Optimization: Applications, Algorithms, and Computation

24 Lectures on Nonlinear Optimization and Beyond

Sven Leyffer

(with help from Pietro Belotti, Christian Kirches, Jeff Linderoth, Jim Luedtke, and Ashutosh Mahajan)

August 30, 2016

To my wife Gwen
my parents Inge and Werner;
and my teacher and mentor Roger.

Contents

List of Figures

List of Tables

Part I

Introduction, Applications, and Modeling

Chapter 1

Introduction to Optimization

We start by stating a general optimization problem, and discussing how optimization problems can be classified by their functional forms, variables types, and structure.

1.1 Objective Function and Constraints

Optimization is the art of finding a best solution from a collection alternatives. It has numerous applications in science, engineering, finance, medicine, economics, and big-data. We review some of these applications in more detail in Chapter 2. In all these applications, we model our decisions using a set of independent variables, which may be constrained to lie in some region. The goal is to identify values of these variables that maximize (or minimize) a performance measure, or objective function that measures the quality of a solution.

Formally, we consider optimization problems of the form

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) && \text{(1.1a)}\\
\text{subject to} \quad & l_c \leq c(x) \leq u_c && \text{(1.1b)}\\
& l_A \leq A^T x \leq u_A && \text{(1.1c)}\\
& l_x \leq \quad x \leq u_x && \text{(1.1d)}\\
& x \in \mathcal{X}. && \text{(1.1e)}
\end{aligned}$$

Here, we are mainly concerned with finite-dimensional optimization problems, where $x \in \mathbb{R}^n$, and the problem functions, $c : \mathbb{R}^n \to \mathbb{R}^m$ and $f : \mathbb{R}^n \to \mathbb{R}$ are smooth (typically twice continuously differentiable over an open set containing $\mathcal{X}$). The set $\mathcal{X} \subset \mathbb{R}^n$ imposes further structural restrictions on the variables that are discussed in subsequent chapters. $A \in \mathbb{R}^{n\times p}$ is a matrix, and the bounds, $l_c, u_c, l_A, u_A, l_x, u_x$, are of suitable dimensions and can include components that are infinite.

To Minimize or To Maximize. It is well known that $\max\ f(x)$ is equivalent to $-\min\ -f(x)$. Hence, we only consider minimization problems in this book.

Notation and Nomenclature. We call $f(x)$ the objective function and $c(x)$ the nonlinear constraint functions . We use subscripts to denote components of vectors, e.g. for $a \in \mathbb{R}^n$, the components are $a = (a_1, \ldots, a_n)^T$, and vectors are assumed to be columns vectors unless otherwise stated. For two vectors, $a, b \in \mathbb{R}^n$, the notation $a \leq b$ means that $a_i \leq b_i$ for all $i = 1, \ldots, n$. We use upper case letters for matrices, lower case letters for vectors (or scalars), and calligraphic type for sets, e.g. $\mathcal{X} \subset \mathbb{R}^n$.

1.2 Classification of Optimization Problems

We distinguish or classify optimization problems by the type of variables, constraints, and functions involved in its definition. A good early classification was given in the NEOS guide, see Figure 1.1. It has since grown considerably with the addition of more complex classes of constraints. Here, we briefly discuss some basic classes of optimization problems, which we classify by constraint type, variable type, and function type. In addition, combinations of these classifications lead to important and challenging optimization problems.

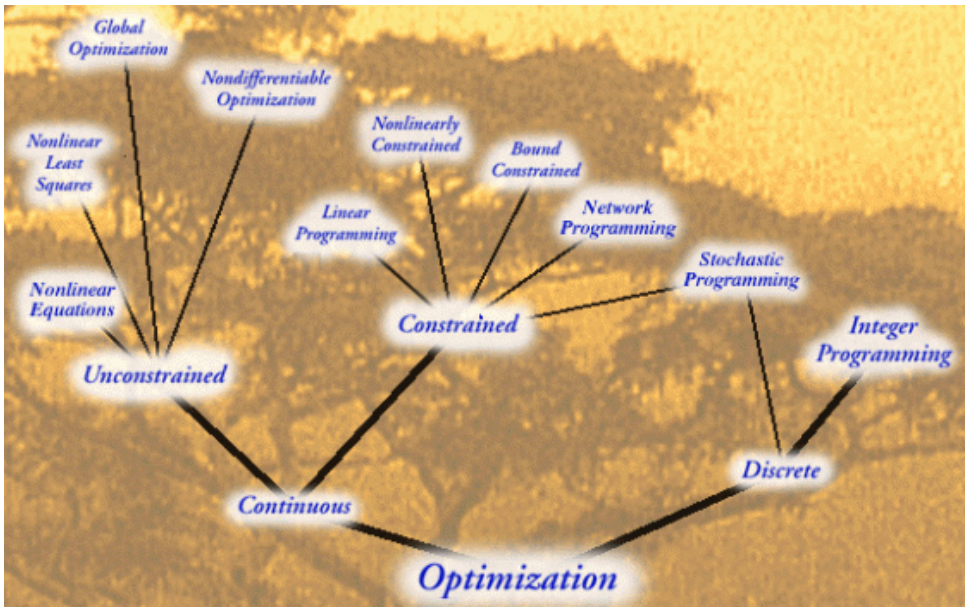


Figure 1.1: NEOS Optimization Tree.

