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

© Cambridge University Press, 2013
Printed in the United Kingdom

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

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

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 $l \le c(x) \le u$. 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'.

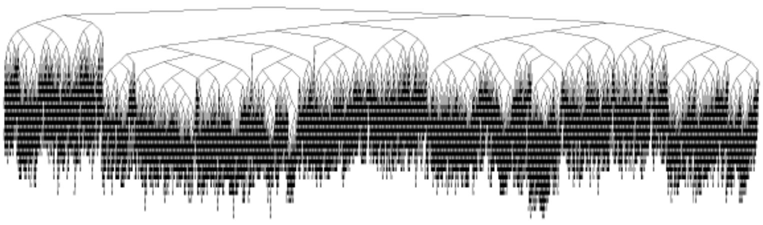

Figure 1.1. A branch-and-bound tree without presolve after 360 s CPU time has more than 10 000 nodes.

of x corresponding to J. In particular, x_I are the integer variables. We also define $C = \{1, \ldots, n\} - I$ 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 $\operatorname{conv}(S)$ and defined as

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

If $X = \{x \in \mathbb{Z}^p : l \le x \le u\}$ and $l \in \mathbb{Z}^p$, $u \in \mathbb{Z}^p$, then $\operatorname{conv}(X) = [l, u]$ is simply the hypercube. In general, however, even when X itself is polyhedral, it is not easy to find $\operatorname{conv}(X)$. 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:

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

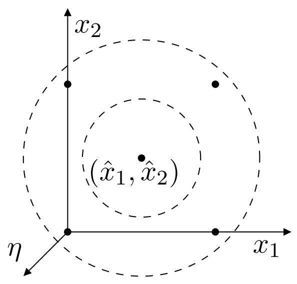

Figure 1.2. Small MINLP to illustrate the need for a linear objective function.

The solution of the continuous relaxation is $x = \left(\frac{1}{2}, \dots, \frac{1}{2}\right)$, 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

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

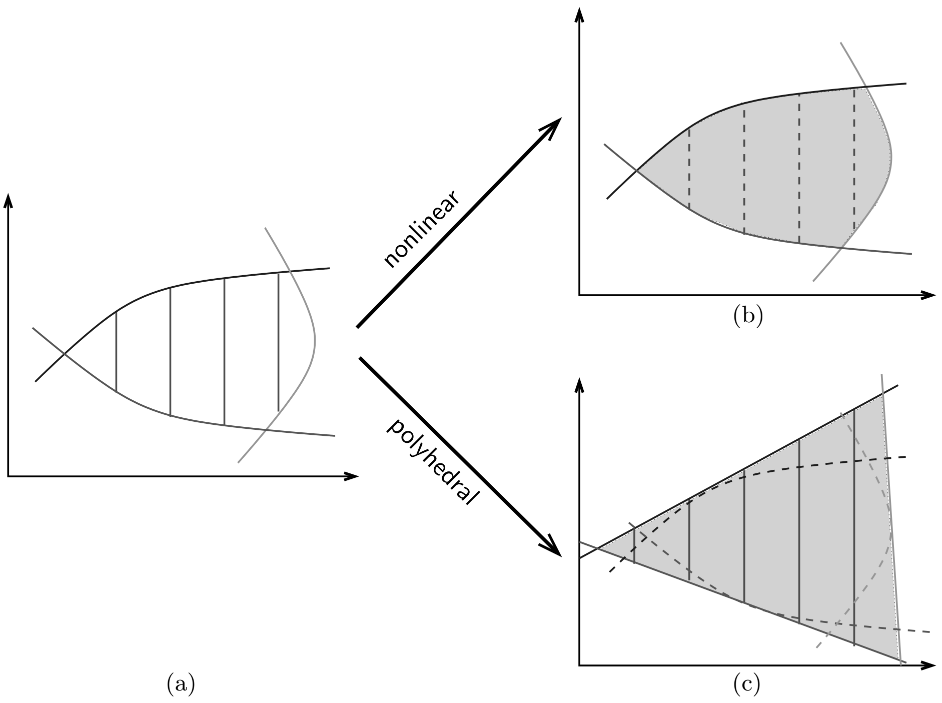

Figure 1.3. Illustration of the two classes of relaxation. (a) The mixed-integer feasible set, (b) the nonlinear relaxation, and (c) the polyhedral relaxation.

