x - b_l|,\ 1 \le l \le n_1, \\ & \|t\|_2 \le t_0 \le d^T x - h_0, \\ & x \ge 0, \\ & x_i \in \mathbb{Z},\ \forall i \in I. \end{cases} \tag{4.39}$$

We denote by S_l the feasible set considering a single component $1 \le l \le n_1$ of the polyhedral conic constraint,

$$S_l := \{x \in \mathbb{R}^n,\ x \ge 0,\ x_i \in \mathbb{Z},\ \forall i \in I,\ t \in \mathbb{R} : t \ge |a_l^T x - b_l|\}. \tag{4.40}$$

A family of valid inequalities for S_l, called conic mixed-integer rounding (MIR) inequalities, is given by the following theorem.

Theorem 4.3 (Atamtürk & Narayanan 2010, Theorem 1). For any $\alpha \neq 0$ the conic mixed-integer rounding (MIR) inequality

$$\sum_{i \in I} \phi_{f_\alpha}(a_i/\alpha) x_i - \phi_{f_\alpha}(b/\alpha) \le (t + x_C^+ + x_C^-)/|\alpha| \tag{4.41}$$

is valid for the set S_l, where x_C^+ and x_C^- aggregate the real variables x_C with positive and negative coefficients a_{lR}, and $\phi_{f_\alpha} : \mathbb{R} \to \mathbb{R}$ is the *conic MIR function* for $0 \leq f_\alpha := b/\alpha - \lfloor b/\alpha \rfloor \leq 1$,

$$\phi_{f_\alpha}(a) := \begin{cases} (1-2f_\alpha)p - (a-p) & \text{if } p \leq a < p + f_\alpha, \\ (1-2f_\alpha)p + (a-p) - 2f_\alpha & \text{if } p + f_\alpha \leq a < p + 1, \end{cases} \quad n \in \mathbb{Z}. \tag{4.42}$$

Moreover, the inequalities are shown to be facet-defining under certain conditions. Furthermore, such inequalities can be used efficiently to cut fractional points off the relaxation of S_l.

Theorem 4.4 (Atamtürk & Narayanan 2010, Proposition 4). Conic mixed-integer equalities with $\alpha = a_{lj}$, $j \in I$ are sufficient to cut off all fractional extreme points of the LP relaxation of S_l.

5. Non-convex MINLP

Non-convex MINLPs are especially challenging because they contain non-convex functions in the objective or the constraints; hence even when the integer decision variables are relaxed to be continuous, the feasible region may be non-convex. Therefore, more work needs to be done to obtain an efficiently solvable (convex) relaxation for use in a branch-and-bound framework.

Non-convex MINLP is closely related to global optimization, a topic that also seeks optimal solutions to optimization problems having non-convex functions, although the focus in global optimization has often been on problems with only continuous decision variables. A huge literature on global optimization exists, including several textbooks (Hansen 1992, Horst and Tuy 1993, Horst, Pardalos and Thoai 1995, Floudas 2000). It is beyond the scope of this paper to provide a comprehensive review of global optimization. Our focus will be on describing techniques that either explicitly consider the integer variables appearing in an MINLP or that are essential to many algorithms for solving non-convex MINLPs.

One approach to solving non-convex MINLPs is to replace the non-convex functions with piecewise linear approximations and solve the corresponding approximation by using mixed-integer linear programming solvers. We discuss this approach in Section 5.1. In the remaining sections we discuss components of methods for directly solving non-convex MINLPs. In Section 5.2, we discuss generic strategies for obtaining convex relaxations of non-convex functions. We then describe in Section 5.3 how spatial branching can be used with these relaxations to obtain a convergent algorithm. Section 5.4 provides a sample of some techniques to obtain improved convex relaxations by exploiting particular types of non-convex structures.

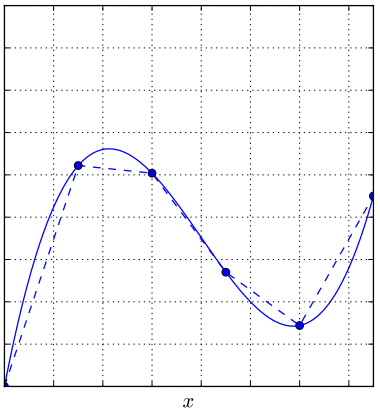


Figure 5.1. Example of a function (solid line) and a piecewise linear approximation to the function (dashed line).

We refer the reader to the works of Tawarmalani and Sahinidis (2002) and Burer and Letchford (2012) for additional surveys focused on non-convex MINLP.

5.1. Piecewise linear modelling

A common approach to the approximate solution of MINLPs with non-convex functions is to replace the nonlinear functions with piecewise linear approximations, leading to an approximation that can be solved by mixed-integer *linear* programming solvers. In fact, the importance of being able to model such functions using binary variables was recognized in some of the earliest work in binary integer programming (Markowitz and Manne 1957, Dantzig 1960). Figure 5.1 shows an example of a non-convex function with a corresponding piecewise linear approximation.

We focus our attention on modelling piecewise linear approximations of a univariate function $f : [l, u] \to \mathbb{R}$, where $l, u \in \mathbb{R}$. A multivariate separable function of the form

$$g(x) = \sum_{i=1}^{K} f_i(x_i)$$

can be approximated by separately obtaining piecewise linear approximations of $f_i(x_i)$. In Section 5.1.3 we briefly introduce extensions of the piecewise linear approximation approach to more general multivariate functions.

Using a piecewise linear modelling technique for solving MINLPs involves two steps: obtaining piecewise linear approximations of the nonlinear functions and modelling the piecewise linear functions in a way that mixed-integer linear programming solvers can handle. We discuss these two steps in Sections 5.1.1 and 5.1.2, respectively. In Section 5.1.3 we provide a brief

overview of how these modelling approaches can be extended to multivariate functions.

Our treatment of this approach is necessarily brief. For more details on this topic we refer the reader to Geissler, Martin, Morsi and Schewe (2012), who provide a recent survey of piecewise linear modelling in MINLP, and to Vielma, Ahmed and Nemhauser (2010), who provide a detailed review of methods for modelling piecewise linear functions using binary variables.

5.1.1. Obtaining a piecewise linear approximation

Given a function $f : [l, u] \to \mathbb{R}$, we seek to obtain a piecewise linear function $\hat{f} : [l, u] \to \mathbb{R}$ such that $\hat{f}(x) \approx f(x)$ for all $x \in [l, u]$. If the piecewise linear function $\hat{f}$ consists of d linear segments, then it may be specified by its *break points* $l =: b^0 < b^1 < \cdots < b^d := u$ and the corresponding function values $y^k = \hat{f}(b^k)$, for $k = 0, 1, \ldots, d$. Then the function $\hat{f}$ is given by

$$\hat{f}(x) = y^{k-1} + \left(\frac{y^k - y^{k-1}}{b^k - b^{k-1}}\right)(x - b^{k-1}), \quad x \in [b^{k-1}, b^k], \text{ for all } k = 1, \ldots, d. \tag{5.1}$$

Alternatively, if for each $k = 1, \ldots, d$ we let $m_k = (y^k - y^{k-1})/(b^k - b^{k-1})$ be the slope of the line segment in interval k, then $a_k = y^{k-1} - m^k b^{k-1}$ is the y-intercept of the line defining the line segment in interval k. Thus we can equivalently write $\hat{f}$ as

$$\hat{f}(x) = a_k + m_k x, \; x \in [b^{k-1}, b^k], \text{ for all } k = 1, \ldots, d. \tag{5.2}$$

Obtaining a piecewise linear approximation involves two competing objectives. The first is to obtain an accurate approximation, where accuracy may be measured in multiple ways. A natural measure of accuracy is the maximum absolute difference between the function and its approximation:

$$\max_{x \in [l,u]} |f(x) - \hat{f}(x)|.$$

The second objective is to obtain an approximation that uses fewer linear segments d. This is important because the time to solve the resulting approximate problem increases with the number of segments used in the approximation, possibly dramatically. Therefore, while one can always obtain an improved approximation by including more segments, this has to be weighed against the computational costs.

The simplest approach to obtaining a piecewise linear approximation is simply to choose a set of break points, for example, uniformly in the interval $[l, u]$, and then let $y^k = f(b^k)$ for each break point b^k. This approach is illustrated in Figure 5.1. For a given number of break points, however, one can obtain significantly better approximations by choosing the location of the break points so that parts of the function that are more nonlinear have

more break points. In addition, using a value of $y^k \neq f(b^k)$ may also yield a better approximation.

The sandwich algorithm is another simple approach to obtaining a piecewise linear approximation. This approach begins with a single linear segment approximating the function by using break points $b^0 = l$ and $b^1 = u$ and function values $y^0 = f(b^0)$ and $y^1 = f(b^1)$. At each iteration $k \geq 1$ we have break points $b^0, \ldots, b^k$, and we use $y^i = f(b^i)$ for $i = 0, \ldots, k$. We then select the index $i \in \{1, \ldots, k\}$ such that the error between the function f and the current approximation over the interval $[x_{i-1}, x_i]$ is largest, according to whatever error measure we are interested in. A new break point in the interval (x_{i-1}, x_i) is then selected and added to the set of break points. Many possible rules for selecting the the new break point exist, such as choosing the point where the error is largest or choosing the mid-point of the interval. Rote (1992) analyses these and two other variants of this algorithm when applied to a function f that is either convex or concave, and he shows that using any of these variants the error after k iterations is $O(1/k^2)$.

Piecewise linear approximation methods appear in diverse fields, and so it is beyond the scope of this paper to provide a thorough review of these techniques. We instead provide a sample of references to other approaches. Geoffrion (1977) studies methods for finding optimal piecewise linear approximations of convex or concave functions for a given number of break points. Bellman (1961) introduces a dynamic programming approach. Boyd and Vandenberghe (2004) consider the problem of finding a piecewise linear *convex* function that best approximates a given finite set of $(x, f(x))$ points, where they assume the break points are given and the problem is to find the function values at the break points. Toriello and Vielma (2012) also consider the setting in which a finite set of $(x, f(x))$ data points is given and provide optimization formulations for finding piecewise linear approximations of these points. Notably, the approach of Toriello and Vielma (2012) does not require that the piecewise linear approximations be convex (although this can be enforced if desired) and that one simultaneously choose the break points and the function values of the approximation.

5.1.2. Modelling piecewise linear functions

This section describes techniques for modelling a piecewise linear function $\hat{f} : [l, u] \to \mathbb{R}$, as defined in (5.1). We introduce a variable y that will be used to model the value of this function. That is, we provide formulations that enforce the condition

$$y = \hat{f}(x).$$

We then use y in place of the function $f(x)$ anywhere it appears in the optimization model. In our development, we assume that the piecewise

linear function $\hat{f}$ is continuous, although many of these formulations can be extended to lower semicontinuous piecewise linear functions (Vielma *et al.* 2010). In describing the different approaches, we follow the names used by Vielma *et al.* (2010).

The first approach we present is the *multiple choice model*, which uses the representation of $\hat{f}(x)$ given in (5.2). In this model, a set of binary variables z_k, $k = 1, \dots, d$, is introduced, where $z_k = 1$ indicates that x is in interval k, $[b^{k-1}, b^k]$. In addition, for each interval k a variable w_k is introduced, where w_k will be equal to x if x is in interval k, and $w_k = 0$ otherwise. The model is as follows:

$$\sum_{k=1}^{d} w^k = x, \quad \sum_{k=1}^{d} (m_k w_k + a_k z_k) = y, \quad \sum_{k=1}^{d} z_k = 1, \tag{5.3a}$$

$$b^{k-1} z_k \le w_k \le b^k z_k, \quad z_k \in \{0, 1\}, \quad k = 1, \dots, d. \tag{5.3b}$$

This model was introduced by Jeroslow and Lowe (1984) and has been studied by Balakrishnan and Graves (1989) and Croxton, Gendron and Magnanti (2003). In computational experiments conducted by Vielma *et al.* (2010), this model frequently yielded the best computational performance when the number of break points was small or moderate (*e.g.*, $d \le 16$).

The second model is the *disaggregated convex combination model*, which uses the representation of $\hat{f}(x)$ given in (5.1). In this model, a set of binary variables z_k, $k = 1, \dots, d$, is introduced, where $z_k = 1$ indicates that $x \in [b^{k-1}, b^k]$. For each interval, this model introduces a pair of continuous variables λ_k and μ_k, which are zero if $z_k = 0$, but otherwise describe x as a convex combination of endpoints of the interval $[b^{k-1}, b^k]$. Consequently, the function value $y = \hat{f}(x)$ can then be described as the same convex combination of y^{k-1} and y^k. The formulation is as follows:

$$\sum_{k=1}^{d} (\lambda_k b^{k-1} + \mu_k b^k) = x, \quad \sum_{k=1}^{d} (\lambda_k y^{k-1} + \mu_k y^k) = y, \tag{5.4a}$$

$$\sum_{k=1}^{d} z_k = 1, \quad \lambda_k + \mu_k = z_k, \quad k = 1, \dots, d, \tag{5.4b}$$

$$\lambda_k \ge 0, \quad \mu_k \ge 0, \quad z_k \in \{0, 1\}, \quad k = 1, \dots, d. \tag{5.4c}$$

The constraints (5.4b) enforce that exactly one interval has $z_k = 1$, that for this interval the variables λ_k and μ_k sum to one, and that $\lambda_i = \mu_i = 0$ for all other intervals $i \ne k$. Thus, the constraints (5.4a) enforce that x and y be written as a convex combination of the end points of the selected interval, and the function values at those end points, respectively. This approach has been presented by Meyer (1976), Jeroslow and Lowe (1984, 1985), Croxton *et al.* (2003) and Sherali (2001).

The next formulation is the *convex combination model*, also sometimes called the lambda method. In this formulation, a single continuous variable is introduced for each break point. These variables are used to express x and y as a convex combination of the break points, and their function values, respectively. As in the disaggregated convex combination model, binary variables are introduced to determine which interval x lies in. These variables are then used to ensure that only the convex combination variables associated with the end points of this interval are positive. The formulation is as follows:

$$\sum_{k=0}^{d} \lambda_k b^k = x, \quad \sum_{k=0}^{d} \lambda_k y^k = y, \tag{5.5a}$$

$$\sum_{j=k}^{d} \lambda_j \leq \sum_{j=k}^{d} z_j, \quad \sum_{j=0}^{k-1} \lambda_j \leq \sum_{j=1}^{k} z_j, \quad k = 1, \ldots, d, \tag{5.5b}$$

$$\sum_{k=0}^{d} \lambda_k = 1, \quad \sum_{k=1}^{d} z_k = 1, \tag{5.5c}$$

$$\lambda_k \geq 0, \quad k = 0, 1, \ldots, d, \quad z_k \in \{0, 1\}, \quad k = 1, \ldots, d. \tag{5.5d}$$

The constraints (5.5b) enforce that when $z_k = 1$, $\lambda_j = 0$ for all $j \notin \{k - 1, k\}$. In most presentations of the convex combination model (Dantzig 1960, Dantzig 1963, Jeroslow and Lowe 1985, Nemhauser and Wolsey 1988, Wilson 1998, Lee and Wilson 2001, Vielma *et al.* 2010, Geissler *et al.* 2012) this condition is instead formulated by using the following constraints:

$$\lambda_0 \leq z_1, \quad \lambda_d \leq z_d, \quad \lambda_k \leq z_k + z_{k+1}, \quad k = 1, \ldots, d - 1. \tag{5.6}$$

However, Padberg (2000) demonstrated that the formulation using (5.6) allows more continuous solutions when the binary constraints on the z_k variables are relaxed, which makes (5.5b) preferable for computational purposes.

The next model we present is the *incremental model*, sometimes referred to as the delta method, which was originally proposed by Markowitz and Manne (1957). In this model, continuous variables δ_k, for each interval $k = 1, \ldots, d$, are introduced to determine what portion of interval k the argument x has 'filled'. Binary variables are introduced to enforce the condition that the intervals are filled in order. This leads to the following formulation:

$$b^0 + \sum_{k=1}^{d} \delta_k (b^k - b^{k-1}) = x, \quad y^0 + \sum_{k=1}^{d} \delta_k (y^k - y^{k-1}) = y, \tag{5.7a}$$

$$\delta_{k+1} \leq z_k \leq \delta_k, \quad k = 1, \ldots, d - 1, \tag{5.7b}$$

$$\delta_1 \leq 1, \quad \delta_d \geq 0, \quad z_k \in \{0, 1\}, \quad k = 1, \ldots, d - 1. \tag{5.7c}$$

If $\delta_i < 1$ for some i, then constraints (5.7b) combined with the binary restrictions on the z variables enforce that $z_i = 0$ and $\delta_{i+1} = 0$ and then recursively that $z_j = \delta_{j+1} = 0$ for all $j > i$, which is precisely the condition that the intervals should be filled in order. As pointed out by Padberg (2000), the convex combination formulation (5.7) and the incremental formulation (5.7) are related by a simple change of variables (*i.e.*, $\delta_k = \sum_{j=k}^{d} \lambda_j$, $k = 1, \ldots, d$, and similarly for the respective z_k variables).

The fifth model we describe does not use binary variables at all. Instead, it uses the concept of a special-ordered set of variables of type II (SOS2) (Beale and Tomlin 1970, Tomlin 1981, Beale and Forrest 1976, Keha, De Farias Jr and Nemhauser 2006). An ordered set of variables $\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_d)$ is said to be SOS2 if at most two of the variables in the set are non-zero and the non-zero variables are adjacent. This is exactly the condition that the binary variables in the convex combination model (5.5) are used to enforce. Therefore, by instead explicitly stating this condition, the following formulation is obtained:

$$\sum_{k=0}^{d} \lambda_k b^k = x, \quad \sum_{k=0}^{d} \lambda_k y^k = y, \tag{5.8a}$$

$$\sum_{k=0}^{d} \lambda_k = 1, \tag{5.8b}$$

$$\lambda_k \geq 0, \quad k = 0, 1, \ldots, d, \quad (\lambda_0, \lambda_1, \ldots, \lambda_d) \text{ is SOS2}. \tag{5.8c}$$

The condition that an ordered set of variables is SOS2 can be declared in most commercial mixed-integer linear programming solvers (IBM 2009, Gurobi 2012, FICO Xpress 2009), similar to declaring that an individual variable is integer or binary. This condition is relaxed to obtain a linear programming relaxation and is progressively enforced through branching, just as integer restrictions are. The main difference is how the branching is done. In an LP relaxation solution, if the SOS2 condition is violated by the relaxation solution $\hat{\lambda}$, then an index $k \in \{1, \ldots, d\}$ is selected such that there exists an index $j_1 < k$ with $\lambda_{j_1} > 0$ and also an index $j_2 > k$ with $\lambda_{j_2} > 0$. Then, two branches are created: one that enforces $\lambda_j = 0$ for all $j < k$ and the other that enforces $\lambda_j = 0$ for all $j > k$. Every solution that satisfies the SOS2 condition is feasible for one of the two branching conditions, and hence this branching strategy does not exclude any feasible solutions. In addition, the current relaxation solution, which violates the SOS2 condition, is eliminated from both branches, enabling the relaxation bound to improve and ensuring that the SOS2 condition will be satisfied after a finite number of branches. One also can derive valid inequalities based on the SOS2 condition, analogous to the use of valid inequalities for mixed-integer programming (Keha *et al.* 2006).

Vielma and Nemhauser (2011) have recently developed a novel approach to modelling piecewise linear functions using binary variables. The interesting feature of this approach is that the number of binary variables required is only logarithmic in the number of segments of the linear function. In contrast, the binary formulations we presented here all require one binary variable for each segment. Note, however, that the number of continuous variables required in the formulations of Vielma and Nemhauser (2011) is still linear in the number of segments. The computational results of Vielma and Nemhauser (2011) suggest that this approach is most beneficial when modelling multivariate piecewise linear functions.

5.1.3. Multivariate functions

One approach to using piecewise linear models for problems with multivariate functions is to attempt to reformulate the problem in such a way that only univariate functions appear, and then apply the univariate modelling techniques. We have already mentioned separable functions of the form $g(x) = \sum_{i=1}^{K} f_i(x_i)$ as one example where this approach can work well. More generally, if an algebraic description of a multivariate function g is available, then the techniques described in Section 5.2.1 can be used to first obtain a reformulated problem that contains only univariate and bivariate functions and then construct piecewise linear approximations of these functions. Using further transformations, one may be able to eliminate even the bivariate functions. For example, suppose we have a constraint

$$y = x_1 x_2$$

in our model, where y, x_1, and x_2 are all decision variables, and because of other constraints in the model we know that $x_1 > 0$ and $x_2 > 0$ in any feasible solution. Then, this constraint is equivalent to

$$\ln(y) = \ln(x_1) + \ln(x_2).$$

Each of the functions $\ln(y)$, $\ln(x_1)$, and $\ln(x_2)$ can then be approximated by using univariate piecewise linear models.

The approach of reducing multivariate functions to univariate functions has a few potential disadvantages, which have been pointed out, for example, by Tomlin (1981), Lee and Wilson (2001) and Vielma *et al.* (2010). First, it may not always be easy to find a systematic approach that reduces a given model to one that contains only univariate functions. In our example above, we required $x_1 > 0$ and $x_2 > 0$ in order for the log transformation to be valid. If this were not the case, an alternative would be required. Second, and probably more important, this process of reformulation may allow the errors introduced by the piecewise linear approximations to accumulate and amplify, potentially leading to an approximate model that is either too inaccurate to be useful or requires so many break points in the piecewise

linear approximations that it becomes intractable to solve. Finally, in some cases a function is not given analytically. Instead, we may have only an oracle, or 'black box', that allows us to obtain function evaluations at given points, for example, by running a complex simulation or even a physical experiment. In this case, we may wish to obtain a piecewise linear function that approximates the data obtained at a set of trial points.

One can also directly model non-separable multivariate piecewise linear functions using binary variables. We refer the reader to the works of Vielma *et al.* (2010) and Geissler *et al.* (2012) for details of these techniques. In particular, the convex combination method has been extended to multivariate functions by Lee and Wilson (2001), and the incremental method has been extended by Wilson (1998). Tomlin (1981) also extended the notion of special-ordered sets for modelling multivariate piecewise linear functions. D'Ambrosio, Lodi and Martello (2010) provide an interesting alternative for approximating multivariate piecewise linear functions, in which the piecewise linear approximation is not fixed *a priori*, but is instead allowed to be chosen 'optimistically' by the optimization algorithm. This modification enables a relatively compact formulation to be derived. We note, however, that the number of pieces required to obtain an acceptable piecewise linear approximation of a given nonlinear function may grow exponentially in the number of arguments to the function. Hence, all these approaches are, in practice, limited to functions with at most a few arguments.

5.2. Generic relaxation strategies

For general non-convex MINLP problems, methods for finding a relaxation exploit the structure of the problem. For a broad class of MINLP problems, the objective function and the constraints are nonlinear but *factorable*, in other words, they can be expressed as the sum of products of unary functions of a finite set $\mathcal{O}_{\text{unary}} = \{\sin, \cos, \exp, \log, |\cdot|\}$ whose arguments are variables, constants, or other functions, which are in turn factorable. In other words, a factorable function can be written by combining a finite number of elements from a set of operators $\mathcal{O} = \{+, \times, /, \hat{}, \sin, \cos, \exp, \log, |\cdot|\}$. This excludes, among others, integral functions $\int_{x_0}^{x} h(x)\mathrm{d}x$ whose antiderivative is unknown, and *black-box* functions whose value can be computed by running a simulation, for instance. The approach described below is therefore suitable when the symbolic information about the objective function and constraints, that is, their expressions, is known.

Factorable functions can be represented by *expression trees*. These are n-ary arborescences whose leaves are constants or variables and whose non-leaf nodes are, in general, n-ary operators whose children are the arguments of the operator, and which are in turn expression trees (Cohen 2003). The expression tree of the function $f(x_1, x_2) = x_1 \log(x_2) + x_2^3$ is depicted in Figure 5.2.

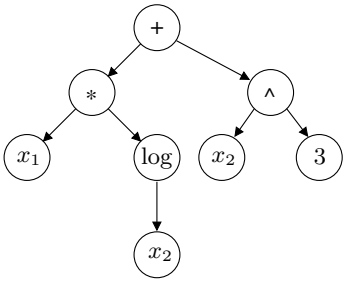


Figure 5.2. The expression tree of $f(x_1, x_2) = x_1 \log(x_2) + x_2^3$. Leaf nodes are for variables x_1 and x_2 and for the constant 3.