1.2.1 Classification by Constraint Type

We can build a hierarchy of classes of optimization problems by considering different sets of constraints. In all these classes, we assume that $\mathcal{X} = \mathbb{R}^n$, and that the functions involved are at least twice continuously differentiable (other variable types and functional forms are considered in the next two sections).

Unconstrained Optimization are problems without any additional constraints, i.e. (1.1b) to (1.1d) are not present. Problems of this class are simply expressed as

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x).$$

Bound Constrained Optimization are problems, where we minimize an objective function subject to bound constraints only. Problems of this class are expressed as

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; f(x) \quad \text{subject to } l \leq x \leq u,$$

where the bounds $l, u \in \mathbb{R}^n$ and either bound can infinite for any component. Note, that if $f(x) = c^T x$ is linear, then this problem can be solved trivially, see Exercise 1.1

Linearly Constrained Optimization are problems, where we minimize a nonlinear function subset to linear constraints, (1.1c) and (1.1d), only. Two special classes of linearly constrained optimization problems that we will study in detail are linear programming or quadratic programming problems, where the objective function is linear, $f(x) = c^T x$, or quadratic, $f(x) = x^T Gx/2 + g^T x$, respectively.

Equality Constrained Optimization are problems, where all constraints are equality constraints, and can be expressed as

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) = 0. \end{array}$$

It is often useful to distinguish problems with only linear constraints, because we can typically project onto linear constraints more easily.

Nonlinearly Constrained Optimization are problems of the most general form, (1.1).

In addition, we can define many special classes of problems, such as nonlinear equations, which are a subset of equality constrained optimization, or least-squares problems, which are a subset of unconstrained optimization for which the objective function is a sum of squares, such as

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \; f(x) = \sum_{j=1}^{m} \left(r_j(x)\right)^2,$$

where the functions $r_j(x)$ define the residual, e.g. $r_j(x) = a_j^T x - b_j$ for the linear least-squares problem.

Recently, new classes of constraints have emerged in practical applications:

Semi-Definite Optimization are problems that involve matrix variables, $X \in \mathbb{R}^{n\times n}$, that are required to be positive semi-definite, i.e. symmetric matrices with nonnegative eigenvalues. Constraints of this form are written as $X \succeq 0$. The semi-definiteness constraint generalizes the nonnegativity constraint, $x \geq 0$, and both can be shown to form cones in $\mathbb{R}^n$ and $\mathbb{R}^{n\times n}$, respectively.

Second-Order Cone Constraints are problems with a special class of quadratic constraints that form a cone in $\mathbb{R}^n$. One example is is the second-order cone defined by the set

$$\{(x_0, x) \in \mathbb{R} \times \mathbb{R}^n \mid x_0 \geq ||x||_2\},$$

also known as the ice-cream cone.

1.2.2 Classification by Variable Type

Within each class of optimization problem, defined by its constraint type, we can further subdivide optimization problems depending on their variable type. In our generic optimization problem, (1.1), the variable type is encoded in the set $\mathcal{X}$. There are two main variable types that we distinguish here:

Continuous Variables are variables that take values in $\mathcal{X} \subset \mathbb{R}^n$. This is the easier class of problems, because it allows us to leverage standard calculus techniques.

Discrete Variables are variables that lie in a discrete subset. The main classes of discrete variables are

- *Binary Variables*, whose values must lie in $\mathcal{X} = \{0, 1\}^n$. Variables of this type model binary or logical decisions, such as the presence or absence of a piece of equipment.
- *Integer Variables*, which take values in $\mathcal{X} = \mathbb{Z}^n$, the integer lattice. Variables of this type typically model the number of equipment, or repeated actions.

Problems of this type are referred to as integer or discrete programming problems.

Many problems contain both continuous and discrete variables, and are referred to as mixed-integer programming problems.

In addition to these two main classes of variable types, there exists further classes of variables that are important in many practical applications.

State and Control Variables appear in optimal control and certain design optimization problems that are governed by differential equation constraints. These variables are infinite dimensional, and are themselves functions of time, t, and or position, $(x, y, z) \in \mathbb{R}^3$.