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

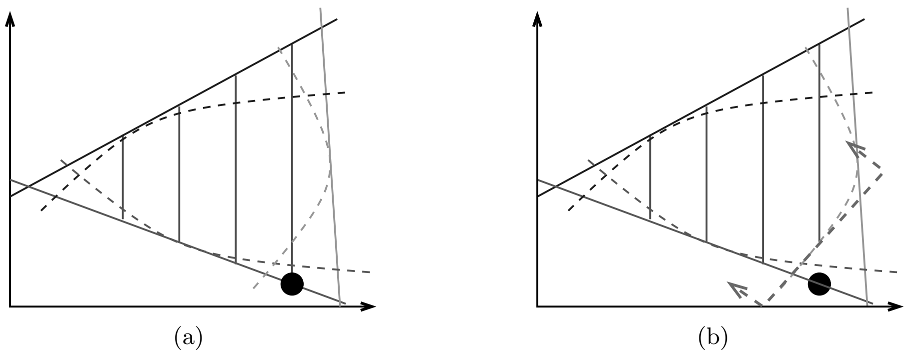

Figure 1.4. Separation of infeasible point (black dot) by adding a separating hyperplane. The dashed line with arrows (b) shows a separating hyperplane with arrows indicating the feasible side.

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

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, $c(\hat{x}) \le 0$: 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

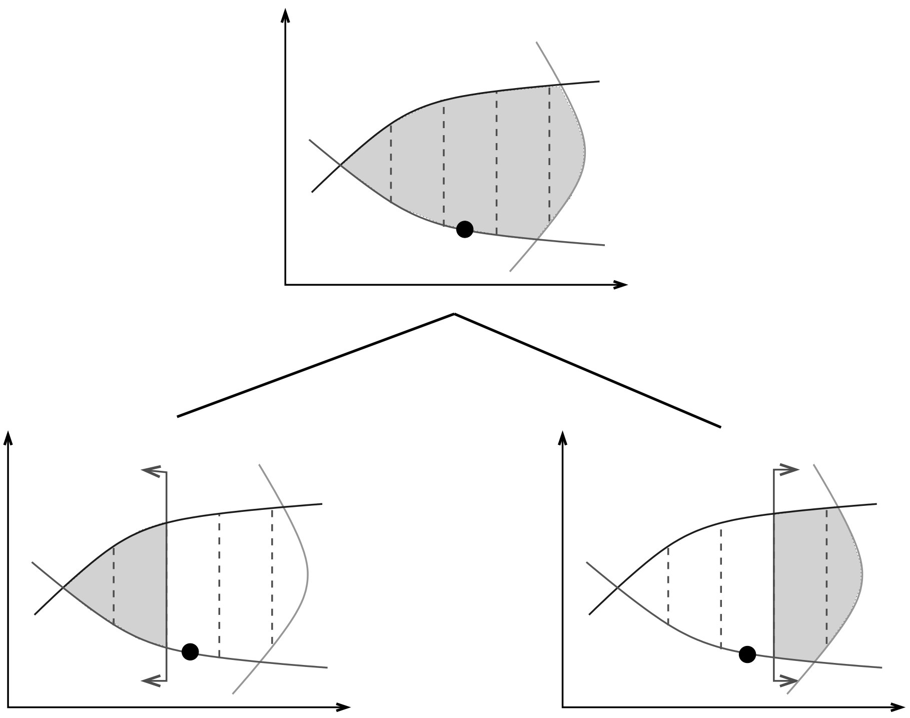

Figure 1.5. Branching on the values of an integer variable creates two new nonlinear subproblems that both exclude the infeasible point, denoted by the black dot.

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

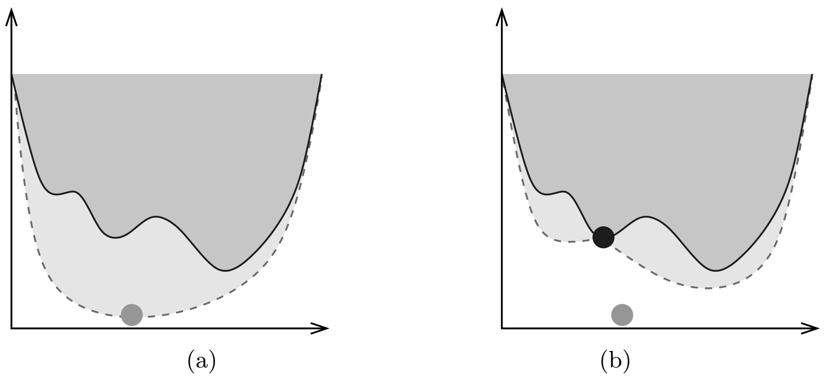

Figure 1.6. Constraint enforcement by using spatial branching for global optimization.

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.