5.2.1. Relaxations of factorable functions

If the objective and the constraints of an MINLP of the form (1.1) are factorable, the problem admits a representation where the expression trees of the objective function and all constraints are combined. The root of each expression is one among $c_1(x), c_2(x), \ldots, c_m(x)$, or $f(x)$, and is associated with a lower and an upper bound: $[-\infty, 0]$ for $c_i(x), i = 1, 2, \ldots, m$, and $[-\infty, \bar{\eta}]$ for $f(x)$, where $\bar{\eta}$ is the objective function value of a feasible solution for (1.1), if available, or ∞. The leaf nodes of all expression trees are replaced by a unique set of nodes representing the variables $x_1, x_2, \ldots, x_n$ of the problem. The result is a *directed acyclic graph* (DAG).

Example. Consider the following MINLP:

$$\begin{cases} \text{minimize} & x_1 + x_2^2, \\ \text{subject to} & x_1 + \sin x_2 \leq 4, \\ & x_1 x_2 + x_2^3 \leq 5, \\ & x_1 \in [-4, 4] \cap \mathbb{Z}, \\ & x_2 \in [0, 10] \cap \mathbb{Z}. \end{cases} \tag{5.9}$$

The DAG of this problem has three nodes without entering arcs (one for the objective function and two for the constraints) and six leaf nodes: two for the variables x_1 and x_2 and four for the constants 2, 3, -4, and -5. It is represented in Figure 5.3.

Factorable problems allow a *reformulation* of problem (1.1) (McCormick 1976, Smith and Pantelides 1997, Tawarmalani and Sahinidis 2002). The reformulation is another MINLP as follows:

$$\begin{cases} \underset{x}{\text{minimize}} & x_{n+q}, \\ \text{subject to} & x_k = \vartheta_k(x), & k = n+1, n+2, \ldots, n+q, \\ & l_i \leq x_i \leq u_i, & i = 1, 2, \ldots, n+q, \\ & x \in X, \\ & x_i \in \mathbb{Z}, \ \forall i \in I, \end{cases} \tag{5.10}$$

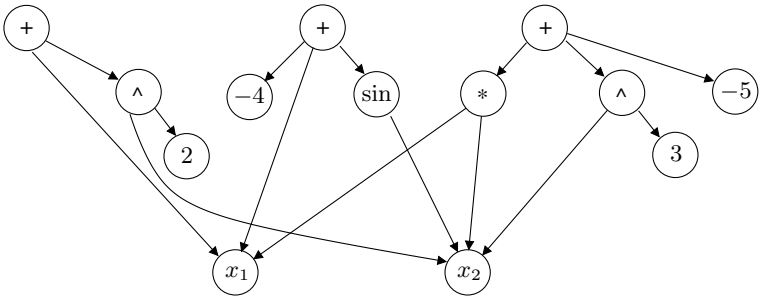


Figure 5.3. The DAG associated with the problem in (5.9). Each node without entering arcs is associated with the root of the expression tree of either a constraint or the objective function. In common with all constraints and the objective are the leaf nodes associated with variables x_1 and x_2.

where ϑ_k is an operator of the set $\mathcal{O}$ introduced above. The bounds on all variables are written explicitly here for the sake of clarity, but they are included in the definition of X; we will use this notation throughout this section. The reformulation contains a set of q new variables known as *auxiliary variables* (or more simply auxiliaries). By convention, the last auxiliary variable replaces the objective function. Each of these variables is constrained to be equal to a function $\vartheta(x)$ such that $\vartheta \in \mathcal{O}$. The lower and upper bounds on each auxiliary variable x_k, as well as its integrality constraint, depend on the operator ϑ_k associated with it and on the bounds on the arguments of ϑ_k (and the integrality of these arguments).

Example. The MINLP shown in (5.9) admits the following reformulation:

$$\begin{cases} \text{minimize } x_9, \\ \text{subject to} \\ \quad x_3 = \sin x_2, \quad x_7 = x_5 + x_6 - 5, \quad 0 \le x_2 \le 10, \quad 0 \le x_6 \le 1000, \\ \quad x_4 = x_1 + x_3 - 4, \quad x_8 = x_2^2, \quad -1 \le x_3 \le 1, \quad -45 \le x_7 \le 0, \\ \quad x_5 = x_1 x_2, \quad x_9 = x_1 + x_8, \quad -9 \le x_4 \le 0, \quad 0 \le x_8 \le 100, \\ \quad x_6 = x_2^3, \quad -4 \le x_1 \le 4, \quad -40 \le x_5 \le 40, \quad -4 \le x_9 \le 104, \\ \quad x_1, x_2, x_5, x_6, x_7, x_8, x_9 \in \mathbb{Z}. \end{cases}$$

Note that x_3, the auxiliary associated with $\sin x_2$, has bounds $[-1, 1]$ because $x_2 \in [0, 10]$, while $x_7 := x_5 + x_6 - 5$, with bounds $[-45, 1035]$, is further constrained by the right-hand side of the constraint $x_1 x_2 + x_2^3 \le 5$. This variable is indeed associated with the root of the expression tree of $c_2(x) = x_1 x_2 + x_2^3 - 5$. The integrality of an auxiliary also depends on whether the function associated with it can return only integer values given the integrality constraints of its arguments. In this case, x_3 and x_4 are not constrained to be integer, while all other variables are because of the integrality constraint on x_1 and x_2.

Problems (5.10) and (1.1) are equivalent in that for any feasible (resp. optimal) solution $\bar{x} \in \mathbb{R}^{n+q}$ of (5.10) one can obtain a feasible (resp. optimal) solution $\tilde{x} \in \mathbb{R}^n$ of (1.1) and *vice versa*. Although still a non-convex MINLP, (5.10) makes it easier to obtain a convex relaxation of (1.1). Consider the non-convex sets

$$\Theta_k = \{x \in \mathbb{R}^{n+q} : x_k = \vartheta_k(x),\ x \in X,\ l \le x \le u,\ x_i \in \mathbb{Z},\ i \in I\},$$

for $k = n+1, n+2, \ldots, n+q$. Note that Θ_k are, in general, non-convex because of the equation $x_k = \vartheta_k(x)$ and the integrality constraints. Suppose that a convex set $\breve{\Theta}_k \supseteq \Theta_k$ exists for each $k = n+1, n+2, \ldots, n+q$. Then the following is a convex relaxation of (5.10) that intersects $\breve{\Theta}_k$ for $k = n+1, n+2, \ldots, n+q$, and hence it is a relaxation of (1.1) (note that the integrality constraints are also relaxed):

$$\begin{cases} \underset{x}{\text{minimize}} & x_{n+q}, & \\ \text{subject to} & x \in \breve{\Theta}_k, & k = n+1, n+2, \ldots, n+q, \\ & l_i \le x_i \le u_i, & i = 1, 2, \ldots, n+q, \\ & x \in X. & \end{cases}$$

Convex sets $\breve{\Theta}_k$ are generally polyhedral, that is, they are described by a system of m_k linear inequalities:

$$\breve{\Theta}_k = \left\{x \in \mathbb{R}^{n+q} : a^k x_k + B^k x \ge d^k,\ x \in X,\ l \le x \le u\right\},$$

where $a^k \in \mathbb{R}^{m_k}$, $B^k \in \mathbb{R}^{m_k \times (n+q)}$, and $d^k \in \mathbb{R}^{m_k}$. Hence several practical MINLP solvers for (1.1) use the following linear relaxation to obtain a lower bound:

$$\begin{cases} \underset{x}{\text{minimize}} & x_{n+q}, & \\ \text{subject to} & a^k x_k + B^k x \ge d^k, & k = n+1, n+2, \ldots, n+q, \\ & l_i \le x_i \le u_i, & i = 1, 2, \ldots, n+q, \\ & x \in X. & \end{cases}$$

Although finding a polyhedral superset of Θ_k is non-trivial, for each operator of a finite set $\mathcal{O}$ we can define a method to find one. We provide two examples. LP relaxations for monomials of odd degree such as $x_k = x_i^{2p+1}$, with $p \in \mathbb{Z}_+$, were proposed by Liberti and Pantelides (2003). For products of two variables $x_k = x_i x_j$, the following four inequalities proposed by McCormick (1976) provide the convex hull of the set

$$\Theta_k = \{(x_i, x_j, x_k) : x_k = x_i x_j, (l_i, l_j, l_k) \le (x_i, x_j, x_k) \le (u_i, u_j, u_k)\},$$

as proved by Al-Khayyal and Falk (1983):

$$\begin{aligned} x_k &\ge l_j x_i + l_i x_j - l_i l_j, & x_k &\le l_j x_i + u_i x_j - u_i l_j, \\ x_k &\ge u_j x_i + u_i x_j - u_i u_j, & x_k &\le u_j x_i + l_i x_j - l_i u_j. \end{aligned} \tag{5.11}$$

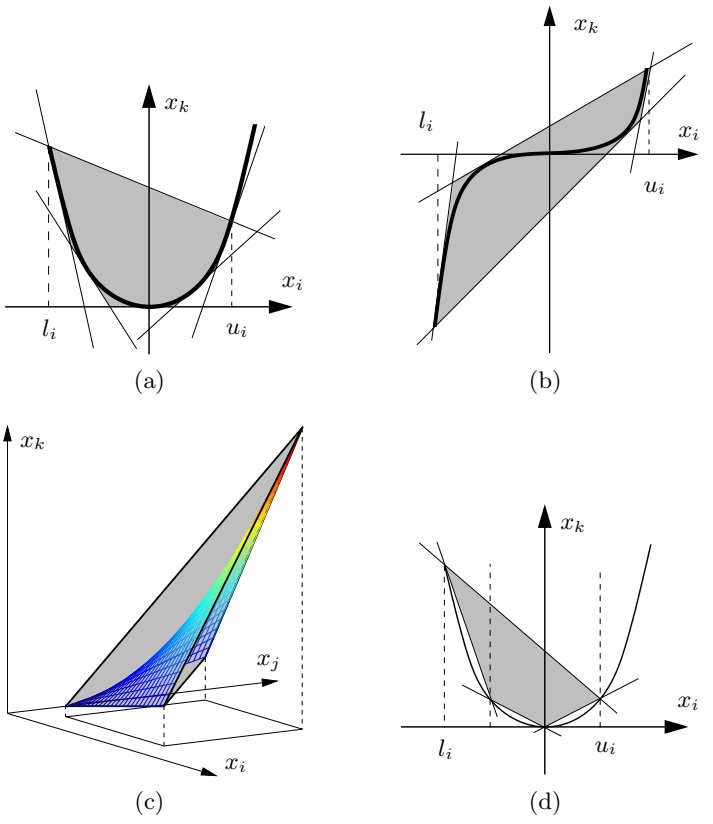


Figure 5.4. Polyhedral relaxations $\breve{\Theta}_k$ for several univariate and bivariate operators: $x_k = x_i^2$, $x_k = x_i^3$, $x_k = x_i x_j$, and $x_k = x_i^2$ with x_i integer. Note that these relaxations are *exact* at the bounds on x_i and x_j.

Figure 5.4 shows polyhedral relaxations for $x_k = x_i^2$, $x_k = x_i^3$, and $x_k = x_i x_j$. If the argument x_i of a function ϑ_k is integer, the linear relaxation can be strengthened. Suppose $x_j = (x_i)^2$ and $x_i \in \mathbb{Z} \cap [-2, 1]$, as in Figure 5.4(d). Then we can add linear inequalities that can be violated by points $(\hat{x}, \hat{x}^2)$ if $\hat{x} \notin \mathbb{Z}$.

Convex relaxations $\breve{\Theta}_k$ of the univariate or multivariate operator ϑ_k are required to be *exact* at the frontier of the bound interval of the arguments of ϑ_k. In particular, let $x_{i_1}, x_{i_2}, \ldots, x_{i_{h(k)}}$ be the arguments of ϑ_k, each of them bounded by the interval $[l_{i_j}, u_{i_j}]$. Define for each $j = 1, 2, \ldots, h(k)$ a value $b_{i_j} \in \{l_{i_j}, u_{i_j}\}$. Then

$$\breve{\Theta}_k \cap \{x \in \mathbb{R}^n : x_{i_j} = b_{i_j}\} = \Theta_k \cap \{x \in \mathbb{R}^n : x_{i_j} = b_{i_j}\}.$$

A direct consequence is the convergence result shown in Section 5.3.

5.2.2. Disjunctive cuts

In the class of convex MINLPs, the only non-convex constraints are represented by integer variables, and these non-convexities are resolved by integer branching, which represents a specific class of *disjunctions*. There are classes of non-convex MINLPs whose nonlinear objective and constraints introduce a different type of disjunction that can be used to create a valid cut. Disjunctive cuts for several classes of convex MINLP problems have been discussed in Section 4.3. For non-convex, factorable MINLP, disjunctions arise from branching rules $x_i \leq b \vee x_i \geq b$ on continuous variables. Other disjunctions that have important applications arise from *complementarity constraints*, that is, constraints of the form $x_i x_j = 0$ that are equivalent to the disjunction $x_i = 0 \vee x_j = 0$. These constraints are the basis of disjunctive cuts developed by Júdice, Sherali, Ribeiro and Faustino (2006).

Disjunctive cuts can be developed by using the branching disjunction $x_i \leq b \vee x_i \geq b$ in MINLPs with factorable functions. Suppose a branching rule is enforced because an auxiliary variable x_k and its related non-convex constraint $x_k = \vartheta_k(x)$ are such that $\hat{x}_k \neq \vartheta_k(\hat{x})$ for a given LP solution $\hat{x}$. A branching rule allows us to refine the LP relaxation of a subproblem as shown in Figure 5.5 (page 79). Figure 5.5(a) depicts the set $\breve{\Theta}_j$ for $x_j = x_i^2$ and shows that $\hat{x}_j \neq \hat{x}_i^2$. The disjunction $x_i \leq b \vee x_i \geq b$ can be used to create two new subproblems, NLP(l^-, u^-) and NLP(l^+, u^+), and consequently two tighter LP relaxations, LP(l^-, u^-) and LP(l^+, u^+), shown in Figure 5.5(b); note that both relaxations exclude $(\hat{x}_i, \hat{x}_j)$. For an example with a different function $x_j = \mathrm{e}^{x_i}$, see Figure 5.5(c). Belotti (2012) creates disjunctive cuts by means of a *cut generating LP* (CGLP) (Balas, Ceria and Cornuéjols 1993), a linear optimization problem used to devise a disjunctive cut that maximizes the violation with respect to $\hat{x}$. The CGLP obtains a disjunctive cut that is valid for both LP(l^-, u^-) and LP(l^+, u^+) and that hence exploits the disjunction to eliminate infeasible solutions without actually creating two subproblems.

This procedure retains the pros and cons of its MILP version: although it allows us to avoid creating two new subproblems and can be effective, each of these cuts is generated by solving very large LPs. Therefore, this procedure can be computationally expensive for large MINLPs.

5.3. Spatial branch-and-bound

The best-known method for solving non-convex MINLP problems is *branch-and-bound* (BB). Most of the global optimization community usually refers to these methods as *spatial BB* (sBB). As outlined in Section 3, a BB method is an implicit enumeration technique that recursively partitions the feasible set. This partitioning yields two or more subproblems whose solution sets are, ideally, *disjoint* from one another, in order to avoid evaluating

a feasible solution in more than one subproblem. Most BB implementations for MINLP use the reformulation scheme outlined in Section 5.2.1 to obtain a lower bound (Sahinidis 1996, Ryoo and Sahinidis 1996, Tawarmalani and Sahinidis 2002, Ryoo and Sahinidis 1995, Tawarmalani and Sahinidis 2004, Smith and Pantelides 1997, Belotti *et al.* 2009).

Other approaches employ a relaxation technique called *α-convexification* (Androulakis, Maranas and Floudas 1995). For a non-convex quadratic function $f(x) = x^T Q x + c^T x$ with $x \in [l, u]$, a valid lower bound is provided by

$$\breve{f}(x) = x^T Q x + c^T x + \alpha \sum_{i=1}^{n} (x_i - l_i)(x_i - u_i).$$

Note that $\breve{f}(x) \leq f(x)$ for $x \in [l, u]$ and that $\breve{f}(x)$ can be rewritten as a quadratic function $x^T P x + d^T x$, where $P = Q + \alpha I$, and $\breve{f}(x)$ is convex if $P \succeq 0$. Therefore it suffices to set $\alpha = -\lambda_{\min}(Q)$, namely, the opposite of the minimum eigenvalue of Q, to obtain a convex relaxation. Some implementations of the α-convexification adopt a quadratic approximation of the objective and the constraints and hence guarantee optimality only for quadratic problems (Nowak, Alperin and Vigerske 2003). This method can be extended to non-quadratic functions while preserving the validity of the lower bound. A generalization of this method is at the base of the MINLP solver GloMIQO (Misener and Floudas 2013).

A BB algorithm requires (i) a procedure to compute a lower bound on the optimal objective function value of a subproblem and (ii) a procedure for partitioning the feasible set of a subproblem. In general, the former consists of obtaining a convex relaxation of a subproblem NLP(l, u) and solving it to optimality, while the latter generates two new subproblems, NLP(l^-, u^-) and NLP(l^+, u^+), by appropriately setting new variable bounds. We discuss these techniques in detail below.

The structure of a branch-and-bound algorithm for non-convex MINLP follows the scheme described in Section 3, and we will not repeat it here. The generic subproblem NLP(l, u) of the BB can be defined as a restriction of the original MINLP (1.1) as follows:

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X, \\ & l_i \leq x_i \leq u_i, \ \forall i = 1, 2, \ldots, n, \\ & x_i \in \mathbb{Z}, \ \forall i \in I. \end{cases} \tag{5.12}$$

At subproblem NLP(l, u), the BB algorithm seeks a lower bound of the optimal value of $f(x)$ by solving a convex relaxation such as the LP

relaxation LP(l, u):

$$\begin{cases} \underset{x}{\text{minimize}} & x_{n+q}, \\ \text{subject to} & a^k x_k + B^k x \geq d^k, \quad k = n+1, n+2, \ldots, n+q, \\ & l_i \leq x_i \leq u_i, \qquad\quad i = 1, 2, \ldots, n+q, \\ & x \in X. \end{cases} \tag{5.13}$$

Suppose an optimal solution $\hat{x}$ of LP(l, u) is found. If $\hat{x}$ is feasible for (5.12) and hence for (1.1), subproblem NLP(l, u) can be eliminated. If $\hat{x}$ is infeasible for (5.12), then at least one of the following two holds.

(1) $\hat{x}$ is not integer feasible, *i.e.*, there exists $i \in I$ such that $\hat{x}_i \notin \mathbb{Z}$.

(2) At least one of the continuous non-convex constraints of the reformulation is violated, that is,

$$\text{there exists } k \in \{n+1, n+2, \ldots, n+q\} \text{ such that } \hat{x}_k \neq \vartheta_k(\hat{x}).$$

In the first case, one can generate two new subproblems NLP(l^-, u^-) and NLP(l^+, u^+), whose feasible sets $\mathcal{F}(l^-, u^-)$ and $\mathcal{F}(l^+, u^+)$ are amended new bounds on x_i through the branching rule $x_i \leq \lfloor \hat{x}_i \rfloor \vee x_i \geq \lceil \hat{x}_i \rceil$. In the second case, branching may be necessary on a continuous variable. In that case, suppose that x_i is among the arguments of function ϑ_k. Then the branching rule $x_i \leq \hat{x}_i \vee x_i \geq \hat{x}_i$ creates two new subproblems whose feasible sets have a non-empty intersection, where $x_i = \hat{x}_i$. This constitutes a strong point of departure with the subclass of pure integer non-convex MINLPs, where all variables are integer, and with convex MINLP discussed in Section 3. For these subclasses, branching is necessary only on integer variables. Finite bounds on integer variables ensures finite termination of the branch-and-bound algorithm.

Consider the feasible set of a subproblem NLP(l, u) of (1.1):

$$\mathcal{F}(l, u) = \{x \in [l, u] : c_i(x) \leq 0, \ \forall i = 1, 2, \ldots, m, x \in X, x_i \in \mathbb{Z}, i \in I\},$$

whose only difference from the feasible set of (1.1) is new bounds on the variables, as dictated by the branching rules. Consider a branching rule on a continuous variable x_i, subdividing the feasible set of subproblem NLP(l, u) into two subproblems NLP(l^-, u^-) and NLP(l^+, u^+), with feasible sets $\mathcal{F}(l^-, u^-)$ and $\mathcal{F}(l^+, u^+)$. A *bounding operation* yields two subproblems NLP(l^-, u^-) and NLP(l^+, u^+), by applying a branching rule, and lower bounds $\lambda_{\mathcal{F}(l^-, u^-)}$, $\lambda_{\mathcal{F}(l^+, u^+)}$ and upper bounds $\mu_{\mathcal{F}(l^-, u^-)}$, $\mu_{\mathcal{F}(l^+, u^+)}$ for the new subproblems. Such a bounding operation is said to be *consistent* if, at every step, subsets $\mathcal{F}(l^-, u^-)$ and $\mathcal{F}(l^+, u^+)$ are either fathomed or can be further refined in such a way that, for any infinite sequence $\{\mathcal{F}_h\}$ resulting from applying bounding operations, one can guarantee that (Horst and Tuy 1993)

$$\lim_{h \to \infty} \mu_{\mathcal{F}_h} - \lambda_{\mathcal{F}_h} = 0.$$

In addition, a bounding operation is *finitely consistent* if any sequence $\{\mathcal{F}_h\}$ of successively refined partitions of $\mathcal{F}$ is finite. Branching on continuous variables does not imply directly that a finite number of branching rules will be used, yet both in theory and in practice BB algorithms do have finite termination properties, as shown by McCormick (1976) and Horst and Tuy (1993).

Theorem 5.1 (McCormick 1976, Horst & Tuy 1993). If the bounding operation in the BB algorithm is finitely consistent, the algorithm terminates in a finite number of steps.

The value of this result can be made clearer if one considers an MINLP with even just one continuous variable x_1: by branching only on integer variables (in a finite number of BB nodes if all integer variables are bounded), one eventually obtains a possibly non-convex continuous optimization problem. Therefore, branching will become necessary on the continuous variable x_1 as well, although termination is no longer guaranteed by integrality. The result by McCormick (1976) states that convergence is still ensured as long as the bounding operation is finitely consistent.

5.3.1. Spatial branching

Partitioning the feasible set of a subproblem $\mathrm{NLP}(l, u)$ yields $h \geq 2$ new subproblems $\mathrm{NLP}(l', u')$, $\mathrm{NLP}(l'', u''), \ldots, \mathrm{NLP}(l^{(h)}, u^{(h)})$, whose lower bounds $\lambda_{\mathrm{NLP}(l',u')}, \lambda_{\mathrm{NLP}(l'',u'')}, \ldots, \lambda_{\mathrm{NLP}(l^{(h)},u^{(h)})}$ are no smaller than the lower bound of $\mathrm{NLP}(l, u)$. We will assume without loss of generality that two new problems $\mathrm{NLP}(l^-, u^-)$ and $\mathrm{NLP}(l^+, u^+)$ are created. Most practical implementations adopt a *variable branching* $x_i \leq b \vee x_i \geq b$. The performance of the BB algorithm depends strongly on the choice of i and b (Belotti *et al.* 2009, Tawarmalani and Sahinidis 2002). An integer variable is obviously a candidate for selection as a branching variable if its value is fractional in the LP solution. In the remainder of this section, we assume that all integrally constrained variables are integer and hence no branching is possible (integer branching has been discussed in Section 3) and that branching is done because of an auxiliary x_k such that $\hat{x}_k \neq \vartheta_k(\hat{x})$.

An ideal choice of i should balance more than one objective: it should (i) increase both lower bounds $\lambda_{\mathrm{NLP}(l^-,u^-)}$ and $\lambda_{\mathrm{NLP}(l^+,u^+)}$; (ii) shrink both feasible sets $\mathcal{F}(l^-, u^-)$ and $\mathcal{F}(l^+, u^+)$; and (iii) allow for a *balanced* BB tree, among other criteria.