Random Variables appear in robust or stochastic optimization problems, and are often called second-stage variables. In this case, we typically optimize the expectation or some other stochastic function.

These variables are typically discretized on a finite-element mesh (or by drawing random samples), and can in principle be treated similarly to finite-dimensional variables (though important differences exist, and care has to be taken to use consistent discretizations).

1.2.3 Classification by Functional Forms

In addition to the classification by constraint and variable type, we also distinguish problems by the type of their functional forms. In most cases, we will assume that the functions are at least twice continuously differentiable. Problems in which the functions are only Lipschitz continuous (e.g. $f(x) = \|x\|_1$) are called *nonsmooth optimization problems*.

Finally, we note, that we can also consider problems with multiple objective functions, which arise when we need to model trade-offs between different goals that cannot easily be converted into one another, such as maximizing take-off weight (measured in tons), while minimizing fuel consumption (measured in \$) for an airliner. Problems of this form are called *multi-objective optimization problems*. We note, that there exist various techniques that convert multi-objective optimization problems into (a series of) single-objective optimization problems, and we will hence assume that we only have a single objective in the remainder of these notes.

1.3 A First Optimization Example

Consider the design of a reinforced concrete beam to support a load (more complex examples, where we consider structures such as houses can be readily derived). We are interested in minimizing the cost of the reinforced beam, which is the sum of the steel reinforcement, and the concrete. The variables are the area of the re-reinforcement, x_1, and the width and depth of the beam, x_2 and x_3.

The full problem can be defined as

$$\begin{array}{lll} \underset{x}{\text{minimize}} & f(x) = 29.4x_1 + 0.6x_2x_3 & \text{cost of beam} \\ \text{subject to} & c(x) = x - 1x_2 - 7.735\frac{x_1^2}{x_2} \geq 180 & \text{load constraint} \\ & x_3 - 4x_2 \leq 0 & \text{width/depth ratio} \\ & 40 \leq x_1 \leq 77,\ x_2 \geq 0,\ x_3 \geq 0 & \text{simple bounds,} \end{array} \tag{1.2}$$

where the objective function adds the cost of the reinforcement and the concrete, the nonlinear constraint describes the load, the linear constraint describes the desired width to depth ratio, and the simple bounds describe positivity and size constraints. In practice, some of the variables such as the size of the reinforcement may be integer, because they have to be chosen from a set of prefabricated units.

1.4 Course Outline

This course is divided into four related parts. In Part I, we introduce general concepts of optimization, such as the optimization problem, its variables, and its constraints. We classify general optimization problems, and provide a range of application examples.

In Part II, we consider unconstrained and bound-constrained optimization problems. We derive their optimality conditions, and consider two general classes of methods, namely quasi-Newton methods and conjugate gradient methods. We also introduce the global convergence mechanisms such as a line search and a trust region.

In Part III, we discuss general nonlinear optimization problems. We start by investigating their optimality conditions, then consider two special classes of problems that form the basis of general nonlinear methods, namely linear and quadratic programs. Next, we introduce three classes of methods: active-set type methods, interior-point methods, and augmented Lagrangian methods. We conclude this part by showing how these methods can be extended to solve more challenging classes of nonlinear problems.

Part IV is dedicated to mixed-integer nonlinear optimization. We introduce modeling techniques for integer variables, discuss the three main classes of methods, and then present methods for dealing with nonconvex constraints. We finish this section by presenting heuristics with mixed-integer optimization, and introducing a new class of problems: mixed-integer PDE-constrained optimization problems.

1.5 Exercises

1.1. Show that

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; c^T x \quad \text{subject to } l \leq x \leq u,$$

can be solved trivially. Hence, write down its solution.

1.2. Create an AMPL model of Example (1.2).

Chapter 2

Applications of Optimization

We briefly review some important applications of optimization, and provide a list of further applications. We focus on applications in power systems and other areas that are of interest to the US Department of Energy.

2.1 Power-System Engineering: Electrical Transmission Planning

Several emerging paradigms are pushing the expansion of the national transmission network. In particular, the successful incorporation of demand-response programs, increasing interconnect exchanges, renewable generation, and storage strongly depends on transmission capabilities. The transmission network facilitates market transactions and thus reduces prices and maximizes social welfare [359]. Planning and installing new transmission facilities is a complex task, however, since it involves long-term and expensive financial decisions. In addition, the resulting network needs to satisfy stringent reliability constraints under myriad uncertain future scenarios. Uncertainty is mainly related to emerging technologies (e.g.; high-voltage direct current, compressed-air storage), spatiotemporal variations in climate and loads, natural gas prices, adoption of carbon emission programs, and wind power adoption [120, 252]. The objective is to have a network that works efficiently under all these scenarios.

The expansion of a transmission system is a multi-stage process where decisions can be made and adapted as uncertainty is resolved. Such a process can be performed systematically by using mixed-integer stochastic optimization. The term "mixed-integer" comes from the fact that the transmission problem contains both integer and continuous variables. Integer variables arise from installation decisions (binary) while continuous variables arise from the physical representation of the transmission network (e.g., Kirchkoff's law).

Deterministic and stochastic mixed-integer optimization methods have been used for transmission planning [15, 109, 290]. Some formulations use static planning (one year decision), while others use multi-year planning. One of the key challenges identified in these works is the need to capture operational details in the planning decisions. For instance, if a line is installed, the expanded network must be guaranteed to work efficiently under real-time operational environments including economic dispatch, AC power flow, and $N-1$ reliability tasks. Since the loads exhibit hourly and seasonal fluctuations throughout the year (see Figure 2.1), capturing the performance of the updated network requires a large number of *time snapshots*. For instance, a single year would require 8760 hourly snapshots. Since each of these snapshots would incorporate a representation of the network, the resulting problem quickly becomes intractable. If uncertain scenarios are added to the picture, the problem becomes even harder. As an order of magnitude estimate, a transmission system with 2,000 buses, 8,760 snapshots, and 100 scenarios would give an optimization problem with 1×10^9 variables. Reducing the number of snapshots to 100 would still give a problem with 1×10^7

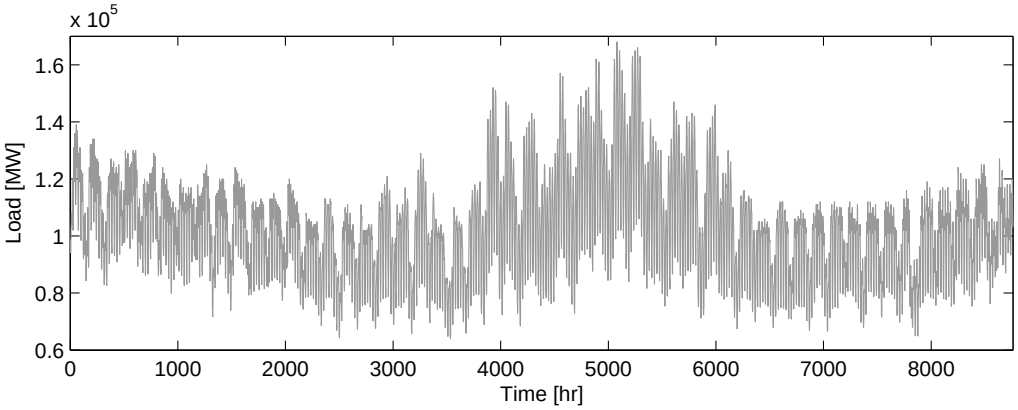


Figure 2.1: Fluctuations of total load for Illinois system in year 2006.

variables. These problems cannot be handled by off-the-shelf optimization solvers and standard personal computers. Furthermore, as noted in a recent workshop on transmission planning software organized by the Federal Energy Regulatory Commission, the ISOs are demanding more complex transmission formulations able to capture market bidding and settlement procedures, AC power flow, forecast errors, ramp constraints, unit commitment, and transmission switching [117, 249, 291].

2.1.1 Transmission Planning Formulation

In this section, we describe canonical transmission planning formulations that will be used throughout the report to explain our main developments.

Power-grid models can be described on graphs or networks. In the power-grid community, arc or edges are referred to as lines, and nodes are referred to as buses. We consider a network with a set of L_I *existing* lines defined over the set $\mathcal{L}_I$, B buses over $\mathcal{B}$, G_j generators over $\mathcal{G}_j$, D_j loads over $\mathcal{D}_j$, and W_j wind generators over $\mathcal{W}_j$. Here, sets subindexed by j indicate subsets connected to bus j. We also define a set of *candidate* lines $\mathcal{L}_C$, a transmission planning horizon with a T years over set $\mathcal{T}$, and K hourly time instants or snapshots over set $\mathcal{K}$. The total number of existing and candidate lines, buses, generator loads, and wind generators is denoted by L, B, G, D, and $\mathcal{W}$, respectively. The objective is to compute an installation schedule for the candidate lines defined by the binary variables $\mathbf{y}^L_{t,i,j} \in [0,1]$ that minimizes the accumulated generation costs over the planning horizon $\mathcal{T}$. The network DC loads are defined by $L^D_{t,k,j}$, the generation flows are $G_{t,k,j}$, the wind power flows are $L^W_{t,k,j}$, the branch flows are $P_{t,k,i,j}$, and the phase angles are $\theta_{t,k,j}$.