Suppose an optimal solution $\hat{x}$ of the LP relaxation $\mathrm{LP}(l, u)$ is found. A continuous variable x_i is a candidate for branching if it is not fixed (*i.e.*, its lower and upper bounds do not coincide), if it is an argument of a function $\vartheta_k(x)$ associated with an auxiliary variable x_k, and $\hat{x}_k \neq \vartheta_k(x)$. For example, if $x_k = \vartheta_k(x) = x_i x_j$, $\hat{x}_k \neq \hat{x}_i \hat{x}_j$, and $l_i < u_i$, then x_i is a candidate for branching.

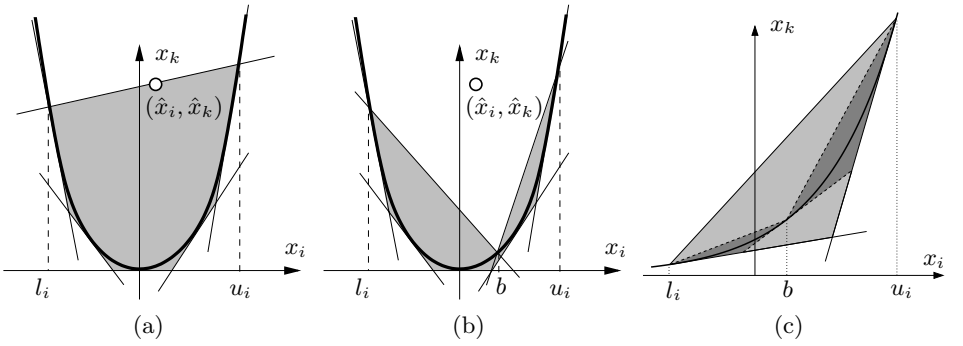


Figure 5.5. Polyhedral relaxations upon branching. (a) The set $\breve{\Theta}_k$ is shown with the components $(\hat{x}_i, \hat{x}_k)$ of the LP solution. Branching on x_i excludes the LP solution: see (b). (c) The LP relaxation before and after branching is shown for $x_k = \mathrm{e}^{x_i}$ in light and dark grey, respectively.

Upon branching, the two generated subproblems will each obtain a lower bound by solving two tighter relaxations than that of their ancestor. Geometrical intuition is provided in Figure 5.5. Suppose the auxiliary x_k is defined as $x_k = \vartheta_k(x_i) = (x_i)^2$ and $x_i \in [l_i, u_i]$ for this subproblem. Because the LP solution $\hat{x}$ is such that $\hat{x}_k \neq (\hat{x}_i)^2$ (see Figure 5.5(a)), one can generate two new subproblems using the branching rule $x_i \leq b \vee x_i \geq b$. The new linear relaxations are the polytopes in Figure 5.5(b), which are disjoint except for the point (b, b^2) and which exclude the point $(\hat{x}_i, \hat{x}_k)$. Figure 5.5(c) provides a similar example for $x_k = \vartheta_k(x_i) = \mathrm{e}^{x_i}$.

Tawarmalani and Sahinidis (2002) introduce *violation transfer* (VT) as a variable selection technique. VT identifies the variable x_i that has the largest impact on the violation of the non-convex constraints $x_k = \vartheta_k(x)$ for all $k = 1, 2, \ldots, n+q$ such that x_i is an argument of ϑ_k. Strong branching, pseudocost branching, and reliability branching, discussed in Section 3, can be applied with little modification to non-convex MINLP and have been implemented in non-convex MINLP solvers (Belotti *et al.* 2009).

The choice of branching point is also crucial, and differs from integer branching in that we have the freedom to choose a branching point for variable x_i that can differ from $\hat{x}_i$. A branching rule should ensure that $\hat{x}$ is infeasible for both $\mathrm{LP}(l^-, u^-)$ and $\mathrm{LP}(l^+, u^+)$; hence the sole branching rule $x_i \leq \hat{x}_i \vee x_i \geq \hat{x}_i$ will not suffice. However, the refined linear relaxations $\mathrm{LP}(l^-, u^-)$ and $\mathrm{LP}(l^+, u^+)$ will be obtained by adding linear inequalities that are violated by $\hat{x}$. While the linear inequalities depend on the new bounds on x_i, setting the branch point to a suitable $b \neq x_i$ does not prevent us from excluding $\hat{x}$.

5.3.2. Bound tightening

Bound tightening (also referred to as bound reduction or domain reduction) is a class of algorithms aiming to reduce the bound intervals on the variables of (1.1). Although these algorithms are optional in a BB solver, they are crucial to obtaining an optimal solution in reasonable time and are therefore implemented in the vast majority of MINLP solvers.

Their importance is directly connected to the LP relaxation (5.13): the tighter the variable bounds, the tighter the linear polyhedra $\breve{\Theta}_k$ for each auxiliary variable x_k and hence the better the lower bound on the objective. Some MINLP solvers use bound reduction as the sole means of obtaining a lower bound on the optimal objective function value of the subproblem (Messine 2004).

The usefulness of bound reduction is not limited to a tighter bound hyperrectangle: as a result of bound reduction, the feasible set might become empty, or the lower bound l_{n+q} on x_{n+q} is above the *cutoff* value of the problem, namely, the objective function value of a feasible solution of (1.1). In these two cases, the procedure proves that the current node is infeasible and can be fathomed without the need to compute a lower bound by solving the convex relaxation.

Consider again the feasible set of an MINLP problem:

$$\mathcal{F} = \{x \in [l, u] : c_i(x) \leq 0,\ \forall i = 1, 2, \ldots, m, x \in X, x_i \in \mathbb{Z}, i \in I\},$$

and suppose that a feasible solution of (1.1) is known with value $\check{z}$. For each variable $x_i, i = 1, 2, \ldots, n$, valid (and possibly tighter) lower and upper bounds are given by

$$l_i' = \min\{x_i : x \in S, f(x) \leq \check{z}\}, \quad u_i' = \max\{x_i : x \in S, f(x) \leq \check{z}\}. \quad (5.14)$$

Solving the $2n$ optimization problems above would yield tighter bounds, but these problems can be as hard as problem (1.1) itself. The two most important bound reduction techniques are *feasibility-based* (FBBT) and *optimality-based bound tightening* (OBBT). Other commonly used techniques are *probing* and *reduced-cost tightening*; we present these techniques below.

Feasibility-based bound tightening. FBBT has been used in the artificial intelligence literature (Davis 1987) and is a strong component of constraint programming solvers. It is part of nonlinear optimization solvers (Messine 2004) as well as of MILP solvers (Andersen and Andersen 1995, Savelsbergh 1994).

FBBT works by inferring tighter bounds on a variable x_i as a result of a changed bound on one or more other variables x_j that depend, directly or indirectly, on x_i. For example, if $x_j = x_i^3$ and $x_i \in [l_i, u_i]$, then the bound interval of x_j can be tightened to $[l_j, u_j] \cap [l_i^3, u_i^3]$. *Vice versa*, a tightened bound l_j' on x_j implies a possibly tighter bound x_i, *i.e.*, $l_i' = \sqrt[3]{l_j'}$. Another

example is given by $x_k = x_i x_j$, with $(1,1,0) \le (x_i, x_j, x_k) \le (5,5,2)$. Lower bounds $l_i = l_j = 1$ imply a tighter lower bound $l_k = l_i l_j = 1 > 0$, while the upper bound $u_k = 2$ implies that $x_i \le \frac{u_k}{l_j}$ and $x_j \le \frac{u_k}{l_i}$, and hence $u'_i = u'_j = 2 < 5$.

Perhaps the best-known example applies to affine functions. Suppose x_k is an auxiliary variable defined as $x_k = a_0 + \sum_{j=1}^{n} a_j x_j$, with $k > n$. Suppose also that

$$J^+ = \{j = 1, 2, \ldots, n : a_j > 0\} \quad \text{and} \quad J^- = \{j = 1, 2, \ldots, n : a_j < 0\}.$$

Then valid bounds on x_k are

$$a_0 + \sum_{j \in J^-} a_j u_j + \sum_{j \in J^+} a_j l_j \le x_k \le a_0 + \sum_{j \in J^-} a_j l_j + \sum_{j \in J^+} a_j u_j.$$

Moreover, explicit bounds $[l_k, u_k]$ on x_k imply new (possibly tighter) bounds l'_j, u'_j on $x_j, j = 1, 2, \ldots, n : a_j \ne 0$:

$$\begin{aligned}
\text{for } a_j > 0, \quad l'_j &= \frac{1}{a_j}\left(l_k - \left(a_0 + \sum_{i \in J^+ \setminus \{j\}} a_i u_i + \sum_{i \in J^-} a_i l_i\right)\right), \\
u'_j &= \frac{1}{a_j}\left(u_k - \left(a_0 + \sum_{i \in J^+ \setminus \{j\}} a_i l_i + \sum_{i \in J^-} a_i u_i\right)\right); \\
\text{for } a_j < 0, \quad l'_j &= \frac{1}{a_j}\left(u_k - \left(a_0 + \sum_{i \in J^+} a_i l_i + \sum_{i \in J^- \setminus \{j\}} a_i u_i\right)\right), \\
u'_j &= \frac{1}{a_j}\left(l_k - \left(a_0 + \sum_{i \in J^+} a_i u_i + \sum_{i \in J^- \setminus \{j\}} a_i l_i\right)\right).
\end{aligned} \tag{5.15}$$

These *implied bounds* are often used as a preprocessing technique (Andersen and Andersen 1995) prior to solving MILP problems. For both MILP and MINLP problems, bound reduction can be obtained by using *pairs* of inequalities (Belotti 2013), specifically through the convex combination of two inequalities $a^T x \ge \alpha$ and $b^T x \ge \beta$ using a parameter $\lambda \in [0,1]$. The resulting inequality $(\lambda a + (1-\lambda)b)^T x \ge \lambda\alpha + (1-\lambda)\beta$ yields bounds on the variables similar to (5.15), but these are a function of λ and can be shown to be tighter than those obtained by single inequalities.

For the general nonlinear case, variable bounds are propagated by using the DAG of the problem. For instance, consider problem (5.9) and bounds $[-4, 4]$ and $[0, 10]$ on x_1 and x_2, respectively. We rewrite the DAG of this problem to reflect these bounds and to show each auxiliary variable next to the root of the expression tree associated to it: see Figure 5.6.

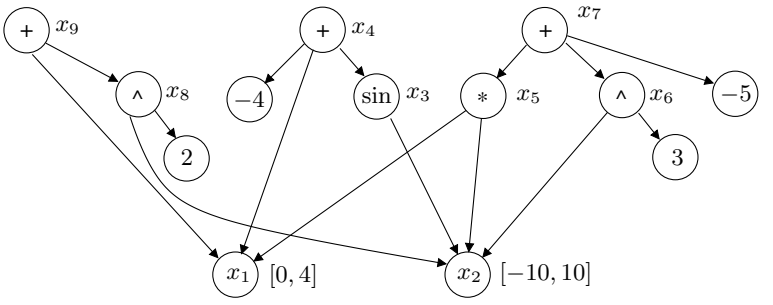


Figure 5.6. Association between auxiliary variables and the nodes of the DAG related to the problem in (5.9).

If a solution $\hat{x}$ is found with $f(\hat{x}) = 10$, then an upper bound on the objective function $x_9 := x_1 + x_8$ and the lower bound on $x_1 \geq -4$ imply that $x_8 \leq 14 < 100$. This in turn is propagated to the expression $x_8 = x_2^2$, which implies that $-\sqrt{14} \leq x_2 \leq \sqrt{14}$, thus tightening x_2. No other variables are tightened because of the new cutoff. In the general case, this procedure *propagates* throughout the DAG of the problem and repeats while there are tightened bounds, terminating when no more bounds are reduced.

FBBT algorithms allow for fast implementation and are commonly used in problems even of very large size. However, they may exhibit convergence issues even at very small scale: consider the trivial problem $\min\{x_1 : x_1 = \alpha x_2, x_2 = \alpha x_1, x_1 \in [-1, 1]\}$, with $\alpha \in \mathbb{R} \setminus \{0, 1\}$. Although by inspection one can see that the only feasible solution $x = (0, 0)$ is also optimal, FBBT will not terminate in a finite number of steps. In fact, a first pass will tighten x_2 to $\left[-\frac{1}{\alpha}, \frac{1}{\alpha}\right]$; this will trigger a reduction of the bound interval of x_1 to $\left[-\frac{1}{\alpha^2}, \frac{1}{\alpha^2}\right]$, which in turn will propagate to yield new bounds $\left[-\frac{1}{\alpha^3}, \frac{1}{\alpha^3}\right]$ on x_2. This procedure does not terminate unless tolerances or iteration limits are imposed, and it does not achieve its *fixed point* in finite time. A linear optimization problem has been proposed for MINLP that achieves the fixed point of a FBBT algorithm applied to the linear relaxation of (1.1) (Belotti, Cafieri, Lee and Liberti 2010).

Optimality-based bound tightening. Solving problems (5.14) is impractical because of the non-convexity of their feasible set, which is the same feasible set of (1.1). A more practical approach considers the feasible set of a convex relaxation of (1.1), such as the one in (5.13):

$$\mathcal{F}(l, u) = \left\{ x \in \mathbb{R}^{n+q} \;\middle|\; \begin{array}{ll} a^k x_k + B^k x \geq d^k, & k = n+1, n+2, \ldots, n+q, \\ l_i \leq x_i \leq u_i, & i = 1, 2, \ldots, n+q, \\ x \in X. & \end{array} \right\}$$

Then the following are valid bounds on variable x_i:

$$l_i' = \min\{x_i : x \in \mathcal{F}(l, u), f(x) \leq \hat{z}\}, \quad u_i' = \max\{x_i : x \in \mathcal{F}(l, u), f(x) \leq \hat{z}\}.$$

Empirical evidence shows that this technique is effective at obtaining tight bounds (Belotti *et al.* 2009), but it requires solution of $2n$ linear programming problems. Thus, its use is limited to the root node or to nodes of small depth.

Probing and reduced-cost bound tightening. Consider the bounds $[l_i, u_i]$ on a variable x_i as defined in (5.13), and set the upper bound to a fictitious value $u'_i < u_i$, regardless of whether u'_i is valid. Then apply a bound-tightening procedure such as FBBT. If the procedure indicates that the tightened bound hyper-rectangle renders the problem infeasible or drives the lower bound l_{n+q} on x_{n+q} above the cutoff value, then we have proof that no optimal solution exists with $x_i \in [l_i, u'_i]$, and the bounds on x_i become $[u'_i, u_i]$.

The same procedure can be repeated by imposing a fictitious lower bound l'_i on x_i and verifying through bound tightening whether tightening x_i to $[l'_i, u_i]$ yields a problem that can be fathomed. Applying this procedure to all variables, possibly in a repeated fashion, can lead to massive reduction in bounds, but it is computationally expensive given that it requires multiple calls to other bound-tightening procedures. Probing is used in MILP (Savelsbergh 1994) and MINLP (Belotti *et al.* 2009, Tawarmalani and Sahinidis 2002) on binary, integer, and continuous variables.

Reduced-cost bound tightening (Ryoo and Sahinidis 1995), akin to the *reduced-cost fixing* technique used in MILP (Nemhauser and Wolsey 1988), uses the solution of an LP relaxation of (1.1) to infer new, and possibly tighter, bounds on the variables. Suppose that an optimal solution $\hat{x}$ of (5.13) has a variable x_i at its lower bound l_i. Suppose also that the optimal solution has an objective function value $z_{\text{LP}} = \hat{x}_{n+q}$ and that a cutoff is known for (1.1) with value $\check{z}$. If the reduced-cost ρ_i of x_i is positive, then increasing x_i by δ yields an increase in the objective function of $\rho_i\delta$. Then a valid upper bound on x_i is $u'_i = l_i + \frac{\check{z}-z_{\text{LP}}}{\rho_i}$, which constitutes a tightening if $u'_i < u_i$. Similarly, if for the optimal solution $x^\star$ we have $x_i = u_i$ and a negative reduced cost ρ_i, then a valid lower bound on x_i is $l'_i = u_i + \frac{\check{z}-z_{\text{LP}}}{\rho_i}$.

5.4. Relaxations of structured non-convex sets

The methods described in Sections 5.2 and 5.3 are broadly applicable. This approach can be used to relax any constraint containing a nonlinear function that can be factored into simpler primitive functions for which we have known relaxations, and then to refine this relaxation after spatial branching. When combined with relaxation and branching on integer variables, this leads to algorithms that can (theoretically) solve almost any MINLP with explicitly given nonlinear constraints. The drawback of this general approach is that the relaxation obtained may be weak compared with the

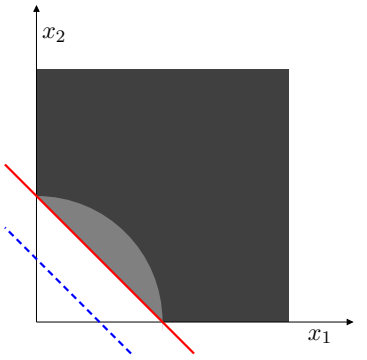


Figure 5.7. The dark shaded area is the feasible region. The convex hull is the entire shaded area, and is defined by the solid line, $x_1 + x_2 \geq 1$. The dashed line corresponds to the weaker inequality $x_1 + x_2 \geq 1/2$.

tightest possible relaxation, the convex hull of feasible solutions, leading to an impractically large branch-and-bound search tree.

As a simple example, consider the non-convex constraint in two variables x_1 and x_2,

$$x_1^2 + x_2^2 \geq 1, \tag{5.16}$$

and suppose $x_1, x_2 \in [0, 2]$. The set defined by these constraints is the dark shaded area in Figure 5.7. One can easily see that the convex hull of this set is given by the bounds on the variables, plus the inequality $x_1 + x_2 \geq 1$ (the solid line). Now consider the relaxation approach of Section 5.2. We first introduce two new decision variables, x_3 and x_4, with $x_3 \leq x_1^2$ and $x_4 \leq x_2^2$, and replace the constraint (5.16) with

$$x_3 + x_4 \geq 1. \tag{5.17}$$

The non-convex constraints $x_3 \leq x_1^2$ and $x_4 \leq x_2^2$ are then relaxed (in the best possible way given the bounds on x_1 and x_2) with $x_3 \leq 2x_1$ and $x_4 \leq 2x_2$. To compare this with the convex hull, we can then eliminate the variables x_3 and x_4 by substituting these inequalities into (5.17), obtaining $2x_1 + 2x_2 \geq x_3 + x_4 \geq 1$, or $x_1 + x_2 \geq 1/2$ (the bold dashed line in the figure). This relaxation is therefore significantly weaker than the convex hull.

Examples like those in the preceding paragraph motivate the study of relaxations that consider more of the problem *simultaneously*, rather than just separately relaxing all components. Of course, considering the entire MINLP feasible region is in general an intractable task. One common strategy, however, is to identify specific structures that may appear in many MINLP problems and study improved relaxations for these structures.

A huge variety of such structures exists, so we cannot provide a complete survey in this paper; but in the following subsections we highlight a couple of examples.

5.4.1. Non-convex quadratic functions

One structure that appears in many MINLP problems is the presence of (non-convex) quadratic or bilinear functions in either the constraints or the objective. These *quadratically constrained quadratic programs* (QCQPs) may also include integer variables and linear constraints. A generic QCQP is a special case of MINLP of the following form:

$$\begin{cases} \underset{x}{\text{minimize}} & x^T Q_0 x + c_0^T x, \\ \text{subject to} & x^T Q_k x + c_k^T x \le b_k, \quad k = 1, \dots, q, \\ & Ax \le b, \\ & 0 \le x \le u, \ x_i \in \mathbb{Z}, \ \forall i \in I, \end{cases} \tag{5.18}$$

where for each $k = 0, 1, \dots, q$, Q_k is an $n \times n$ symmetric matrix, and A is an $m \times n$ matrix. The matrices Q_k are not assumed to be positive semidefinite, so this problem is non-convex even when the integrality constraints are relaxed. It is also possible to have non-zero lower bounds $x \ge \ell$, but we assume here $x \ge 0$ to simplify exposition.

Many relaxation strategies for QCQPs are based on introducing additional variables X_{ij} for all i, j pairs, and reformulating (5.18) as follows:

$$\begin{cases} \underset{x}{\text{minimize}} & Q_0 \bullet X + c_0^T x, \\ \text{subject to} & Q_k \bullet X + c_k^T x \le b_k, \quad k = 1, \dots, q, \\ & Ax \le b, \\ & 0 \le x \le u, \ x_i \in \mathbb{Z}, \ \forall i \in I, \\ & X = xx^T, \end{cases} \tag{5.19}$$

where X is the $n \times n$ matrix containing all the X_{ij} variables, so that the constraint $X = xx^T$ records the non-convex constraints $X_{ij} = x_i x_j$ for all $i, j = 1, \dots, n$. Observe that these constraints are the only nonlinear constraints in this reformulation. Obtaining a relaxation of (5.19) can then be accomplished by relaxing the constraint $X = xx^T$. We discuss two general approaches for relaxing this constraint: the reformulation–linearization technique (RLT) (Adams and Sherali 1986, Sherali and Alameddine 1992, Sherali and Adams 1998) and semidefinite programming. Both approaches have been widely studied, and an exhaustive literature review is beyond the scope of this work. Instead, we introduce the basic idea of each as an example of how *structured* non-convex constraints can be relaxed.

In its most basic form, RLT relaxes the constraint $X_{ij} = x_i x_j$ for any fixed i, j by first deriving the following nonlinear non-convex constraints,

based on multiplying pairs of the non-negative quantities $x_i, x_j, u_i - x_i$, and $u_j - x_j$:

$$x_i x_j \geq 0, \quad (u_i - x_i)(u_j - x_j) \geq 0, \quad x_i(u_j - x_j) \geq 0, \quad (u_i - x_i)x_j \geq 0.$$

These inequalities are then *linearized* by replacing the products $x_i x_j$ with the variable X_{ij}, yielding

$$X_{ij} \geq 0, \quad X_{ij} \geq u_i x_j + u_j x_i - u_i u_j, \quad X_{ij} \leq u_j x_i, \quad X_{ij} \leq u_i x_j.$$

Observe that these inequalities are exactly the special case of the inequalities (5.11) where the lower bounds on the variables being multiplied are 0. Using these inequalities in place of $X = xx^T$ yields a polyhedral relaxation of (5.19). Two other techniques are commonly used to further strengthen the RLT relaxation. First, if a decision variable x_i, for $i \in I$, is binary (*i.e.*, $u_i = 1$), then it holds that $x_i^2 = x_i$, and hence the linear constraint $X_{ii} = x_i$ is added to the relaxation. A generalization of this technique to general integer variables, based on Lagrange interpolating polynomials, has been proposed as well (Adams 2011). The second major technique for further improving the RLT relaxation is to multiply linear constraints together to obtain additional quadratic constraints that can then be linearized. For example, multiplying a non-negative decision variable x_i with a linear constraint $b_t - \sum_{j=1}^n a_{tj} x_j \geq 0$ yields the inequality

$$b_t x_i - \sum_{j=1}^n a_{tj} x_i x_j \geq 0,$$

which can then be linearized as

$$b_t x_i - \sum_{j=1}^n a_{tj} X_{ij} \geq 0.$$

Similar inequalities can be derived by multiplying linear constraints with the non-negative terms $(u_i - x_i)$, and also by multiplying linear constraints with each other, although deriving inequalities from all possible pairs of linear inequalities may yield a very large linear program.

The other general technique for relaxing the constraint $X = xx^T$ in (5.19) is via semidefinite programming. The key observation is that this constraint $X - xx^T$ can be relaxed to the constraint $X - xx^T \succeq 0$, which is equivalent to

$$\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \succeq 0,$$

and hence this yields a semidefinite programming relaxation. Just as in RLT, the corresponding relaxation can be improved by including the constraint $X_{ii} = x_i$ for binary variables x_i. When the QCQP contains linear *equations*, such as $a_t x = b_t$, additional linear constraints for the SDP relaxation can be obtained by 'squaring' the constraints and then linearizing

the variables (Anstreicher 2009):

$$a_t^T X a_t = b_t^2.$$

SDP relaxations can be used within a branch-and-bound algorithm to solve QCQPs to optimality (see Burer and Vandenbussche 2009 and Buchheim and Wiegele 2013).

Anstreicher (2009) compares the relaxations obtained using the RLT and SDP approaches, finding that neither strictly dominates the other, and that combining the two approaches can yield a relaxation significantly better than obtained by using either approach individually. Additional linear inequalities related to those obtained for the Boolean Quadric Polytope (Padberg 1989) can also be added to further strengthen the relaxation (Yajima and Fujie 1998, Burer and Letchford 2009). Anstreicher (2012) demonstrated that the resulting relaxations can be exceptionally tight, very often yielding a bound equal to the optimal objective value. Unfortunately, the resulting relaxations, although convex, may be computationally demanding. Consequently, linear cuts have been introduced by Sherali and Fraticelli (2002) and further refined by Qualizza, Belotti and Margot (2012) to provide a polyhedral approximation of the SDP constraint.

For both the RLT and SDP relaxation approaches, improved relaxations can also be obtained by further multiplying linear constraints with each other, obtaining higher-order polynomial constraints. Additional variables can then be introduced, as in (5.19), to linearize these constraints, and then the constraints defining these variables (such as $X_{i,j,k} = x_i x_j x_k$) can be relaxed using an approach similar to that used for (5.19). This approach leads to a hierarchy of relaxations of improving quality (Adams and Sherali 2005, Lasserre 2000, Lasserre 2001). The drawback of this approach is that the size of these formulations grows dramatically, limiting their current practical use. Indeed, even the formulation (5.19) may be significantly larger than the original formulation (5.18), leading Saxena, Bonami and Lee (2011) to study relaxations of QCQPs that do not use the additional variables X_{ij}.

For MINLPs with only a quadratic objective (*i.e.*, problem (5.18) with $q = 0$), Burer (2009) derived another important link with conic optimization. In particular, he showed that an *exact* reformulation of such a problem can be obtained using a reformulation similar to (5.19) with certain enhancements (such as including the constraints $X_{ii} = x_i$ for binary variables) and replacing the constraint $X = xx^T$ with the conic constraint

$$\begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \in \mathcal{C},$$

where $\mathcal{C}$ is the *completely positive cone*, the set of matrices Y such that $Y = BB^T$ for some matrix B that is componentwise non-negative.

We close this section by mentioning a few other approaches for solving or relaxing QCQPs. Saxena, Bonami and Lee (2010) study techniques for deriving disjunctive cuts based on the non-convex constraint $xx^T - X \succeq 0$, which is also implied by the constraint $X = xx^T$. Vandenbussche and Nemhauser (2005*b*, 2005*a*) studied polyhedral relaxations of box-constrained QPs, in which a general quadratic function is to be minimized subject to bound constraints on the continuous decision variables. Linderoth (2005) studied a simplicial branch-and-bound algorithm for QCQP. Burer and Letchford (2013) study relaxations of QCQPs with unbounded integer decision variables. The general α-BB relaxation approach (Androulakis *et al.* 1995) has also been used for solving QCQPs, although Anstreicher (2012) showed that the bounds obtained from SDP relaxation are always at least as good as those obtained from the α-BB approach. Misener and Floudas (2012) study the additional use of piecewise linear and edge-concave relaxations of QCQPs. Bao, Sahinidis and Tawarmalani (2009) and Luedtke, Namazifar and Linderoth (2012) consider related approaches for relaxing *multilinear* functions in which the decision variables are bounded, which can also be applied to QCQPs.

5.4.2. Bilinear covering sets

Tawarmalani, Richard and Chung (2010) study a framework for generating valid inequalities for MINLPs that have a certain 'orthogonal disjunction' structure. We do not review this general work but instead highlight the results they obtain by applying this framework to sets they refer to as *bilinear covering sets*.

First consider a pure integer covering set defined by

$$B^{\mathrm{I}} := \left\{ (x, y) \in \mathbb{Z}^n_+ \times \mathbb{Z}^n_+ \mid \sum_{i=1}^{n} x_i y_i \geq r \right\},$$

where r is a positive number. Note that, for any i, the convex hull of the two variable integer set

$$B^{\mathrm{I}}_i := \{(x_i, y_i) \in \mathbb{Z}_+ \times \mathbb{Z}_+ \mid x_i y_i \geq r\}$$

is a polyhedron defined by $d \leq \lceil r \rceil + 1$ linear inequalities. Let us denote the inequalities defining the convex hull of B^{I}_i by

$$a^k x_i + b^k y_i \geq 1, \quad k = 1, \ldots, d, \tag{5.20}$$

where we can assume (by scaling) that each inequality has a right-hand side 1. In particular, these inequalities include the constraints $x_i \geq 1$ and $y_i \geq 1$. The remaining inequalities can be computed, for example, by finding all inequalities of the form $ax_i + by_i \geq 1$ that do not cut off any of the points $(x^t_i, y^t_i) = (t, \lceil r/t \rceil)$ for $t = 1, \ldots, \lceil r \rceil$ and that are exactly satisfied by two of these points.

Now, let Π be the collection of all possible mappings of the form $\pi : \{1,\dots,n\} \to \{1,\dots,d\}$. That is, if $\pi \in \Pi$, then for each $i \in 1,\dots,n$, $\pi(i)$ selects an inequality in the description of $\mathrm{conv}(B_i^{\mathrm{I}})$. Then, $\mathrm{conv}(B^{\mathrm{I}})$ is characterized as follows.

Theorem 5.2 (Tawarmalani *et al.* 2010, Proposition 6). The convex hull of B^{I} is given by the set of $x \in \mathbb{R}^n_+, y \in \mathbb{R}^n_+$ that satisfy the inequalities

$$\sum_{i=1}^{n} (a^{\pi(i)} x_i + b^{\pi(i)} y_i) \geq 1, \quad \text{for all } \pi \in \Pi. \tag{5.21}$$

While there are an exponential number of inequalities in (5.21), given a point $(\hat{x}, \hat{y}) \in \mathbb{R}^n_+ \times \mathbb{R}^n_+$, separation can be accomplished efficiently by independently considering each $\hat{x}_i, \hat{y}_i$ pair and setting $\pi(i)$ to the index of the most violated constraint in (5.20). Tawarmalani *et al.* (2010) provide a very similar result for the case of a bilinear covering set similar to B^{I}, but where one of the sets of variables is continuous.

We now turn to the case of a continuous bilinear covering set of the form

$$B^{\mathrm{C}} := \left\{ (x,y) \in \mathbb{R}^n_+ \times \mathbb{R}^n_+ \mid \sum_{i=1}^{n} (a_i x_i y_i + b_i x_i + c_i y_i) \geq r \right\},$$

where $r > 0$ and $a_i, b_i, c_i > 0$ for $i = 1,\dots,n$.

Theorem 5.3 (Tawarmalani *et al.* 2010, Proposition 9). The convex hull of B^{C} is given by the set of $x \in \mathbb{R}^n_+, y \in \mathbb{R}^n_+$ that satisfy the inequality

$$\frac{1}{2} \sum_{i=1}^{n} \left(b_i x_i + c_i y_i + \sqrt{(b_i x_i + c_i y_i)^2 + 4 a_i r x_i y_i} \right) \geq r.$$

Unfortunately, when the bounds on the variables are also considered (*e.g.*, if x is a set of binary variables), the corresponding covering sets become much more difficult to analyse. In particular, Chung, Richard and Tawarmalani (2011) show that optimizing a linear function over such a set is $\mathcal{NP}$-hard. Chung *et al.* (2011) did, however, study valid inequalities for this set that may be useful in strengthening the continuous relaxation.

6. Heuristics for solving MINLPs

Some real-world applications cannot be solved to global optimality by using the methods described in Sections 3–5, because the problems are too large, generate a huge search tree, or must be solved in real time. In these situations it is more desirable to obtain a good solution quickly than to wait for an optimal solution. In such situations, we may resort to heuristic search techniques that provide a feasible point without any optimality guarantees. Heuristics can also accelerate deterministic techniques by quickly

identifying an incumbent with a low value of the objective function. This upper bound can then be used to prune a larger number of the nodes in the branch-and-bound algorithm of Section 3.1 and in the branch-and-cut algorithm of Section 3.1.4. An incumbent solution may additionally be used for 'guided dives' (Danna, Rothberg and LePape 2005), that is, selecting a child node during a dive. Bound tightening, described in Section 5.3.2, can also be applied to the objective function more effectively if a tight upper bound is known.

We distinguish two classes of heuristic search techniques: probabilistic search and deterministic search. Probabilistic search refers to techniques that require at each iteration a random choice of a candidate solution or parameters that determine a solution. Simulated annealing (Kirkpatrick, Gelatt and Vecchi 1983), ant colony optimization (Dorigo, Maniezzo and Colorni 1996), particle-swarm optimization (Kennedy and Eberhart 1995), cross-entropy (Rubinstein and Kroese 2004), tabu search (Glover 1989, Glover 1990) and genetic algorithms (Goldberg 1989) are some methods that fall into this category. Although simple to design and applicable to many combinatorial optimization problems, these methods require implementation and modifications specific to the structure of the problem being solved. We therefore focus only on more general approaches.

We use the term 'deterministic' rather loosely, since the methods in this category may also sometimes need randomization in certain iterations. To begin with, all deterministic techniques discussed in Sections 3–5 can be run as heuristics. For example, we can run branch-and-bound for a fixed time or fixed number of nodes or until it finds its first incumbent. In this section, we discuss more efficient alternatives to such simple heuristics. These heuristics can be classified into two types: search heuristics, which search for a solution without the help of any known solutions, and improvement heuristics, which improve upon a given solution or a set of solutions.

Notation. Throughout this section we use a unified notation to refer to incumbents and solutions of the various steps of the heuristics: x^* refers to the current incumbent, which is feasible in (1.1); x' refers to a (local) solution of the continuous relaxation (1.3); $x^\diamond$ denotes the solution to a polyhedral relaxation of (1.1); and $x^{(j)}$ is a (local) solution of an NLP with fixed integer variables, (NLP($x_I^{(j)}$)). Given this notation, we can now describe the deterministic search techniques within a unified terminology.

6.1. Search heuristics

Several heuristics to search for a feasible solution of an MINLP have been proposed recently. They all make clever use of LP, MILP, and NLP solvers to solve problems easier than the MINLP to obtain a feasible point. Some of these heuristics may completely ignore the objective function and focus on

finding only a feasible solution. They may use the solution of the relaxation at any node in the branch-and-bound as a starting point and hence try to make up for the lack of focus on the objective function of the MINLP.

6.1.1. MILP-based rounding

Finding a locally optimal solution to the continuous relaxation (1.3) of the MINLP (1.1) is usually easier and computationally faster than solving the MINLP itself. Given a solution x' of the continuous relaxation, one can try rounding fractional values of integer-constrained variables. Unfortunately, such a simple rounding will usually not produce a feasible solution. Nannicini and Belotti (2012) propose overcoming this difficulty by solving an MILP. The constraints of the MILP are linear relaxations of the original MINLP obtained by the methods described in Section 5.2. The objective function is the ℓ_1 norm $\|x - x'\|_1$. The solution $x^\diamond$ of the MILP satisfies the integrality constraints but not necessarily the nonlinear constraints. Another NLP is now solved, this time with all integer variables fixed to the values in $x^\diamond_I$. The process is repeated until we obtain a feasible solution or reach termination criteria (limits on time or iterations).

Nannicini and Belotti (2012) suggest several practical measures for implementing the above general scheme. First, it is not necessary to fully solve the MILP. We can stop as soon as a feasible point of the MILP is found. Second, different initial points x' can be used to initialize the heuristic. In particular, if we are solving the NLP with an interior point method, we can stop the NLP if it finds a feasible point even when the log-barrier coefficient is not close to zero. Third, no-good cuts are added to ensure that the MILP does not include integer solutions ($x^\diamond$) found in previous iterations of the heuristic. A no-good cut is used to model the constraint

$$\sum_{i \in I} |x_i - x^\diamond_i| \geq 1. \tag{6.1}$$

When all integer variables are binary, the constraint (6.1) is simplified to

$$\sum_{i \in I : x^\diamond_i = 0} x_i + \sum_{i \in I : x^\diamond_i = 1} (1 - x_i) \geq 1. \tag{6.2}$$

Auxiliary binary variables may need to be introduced when some variables are general integers instead of binary.

6.1.2. Feasibility pump

The feasibility pump heuristic was introduced by Fischetti, Glover and Lodi (2005) in the context of MILP, and improved by Achterberg and Berthold (2007) and Fischetti and Salvagnin (2009). Bonami, Cornuéjols, Lodi and Margot (2009) have extended this idea to MINLPs. The main idea, like that in MILP-based rounding described above, is that an NLP solver can be used

to find a solution that satisfies nonlinear constraints. Integrality is enforced by solving an MILP. An alternating sequence of NLP and MILP is solved that may lead to a solution feasible for the MINLP. The main difference from the rounding approach of Section 6.1.1 is the way MILP is set up. Suppose x' is a locally optimal solution of the NLP relaxation of the MINLP (1.1). Bonami *et al.* (2009) obtain the MILP using the linearization (3.4) used for outer approximation. Thus, if we have a convex MINLP, the MILP is a relaxation. Otherwise, it is only an approximation. The objective function is again the ℓ_1 norm $\|x - x'\|_1$.

If the solution $x^\diamond$ to the MILP satisfies the nonlinear MINLP constraints, $c(x^\diamond) \leq 0$, then we have found a new incumbent. Otherwise, we solve an NLP with all the integer variables fixed to the values of $x^\diamond$. The objective of this NLP is $\|x - x^\diamond\|_2$. Now, x' is updated to the solution of this NLP. If x' does not satisfy integrality constraints, new linearizations are added to the previous MILP.

For a convex MINLP, the more restricted MILP does not contain $x^\diamond$, and hence the no-good cuts are not required. In addition, Bonami *et al.* (2009) add the valid inequality

$$(x' - x^\diamond)(x - x') \geq 0$$

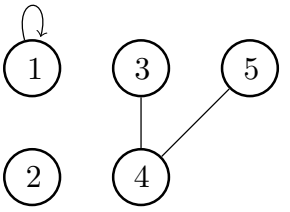
to the MILP. They also prove that in the case of a convex MINLP the heuristic does not cycle, and always terminates in a finite number of iterations. They call their version of the feasibility pump for convex MINLPs the 'enhanced feasibility pump'. A simpler version of the feasibility pump (Bonami and Gonçalves 2012) does not involve solving an MILP. Instead one just rounds x' to the nearest point satisfying integrality constraints. This version requires random changes in x' whenever cycling occurs, similar to the work of Fischetti *et al.* (2005).

D'Ambrosio, Frangioni, Liberti and Lodi (2012) note that feasibility-pump-based heuristics can be viewed as applications of the sequential projection method (SPM) or the alternating projection methods. The SPM has been used extensively for solving convex feasibility problems. We refer to the survey by Bauschke and Borwein (1996) for theory and algorithms. The feasibility of MINLPs is not a convex problem, and so the heuristics along the lines of these algorithms cannot be expected to have a similar performance or even to converge. D'Ambrosio *et al.* (2012) consider two sets related to the feasible region of the MINLP (1.1):

$$A = \big\{x \mid c(x) \leq 0,\ x \in X\big\}, \quad \text{and} \tag{6.3}$$

$$B = \big\{x \mid c_C(x) \leq 0,\ x \in X,\ x_i \in \mathbb{Z},\ \forall\, i \in I\big\}, \tag{6.4}$$

where c_C refers to the convex constraints in the MINLP. Set A is in general non-convex, while B is a set of feasible points of a convex MINLP. One can now solve optimization problems over A and B repeatedly in order to obtain



(a)

$$\begin{cases} \text{minimize}_x & x_1^3 + x_2 + x_3 x_4 + x_5, \\ \text{subject to} & x_4 x_5, \\ & x_1, x_2, x_3, x_4, x_5 \in [0, 10]. \end{cases}$$

(b)

Figure 6.1. (a) Graph denoting the non-zero structure of the Hessian of the example in (b).

two sequences of solutions $\bar{x}_i$ and $\hat{x}_i$ as follows:

$$\bar{x}_i = \min_{x \in A} \|x - \hat{x}_{i-1}\|,$$
$$\hat{x}_i = \min_{x \in B} \|x - \bar{x}_{i-1}\|.$$

Both the above problems are NP-hard, and D'Ambrosio *et al.* (2012) suggest solving them using well-known heuristics. For instance, a locally optimal solution of the first problem can be obtained by most NLP solvers. The latter problem can be solved as a convex MINLP by methods described in Section 3. By selecting different methods to solve these problems, one can obtain several variants of the feasibility pump.

6.1.3. Undercover

The 'undercover' heuristic (Berthold and Gleixner 2012) is specially designed for non-convex MINLPs. The basic idea is to fix certain variables in the problem to specific values so that the resulting restriction becomes easier to solve. The restriction that Berthold and Gleixner (2012) obtain is an MILP. This MILP can then be solved either exactly or heuristically. Since the MILP is a restriction of the MINLP, any feasible solution of MILP will also satisfy the MINLP.

In order to be successful, the heuristic should fix a minimal number of variables lest the reduction be too restrictive and good solutions be cut off. The sparsity pattern of the Hessian of the Lagrangian tells us which variables appear in nonlinear functions. Berthold and Gleixner create a graph $G(V, E)$ where each vertex $v_i \in V, i = 1, \ldots, n$ denotes a variable of the MINLP. An edge e_{ij} is added to the graph if the Hessian of the Lagrangian of the NLP relaxation has a non-zero entry (i, j). G may also contain loops if a diagonal entry in the Hessian is non-zero. We illustrate a graph for a simple example in Figure 6.1. To make the problem linear, we can fix the variables x_1, x_3, and x_5. However, this choice is not minimal because we can also fix x_1 and x_4 to obtain a linear problem. Berthold and

Gleixner (2012) observe that a minimal vertex-cover of G is also a minimal set of variables that can be fixed to make the problem linear. They solve the minimal vertex-cover problem using an MILP solver that is usually available in an MINLP framework.

The fixed values of variables in a cover are obtained from a solution of the NLP relaxation or an LP relaxation. The variables are fixed sequentially. Each time a variable is fixed, domain propagation is invoked to tighten bounds or to fix other variables. This heuristic works particularly well for problems with quadratic constraints and objective function.

6.1.4. RENS: relaxation enforced neighbourhood search

The RENS (Berthold 2012) heuristic searches for a feasible solution of an MINLP around a point that does not satisfy integrality constraints. Consider an NLP solution x' and suppose $F = \{i \in I \mid x'_i \notin \mathbb{Z}\}$ is the set of all variables that violate integrality constraints at x'. Keeping the values of the variables in the set $I \setminus F$ fixed to those of x', we have $2^{|F|}$ ways of rounding up or down x' so that it satisfies integrality constraints. The RENS heuristic seeks to systematically search through these $2^{|F|}$ combinations to find a feasible solution of the MINLP.

Berthold (2012) creates a possibly much smaller MINLP in order to search for the feasible solution. First, he fixes all variables in the set I that have an integer value in x'. Next, the bounds of each variable $x_i, i \in F$ are changed to $[\lfloor x'_i \rfloor, \lceil x'_i \rceil]$. The resulting MINLP is then solved by using an MINLP solver. If the restricted MINLP is considerably smaller than the original, then the solver can solve it quickly. Additional limits need to be imposed on the solver so that the heuristic does not take excessive time. By means of this heuristic, the authors show that more than half of the test-instances have solutions that could be obtained by rounding.

6.1.5. Diving

The fundamental idea of all diving heuristics is to conduct a depth-first exploration of a possible path from the root node to a leaf node of the branch-and-bound tree, before searching other branches. The hope is that this will lead to a feasible solution and hence an upper bound early in the MINLP solution process.

Bonami and Gonçalves (2012) propose starting the diving process by solving the NLP relaxation of the MINLP to obtain a relaxed solution $x' \in \mathbb{R}^n$. They then fix $x'_i \neq \mathbb{Z}$, $i \in I$ to $\lfloor x'_i \rfloor$ or $\lceil x'_i \rceil$ and re-solve the modified NLP. This process is iterated until all integer variables have been fixed. This heuristic is successful if the obtained leaf NLP is feasible for the MINLP. It reports a failure if the obtained leaf NLP is infeasible, the objective function exceeds the incumbent bound, or an NLP solver termination criterion is met such as a limit on the number of iterations.

Selection of the variable to be fixed and the side to which it is fixed leaves room for some tailoring of diving heuristics. Bonami and Gonçalves (2012) describe *fractional diving*, *vector length diving*, and a modification of these heuristics we refer to as *nonlinear diving*:

- In *fractional diving*, the variable to be rounded is selected from the set of smallest values $|x'_j - [x_j]|$, where the bracket $[\cdot]$ indicates rounding to the nearest integer. The selected variable is fixed to the nearest integer.
- In *vector length diving*, the variable is selected from the set of smallest ratios

$$\frac{(\lceil x'_j \rceil - x'_j)g_j + \varepsilon}{A_j + 1} \text{ if } g_j \geq 0, \quad \text{and} \quad \frac{(\lfloor x'_j \rfloor - x'_j)g_j + \varepsilon}{A_j + 1} \text{ otherwise,}$$

  for $j \in I$. Here, $g_j = \frac{\partial f(x')}{\partial x_j}$, the constant A_j indicates the number of problem functions for which x_j, $j \in I$ has a non-zero coefficient in the linearization, and ε is chosen to be a small positive constant, for example $\varepsilon = 10^{-6}$. The selected variable is rounded up if the gradient with respect to x_j is non-negative, and it is rounded down otherwise. The selection favours rounding of a variable that incurs a small objective function change but affects a large number of problem constraints.

The nonlinear diving heuristic may be applied to either of the above criteria. Here, x_i, $i \in I$ is selected from the subset of nonlinear fractional variables only, with the aim of obtaining an MILP that can then be solved by a (black-box) MILP solver. In a leaf node where all nonlinear integer variables have been fixed and at least one linear integer variable is still fractional, this MILP is obtained by fixing all continuous nonlinear variables to their current value.

Diving heuristics need to take into account the computational effort required for repeated re-solves of the modified NLP. To mitigate this cost, Bonami and Gonçalves (2012) propose to fix $K > 1$ variables at the same time before resolving the modified NLP. Mahajan *et al.* (2012) propose solving QPs instead of NLPs, which speeds up the diving process by exploiting the warm-starting capabilities of active set QP solvers.

6.2. Improvement heuristics

Improvement heuristics start with a given feasible point x^* of the MINLP and try to find a better point. Two well-known heuristics for searching a better solution in the neighbourhood of a known solution have been adapted from MILP to MINLP. We describe them next.

6.2.1. Local branching

Local branching is a heuristic for MINLPs where all integer variables are binary, that is, $x_i \in \{0,1\}$ for all $i \in I$. It was first introduced in the context of MILP by Fischetti and Lodi (2003) and generalizes readily to convex MINLPs. We start by describing local branching for convex MINLPs and then describe an extension to non-convex MINLPs.

The main idea behind local branching is to use a generic MILP solver at a tactical level that is controlled at a strategic level by a simple external branching framework. Assume that we are given a feasible incumbent x^* of (1.1), and consider the following disjunction (generalized branching) for a fixed constant $k \in \mathbb{Z}$:

$$\|x_I - x_I^*\|_1 \leq k \ \text{(left branch)} \quad \text{or} \quad \|x_I - x_I^*\|_1 \geq k+1 \ \text{(right branch)}. \tag{6.5}$$

This disjunction corresponds to the Hamming distance of x_I from x_I^*, and the left branch can also be interpreted as an ℓ_1 trust region around the incumbent. In the case of binary variables, we can rewrite (6.5) as two linear constraints:

$$\begin{aligned} \sum_{i \in I: x_i^* = 0} x_i + \sum_{i \in I: x_i^* = 1} (1 - x_i) \leq k \ \ \text{(left)} \quad \text{or} \\ \sum_{i \in I: x_i^* = 0} x_i + \sum_{i \in I: x_i^* = 1} (1 - x_i) \geq k + 1 \ \ \text{(right)}. \end{aligned} \tag{6.6}$$

The left branch is constructed in such a way that it is much easier to solve than (1.1), typically by choosing $k \in [10, 20]$. We start by solving the left branch using any of the methods introduced in Section 3 and obtain a new incumbent. We can then either solve the right branch or again divide the right branch using a new disjunction (6.6). This creates an outer branch-and-bound tree where each node corresponds to an MINLP. If this local branching tree has been searched to completion, we have solved the MINLP. In general, however, we do not run local branching to completion, because it would be inefficient to regenerate pseudocosts for every MINLP solve, for example. Fischetti and Lodi (2003) propose two enhancements: first they impose a time limit or node limit on each MINLP solve, and second they introduce a diversification mechanism in case the left branch does not improve the solution. If the left branch is not solved completely, one can still obtain a complete search algorithm by modifying the local branching strategy.