The susceptance of the lines is given by $b_{i,j}$. The formulation is presented in equation (2.1).

$$\min \quad \sum_{t\in\mathcal{T}} \sum_{(i,j)\in\mathcal{L}^C} c^L_{t,i,j}(\mathbf{y}^L_{t+1,i,j} - \mathbf{y}^L_{t,i,j}) + \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{G}} c^G_{t,j} \cdot G_{t,k,j} \tag{2.1a}$$

$$\text{s.t.} \quad \mathbf{y}^L_{t+1,i,j} \geq \mathbf{y}^L_{t,i,j},\ t \in \mathcal{T}, (i,j) \in \mathcal{L}_C \tag{2.1b}$$

$$\mathbf{y}^L_{t,i,j} \in [0,1],\ t \in \mathcal{T}, (i,j) \in \mathcal{L}_C \tag{2.1c}$$

$$|P_{t,k,i,j}| \leq P^{max}_{i,j} \cdot \mathbf{y}^L_{t,i,j},\ t \in \mathcal{T}, k \in \mathcal{K}, (i,j) \in \mathcal{L}_C \tag{2.1d}$$

$$|P_{t,k,i,j}| \leq P^{max}_{i,j},\ t \in \mathcal{T}, k \in \mathcal{K}, (i,j) \in \mathcal{L}_I \tag{2.1e}$$

$$|P_{t,k,i,j} - b_{i,j}(\theta_{t,k,i} - \theta_{t,k,j})| \leq M_{i,j} \cdot (1 - \mathbf{y}^L_{t,i,j}), t \in \mathcal{T}, k \in \mathcal{K}, (i,j) \in \mathcal{L}_C \tag{2.1f}$$

$$P_{t,k,i,j} = b_{i,j}(\theta_{t,k,i} - \theta_{t,k,j}), t \in \mathcal{T}, k \in \mathcal{K}, (i,j) \in \mathcal{L}_I \tag{2.1g}$$

$$\sum_{(i,j)\in\mathcal{L}_j} P_{t,k,i,j} + \sum_{i\in\mathcal{W}_j} L^W_{t,k,i} + \sum_{i\in\mathcal{G}_j} G_{t,k,i} = \sum_{i\in\mathcal{D}_j} L^D_{t,k,i},\ t, \in \mathcal{T}, k \in \mathcal{K}, j \in \mathcal{B} \tag{2.1h}$$

$$0 \leq G_{t,k,j} \leq G^{max}_j,\ t \in \mathcal{T}, k \in \mathcal{K}, j \in \mathcal{G} \tag{2.1i}$$

$$|\theta_{t,k,j}| \leq \theta^{max}_j,\ t \in \mathcal{T}, k \in \mathcal{K}, j \in \mathcal{B}. \tag{2.1j}$$

This is a mixed-integer linear optimization problem. The objective function comprises installation and generation costs. Constraint (2.1b) reflects the logic that if the installation of a line starts at year t, then the line remains installed during the remaining of the horizon. Constraints (2.1f)-(2.1g) are the DC power flow equations for the candidate and installed lines. Constraints (2.1e),(2.1d) are the congestion constraints for the candidate and installed lines. Constraint (2.1h) is Kirchoff's law and constraints (2.1i),(2.1j) are bounds for the generation flows and the phase angles.

2.2 Other Power Systems Applications

The power grid poses a number of computational challenges, ripe with optimization applications. The main challenges are:

- **Size:** The US grid contains around 100k lines and buses (generators) and has been called the "most complex machine ever built".
- **Complexity:** The power-grid is very complex. The power-flow equations are nonlinear nonlinear, the models contains hierarchical decisions, and some of the decisions are discrete (e.g. unit commitment).
- **Uncertainty:** Both demand and supply (in the case of renewable energy such as solar- and wind-power) are uncertain, and we must take these uncertainties into account in our models.

Figure 2.2 shows the power grid of th United States.

2.2.1 The Power-Flow Equations and Network Expansion

The power-flow equations arise in a range of operational and design problems within the power grid such as optimal power flow, transmission switching, network expansion. Discrete decisions arise in the unit commitment problem (which power plant to run) and network expansion problems, such as (2.1).