Local branching has been extended to non-convex MINLPs (Nannicini, Belotti and Liberti 2008). The extension is based on solving an alternating sequence of (local) NLP relaxations and fixings and (global) MILP outer approximations, and is closely related to the feasibility pump described in Section 6.1.2. The local branching is used only as a no-good cut (6.1) and

is not viewed as a strategic (outer) branching technique. A related heuristic is RECIPE (Liberti, Mladenović and Nannicini 2011).

6.2.2. RINS: relaxation-induced neighbourhood search

In the RINS (Danna *et al.* 2005) heuristic, one searches for better solutions in the neighbourhood of an already known solution, much like local branching. However, instead of imposing a distance constraint (6.5) to determine a neighbourhood, variables are fixed to certain values. The variables to be fixed are selected on the basis of the solution of the relaxation x', and the already known incumbent x^*. For all $i \in I$ the variables are fixed to x_i^*, $x'_i = x_i^*$. If the fixing reduces the problem size considerably, then it can be solved by calling the solver again.

Bonami and Gonçalves (2012) extend this idea from MILP to the NLP-based branch-and-bound algorithm of Section 3.1 for convex MINLP. Once they fix the integer variables as above, they solve the smaller MINLP using the LP/NLP-BB algorithm mentioned in Section 3.3.1. They show that the LP/NLP-BB algorithm is much faster on the smaller problems than is the NLP-based branch-and-bound algorithm.

7. Mixed-integer optimal control problems

In this section, we are interested in mixed-integer nonlinear optimization problems constrained by a system of ordinary differential equations, also called mixed-integer optimal control problems (MIOCPs). In principle, these problems can be discretized in time and solved by using any of the techniques of Sections 3–5. Unfortunately, such a discretization destroys the structure of the problem and does not provide successful algorithms. We present an alternative partial outer convexification approach that exploits problem structures and provides an efficient approach to this class of problems.

7.1. Mixed-integer optimal control problem class

We start by defining a class of MIOCPs:

$$\left\{\begin{array}{lll} \underset{x(\cdot),u(\cdot),v(\cdot)}{\text{minimize}} & J(x(T)), & \\ \text{subject to} & \dot{x}(t) = f_{\text{ODE}}(x(t),u(t),v(t)), & \forall t \in [0,T], \\ & x(0) = x_0, & \\ & u(t) \in \mathcal{U}, & \forall t \in [0,T], \\ & 0 \le c(x(t),u(t)), & \forall t \in [0,T], \\ & v(t) \in \Omega, & \forall t \in [0,T], \end{array}\right. \tag{MIOCP}$$

where we strive to determine differential states

$$x : [0,T] \to \mathbb{R}^{n_x},$$

continuous controls

$$u : [0, T] \to \mathcal{U} \subset \mathbb{R}^{n_{\mathrm{u}}},$$

and integer controls

$$v : [0, T] \to \Omega \subset \mathbb{R}^{n_{\mathrm{v}}}$$

on the time horizon $[0, T]$ such that a Mayer term $J : \mathbb{R}^{n_{\mathrm{x}}} \to \mathbb{R}$ evaluated at the end of this horizon is minimized subject to a nonlinear system

$$f_{\mathrm{ODE}} : \mathbb{R}^{n_{\mathrm{x}}} \times \mathbb{R}^{n_{\mathrm{u}}} \times \Omega \to \mathbb{R}^{n_{\mathrm{x}}}$$

of ordinary differential equations (ODEs) with fixed initial values $x_0 \in \mathbb{R}^{n_{\mathrm{x}}}$. In addition, we consider nonlinear path constraints

$$c : \mathbb{R}^{n_{\mathrm{x}}} \times \mathcal{U} \to \mathbb{R}^{n_{\mathrm{c}}}$$

independent of the integer control. The set $\mathcal{U}$ describes a feasible domain for the continuous control $u(\cdot)$, and the set $\Omega := \{v^1, v^2, \ldots, v^{n_\omega}\}$ contains $n_\omega < \infty$ discrete choices $v^i \in \mathbb{R}^{n_{\mathrm{v}}}$, $i \in I := \{1, \ldots, n_\omega\}$ admissible for the integer control.

Mixed-integer optimal control problems can be discretized in time by using, for example, direct collocation or direct shooting discretizations and can then be solved by any of the MINLP algorithms in Sections 3–5. The discretized MINLPs, however, are typically challenging. Strong nonlinear and transient phenomena inherent in the dynamic model $f_{\mathrm{ODE}}(\cdot)$ may require a fine-grained time discretization, which leads to a large number of integer variables in the discretized MINLP. While this issue is typically successfully addressed by error estimation and adaptivity, incorporating adaptive discretizations into the MINLP model is difficult. Moreover, simply relaxing the integrality in (MIOCP) to $v(t) \in \operatorname{conv} \Omega$ yields a lower bound on the optimal solution that can be obtained by solving a continuous optimal control problem. Unfortunately, this relaxation in general does not give any guarantees on integer feasibility or on the distance to an integer optimal solution. In addition, the ODE constraint poses a particular challenge since the dynamics need not remain well-defined on the continuous relaxation.

To address these issues, we review an NLP-MILP decomposition approach for mixed-integer optimal control, as developed by Sager (2005), Sager, Reinelt and Bock (2009) and Sager, Bock and Diehl (2012). The fundamental insight here is that it is indeed sufficient to compute a relaxed optimal solution for a special convexified reformulation of the mixed-integer optimal control problem. With a direct method for continuous optimal control, this essentially amounts to solving an NLP. From this, we obtain a lower bound on the MIOCP solution, which, unlike in MINLP, can be shown to be tight. An integer feasible solution can be obtained constructively by applying a certain rounding scheme or by solving an MILP.

7.2. Partial outer convexification

To tighten the lower bound obtained from a relaxation and to overcome the problem of ill-defined dynamics, Sager (2005) proposes convexifying the problem with respect to the integer control $v(\cdot)$. In doing so, binary multiplier functions $\omega_i(\cdot) \in \{0,1\}$ are introduced for each admissible choice of $v(t)$. We represent the integer control as a convex combination,

$$\forall t \in [0,T], \quad v(t) = v^i \;\Leftrightarrow\; \omega_i(t) = 1,\; \omega_j(t) = 0,\; \forall j \in I,\; j \neq i. \tag{7.1}$$

This gives rise to a partially convexified MIOCP:

$$\left\{\begin{array}{lll} \underset{x(\cdot),u(\cdot),\omega(\cdot)}{\text{minimize}} & J(x(T)), & \\ \text{subject to} & \dot{x}(t) = \sum_{i\in I} \omega_i(t) f_{\text{ODE}}(x(t),u(t),v^i), & \forall t \in [0,T], \\ & x(0) = x_0, & \\ & u(t) \in \mathcal{U}, & \forall t \in [0,T], \\ & 0 \leq c(x(t),u(t)), & \forall t \in [0,T], \\ & 1 = \sum_{i\in I} \omega_i(t), & \forall t \in [0,T], \\ & \omega(t) \in \{0,1\}^{n_\omega}, & \forall t \in [0,T]. \end{array}\right. \tag{MIOCP-BC}$$

This problem is convex only if f_{ODE} and c are convex in x and u. Clearly, the sets of optimal solutions of (MIOCP) and (MIOCP-BC) coincide. Given optimal binary multiplier functions $\omega(\cdot)$, the corresponding original integer control is easily determined. If $v(\cdot)$ is binary and enters linearly, then (MIOCP) is already in the convexified form. Otherwise the special-ordered set of type-1, SOS1, may be used. Introduced by Beale and Tomlin (1970) in the context of MILP, an SOS1 set can be used to model the condition (7.1) by means of the equations

$$\begin{aligned} v(t) &= \sum_{i\in I} v^i \omega_i(t), \quad \text{for all } t \in [0,T], \\ \sum_{i\in I} \omega_i(t) &= 1, \qquad\qquad \text{for all } t \in [0,T]. \end{aligned} \tag{7.2}$$

We are interested in the relaxation of (MIOCP-BC) obtained by replacing $\omega_i(\cdot)$ by relaxed convex multiplier functions $\alpha_i(\cdot) \in [0,1]$. This relaxation is a purely continuous optimal control problem, denoted by (MIOCP-RC). Direct and all-at-once methods have emerged as the methods of choice to solve it efficiently.

As an illustration, we consider an energy-optimal rocket-car control problem in its integer nonlinear formulation on the left and its partially con-

vexified formulation on the right for $t \in [0, 32]$:

$$\begin{cases} \underset{x(\cdot),v(\cdot)}{\text{minimize}} & x_2(32), \\ \text{subject to} & \dot{x}_0(t) = x_1(t), \quad \forall t, \\ & \dot{x}_1(t) = v(t), \quad \forall t, \\ & \dot{x}_2(t) = v^2(t), \quad \forall t, \\ & x(0) = (0, 0, 0), \\ & x_0(32) = 300, \\ & v(t) \in \{-2, 1\}, \quad \forall t, \end{cases} \tag{7.3}$$

$$\iff \begin{cases} \underset{x(\cdot),\omega(\cdot)}{\text{minimize}} & x_2(32), \\ \text{subject to} & \dot{x}_0(t) = x_1(t), & \forall t, \\ & \dot{x}_1(t) = -2\omega_1(t) + \omega_2(t), & \forall t, \\ & \dot{x}_2(t) = 4\omega_1(t) + \omega_2(t), & \forall t, \\ & x(0) = (0, 0, 0), \\ & x_0(32) = 300, \\ & \omega(t) \in \{0, 1\}^2, & \forall t, \\ & \omega_1(t) + \omega_2(t) = 1, & \forall t. \end{cases}$$

The multiplier function $\omega_1(t)$ lends itself to elimination by using the SOS1 constraint.

For our example, we use the multiple shooting code MUSCOD-II for direct mixed-integer optimal control (Bock and Plitt 1984, Leineweber *et al.* 2003, Sager 2005, Kirches and Leyffer 2011) available on NEOS (Czyzyk, Mesnier and Moré 1998, Dolan, Fourer, Moré and Munson 2002).

7.3. Relaxation of the partially convexified MIOCP

Optimal solutions to control problems permit a classification into arcs, or sections, of $[0, T]$ on which the solution has a particular structure: see, for example, Bryson and Ho (1975) for an introduction. Of interest for (MIOCP-RC) are arcs on which the optimal solution to the relaxed multipliers $\alpha(\cdot)$ comes to lie on the boundary of the feasible set and is hence feasible also for (MIOCP-BC). One example of such behaviour is *bang-bang* arcs. For particular problem classes, such as time-optimal linear control problems, a bang-bang principle proves the existence of an optimal solution that consists of bang-bang arcs only; see the work of Kirches *et al.* (2010) for a MIOCP example. Solving (MIOCP-RC) then already yields an optimal solution of (MIOCP). In example (7.3), (MIOCP-RC) has a bang-bang arc on $[0, 3.5]$ approximately: see Figure 7.1(e).

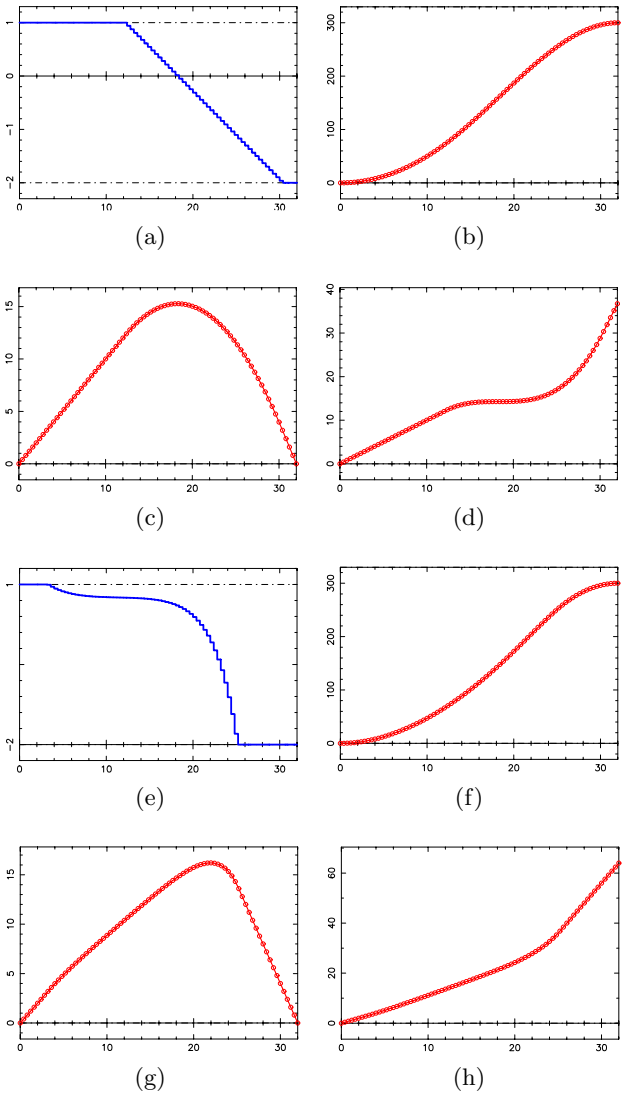


Figure 7.1. The nonlinear relaxed solution to the energy-optimal rocket-car example yields a weak lower bound $x_2(32) \approx 37$ to the MIOCP's optimal objective. The figure shows $v(\cdot) \in [-2, 1]$ relaxed (a), and partial outer convexification $v(\cdot) = -2\alpha_1(t) + 1\alpha_2(t)$ (e), as well as the resulting differential state trajectories $x_0(\cdot)$, $x_1(\cdot)$, $x_2(\cdot)$ (b–d, f–h) for an equidistant piecewise constant control discretization of $N = 80$ intervals.

Arcs on which $\alpha(\cdot)$ is not binary feasible require application of a rounding scheme to construct a binary feasible control $\omega(\cdot)$. We postpone discussion of such rounding schemes to the next section, and first consider the work of Sager *et al.* (2009, 2012), who investigate the difference between the state trajectories generated by relaxed multipliers $\alpha(\cdot)$ and binary ones $\omega(\cdot)$. We consider the ODE right-hand side $\tilde{f}(x,u)$ composed from columns $f_{\text{ODE}}(x,u,v^i)$, $i \in I$, and we fix a measurable control $u^*(\cdot)$.

Theorem 7.1 (Sager *et al.* 2012, Theorem 2 and Corollary 6). Let $x_\alpha(\cdot)$ and $x_\omega(\cdot)$ be the solution of the initial value problems

$$\begin{cases} \dot{x}_\alpha(t) = \tilde{f}(x_\alpha(t), u^*(t)) \cdot \alpha(t), & x_\alpha(0) = x_0, \\ \dot{x}_\omega(t) = \tilde{f}(x_\omega(t), u^*(t)) \cdot \omega(t), & x_\omega(0) = x_0, \end{cases} \tag{7.4}$$

with $t \in [0,T]$, $\alpha, \omega : [0,T] \to [0,1]^{n_\omega}$ measurable functions and $\tilde{f} : \mathbb{R}^{n_\text{x}+1} \to \mathbb{R}^{n_\text{x}} \times \mathbb{R}^{n_\omega}$ essentially bounded on $[0,T]$ by $M \in \mathbb{R}^+$ and t-differentiable almost everywhere. If there exist C, $L \in \mathbb{R}^+$ such that

$$\begin{aligned} \left\| \frac{\mathrm{d}\tilde{f}}{\mathrm{d}t}(x(t), u^*(t)) \right\| &\leq C, \\ \|\tilde{f}(x_\alpha(t), u^*(t)) - \tilde{f}(x_\omega(t), u^*(t))\| &\leq L\|x_\alpha(t) - x_\omega(t)\|, \end{aligned} \tag{7.5}$$

for $t \in [0,T]$ almost everywhere, and there exists $\varepsilon \in \mathbb{R}^+$ such that

$$\left\| \int_0^T \alpha(t) - \omega(t)\, \mathrm{d}t \right\| \leq \varepsilon, \tag{7.6}$$

then for all $t \in [0,T]$ it holds that

$$\|x_\alpha(t) - x_\omega(t)\| \leq (M + Ct)\varepsilon \exp(Lt). \tag{7.7}$$

This theorem differs markedly from the MINLP setting. Given a relaxed optimal solution $(x_\alpha(\cdot), u^*(\cdot), \alpha(\cdot))$, if one can construct a binary solution $\omega(\cdot)$ that satisfies assumption (7.6) on the control deviation bound ε, then the deviation of the resulting state trajectory $x_\omega(\cdot)$ from the relaxed optimal one $x_\alpha(\cdot)$ is bounded linearly in ε.

7.4. Constructing the integer control

A sum-up rounding scheme for construction of $\omega(\cdot)$ from a relaxed optimal solution $\alpha(\cdot)$ is shown to satisfy assumption (7.6) by Sager *et al.* (2012). We first present the case for an affine-linear binary control function $v(\cdot)$, when the SOS1 constraint is absent.

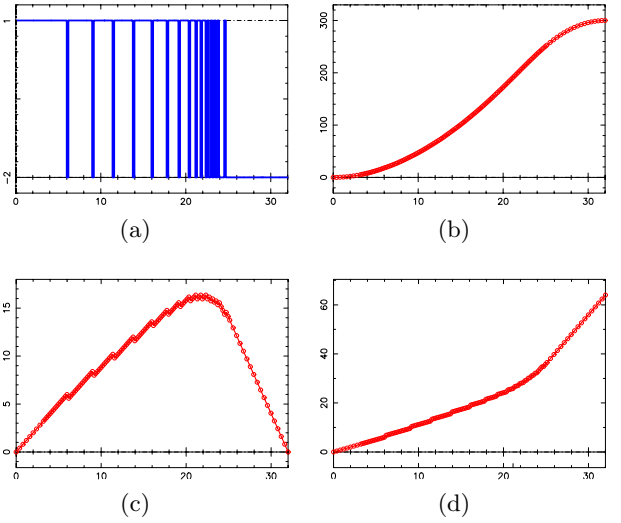


Figure 7.2. Integer feasible sum-up rounding solutions to the energy-optimal rocket-car example. The objective is $x_2(32) = 64$, and the lower bound provided by the convexified relaxed solution is exact in this case (*cf.* Figure 7.1(e–h)). The figure shows the integer control $v(\cdot)$ (a) and the resulting state trajectories $x_0(\cdot)$, $x_1(\cdot)$, $x_2(\cdot)$ (b–d).

Theorem 7.2 (Sager *et al.* 2012, Theorem 3). Let $\alpha\colon [0,T] \to [0,1]^{n_\omega}$ and a time grid $0 = t_0 < t_1 < \cdots < t_N = T$ be given, and define $\omega : [0,T] \to \{0,1\}^{n_\omega}$ by $\omega(t) := p_i$ for $t \in [t_i, t_{i+1})$, $0 \le i < N$, where

$$\begin{aligned} \hat{p}_{i,j} &:= \int_0^{t_i+1} \alpha_j(s)\,\mathrm{d}s - \sum_{k=0}^{i-1} p_{k,j}(t_{k+1} - t_k), \\ p_{i,j} &:= \begin{cases} 1 & \text{if } \hat{p}_{i,j} \ge \frac{1}{2}(t_{i+1} - t_i), \\ 0 & \text{else,} \end{cases} \qquad j \in I. \end{aligned} \tag{7.8}$$

Then it holds that

$$\left\| \int_0^T \alpha(t) - \omega(t)\,\mathrm{d}t \right\| \le \tfrac{1}{2} \max_{0 \le i < N} \{t_{i+1} - t_i\} =: \varepsilon. \tag{7.9}$$

After elimination of $\alpha_1(\cdot)$, example (7.3) is covered by this theorem. Figure 7.2 shows the sum-up rounding solution constructed from the convexified relaxed solution of Figure 7.1(e–h).

When the SOS1 constraint is present in (MIOCP-RC), $\omega(\cdot)$ constructed by the above rounding may violate it. Hence, a more elaborate rounding scheme is required.

Theorem 7.3 (Sager *et al.* 2012, Theorem 5). Let $\alpha\colon [0,T] \to [0,1]^{n_\omega}$ and a time grid $0 = t_0 < t_1 < \cdots < t_N = T$ be given, and define $\omega : [0,T] \to \{0,1\}^{n_\omega}$ by $\omega(t) := p_i$ for $t \in [t_i, t_{i+1})$, $0 \le i < N$, where

$$p_{i,j} := \begin{cases} 1 & \text{if } \forall k \neq j : \hat{p}_{i,j} \geq \hat{p}_{i,k} \text{ and } \forall k \neq j,\ \hat{p}_{i,j} = \hat{p}_{i,k} : j < k, \\ 0 & \text{else,} \end{cases} \quad j \in I, \tag{7.10}$$

and the $\hat{p}_{i,j}$ are defined as above. Then it holds that

$$\left\| \int_0^T \alpha(t) - \omega(t)\, \mathrm{d}t \right\| \leq \gamma(n^\omega) \max_{0 \le i < N} \{t_{i+1} - t_i\} =: \varepsilon. \tag{7.11}$$

For the constant $\gamma(n^\omega)$, the best-known result currently is $\gamma(n^\omega) := n^\omega - 1$ by Sager *et al.* (2012), who conjecture $\gamma(n^\omega) \in O(\log n^\omega)$. The practical implication of these theorems is that the bound ε of Theorem 7.1 can be made arbitrarily small by using a sufficiently fine time discretization, that is, N large enough to allow for $\max_{0 \le i < N}\{t_{i+1} - t_i\}$ to be sufficiently small. As a consequence of Theorems 7.1 and 7.2 or 7.3, the increase in the objective function value and the constraint infeasibility are bounded as well.

Theorem 7.4 (Sager *et al.* 2012, Corollary 8). Assume that J and c are continuous functions, and that the assumptions of Theorem 7.1 hold. Then for all $\delta > 0$ there exists $\Delta t := \max_{0 \le i < N}\{t_{i+1} - t_i\}$ sufficiently small that

$$|J(x_\alpha(T)) - J(x_\omega(T))| \leq \delta, \tag{7.12}$$

$$|c_i(x_\alpha(t), u^*(t)) - c_i(x_\omega(t), u^*(t))| \leq \delta, \quad 1 \le i \le n_\mathrm{c}. \tag{7.13}$$

The binary control $\omega(\cdot)$ obtained from the constructive sum-up rounding process may switch frequently, and the number of switches may increase along with N and need not be bounded, as an example due to Fuller (1963) in Figure 7.3 shows. Sager, Jung and Kirches (2011) hence consider this rounding scheme as a special-case solution to the following switch cost MILP, in which $\sigma_{k,\max}$ is a fixed upper bound on the number of switches allowed in the integer control,

$$\begin{cases} \underset{p}{\text{minimize}}\ \underset{0 \le i < N, k \in I}{\text{maximize}} & \sum_{j=1}^{i} |(\alpha_{jk} - p_{jk})(t_{j+1} - t_j)|, & \\ \text{subject to} & \sigma_{k,\max} \geq \sum_{i=0}^{N-1} |p_{ik} - p_{i+1,k}|, & k \in I, \\ & p_i \in \{0,1\}^{n_\omega}, & 0 \le i < N. \end{cases} \tag{7.14}$$

Here, $\omega(\cdot)$ is determined by the solution of this MILP, rather than by Theorem 7.2 or 7.3. If the switch cost constraint is inactive, these theorems give the linear-time solution to this MILP.

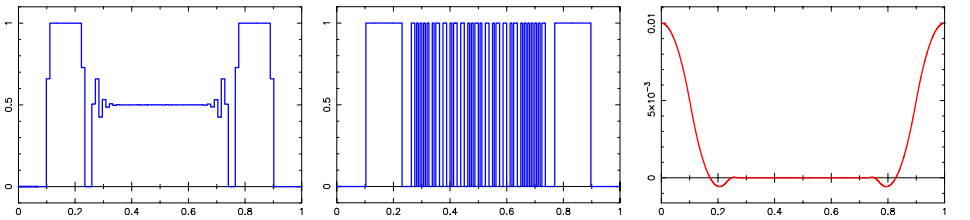


Figure 7.3. Fuller's problem (Fuller 1963) has an optimal solution that, in the limit $N \to \infty$, switches infinitely often on a finite time horizon. Relaxed optimal control (a), sum-up rounding control with optimized switching times (b), and optimal state trajectory (c) for $N = 80$.

Partial outer convexification, relaxation, and sum-up rounding can hence be seen as an NLP-MILP decomposition approach to solving a time-discretized MIOCP that enjoys ε-optimality and ε-feasibility certificates. The computational advantage of this approach over MINLP branch-and-bound algorithms for MIOCP can hence be arbitrarily large, as demonstrated by Kirches *et al.* (2010) for a benchmark problem due to Gerdts (2005).

7.5. Extensions to the presented theory

Problem class (MIOCP) is a special case of a larger class of mixed-integer optimal control problems also covered by the presented theory. An integral Lagrange term in the objective, time-dependent ODE systems, and free continuous model parameters constant in time are easily covered by introducing additional artificial ODE states. Free initial and final times subject to optimization can be realized by introducing two free parameters and an affine-linear transformation of the free horizon. Problem class (MIOCP) is also easily extended to allow for free initial values, for point constraints on the state $x(\cdot)$, including end-point constraints, and for constraints coupled in time, such as periodicity constraints.

Mixed-integer optimal control of systems described by differential-algebraic equations is discussed by Gerdts and Sager (2012). Because of time discretization, the switching times are determined up to the grid granularity only. Sager (2005) uses a switching-time optimization approach to refine the integer solution once the optimal switching structure has been determined and fixed. When (MIOCP) contains integer parameters subject to optimization, a combination of MIOCP techniques and MINLP techniques has to be used.

Constraints that depend on the integer control $v(t)$ are not immediately covered by (MIOCP) and the approximation theory based on Theorem 7.1, because violations of constraints after rounding cannot be ruled out. Kirches

Table 8.1. MINLP solvers (convex). A '—' means that we do not know the language the solver is written in. 'OS' stands for 'open source'.

Name	Algorithm(s)	Interfaces	Language	OS
α-ECP	extended cutting-plane	customized, GAMS	C	no
BONMIN	NLP-BB, LP/NLP-BB, OA, Hybrid	AMPL, C++, GAMS, Matlab	C++	yes
DICOPT	OA	GAMS	—	no
FilMINT	LP/NLP-BB	AMPL	C	no
KNITRO	NLP-BB, LP/NLP-BB	C, C++, Microsoft Excel, AMPL, AIMMS, GAMS, *etc.*	C, C++	no
MILANO	NLP-BB, OA	Matlab	Matlab	yes
MINLPBB	NLP-BB	AMPL, Fortran	Fortran-77	no
MINOPT	OA	customized	C	no
MINOTAUR	NLP-BB, QP-diving	AMPL, C++	C++	yes
SBB	NLP-BB	GAMS	—	no

(2011) discusses this case as well as extensions to mixed-integer nonlinear feedback control. An online benchmark library of mixed-integer optimal control problems is presented by Sager (2012).

8. Software for MINLP

The availability as well as the maturity of software for modelling and solving MINLP has increased significantly in the past fifteen years, and now includes a number of open-source and commercial solvers. We briefly survey the solvers currently available and describe their salient features. Recent surveys of Bussieck and Vigerske (2010) and D'Ambrosio and Lodi (2011) also provide an excellent description of available MINLP solvers. We divide the solvers into those for convex and non-convex MINLPs; little intersection exists between the two categories.

Key characteristics and features of the solvers are summarized in Tables 8.1 and 8.2, where we use the following abbreviations to indicate the type of MINLP method that the solver implements: NLP-BB for nonlinear branch-and-bound (Section 3.1); LP/NLP-BB for LP/NLP-based branch-and-bound (Section 3.3.1); OA for outer approximation (Section 3.2.1); Hybrid is a hybrid between OA and LP/NLP-BB; QP-diving (Section 3.1.3); α-BB for α-branch-and-bound (Section 5.3); LP-BB for LP-based branch-and-bound (Section 5.3).

Table 8.2. MINLP solvers (non-convex). A '—' means that we do not know the language the solver is written in. 'OS' stands for 'open source'.

Name	Algorithm(s)	Interfaces	Language	OS
α-BB	α-BB	customized	—	no
BARON	LP-BB	AIMMS, GAMS	—	no
COCONUT	LP-BB	AMPL, GAMS	C	yes
COUENNE	LP-BB	AMPL, C++, GAMS	C++	yes
GloMIQO	LP-BB	C++, GAMS	C++	no
LGO	sampling, heuristics	AIMMS, AMPL, GAMS, Mathematica	C	no
LindoGlobal	LP-BB	C++, GAMS, Lindo	—	no
SCIP	LP-BB	AMPL, C, GAMS, Matlab, OSiL, ZIMPL	C	yes

We are aware of two software packages, MUSCOD-II and MINOPT, that can solve MIOCPs. While MINOPT relies on MINLP techniques, MUSCOD-II relies on the partial outer convexification approach reviewed in Section 7. We briefly describe it in Section 8.3. Expressing and modelling MINLPs is significantly different from LPs or MILPs because general nonlinear functions are much more difficult to represent with data structures. Hence, good modelling tools are indispensable for MINLPs. In Section 8.4 we describe some tools available for modelling and reformulating MINLPs.

In Appendix A we list the websites of some of the software described below, along with other online resources.

8.1. Convex MINLP solvers

α-ECP is a solver written by Westerlund and Lundqvist (2005) to solve convex MINLPs by using the extended cutting-plane method (Section 3.2.3). It also implements methods for MINLPs with pseudoconvex functions (Westerlund and Pörn 2002). The solver can read MINLPs in an extended LP format and can be called through the GAMS (Brooke, Kendrick, Meeraus and Raman 1992) modelling system. It requires the user to specify an MILP solver for solving the MILP in each iteration. It also provides a graphical user interface for the Microsoft Windows operating system.

BONMIN stands for 'basic open-source nonlinear mixed-integer' optimizer. It is an open-source solver available from the COIN-OR website (Bonami *et al.* 2008). It uses nonlinear branch-and-bound (Section 3.1), LP/NLP-based branch-and-bound (Section 3.3.1), and outer approximation (Section 3.2.1) algorithms. It also implements a hybrid of outer approximation

and LP/NLP-based branch-and-bound. It features several primal heuristics, including the feasibility pump, diving, and RINS. It uses the CBC solver to perform all the MILP operations, such as management of cuts and tree search. It can solve NLPs using IPOPT (Wächter and Biegler 2006) or Filter-SQP (Fletcher and Leyffer 1998). The source code and documentation are available on the project website.

DICOPT stands for 'discrete and continuous optimizer'. It implements a variant of outer approximation (Section 3.2.1), which has been generalized to tackle non-convex MINLPs through a penalty function heuristic; see Viswanathan and Grossmann (1990). The problem is input through the GAMS modelling system. The user can specify options to select both the MILP solver and the NLP solver at each iteration of the algorithm.

FilMINT (Abhishek *et al.* 2010) implements the LP/NLP-BB algorithm (Section 3.3.1) with several practical improvements. It exploits the presolve, cutting planes, searching rules, and other MILP tools of the MINTO solver (Nemhauser, Savelsbergh and Sigismondi 1994). Filter-SQP (Fletcher and Leyffer 1998) is used to solve NLPs. It also implements some of the disjunctive cuts described in Section 4.3 and the feasibility pump heuristic (Section 6.1.2).

KNITRO (Byrd, Nocedal and Richard 2006) was initially designed as an NLP solver. Two branch-and-bound-based algorithms (Sections 3.1, 3.3.1) were recently added for solving convex MINLPs (KNITRO 2012).

MILANO is a Matlab-based solver for convex MINLPs. It implements nonlinear branch-and-bound (Section 3.1) and outer approximation (Section 3.2.1). The source code of MILANO is available on the project website. The main focus of this code is to develop efficient warm-starting methods for interior-point methods (Benson 2011, 2012) so as to make them more effective at solving MINLPs.

MINLPBB (Leyffer 1998) is a Fortran-based nonlinear branch-and-bound solver (Section 3.1) for convex MINLPs. Filter-SQP (Fletcher and Leyffer 1998) is used to solve the NLP relaxations. It also has the ability to restart the NLP iterations from different remote points in order to ensure better solutions for non-convex MINLPs, and it provides options for choosing different branching rules and tree search strategies: see Sections 3.1.1 and 3.1.2.

MINOPT is a framework for both modelling and solving MINLPs and MIOCPs developed in 1998 (Schweiger 1999). It uses generalized Benders decomposition (Section 3.2.2) and outer approximation (Section 3.2.1). MINOPT requires linking with an NLP solver and an MILP solver, for which it has built-in routines for different solvers. A license for MINOPT can be

obtained by contacting the authors. More information, a reference manual and examples are available on the project website.

MINOTAUR stands for 'mixed-integer nonlinear optimization toolkit: algorithms, underestimators and relaxations'. It is a new open-source toolkit for MINLPs. Currently it only implements nonlinear branch-and-bound (Section 3.1) and QP-diving (Section 3.1.3) for convex MINLPs. It has interfaces to NLP, QP and LP solvers. MINOTAUR has the ability to create and modify computational graphs of nonlinear functions. It can be used to reformulate nonlinear constraints and objective functions. The source code and documentation are available online.

MISQP is a solver designed for practical problems where the nonlinear functions cannot be evaluated when the variables $x_i, i \in I$ are not integers. This solver evaluates the functions and derivates at the integer points only. The algorithm does not guarantee an optimal solution even for convex MINLP. It generalizes the sequential–quadratic programming method with a trust region (Exler and Schittkowski 2007) to MINLPs. Exler, Lehmann and Schittkowski (2012) provide documentation of the solver along with examples. More information is available online.

SBB stands for 'simple branch-and-bound'. Implemented in GAMS, it allows the user to choose an NLP solver for the NLP-BB algorithm (Section 3.1). It can also handle SOS1 and SOS2 constraints (see (7.2) and (5.8)). A user manual is available online.

8.2. Non-convex MINLP solvers

Most modern MINLP solvers designed for non-convex problems utilize a combination of the techniques outlined in the previous sections; in particular, they are branch-and-bound algorithms with at least one rudimentary bound-tightening technique and a lower-bounding procedure. The technique for factorable functions described in Section 5.2.1 is most commonly used.

α-BB is a branch-and-bound solver that uses α-convexification (Adjiman, Androulakis and Floudas 1998) and its variants to obtain quadratic underestimators of non-convex functions in the constraints and objective. A number of special underestimators are available for specific commonly used functions. More information is available on the project website.

BARON is the acronym for 'branch and reduce optimization navigator' (Sahinidis 1996). It implements a branch-and-bound algorithm that computes a lower bound at each subproblem by means of a linear relaxation of (1.1), as discussed in Section 5.2. It includes various bound-tightening techniques (Section 5.3.2) such as probing and the *violation transfer* outlined in Section 5.3.1. It is available through the GAMS and AIMMS modelling

systems. More information, examples, and documentation are available on the project website.

COCONUT is an open-source environment for global optimization problems (COCONUT 2004, Schichl 2004). Although it does not solve problems with integer variables, we include it in this section because it uses various techniques common to MINLP solvers: bound tightening (Section 5.3.2), reformulation (Section 5.2), and heuristics. The source code and documentation are available at the project homepage.

COUENNE, or 'convex over- and under-envelopes for nonlinear estimation' (Belotti 2009), is an open-source branch-and-bound algorithm that, similarly to BARON, obtains a lower bound through an LP relaxation using the reformulation technique outlined in Section 5.2.1. It also implements several bound-tightening procedures (Section 5.3.2) as well as a recently introduced feasibility-pump heuristic (Section 6.1.2), a separator of disjunctive cuts (Section 5.2.2), and different branching schemes including strong, pseudo-cost, and reliability branching (Section 5.3.1). Recently, it also introduced the linear cuts described by Qualizza *et al.* (2012) and briefly discussed in Section 5.4.1. The source code and documentation are available online.

GloMIQO is an evolution of α-BB with additional algorithms based on the work of Misener and Floudas (2013). It dramatically improves the lower-bounding procedure used originally in the α-BB method. It can solve only quadratically constrained quadratic problems. The solver is available via the GAMS modelling system. Related publications and more information are available on the project website.

LaGO is a branch-and-bound algorithm that is guaranteed to return the global optimum for mixed-integer quadratic problems (Nowak *et al.* 2003). It uses α-convexification (see Section 5.3) to obtain a lower bound on each subproblem. Though this technique is guaranteed only to solve quadratic problems to global optimality, LaGO can be used as a heuristic in other cases. The source code and documentation are available online.

LGO, or the 'Lipschitz (continuous) global optimizer', implements a set of heuristics and exact methods for global optimization. A constrained local optimization approach is used to obtain upper bounds on the objective value. Lower bounds are estimated through sampling. LGO assumes that the functions in the objective and the constraints of the problem are Lipschitz-continuous. Thus, it can find a global solution when the Lipschitz constants for all functions in the problem are known. One advantage of LGO is that it does not require gradients of the functions and hence can be applied to problems where the functions are not explicitly known: they could come from a black box or a simulation. A license for LGO may be purchased from the project website.

LindoGlobal is the MINLP solver of the LINGO modelling suite (Lin and Schrage 2009). It is in the same class as BARON and COUENNE in that it employs a lower-bounding technique for factorable functions, as described in Section 5.2.1. This solver may be purchased through LINDO Systems.

SCIP started as an MILP solver (Achterberg 2005) but evolved first into a solver for MINLPs with quadratic objective function and constraints (Berthold *et al.* 2010) and, more recently, into a solver for non-convex MINLP (Berthold *et al.* 2012). Following the basic approach of BARON and COUENNE, it implements a branch-and-bound algorithm (Section 5.2.1) with linear relaxation, various heuristics, and bound-tightening procedures. The source code and documentation are available on the project website.

8.3. An MIOCP solver

MUSCOD-II started as a reference implementation of direct multiple shooting, a direct and all-at-once method for ODE-constrained optimal control problems (Bock and Plitt 1984), and was later extended to DAE-constrained problems (Leineweber *et al.* 2003) and mixed-integer optimal control problems (Sager 2005). Kirches and Leyffer (2011) propose an extension for modelling MIOCPs in the symbolic modelling language AMPL and present an interface to MUSCOD-II. More information about this solver is available from its website on NEOS.

8.4. Modelling languages and online resources

Diverse modelling languages and online resources make it easy to specify and solve MINLP problems without needing to install software, code nonlinear functions, or derivatives. The wide availability of these tools means that MINLP has become accessible to the broader scientific and engineering community. In this section, we briefly summarize these tools.

Modelling languages enable scientists and engineers to express optimization problems in a more natural algebraic form that is close to a mathematical representation of the problem. Most modelling languages include automatic differentiation tools (Griewank 2000) that provide (exact) first and second derivatives of the problem functions and relieve the user of the error-prone tasks of coding derivatives. The most popular modelling languages are AIMMS (Bisschop and Entriken 1993), AMPL (Fourer, Gay and Kernighan 1993), GAMS (Brooke *et al.* 1992), MOSEL (Colombani and Heipcke 2002), TomLab (Holmström and Edvall 2004, Holmström, Göran and Edvall 2010) and YALMIP (Löfberg 2004). TomLab and YALMIP are built on top of Matlab, while the other systems are domain-specific languages that define a syntax for specifying optimization problems that can be parsed by the respective system to provide function and derivative information to

the solver through a back-end. YALMIP can also solve small instances of convex and non-convex MINLPs using inbuilt algorithms. Recently, an open-source modelling system, 'Pyomo', has been developed (Hart, Watson and Woodruff 2011), which enables users to express MINLPs using the PYTHON scripting language. Opti Toolbox (Currie and Wilson 2012) is another open-source modelling tool. It enables users to express and solve MINLPs from within the Matlab environment. Currently, users can call BONMIN and SCIP solvers from this toolbox.

Online resources for optimization have grown dramatically over the past 15 years. There exist many libraries of test or benchmark problems in AMPL and GAMS (see Appendix A). Arguably the most important factor in making optimization solvers widely available has been the NEOS server (Czyzyk *et al.* 1998). Many of the solvers described above are now available on NEOS. NEOS provides a collection of state-of-the-art optimization software. Optimization problems are submitted through a web interface (or from within a modelling language session) and solved remotely. The MINLP solvers available on NEOS are α-ECP, BARON, BONMIN, COUENNE, DICOPT, FilMINT, LindoGlobal, MINLPBB, SBB, and SCIP.

A. Online resources

Convex and linear solvers

BONMIN	https://projects.coin-or.org/Bonmin
CBC solver	https://projects.coin-or.org/Cbc
KNITRO	http://www.ziena.com/knitro.htm
MILANO	http://www.pages.drexel.edu/∼hvb22/milano
MINOPT	http://titan.princeton.edu/MINOPT
MINOTAUR	http://wiki.mcs.anl.gov/minotaur/
MISQP	http://www.ai7.uni-bayreuth.de/misqp.htm
SBB	http://www.gams.com/dd/docs/solvers/sbb.pdf

Non-convex solvers

α-BB	http://titan.princeton.edu/tools
BARON	http://archimedes.cheme.cmu.edu/?q=baron
COCONUT	http://www.mat.univie.ac.at/∼coconut/coconut-environment
COUENNE	https://projects.coin-or.org/Couenne
GloMIQO	http://helios.princeton.edu/GloMIQO/publications.html
LaGO	https://projects.coin-or.org/LaGO
LGO	http://www.pinterconsulting.com
LindoGlobal	http://www.lindo.com
SCIP	http://scip.zib.de
MUSCOD-II	http://www.neos-server.org/neos/solvers/miocp:MUSCOD-II/AMPL.html

Online resources and model libraries

AMPL models	http://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP http://minlp.org/
GAMS models	http://www.gamsworld.org/minlp/ http://minlp.org/
NEOS	http://www.neos-server.org/neos/

B. Optimization subproblems

Acknowledgements

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, US Department of Energy, under Contract DE-AC02-06CH11357, and by the US Department of Energy through grant DE-FG02-05ER25694. This work was also supported through NSF grant CCF-0830035.

REFERENCES[2]

K. Abhishek, S. Leyffer and J. T. Linderoth (2010), 'FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs', *INFORMS J. Comput.* **22**, 555–567.

P. Abichandani, H. Y. Benson and M. Kam (2008), Multi-vehicle path coordination under communication constraints. In *American Control Conference*, IEEE Conference Publications, pp. 650–656.

M. A. Abramson (2004), 'Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm', *Optim. Engng* **5**, 157–177.

[2] The URLs cited in this work were correct at the time of going to press, but the publisher and the authors make no undertaking that the citations remain live or are accurate or appropriate.

M. Abramson, C. Audet, J. Chrissis and J. Walston (2009), 'Mesh adaptive direct search algorithms for mixed variable optimization', *Optim. Lett.* **3**, 35–47.

T. Achterberg (2005), SCIP: A framework to integrate constraint and mixed integer programming. ZIB-Report 04-19, Zuse Institut Berlin.

T. Achterberg and T. Berthold (2007), 'Improving the feasibility pump', *Discrete Optim.* **4**, 77–86.

T. Achterberg, T. Koch and A. Martin (2004), 'Branching rules revisited', *Oper. Res. Lett.* **33**, 42–54.

W. Adams (2011), Use of Lagrange interpolating polynomials in the RLT. In *Wiley Encyclopedia of Operations Research and Management Science.*

W. Adams and H. Sherali (1986), 'A tight linearization and an algorithm for zero-one quadratic programming problems', *Management Sci.* **32**, 1274–1290.

W. Adams and H. Sherali (2005), 'A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems', *Ann. Oper. Res.* **140**, 21–47.

C. S. Adjiman, I. Androulakis and C. Floudas (1998), 'A global optimization method, αBB, for general twice-differentiable constrained NLPs, II: Implementation and computational results', *Comput. Chem. Engng* **22**, 1159–1179.

I. Akrotirianakis, I. Maros and B. Rustem (2001), 'An outer approximation based branch-and-cut algorithm for convex 0–1 MINLP problems', *Optim. Methods Software* **16**, 21–47.

F. A. Al-Khayyal and J. E. Falk (1983), 'Jointly constrained biconvex programming', *Math. Oper. Res.* **8**, 273–286.

M. Altunay, S. Leyffer, J. T. Linderoth and Z. Xie (2011), 'Optimal security response to attacks on open science grids', *Computer Networks* **55**, 61–73.

E. D. Andersen and K. D. Andersen (1995), 'Presolving in linear programming', *Math. Program.* **71**, 221–245.

I. P. Androulakis, C. D. Maranas and C. A. Floudas (1995), 'αBB : A global optimization method for general constrained nonconvex problems', *J. Global Optim.* **7**, 337–363.

K. Anstreicher (2012), 'On convex relaxations for quadratically constrained quadratic programming', *Math. Program.* **136**, 233–251.

K. M. Anstreicher (2009), 'Semidefinite programming versus the reformulation–linearization technique for nonconvex quadratically constrained quadratic programming', *J. Global Optim.* **43**, 471–484.

A. Atamtürk and V. Narayanan (2010), 'Conic mixed-integer rounding cuts', *Math. Program. A* **122**, 1–20.

C. Audet and J. E. Dennis, Jr (2000), 'Pattern search algorithms for mixed variable programming', *SIAM J. Optim.* **11**, 573–594.

R. Bacher (1997), The Optimal Power Flow (OPF) and its solution by the interior point approach. EES-UETP Madrid, short course.

M. Baes, A. Del Pia, Y. Nesterov, S. Onn and R. Weismantel (2012), 'Minimizing Lipschitz-continuous strongly convex functions over integer points in polytopes', *Math. Program.* **134**, 305–322.

A. Balakrishnan and S. Graves (1989), 'A composite algorithm for a concave-cost network flow problem', *Networks* **19**, 175–202.

P. Balaprakash, S. M. Wild and P. D. Hovland (2011), 'Can search algorithms save large-scale automatic performance tuning?', *Procedia Comput. Sci.* (ICCS 2011) **4**, 2136–2145.

E. Balas, S. Ceria and G. Cornuéjols (1993), 'A lift-and-project cutting plane algorithm for mixed 0–1 programs', *Math. Program.* **58**, 295–324.

E. Balas, S. Ceria and G. Cornuéjols (1996), 'Mixed 0–1 programming by lift-and-project in a branch-and-cut framework', *Management Sci.* **42**, 1229–1246.

X. Bao, N. Sahinidis and M. Tawarmalani (2009), 'Multiterm polyhedral relaxations for nonconvex quadratically constrained quadratic programs', *Optim. Methods Software* **24**, 485–504.

S. Bartelt-Hunt, T. Culver, J. Smith, L. S. Matott and A. Rabideau (2006), 'Optimal design of a compacted soil liner containing sorptive amendments', *J. Environmental Engng* **132**, 769–776.

E. F. Bartholomew, R. P. O'Neill and M. C. Ferris (2008), 'Optimal transmission switching', *IEEE Trans. Power Systems* **23**, 1346–1355.

H. H. Bauschke and J. M. Borwein (1996), 'On projection algorithms for solving convex feasibility problems', *SIAM Rev.* **38**, 367–426.

E. Beale and J. Tomlin (1970), Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In *Proc. 5th International Conference on Operations Research* (J. Lawrence, ed.), Tavistock Publications, pp. 447–454.

E. M. L. Beale and J. J. H. Forrest (1976), 'Global optimization using special ordered sets', *Math. Program.* **10**, 52–69.

R. Bellman (1961), 'On the approximation of curves by line segments using dynamic programming', *Commun. Assoc. Comput. Mach.* **4**, 284.

P. Belotti (2009), Couenne: A user's manual. Technical report, Lehigh University.

P. Belotti (2012), Disjunctive cuts for non-convex MINLP. In *Mixed Integer Nonlinear Programming*, Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer, pp. 117–144.

P. Belotti (2013), 'Bound reduction using pairs of linear inequalities', *J. Global Optim.*, to appear.

P. Belotti, S. Cafieri, J. Lee and L. Liberti (2010), Feasibility-based bounds tightening via fixed points. In *Combinatorial Optimization and Applications* (W. Wu and O. Daescu, eds), Vol. 6508 of *Lecture Notes in Computer Science*, Springer, pp. 65–76.