The nonlinear (AC) power flow model can be described by:

$$F(U_k, U_l, \theta_k, \theta_l) := b_{kl}U_kU_l \sin(\theta_k - \theta_l) + g_{kl}U_k^2 - g_{kl}U_kU_l \cos(\theta_k - \theta_l),$$

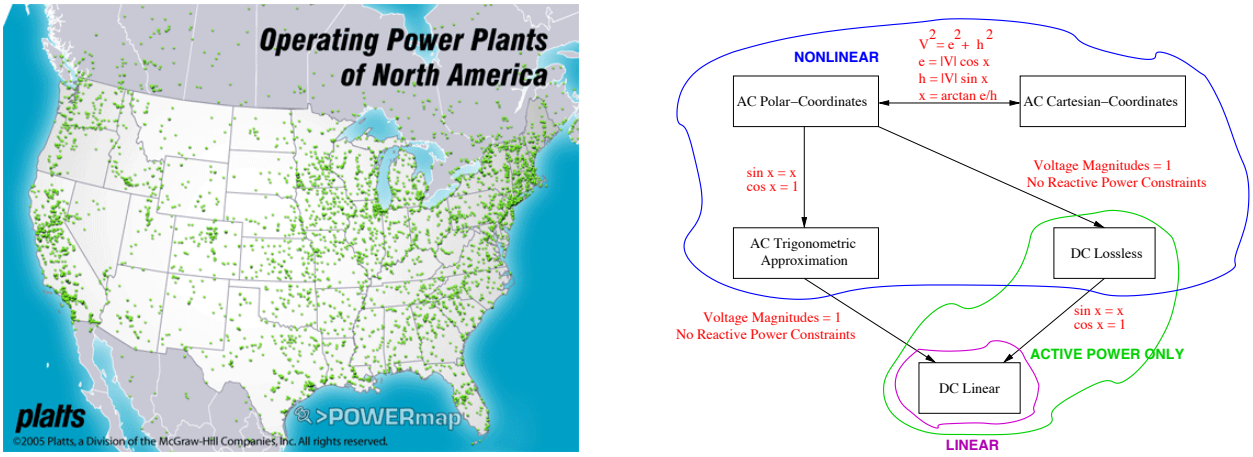


Figure 2.2: Power plants of the US (left), and hierarchy of power-flow models (right).

which describes the flow of power from node k to l, where U_k, U_l are the voltage magnitudes, and θ_k, θ_l are the phase of the (complex) voltage at node k and l, respectively. In the traditional approach to network expansion, we simplify this nonlinear equation by using a first-order Taylor-series approximation to $\sin(x)$ and $\cos(x)$:

$$\sin(x) \simeq x \quad \text{and} \quad \cos(x) \simeq 1 \quad \text{and} \quad U \simeq 1$$

to obtain the linearized (DC) power flow model.

$$F(U_k, U_l, \theta_k, \theta_l) := b_{kl}(\theta_k - \theta_l)$$

Binary variables are used to introduce new lines into the network, giving rise to models that involve

- $-M(1 - z_{k,l}) \leq f_{k,l} - F(U_k, U_l, \theta_k, \theta_l) \leq M(1 - z_{k,l})$
- $z_{k,l} \in \{0, 1\}$ switches lines on/off; $M > 0$ constant

An interesting question is whether the nonlinearities matter. To this end, we compare the results for a network expansion model for linear versus nonlinear power flow models.

The results of this study are shown in Figure 2.3. The blue and green lines are the additional lines for the linear and nonlinear formulation, respectively. This small example shows that there exists a significant difference between the DC and AC solution. In fact, we can show that the linearized DC model is not even feasible in the nonlinear AC power flow model. The reason for this observation is that the Taylor expansion that gave rise to the DC model is only valid, if the phase difference does not change too much. Unfortunately, this assumption does not hold, if the topology of the network changes (which is the whole point of our network expansion study).

2.2.2 Blackout Prevention in National Power Grid

Another important application of optimization is blackout prevention. The 2003 blackout cost $4-10 billion and affected 50 million people (see Figure 2.4). Contingency analysis is a systematic way to prevent blackouts. In this optimization problem, we seek the least number of transmission lines whose removal results in failure. We use binary variables to model the removal of lines, and nonlinearities to model the power flow. This approach results in large integer optimization problem.

Figure 2.3: Comparison of linear and nonlinear power flow equations.