P. Belotti, J. Góez, I. Pólik, T. Ralphs and T. Terlaky (2012), A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization. Technical report 12T-009, Department of Industrial and Systems Engineering, Lehigh University.
http://www.optimization-online.org/DB_FILE/2012/06/3494.pdf

P. Belotti, J. Lee, L. Liberti, F. Margot and A. Wächter (2009), 'Branching and bounds tightening techniques for non-convex MINLP', *Optim. Methods Software* **24**, 597–634.

A. Ben-Tal and A. Nemirovski (1995), 'Optimal design of engineering structures', *Optima* **47**, 4–8.

A. Ben-Tal and A. Nemirovski (2001), 'On polyhedral approximations of the second-order cone', *Math. Oper. Res.* **26**, 193–205.

H. Y. Benson (2011), 'Mixed integer nonlinear programming using interior point methods', *Optim. Methods Software* **26**, 911–931.

H. Y. Benson (2012), Using interior-point methods within an outer approximation framework for mixed integer nonlinear programming. In *Mixed Integer Nonlinear Programming*, Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer, pp. 225–243.

T. Berthold (2012), RENS: The optimal rounding. ZIB-Report 12-17, Zuse Institut Berlin.

T. Berthold and A. M. Gleixner (2012), Undercover: A primal MINLP heuristic exploring a largest sub-MIP. ZIB-Report 12-07, Zuse Institut Berlin.

T. Berthold, G. Gamrath, A. Gleixner, S. Heinz, T. Koch and Y. Shinano (2012), Solving mixed integer linear and nonlinear problems using the SCIP optimization suite. ZIB-Report 12-27, Zuse Institut Berlin.

T. Berthold, A. Gleixner, S. Heinz and S. Vigerske (2010), Extending SCIP for solving MIQCPs. In *Proc. European Workshop on Mixed Integer Nonlinear Programming*, pp. 181–196.
http://www.lix.polytechnique.fr/~liberti/ewminlp/ewminlp-proceedings.pdf

D. Bertsekas and R. Gallager (1987), *Data Networks*, Prentice Hall.

R. Bhatia, A. Segall and G. Zussman (2006), 'Analysis of bandwidth allocation algorithms for wireless personal area networks', *Wireless Networks* **12**, 589–603.

D. Bienstock (1996), 'Computational study of a family of mixed-integer quadratic programming problems', *Math. Program.* **74**, 121–140.

D. Bienstock and S. Mattia (2007), 'Using mixed-integer programming to solve power grid blackout problems', *Discrete Optim.* **4**, 115–141.

V. M. Bier (2005), Game-theoretic and reliability methods in counterterrorism and security. In *Mathematical and Statistical Methods in Reliability, Series on Quality, Reliability and Engineering Statistics* (A. Wilson, N. Limnios, S. Keller-McNulty and Y. Armijo, eds), World Scientific, pp. 17–28.

V. M. Bier, A. Nagaraj and V. Abhichandani (2005), 'Protection of simple series and parallel systems with components of different values', *Reliability Engineering System Safety* **87**, 315–323.

V. M. Bier, S. Oliveros and L. Samuelson (2007), 'Choosing what to protect', *J. Public Economic Theory* **9**, 563–587.

J. Bisschop and R. Entriken (1993), *AIMMS: The Modeling System*, Paragon Decision Technology.

H. Bock and R. Longman (1985), 'Computation of optimal controls on disjoint control sets for minimum energy subway operation', *Adv. Astronaut. Sci.* **50**, 949–972.

H. Bock and K. Plitt (1984), A multiple shooting algorithm for direct solution of optimal control problems. In *Proc. 9th IFAC World Congress*, Pergamon Press, pp. 242–247.

P. Bonami (2011), Lift-and-project cuts for mixed integer convex programs. In *Integer Programming and Combinatorial Optimization* (O. Günlük and G. Woeginger, eds), Vol. 6655 of *Lecture Notes in Computer Science*, Springer, pp. 52–64.

P. Bonami and J. P. M. Gonçalves (2012), 'Heuristics for convex mixed integer nonlinear programs', *Comput. Optim. Appl.* **51**, 729–747.
P. Bonami, L. Biegler, A. Conn, G. Cornuéjols, I. Grossmann, C. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya and A. Wächter (2008), 'An algorithmic framework for convex mixed integer nonlinear programs', *Discrete Optim.* **5**, 186–204.
P. Bonami, G. Cornuéjols, A. Lodi and F. Margot (2009), 'A feasibility pump for mixed integer nonlinear programs', *Math. Program.* **119**, 331–352.
P. Bonami, M. Kılınç and J. T. Linderoth (2012), Algorithms and software for convex mixed integer nonlinear programs. In *Mixed Integer Nonlinear Programming*, Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer, pp. 61–92.
P. Bonami, J. Lee, S. Leyffer and A. Wächter (2011), More branch-and-bound experiments in convex nonlinear integer programming. Preprint ANL/MCS-P1949-0911, Mathematics and Computer Science Division, Argonne National Laboratory.
I. Bongartz, A. R. Conn, N. I. M. Gould and P. L. Toint (1995), 'CUTE: Constrained and unconstrained testing environment', *ACM Trans. Math. Software* **21**, 123–160.
R. Boorstyn and H. Frank (1977), 'Large-scale network topological optimization', *IEEE Trans. Communications* **25**, 29–47.
B. Borchers and J. E. Mitchell (1994), 'An improved branch and bound algorithm for mixed integer nonlinear programs', *Comput. Oper. Res.* **21**, 359–368.
S. Boyd and L. Vandenberghe (2004), *Convex Optimization*, Cambridge University Press.
C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi and P. Toth (2006), An MINLP solution method for a water network problem. In *Algorithms: ESA 2006, 14th Annual European Symposium*, Springer, pp. 696–707.
C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi and P. Toth (2012), 'On the optimal design of water distribution networks: A practical MINLP approach', *Optim. Engng* **13**, 219–246.
A. Brooke, D. Kendrick, A. Meeraus and R. Raman (1992), *GAMS: A User's Guide*, GAMS Development Corporation.
A. Bryson and Y.-C. Ho (1975), *Applied Optimal Control*, Wiley.
C. Buchheim and A. Wiegele (2013), 'Semidefinite relaxations for non-convex quadratic mixed-integer programming', *Math. Program.*, to appear.
S. Burer (2009), 'On the copositive representation of binary and continuous nonconvex quadratic programs', *Math. Program.* **120**, 479–495.
S. Burer and A. Letchford (2009), 'On nonconvex quadratic programming with box constraints', *SIAM J. Optim.* **20**, 1073–89.
S. Burer and A. Letchford (2012), 'Non-convex mixed-integer nonlinear programming: A survey', *Surv. Oper. Res. Management Sci.* **17**, 97–106.
S. Burer and A. Letchford (2013), 'Unbounded convex sets for non-convex mixed-integer quadratic programming', *Math. Program.*, to appear.
S. Burer and D. Vandenbussche (2009), 'Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound', *Comput. Optim. Appl.* **43**, 181–195.

J. Burgschweiger, B. Gnädig and M. Steinbach (2008), 'Optimization models for operative planning in drinking water networks', *Optim. Engng* **10**, 43–73.

M. R. Bussieck and S. Vigerske (2010), MINLP solver software. In *Wiley Encyclopedia of Operations Research and Management Science* (J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, P. Jeffrey and J. C. Smith, eds), Wiley.

R. H. Byrd, J. Nocedal and W. A. Richard (2006), KNITRO: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization* (G. Pillo and M. Roma, eds), Vol. 83 of *Nonconvex Optimization and its Applications*, Springer, pp. 35–59.

S. Callegari, F. Bizzarri, R. Rovatti and G. Setti (2010), 'On the approximate solution of a class of large discrete quadratic programming problems by $\Delta\Sigma$ modulation: The case of circulant quadratic forms', *IEEE Trans. Signal Process.* **58**, 6126–6139.

I. Castillo, J. Westerlund, S. Emet and T. Westerlund (2005), 'Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods', *Comput. Chem. Engng* **30**, 54–69.

M. Çezik and G. Iyengar (2005), 'Cuts for mixed 0–1 conic programming', *Math. Program. A* **104**, 179–202.

S. Ceria and J. Soares (1999), 'Convex programming for disjunctive optimization', *Math. Program.* **86**, 595–614.

K. Chi, X. Jiang, S. Horiguchi and M. Guo (2008), 'Topology design of network-coding-based multicast networks', *IEEE Trans. Mobile Comput.* **7**, 1–14.

K. Chung, J.-P. Richard and M. Tawarmalani (2011), Lifted inequalities for 0–1 mixed-integer bilinear covering sets.
http://www.optimization-online.org/DB_FILE/2011/03/2949.pdf

V. Chvátal (1973), 'Edmonds polytopes and a hierarchy of combinatorial problems', *Discrete Math.* **4**, 305–337.

COCONUT (2004), The COCONUT benchmark: A benchmark for global optimization and constraint satisfaction.
http://www.mat.univie.ac.at/~neum/glopt/coconut/benchmark.html

J. S. Cohen (2003), *Computer Algebra and Symbolic Computation: Elementary Algorithms*, Universities Press.

Y. Colombani and S. Heipcke (2002), Mosel: An extensible environment for modeling and programming solutions. In *Proc. Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems: CP-AI-OR'02* (N. Jussien and F. Laburthe, eds), Ecole des Mines, Nantes, pp. 277–290.

E. Costa-Montenegro, F. J. González-Castaño, P. S. Rodriguez-Hernández and J. C. Burguillo-Rial (2007), Nonlinear optimization of IEEE 802.11 mesh networks. In *ICCS 2007, Part IV*, Springer, pp. 466–473.

K. Croxton, B. Gendron and T. Magnanti (2003), 'A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems', *Management Sci.* **49**, 1268–73.

J. Currie and D. I. Wilson (2012), OPTI: Lowering the barrier between open source optimizers and the industrial MATLAB user. In *Foundations of Computer-Aided Process Operations* (N. Sahinidis and J. Pinto, eds).
http://focapo.org

J. Czyzyk, M. Mesnier and J. Moré (1998), 'The NEOS server', *IEEE J. Comput. Sci. Engng* **5**, 68–75.

D. Dadush, S. Dey and J. P. Vielma (2011*a*), 'The split closure of a strictly convex body', *Oper. Res. Lett.* **39**, 121–126.

D. Dadush, S. S. Dey and J. P. Vielma (2011*b*), 'The Chvátal–Gomory closure of a strictly convex body', *Math. Oper. Res.* **36**, 227–239.

D. Dadush, S. S. Dey and J. P. Vielma (2011*c*), On the Chvátal–Gomory closure of a compact convex set. In *Integer Programming and Combinatorial Optimization* (O. Günlük and G. Woeginger, eds), Vol. 6655 of *Lecture Notes in Computer Science*, Springer, pp. 130–142.

D. Dadush, C. Peikert and S. Vempala (2011*d*), Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *IEEE 52nd Annual Symposium on Foundations of Computer Science: FOCS*, pp. 580–589.

R. J. Dakin (1965), 'A tree search algorithm for mixed programming problems', *Comput. J.* **8**, 250–255.

C. D'Ambrosio and A. Lodi (2011), 'Mixed integer nonlinear programming tools: A practical overview', *4OR* **9**, 329–349.

C. D'Ambrosio, A. Frangioni, L. Liberti and A. Lodi (2012), 'A storm of feasibility pumps for nonconvex MINLP', *Math. Program. B* **136**, 375–402.

C. D'Ambrosio, A. Lodi and S. Martello (2010), 'Piecewise linear approximation of functions of two variables in MILP models', *Oper. Res. Lett.* **38**, 39–46.

E. Danna, E. Rothberg and C. LePape (2005), 'Exploring relaxation induced neighborhoods to improve MIP solutions', *Math. Program.* **102**, 71–90.

G. B. Dantzig (1960), 'On the significance of solving linear programming problems with some integer variables', *Econometrica* **28**, 30–44.

G. B. Dantzig (1963), *Linear Programming and Extensions*, Princeton University Press.

E. Davis (1987), 'Constraint propagation with interval labels', *Artificial Intelligence* **32**, 281–331.

E. Davis and M. Ierapetritou (2009), 'A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions', *J. Global Optim.* **43**, 191–205.

J. A. De Loera, R. Hemmecke, M. Koppe and R. Weismantel (2006), 'Integer polynomial optimization in fixed dimension', *Math. Oper. Res.* **31**, 147–153.

S. S. Dey and D. A. Morán (2013), 'Some properties of convex hulls of integer points contained in general convex sets', *Math. Program.*, to appear.

S. S. Dey and J. P. Vielma (2010), The Chvátal–Gomory closure of an ellipsoid is a polyhedron. In *Integer Programming and Combinatorial Optimization*, Vol. 6080 of *Lecture Notes in Computer Science*, Springer, pp. 327–340.

E. Dolan and J. Moré (2002), 'Benchmarking optimization software with performance profiles', *Math. Program.* **91**, 201–213.

E. Dolan, R. Fourer, J. Moré and T. Munson (2002), 'Optimization on the NEOS server', *SIAM News* **35**, 8–9.

V. Donde, V. Lopez, B. Lesieutre, A. Pinar, C. Yang and J. Meza (2005), Identification of severe multiple contingencies in electric power networks, in *Proc. 37th North American Power Symposium*, IEEE.

M. Dorigo, V. Maniezzo and A. Colorni (1996), 'The ant system: Optimization by a colony of cooperating agents', *IEEE Trans. Systems, Man and Cybernetics B* **26**, 1–13.

S. Drewes (2009), Mixed integer second order cone programming. PhD thesis, Technische Universität Darmstadt.

S. Drewes and S. Ulbrich (2012), Subgradient based outer approximation for mixed integer second order cone programming. In *Mixed Integer Nonlinear Programming*, Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer, pp. 41–59.

M. A. Duran and I. Grossmann (1986), 'An outer-approximation algorithm for a class of mixed-integer nonlinear programs', *Math. Program.* **36**, 307–339.

J. Eckstein (1994), 'Parallel branch-and-bound algorithms for general mixed integer programming on the CM-5', *SIAM J. Optim.* **4**, 794–814.

G. Eiger, U. Shamir and A. Ben-Tal (1994), 'Optimal design of water distribution networks', *Water Resources Research* **30**, 2637–2646.

S. Elhedhli (2006), 'Service system design with immobile servers, stochastic demand, and congestion', *Manufacturing & Service Operations Management* **8**, 92–97.

A. M. Eliceche, S. M. Corvalán and P. Martínez (2007), 'Environmental life cycle impact as a tool for process optimisation of a utility plant', *Comput. Chem. Engng* **31**, 648–656.

A. Elwalid, D. Mitra and Q. Wang (2006), 'Distributed nonlinear integer optimization for data-optical internetworking', *IEEE J. Selected Areas in Communications* **24**, 1502–1513.

M. Engelhart, J. Funke and S. Sager (2013), 'A decomposition approach for a new test-scenario in complex problem solving', *J. Comput. Sci.*, in press.

O. Exler and K. Schittkowski (2007), 'A trust region SQP algorithm for mixed-integer nonlinear programming', *Optim. Lett.* **1**, 269–280.

O. Exler, T. Lehmann and K. Schittkowski (2012), MISQP: A Fortran subroutine of a trust region SQP algorithm for mixed-integer nonlinear programming, user's guide. Technical report, Department of Computer Science, University of Bayreuth.

FICO Xpress (2009), *FICO Xpress optimization suite: Xpress-BCL reference manual*, Fair Isaac Corporation.

M. Fischetti and A. Lodi (2003), 'Local branching', *Math. Program.* **98**, 23–47.

M. Fischetti and D. Salvagnin (2009), 'Feasibility pump 2.0', *Math. Program. Comput.* **1**, 201–222.

M. Fischetti, F. Glover and A. Lodi (2005), 'The feasibility pump', *Math. Program.* **104**, 91–104.

R. Fletcher (1987), *Practical Methods of Optimization*, Wiley.

R. Fletcher and S. Leyffer (1994), 'Solving mixed integer nonlinear programs by outer approximation', *Math. Program.* **66**, 327–349.

R. Fletcher and S. Leyffer (1998), User manual for filterSQP. University of Dundee Numerical Analysis Report NA-181.

R. Fletcher and S. Leyffer (2003), Filter-type algorithms for solving systems of algebraic equations and inequalities. In *High Performance Algorithms and*

Software for Nonlinear Optimization (G. di Pillo and A. Murli, eds), Kluwer, pp. 259–278.
A. Flores-Tlacuahuac and L. T. Biegler (2007), 'Simultaneous mixed-integer dynamic optimization for integrated design and control', *Comput. Chem. Engng* **31**, 648–656.
C. Floudas (1995), *Nonlinear and Mixed-Integer Optimization*, Topics in Chemical Engineering, Oxford University Press.
C. A. Floudas (2000), *Deterministic Global Optimization: Theory, Algorithms and Applications*, Kluwer.
R. Fourer, D. M. Gay and B. W. Kernighan (1993), *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press.
K. R. Fowler, J. P. Reese, C. E. Kees, J. E. Dennis, C. T. Kelley, C. T. Miller, C. Audet, A. J. Booker, G. Couture, R. W. Darwin, M. W. Farthing, D. E. Finkel, J. M. Gablonsky, G. A. Gray and T. G. Kolda (2008), 'A comparison of derivative-free optimization methods for water supply and hydraulic capture community problems', *Adv. Water Resources* **31**, 743–757.
A. Frangioni and C. Gentile (2006), 'Perspective cuts for a class of convex 0–1 mixed integer programs', *Math. Program.* **106**, 225–236.
A. Fügenschuh, M. Herty, A. Klar and A. Martin (2006), 'Combinatorial and continuous models for the optimization of traffic flows on networks', *SIAM J. Optim.* **16**, 1155–1176.
A. Fuller (1963), 'Study of an optimum nonlinear control system', *J. Electronics Control* **15**, 63–71.
L. L. Garver (1997), 'Transmission network estimation using linear programming', *IEEE Trans. Power Apparatus Systems* **89**, 1688–1697.
D. M. Gay (1991), Automatic differentiation of nonlinear AMPL models. In *Automatic Differentiation of Algorithms: Theory, Implementation, and Application* (A. Griewank and G. F. Corliss, eds), SIAM, pp. 61–73.
B. Geissler, A. Martin, A. Morsi and L. Schewe (2012), Using piecewise linear functions for solving MINLPs. In *Mixed Integer Nonlinear Programming* (J. Lee and S. Leyffer, eds), Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer, pp. 287–314.
I. Gentilini, F. Margot and K. Shimada (2013), 'The travelling salesman problem with neighbourhoods: MINLP solution', *Optim. Methods Software* **28**, 364–378,
A. M. Geoffrion (1972), 'Generalized Benders decomposition', *J. Optim. Theory Appl.* **10**, 237–260.
A. M. Geoffrion (1977), 'Objective function approximations in mathematical programming', *Math. Program.* **13**, 23–37.
M. Gerdts (2005), 'Solving mixed-integer optimal control problems by branch& bound: A case study from automobile test-driving with gear shift', *Optimal Control Appl. Methods* **26**, 1–18.
M. Gerdts and S. Sager (2012), Mixed-integer DAE optimal control problems: Necessary conditions and bounds. In *Control and Optimization with Differential-Algebraic Constraints* (L. Biegler, S. Campbell and V. Mehrmann, eds), SIAM, pp. 189–212.
F. Glover (1989), 'Tabu search, part I', *ORSA J. Comput.* **1**, 190–206.

F. Glover (1990), 'Tabu search, part II', *ORSA J. Comput.* **2**, 4–32.
D. E. Goldberg (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley.
N. Goldberg, S. Leyffer and I. Safro (2012), Optimal response to epidemics and cyber attacks in networks. Preprint ANL/MCS-1992-0112, Mathematics and Computer Science Division, Argonne National Laboratory.
R. E. Gomory (1958), 'Outline of an algorithm for integer solutions to linear programs', *Bull. Amer. Math. Monthly* **64**, 275–278.
R. E. Gomory (1960), An algorithm for the mixed integer problem. Technical report RM-2597, The RAND Corporation.
N. I. M. Gould and S. Leyffer (2003), An introduction to algorithms for nonlinear optimization. In *Frontiers in Numerical Analysis* (J. Blowey, A. Craig and T. Shardlow, eds), Springer, pp. 109–197.
N. I. M. Gould, S. Leyffer and P. L. Toint (2004), 'A multidimensional filter algorithm for nonlinear equations and nonlinear least squares', *SIAM J. Optim.* **15**, 17–38.
J.-P. Goux and S. Leyffer (2003), 'Solving large MINLPs on computational grids', *Optim. Engng* **3**, 327–354.
A. Griewank (2000), *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Vol. 19 of *Frontiers in Applied Mathematics*, SIAM.
A. Griewank and P. L. Toint (1984), 'On the existence of convex decompositions of partially separable functions', *Math. Program.* **28**, 25–49.
I. Griva, S. G. Nash and A. Sofer (2009), *Linear and Nonlinear Optimization*, second edition, SIAM.
I. E. Grossmann (2002), 'Review of nonlinear mixed-integer and disjunctive programming techniques', *Optim. Engng* **3**, 227–252.
I. E. Grossmann and Z. Kravanja (1997), Mixed-integer nonlinear programming: A survey of algorithms and applications. In *Large-Scale Optimization with Applications, Part II: Optimal Design and Control* (L. T. Biegler, T. F. Coleman, A. R. Conn and F. N. Santosa, eds), Springer.
I. E. Grossmann and R. W. H. Sargent (1979), 'Optimal design of multipurpose batch plants', *Indust. Engng Chem. Process Design and Development* **18**, 343–348.
A. Guerra, A. M. Newman and S. Leyffer (2011), 'Concrete structure design using mixed-integer nonlinear programming with complementarity constraints', *SIAM J. Optim.* **21**, 833–863.
O. Günlük and J. T. Linderoth (2010), 'Perspective relaxation of mixed integer nonlinear programs with indicator variables', *Math. Program. B* **104**, 186–203.
O. Günlük and J. T. Linderoth (2012), Perspective reformulation and applications. In *Mixed Integer Nonlinear Programming*, Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer, pp. 61–92.
O. K. Gupta and A. Ravindran (1985), 'Branch and bound experiments in convex nonlinear integer programming', *Management Sci.* **31**, 1533–1546.
Gurobi (2012), *Gurobi Optimizer Reference Manual, Version 5.0*, Gurobi Optimization.
E. Hansen (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker.

I. Harjunkoski, T. Westerlund, R. Pörn and H. Skrifvars (1998), 'Different transformations for solving non-convex trim loss problems by MINLP', *European J. Oper. Res.* **105**, 594–603.

W. E. Hart, J.-P. Watson and D. L. Woodruff (2011), 'Pyomo: modeling and solving mathematical programs in Python', *Math. Program. Comput.* **3**, 219–260.

K. W. Hedman, R. P. O'Neill, E. B. Fisher and S. S. Oren (2008), 'Optimal transmission switching: Sensitivity analysis and extensions', *IEEE Trans. Power Systems* **23**, 1469–1479.

S. Heinz (2005), 'Complexity of integer quasiconvex polynomial optimization', *J. Complexity* **21**, 543–556.

E. Hellström, M. Ivarsson, J. Aslund and L. Nielsen (2009), 'Look-ahead control for heavy trucks to minimize trip time and fuel consumption', *Control Engng Practice* **17**, 245–254.

T. Hemker (2008), Derivative free surrogate optimization for mixed-integer nonlinear black box problems in engineering. PhD thesis, Technischen Universität Darmstadt, Darmstadt, Germany.

T. Hemker, K. Fowler, M. Farthing and O. von Stryk (2008), 'A mixed-integer simulation-based optimization approach with surrogate functions in water resources management', *Optim. Engng* **9**, 341–360.

R. Hemmecke, S. Onn and R. Weismantel (2011), 'A polynomial oracle-time algorithm for convex integer minimization', *Math. Program.* **126**, 97–117.

H. Hijazi, P. Bonami and A. Ouorou (2010), An outer–inner approximation for separable MINLPs. Technical report, LIF, Faculté des Sciences de Luminy, Université de Marseille.

R. Hildebrand and M. Köppe (2013), 'A new Lenstra-type algorithm for quasiconvex polynomial integer minimization with complexity $2^{O(n \log n)}$', *Discrete Optim.* **10**, 69–84.

K. Holmström and M. Edvall (2004), The TOMLAB optimization environment. In *Modeling Languages in Mathematical Optimization* (J. Kallrath, ed.), Kluwer Academic, pp. 369–378. http://tomopt.com/tomlab/

K. Holmström, A. O. Göran and M. M. Edvall (2010), *User's Guide for TOMLAB 7*, Tomlab Optimization Inc.

H. Horst, P. M. Pardalos and V. Thoai (1995), *Introduction to Global Optimization*, Kluwer.

R. Horst and H. Tuy (1993), *Global Optimization*, Springer.

IBM Ilog CPLEX (2009), *IBM Ilog CPLEX V12.1: User's Manual for CPLEX*, IBM.

R. Jeroslow and J. Lowe (1984), 'Modelling with integer variables', *Math. Program. Studies* **22**, 167–84.

R. Jeroslow and J. Lowe (1985), 'Experimental results on the new techniques for integer programming formulations', *J. Oper. Res. Soc.* **36**, 393–403.

R. G. Jeroslow (1973), 'There cannot be any algorithm for integer programming with quadratic constraints', *Oper. Res.* **21**, 221–224.

N. J. Jobst, M. D. Horniman, C. A. Lucas and G. Mitra (2001), 'Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints', *Quant. Finance* **1**, 489–501.

J. J. Júdice, H. D. Sherali, I. M. Ribeiro and A. M. Faustino (2006), 'A complementarity-based partitioning and disjunctive cut algorithm for mathematical programming problems with equilibrium constraints', *J. Global Optim.* **36**, 89–114.

R. Kannan and C. Monma (1978), On the computational complexity of integer programming problems. In *Optimization and Operations Research* (R. Henn, B. Korte and W. Oettli, eds), Vol. 157 of *Lecture Notes in Economics and Mathematical Systems*, Springer, pp. 161–172.

R. Karuppiah and I. E. Grossmann (2006), 'Global optimization for the synthesis of integrated water systems in chemical processes', *Comput. Chem. Engng* **30**, 650–673.

A. B. Keha, I. R. De Farias Jr and G. L. Nemhauser (2006), 'A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization', *Oper. Res.* **54**, 847–858.

J. E. Kelley (1960), 'The cutting plane method for solving convex programs', *J. SIAM* **8**, 703712.

J. Kennedy and R. Eberhart (1995), Particle swarm optimization. In *IEEE International Conference on Neural Networks*, Vol. 4, pp. 1942–1948.

L. Khachiyan and L. Porkolab (2000), 'Integer optimization on convex semialgebraic sets', *Discrete Comput. Geom.* **23**, 207–224.

M. Kılınç (2011), Disjunctive cutting planes and algorithms for convex mixed integer nonlinear programming. PhD thesis, Department of Industrial and Systems Engineering, University of Wisconsin–Madison.

M. Kılınç, J. T. Linderoth and J. Luedtke (2010), Effective separation of disjunctive cuts for convex mixed integer nonlinear programs. Technical report 1681, Computer Sciences Department, University of Wisconsin–Madison.

C. Kirches (2011), Fast numerical methods for mixed-integer nonlinear model-predictive control. In *Advances in Numerical Mathematics* (H. Bock, W. Hackbusch, M. Luskin and R. Rannacher, eds), Springer Vieweg. PhD thesis, Ruprecht-Karls-Universität Heidelberg.

C. Kirches and S. Leyffer (2011), TACO: A toolkit for AMPL control optimization. Preprint ANL/MCS-P1948-0911, Mathematics and Computer Science Division, Argonne National Laboratory.

C. Kirches, S. Sager, H. Bock and J. Schlöder (2010), 'Time-optimal control of automobile test drives with gear shifts', *Optimal Control Appl. Methods* **31**, 137–153.

S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi (1983), 'Optimization by simulated annealing', *Science* **220**, 671–680.

J. L. Klepeis and C. A. Floudas (2003), 'ASTRO-FOLD: A combinatorial and global optimization framework for *ab initio* prediction of three-dimensional structures of proteins from the amino acid sequence', *Biophysical J.* **85**, 2119–2146.

KNITRO (2012), *KNITRO Documentation*. Ziena Optimization.

G. R. Kocis and I. E. Grossmann (1988), 'Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis', *Indust. Engng Chem. Research* **27**, 1407–1421.

P. A. Krokhmal and P. Soberanis (2010), 'Risk optimization with p-order conic constraints: A linear programming approach', *European J. Oper. Res.* **201**, 653–671.
S. Lakhera, U. V. Shanbhag and M. McInerney (2011), 'Approximating electrical distribution networks via mixed-integer nonlinear programming', *Internat. J. Electric Power and Energy Systems* **33**, 245–257.
A. H. Land and A. G. Doig (1960), 'An automatic method for solving discrete programming problems', *Econometrica* **28**, 497–520.
J. Lasserre (2000), Convergent LMI relaxations for nonconvex quadratic programs. In *Proc. 39th IEEE Conference on Decision and Control*, Vol. 5, IEEE, pp. 5041–5046.
J. Lasserre (2001), An explicit exact SDP relaxation for nonlinear 0–1 programs. In *Integer Programming and Combinatorial Optimization 2001* (K. Aardal and A. Gerards, eds), Vol. 2081 of *Lecture Notes in Computer Science*, Springer, pp. 293–303.
E. L. Lawler and D. E. Woods (1966), 'Branch-and-bound methods: A survey', *Oper. Res.* **14**, 699–719.
J. Lee and S. Leyffer, eds (2012), *Mixed Integer Nonlinear Programming*, Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer.
J. Lee and D. Wilson (2001), 'Polyhedral methods for piecewise-linear functions I: The lambda method', *Discrete Appl. Math.* **108**, 269–285.
D. Leineweber, I. Bauer, A. Schäfer, H. Bock and J. Schlöder (2003), 'An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization (Parts I and II)', *Comput. Chem. Engng* **27**, 157–174.
J. H. W. Lenstra (1983), 'Integer programming with a fixed number of variables', *Math. Oper. Res.* **8**, 538–548.
S. Leyffer (1998), User manual for MINLP-BB. University of Dundee.
S. Leyffer (2001), 'Integrating SQP and branch-and-bound for mixed integer nonlinear programming', *Comput. Optim. Appl.* **18**, 295–309.
S. Leyffer (2003), MacMINLP: Test problems for mixed integer nonlinear programming. http://www.mcs.anl.gov/~leyffer/macminlp
L. Liberti and C. C. Pantelides (2003), 'Convex envelopes of monomials of odd degree', *J. Global Optim.* **25**, 157–168.
L. Liberti, N. Mladenović and G. Nannicini (2011), 'A recipe for finding good solutions to MINLPs', *Math. Program. Comput.* **3**, 349–390.
Y. Lin and L. Schrage (2009), 'The global solver in the LINDO API', *Optim. Methods Software* **24**, 657–668.
J. T. Linderoth (2005), 'A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs', *Math. Program. B* **103**, 251–282.
J. T. Linderoth and M. W. P. Savelsbergh (1999), 'A computational study of search strategies in mixed integer programming', *INFORMS J. Comput.* **11**, 173–187.
G. Liuzzi, S. Lucidi and F. Rinaldi (2012), 'Derivative-free methods for bound constrained mixed-integer optimization', *Comput. Optim. Appl.* **53**, 505–526.
J. Löfberg (2004), YALMIP: A toolbox for modeling and optimization in MATLAB. In *IEEE International Symposium on Computer Aided Control Systems Design*, pp. 284–289.

J. Luedtke, M. Namazifar and J. T. Linderoth (2012), 'Some results on the strength of relaxations of multilinear functions', *Math. Program.* **136**, 325–351.

A. Mahajan, S. Leyffer and C. Kirches (2012), Solving mixed-integer nonlinear programs by QP-diving. Preprint ANL/MCS-2071-0312, Mathematics and Computer Science Division, Argonne National Laboratory.

A. Mahajan, S. Leyffer, J. T. Linderoth, J. Luedtke and T. Munson (2011), MINOTAUR: A toolkit for solving mixed-integer nonlinear optimization. Wiki page. http://wiki.mcs.anl.gov/minotaur

L. Maonan and H. Wenjun (1991), The study of choosing optimal plan of air quantities regulation of mine ventilation network. In *Proc. 5th US Mine Ventilation Symposium*, pp. 427–421.

J. Maria, T. T. Truong, J. Yao, T.-W. Lee, R. G. Nuzzo, S. Leyffer, S. K. Gray and J. A. Rogers (2009), 'Optimization of 3D plasmonic crystal structures for refractive index sensing', *J. Phys. Chem. C* **113**, 10493–10499.

H. M. Markowitz and A. S. Manne (1957), 'On the solution of discrete programming problems', *Econometrica* **25**, 84–110.

A. Martin, M. Möller and S. Moritz (2006), 'Mixed integer models for the stationary case of gas network optimization', *Math. Program.* **105**, 563–582.

S. Masihabadi, S. Sanjeevi and K. Kianfar (2011), 'n-step conic mixed integer rounding inequalities', *Optimization Online*. http://www.optimization-online.org/DB_HTML/2011/11/3251.html

G. P. McCormick (1976), 'Computability of global solutions to factorable nonconvex programs I: Convex underestimating problems', *Math. Program.* **10**, 147–175.

F. Messine (2004), 'Deterministic global optimization using interval constraint propagation techniques', *RAIRO-RO* **38**, 277–294.

R. Meyer (1976), 'Mixed integer minimization models for piecewise-linear functions of a single variable', *Discrete Math.* **16**, 163–71.

R. Miller, Z. Xie, S. Leyffer, M. Davis and S. Gray (2010), 'Surrogate-based modeling of the optical response of metallic nanostructures', *J. Phys. Chem. C* **114**, 20741–20748.

R. Misener and C. Floudas (2012), 'Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations', *Math. Program.* **136**, 155–182.

R. Misener and C. Floudas (2013), 'GloMIQO: Global mixed-integer quadratic optimizer', *J. Global Optim.*, to appear.

J. Momoh, R. Koessler, M. Bond, B. Stott, D. Sun, A. Papalexopoulos and P. Ristanovic (1997), 'Challenges to optimal power flow', *IEEE Trans. Power Systems* **12**, 444–455.

J. Müller (2012), Surrogate model algorithms for computationally expensive black-box global optimization problems. PhD thesis, Tampere University of Technology, Finland.

J. Müller, C. A. Shoemaker and R. Piché (2013), 'SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems', *Comput. Oper. Res.* **40**, 1383–1400.

G. Nannicini and P. Belotti (2012), 'Rounding-based heuristics for nonconvex MINLPs', *Math. Program. Comput.* **4**, 1–31.

G. Nannicini, P. Belotti and L. Liberti (2008), A local branching heuristic for MINLPs. http://arxiv.org/abs/0812.2188

G. Nemhauser and L. A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley.

G. L. Nemhauser, M. W. P. Savelsbergh and G. C. Sigismondi (1994), 'MINTO: A Mixed INTeger Optimizer', *Oper. Res. Lett.* **15**, 47–58.

J. Nocedal and S. Wright (1999), *Numerical Optimization*, Springer.

I. Nowak, H. Alperin and S. Vigerske (2003), LaGO: An object oriented library for solving MINLPs. In *Proc. 1st Global Optimization and Constraint Satisfaction Workshop: COCOS 2002* (C. Bliek, C. Jermann and A. Neumaier, eds), Vol. 2861 of *Lecture Notes in Computer Science*, Springer, pp. 32–42.

J. Oldenburg, W. Marquardt, D. Heinz and D. Leineweber (2003), 'Mixed logic dynamic optimization applied to batch distillation process design', *AIChE J.* **49**, 2900–2917.

M. Padberg (1989), 'The Boolean Quadric Polytope: Some characteristics, facets and relatives', *Math. Program. B* **45**, 139–172.

M. Padberg (2000), 'Approximating separable nonlinear functions via mixed zero-one programs', *Oper. Res. Lett.* **27**, 1–5.

R. Powell (2007), 'Defending against terrorist attacks with limited resources', *Amer. Political Sci. Review* **101**, 527–541.

A. Prata, J. Oldenburg, A. Kroll and W. Marquardt (2008), 'Integrated scheduling and dynamic optimization of grade transitions for a continuous polymerization reactor', *Comput. Chem. Engng* **32**, 463–476.

K. A. Pruitt, S. Leyffer, A. M. Newman and R. Braun (2012), Optimal design and dispatch of distributed generation systems. Preprint ANL/MCS-2004-0112, Mathematics and Computer Science Division, Argonne National Laboratory.

A. Qualizza, P. Belotti and F. Margot (2012), Linear programming relaxations of quadratically constrained quadratic programs. In *Mixed Integer Nonlinear Programming*, Vol. 154 of *The IMA Volumes in Mathematics and its Applications*, Springer, pp. 407–426.

I. Quesada and I. E. Grossmann (1992), 'An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems', *Comput. Chem. Engng* **16**, 937–947.

A. J. Quist, R. van Gemeert, J. E. Hoogenboom, T. Ílles, C. Roos and T. Terlaky (1998), 'Application of nonlinear optimization to reactor core fuel reloading', *Ann. Nuclear Energy* **26**, 423–448.

K. Rashid, S. Ambani and E. Cetinkaya (2013), 'An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization', *Engng Optim.* **45**, 185–206.

R. Romero, A. Monticelli, A. Garcia and S. Haffner (2002), 'Test systems and mathematical models for transmission network expansion planning', *IEEE Proceedings: Generation, Transmission and Distribution* **149**, 27–36.

G. Rote (1992), 'The convergence rate of the sandwich algorithm for approximating convex functions', *Computing* **48**, 337–61.

R. Y. Rubinstein and D. P. Kroese (2004), *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*, Springer.

H. S. Ryoo and N. V. Sahinidis (1995), 'Global optimization of nonconvex NLPs and MINLPs with applications in process design', *Comput. Chem. Engng* **19**, 552–566.

H. S. Ryoo and N. V. Sahinidis (1996), 'A branch-and-reduce approach to global optimization', *J. Global Optim.* **8**, 107–139.

S. Sager (2005), *Numerical Methods for Mixed-Integer Optimal Control Problems*, Der Andere Verlag.

S. Sager (2012), A benchmark library of mixed-integer optimal control problems. In *Mixed Integer Nonlinear Programming* (J. Lee and S. Leyffer, eds), Springer, pp. 631–670.

S. Sager, H. Bock and M. Diehl (2012), 'The integer approximation error in mixed-integer optimal control', *Math. Program. A* **133**, 1–23.

S. Sager, M. Diehl, G. Singh, A. Küpper and S. Engell (2007), Determining SMB superstructures by mixed-integer control. In *Proc. OR2006*, Springer, pp. 37–44.

S. Sager, M. Jung and C. Kirches (2011), 'Combinatorial integral approximation', *Math. Methods Oper. Res.* **73**, 363–380.

S. Sager, G. Reinelt and H. Bock (2009), 'Direct methods with maximal lower bound for mixed-integer optimal control problems', *Math. Program.* **118**, 109–149.

N. V. Sahinidis (1996), 'BARON: A general purpose global optimization software package', *J. Global Optim.* **8**, 201–205.

T. Sandler and D. G. Arce M (2003), 'Terrorism and game theory', *Simulation Gaming* **34**, 319–337.

T. Sandler and K. Siqueira (2006), 'Global terrorism: Deterrence versus preemption', *Canad. J. Economics* **39**, 1370–1387.

M. W. P. Savelsbergh (1994), 'Preprocessing and probing techniques for mixed integer programming problems', *ORSA J. Comput.* **6**, 445–454.

A. Saxena, P. Bonami and J. Lee (2010), 'Convex relaxations of non-convex mixed integer quadratically constrained programs: Extended formulations', *Math. Program.* **124**, 383–411.

A. Saxena, P. Bonami and J. Lee (2011), 'Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations', *Math. Program.* **130**, 359–413.

H. Schichl (2004), Global optimization in the COCONUT project. In *Numerical Software with Result Verification*, Springer, pp. 243–249.

A. Schrijver (1986), *Theory of Linear and Integer Programming*, Wiley.

C. A. Schweiger (1999), Process synthesis, design, and control: Optimization with dynamic models and discrete decisions. PhD thesis, Princeton University.

O. Shaik, S. Sager, O. Slaby and D. Lebiedz (2008), 'Phase tracking and restoration of circadian rhythms by model-based optimal control', *IET Systems Biology* **2**, 16–23.

W. Sheikh and A. Ghafoor (2010), 'An optimal bandwidth allocation and data droppage scheme for differentiated services in a wireless network', *Wireless Communications and Mobile Comput.* **10**, 733–747.

H. Sherali and W. Adams (1998), *A Reformulation–Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer.

H. Sherali and A. Alameddine (1992), 'A new reformulation–linearization technique for bilinear programming problems', *J. Global Optim.* **2**, 379–410.
H. Sherali and E. Smith (1997), 'A global optimization approach to a water distribution network design problem', *J. Global Optim.* **11**, 107–132.
H. D. Sherali (2001), 'On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions', *Oper. Res. Lett.* **28**, 155–160.
H. D. Sherali and B. M. P. Fraticelli (2002), 'Enhancing RLT relaxations via a new class of semidefinite cuts', *J. Global Optim.* **22**, 233–261.
H. D. Sherali, S. Subramanian and G. V. Loganathan (2001), 'Effective relaxations and partitioning schemes for solving water distribution network design problems to global optimality', *J. Global Optim.* **19**, 1–26.
R. Sinha, A. Yener and R. D. Yates (2002), 'Noncoherent multiuser communications: Multistage detection and selective filtering', *EURASIP J. Appl. Signal Processing* **12**, 1415–1426.
H. Skrifvars, S. Leyffer and T. Westerlund (1998), 'Comparison of certain MINLP algorithms when applied to a model structure determination and parameter estimation problem', *Comput. Chem. Engng* **22**, 1829–1835.
E. M. B. Smith and C. C. Pantelides (1997), 'Global optimization of nonconvex MINLPs', *Comput. Chem. Engng* **21**, S791–S796.
M. Soleimanipour, W. Zhuang and G. H. Freeman (2002), 'Optimal resource management in wireless multimedia wideband CDMA systems', *IEEE Trans. Mobile Computing* **1**, 143–160.
M. Soler, A. Olivares, E. Staffetti and P. Bonami (2011), En-route optimal flight planning constrained to pass through waypoints using MINLP. In *Proc. 9th USA/Europe Air Traffic Management Research and Development Seminar*, Berlin.
C. Still and T. Westerlund (2006), 'Solving convex MINLP optimization problems using a sequential cutting plane algorithm', *Comput. Optim. Appl.* **34**, 63–83.
R. Stubbs and S. Mehrotra (1999), 'A branch-and-cut method for 0–1 mixed convex programming', *Math. Program.* **86**, 515–532.
R. Stubbs and S. Mehrotra (2002), 'Generating convex polynomial inequalities for mixed 0–1 programs', *J. Global Optim.* **24**, 311–332.
M. Tawarmalani and N. V. Sahinidis (2002), *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer.
M. Tawarmalani and N. V. Sahinidis (2004), 'Global optimization of mixed integer nonlinear programs: A theoretical and computational study', *Math. Program.* **99**, 563–591.
M. Tawarmalani and N. V. Sahinidis (2005), 'A polyhedral branch-and-cut approach to global optimization', *Math. Program.* **103**, 225–249.
M. Tawarmalani, J.-P. Richard and K. Chung (2010), 'Strong valid inequalities for orthogonal disjunctions and bilinear covering sets', *Math. Program.* **124**, 481–512.
S. Terwen, M. Back and V. Krebs (2004), Predictive powertrain control for heavy duty trucks. In *Proc. IFAC Symposium in Advances in Automotive Control* (G. Rizzo, L. Glielmo, C. Pianese and F. Vasca, eds), Elsevier, pp. 451–457.

J. Tomlin (1981), 'A suggested extension of special ordered sets to non-separable non-convex programming problems', *Ann. Discrete Math.* **11**, 359–370.
A. Toriello and J. P. Vielma (2012), 'Fitting piecewise linear continuous functions', *European J. Oper. Res.* **219**, 86–95.
M. Türkay and I. E. Grossmann (1996), 'Logic-based MINLP algorithms for the optimal synthesis of process networks', *Comput. Chem. Engng* **20**, 959–978.
T. J. Van Roy (1983), 'Cross decomposition for mixed integer programming', *Math. Program.* **25**, 145–163.
D. Vandenbussche and G. L. Nemhauser (2005*a*), 'A branch-and-cut algorithm for nonconvex quadratic programs with box constraints', *Math. Program.* **102**, 559–575.
D. Vandenbussche and G. L. Nemhauser (2005*b*), 'A polyhedral study of nonconvex quadratic programs with box constraints', *Math. Program.* **102**, 531–557.
J. P. Vielma, S. Ahmed and G. Nemhauser (2010), 'Mixed-integer models for non-separable piecewise-linear optimization: Unifying framework and extensions', *Oper. Res.* **58**, 303–315.
J. P. Vielma, S. Ahmed and G. L. Nemhauser (2008), 'A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs', *INFORMS J. Comput.* **20**, 438–450.
J. P. Vielma and G. Nemhauser (2011), 'Modeling disjunctive constraints with a logarithmic number of binary variables and constraints', *Math. Program.* **128**, 49–72.
J. Viswanathan and I. E. Grossmann (1990), 'A combined penalty function and outer-approximation method for MINLP optimization', *Comput. Chem. Engng* **14**, 769–782.
A. Wächter and L. T. Biegler (2006), 'On the implementation of a primal–dual interior point filter line search algorithm for large-scale nonlinear programming', *Math. Program.* **106**, 25–57.
T. Westerlund and K. Lundqvist (2005), Alpha-ECP, version 5.101: An interactive MINLP-solver based on the extended cutting plane method. Technical report 01-178-A, Process Design Laboratory at Åbo University.
T. Westerlund and F. Pettersson (1995), 'A cutting plane method for solving convex MINLP problems', *Comput. Chem. Engng* **19**, s131–s136.
T. Westerlund and R. Pörn (2002), 'Solving pseudo-convex mixed integer optimization problems by cutting plane techniques', *Optim. Engng* **3**, 253–280.
H. P. Williams (1999), *Model Building in Mathematical Programming*, Wiley.
D. L. Wilson (1998), Polyhedral methods for piecewise-linear functions. PhD thesis, University of Kentucky.
D. D. Wolf and Y. Smeers (2000), 'The gas transmission problem solved by an extension of the simplex algorithm', *Management Sci.* **46**, 1454–1465.
L. A. Wolsey (1998), *Integer Programming*, Wiley.
X. Wu, E. Topuz and M. Karfakis (1991), Optimization of ventilation control device locations and sizes in underground mine ventilation systems. In *Proc. 5th US Mine Ventilation Symposium*, Society for Mining, Metallurgy, and Exploration, pp. 391–399.
Y. Yajima and T. Fujie (1998), 'Polyhedral approach for nonconvex quadratic programming problems with box constraints', *J. Global Optim.* **13**, 151–170.

F. You and S. Leyffer (2010), 'Oil spill response planning with MINLP', *SIAG/OPT Views-and-News* **21**, 1–8.

F. You and S. Leyffer (2011), 'Mixed-integer dynamic optimization for oil-spill response planning with integration of a dynamic oil weathering model', *AIChe J.* **57**, 3555–3564.

Y. Zhu and T. Kuno (2006), 'A disjunctive cutting-plane-based branch-and-cut algorithm for 0–1 mixed-integer convex nonlinear programs', *Indust. Engng Chem. Research* **45**, 187–196.

J. Zhuang (2008), Modeling secrecy and deception in homeland security resource allocation. PhD thesis, University of Wisconsin–Madison.

J. Zhuang and V. M. Bier (2007*a*), 'Balancing terrorism and natural disasters: Defensive strategy with endogenous attacker effort', *Oper. Res.* **55**, 976–991.

J. Zhuang and V. M. Bier (2007*b*), 'Investment in security', *Industrial Engineer* **39**, 53–54.

J. Zhuang and V. M. Bier (2010), 'Reasons for secrecy and deception in homeland-security resource allocation', *Risk Analysis* **30**, 1737–1743.