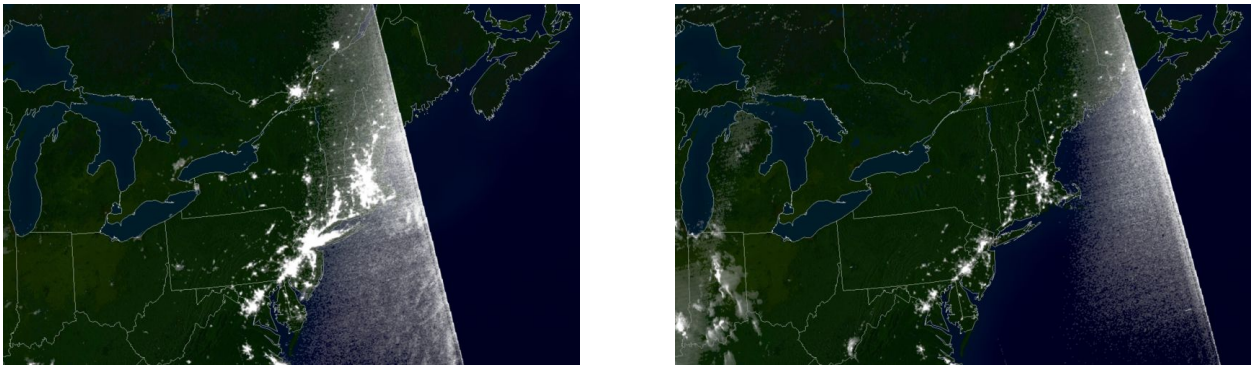


Figure 2.4: Satellite images of the 2003 blackout: before (left) and during(right).

An alternative approach is the $N-k$ contingency analysis which gives rise to a combinatorial number of simulations. The goal is to estimate the vulnerability of the grid to disruption by running "N choose k" scenarios of k out of N lines being shut. This is clearly prohibitive for large N.

Recently, there has been a new optimization-based approach that exploits a bilevel optimization approach. It finds the collection of lines that produce the maximum disruption, by modeling an "attacker" who can decrease the line admittance to disrupt the network. The system operator responds to the attacker by adjusting demands and generation.

2.2.3 Optimal Unit Commitment for Power-Grid

We can account for the uncertainty in renewable energy supply, by, for example, including wind uncertainty in a stochastic optimization problem, where we minimize the expected cost:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) + \mathbb{E}_\omega \left(\min_z \; h(x,z;\omega) \text{ subject to } g(x,z;\omega) \geq 0 \right) \\ \text{subject to} & c(x) \geq 0, \end{array}$$

where, x are here-and-now decisions, the unit commitment, and z are second-stage decisions that correspond to different wind scenarios that model the random realizations of wind for $\omega \in \Omega$ random parameters. This

approach gives rise to huge optimization problems that are solved n leading-edge super computers.

Remark 2.2.1 (Take-Home Message from Power-Grid) *The power grid is a growing area of optimization with an increasing number of important applications driven by*

- *The switch to renewable energy, which introduces uncertainties into our models.*
- *The deregulation and design of electricity markets, which gives rise to bilevel optimization problems, and leader-follower, or Stackelberg games.*
- *The advent of the smart-grid and smart meters, which introduce cybersecurity concerns.*
- *A shift in consumption towards electric vehicles and distributed generation, which disrupts traditional demand and supply patterns.*

In addition, there is an increase in the amount of data that we gather from consumers and on the grid itself, making this a rich source of big data problems.

2.3 A Control Application: Optimal Transition to Clean Energy

This model is an economics model that aims to formulate policy goals on how to transition from our current fossil-fuel based economy to a cleaner economy that avoids the emission of greenhouse gasses. The model is largely theoretical, but allows economists to argue about the impact of policy decision on the transition path.

Our goal is to compute an optimal transition from conventional (old) to low-emission (new) technology for energy production. The new technology has higher costs but a lower emission rate of greenhouse gases, making it possible to reduce emissions without substantial reductions in energy consumption that would be necessary using only the old technology.

We are interested in the socially optimal output schedules of both technologies; these tell us the best possible scenario that could be achieved if the entire energy industry were controlled by a (benevolent and omniscient) single agency. In reality, the energy industry consists of many independent firms, which have to be motivated by policy measures to adopt the new technology. Our model could generate several important inputs to construct such a policy. First, our model determines the optimal output schedule, or transition path, that serves as the ultimate goal of the policy. Second, our model can be used to compute the exact amount of policy intervention (tax rate, emission quota, etc.) Third, if a policy fails to achieve the optimal transition, our model can show how large the shortfall is and whether correcting it justifies the additional effort.

We seek to develop a model that provides a realistic transition path; in other words, the output of either technology should be a continuous, but not necessarily smooth, function of time. In addition, the total energy output should be increasing over time.

The increasing energy output is considered necessary given the continued growth of world population and economic well-being. Gradual transition is motivated by historic data on actual technology transitions that find the penetration rate of the new technology to be an S-shaped function of time—although it is increasing throughout the entire transition period, the penetration rate is convex during the early stages of transition, then passes through an inflection point, and turns concave as it approaches full adoption. see Jensen [261] and Geroski [210]. To be consistent with these results, we develop models with various features such as learning by doing in the new technology, which reduces the unit cost as the cumulative output increases; transition costs, which penalize fast changes in output; and capital investment.

Our paper considers the following economic concepts: technology diffusion, environmental policy, and learning by doing. The model we develop demonstrate different forms of transition behavior, suggesting

that our optimal transition paths are potentially implementable by policy intervention. From an economic standpoint, we demonstrate that the realistic gradual transition can be generated without resorting to multiple agents and is, in fact, socially optimal under reasonable assumptions. We find that learning by doing alone leads to discontinuous instant transition; adding adjustment costs ensures continuous and smooth transition.

2.3.1 Model Description and Background

We solve the social planner's problem of maximizing the social welfare, with the condition that total accumulated emissions at a certain point in time will not exceed the specified level. Our variables are energy output schedules of old and new technology, where the new technology has lower emissions but is more expensive, although its costs are expected to decrease with wider adoption. We can formulate a number of models with increasing detail and features, but concentrate on the easiest model here.

We first define the components of our model. Specific functional forms are chosen for demonstration; however, our approach is not limited to these, and other choices are possible.

Time. Our model is dynamic, with continuous time and finite horizon: $t \in [0, T]$. We denote functions of time as $x(t)$ and their derivatives as

$$\dot{x}(t) = \frac{dx(t)}{dt}.$$

We use continuous discounting with the rate $r > 0$.

Energy Output. There are two technologies, old and new, and their energy output at time t is denoted $q^o(t)$ and $q^n(t)$, respectively. We also define the total output to be $Q(t) = q^o(t) + q^n(t)$.

Demand and Consumer Surplus. The benefit of energy to society is represented by the consumer surplus $\tilde{S}(Q, t)$, computed as the integral of demand and scaled by the demand growth rate (hence the dependence on time). In our model, we use the following functional form for the consumer surplus:

$$\tilde{S}(Q, t) = e^{bt} S(Qe^{-bt}), \tag{2.2}$$

where $b > 0$ is the growth rate of demand. The consumer surplus is derived from the constant elasticity of substitution (CES) utility.

$$S(Q) = \int p(q)dq\Big|_{q=Q} = \int \frac{S_0}{q^\sigma} dq\Big|_{q=Q} = \begin{cases} S(Q) = S_0 \ln Q, & \text{if } \sigma = 1 \\ \frac{S_0}{1-\sigma} Q^{1-\sigma}, & \text{otherwise,} \end{cases}$$

where $\sigma > 0$ is the demand parameter. The functional form of (2.2) is due to the fact that the growth factor e^{bt} is applied to the direct demand function $q(p)$ rather than the inverse demand $p(q)$.

Production Costs. We assume constant marginal costs. Each unit of energy is produced with the old technology costs c_o. The cost of the new technology is subject to learning by doing; that is, the unit cost $c_n(x(t))$ is a decreasing function of cumulative output, which we define as

$$x(t) = \int_0^t q^n(\tau) d\tau. \tag{2.3}$$

Following the economic literature [445], we let

$$c_n(x) = c_n^0 \left[\frac{x}{\bar{X}} + 1 \right]^{\log_2 \gamma}, \tag{2.4}$$

where the parameters $\bar{X}$ and γ are described in Table 2.1.

Greenhouse Gases Emissions. Producing energy generates greenhouse gases at the unit rate of $b_o > 0$ with the old technology and $b_n \in (0, b_o)$ with the new technology. We are interested in limiting cumulative emissions at the end of the modeling period. Since earlier emissions do more damage (for example, by the irreversible reduction of glaciers), we discount the emissions at the environmental time preference rate a. We use $a \in (0, r)$, but there also exist economic justifications for $a = 0$ or $a > r$. The constraint on cumulative emission is

$$\int_0^T e^{-at}\big(b_o q^o(t) + b_n q^n(t)\big)dt \le z_T. \tag{2.5}$$

Units and Parameters. We make our definitions more precise by setting specific units and parameter values. The quantities ($q(t)$'s, $x(t)$) are in "quads," or billion of millions (10^{15}) of BTUs[1]; monetary amounts (objective, $S(Q)$, etc.) are in billions of dollars; and emissions are measured in billion tons of carbon (tC). See Table 2.1.

Table 2.1: Parameter values common to all models

Parameter	Unit	Notation	Value
Discount rate	-	r	0.05
Demand exponent	-	σ	2.0
Demand scale	\$B	S_0	98,000
Demand growth rate	-	b	0.015
Environmental rate	-	a	0.02
Emissions, old tech.	tC/mBTU	b_o	0.02
Emissions, new tech.	tC/mBTU	b_n	0.001
Unconstrained emissions	BtC	$Z_{\max}$	61.9358
Emission reduction %	-	ζ	0.5
Production cost, old tech.	\$/mBTU	c_o	20
Starting cost, new tech.	\$/mBTU	c_n^0	50
Learning rate	-	γ	0.85
Initial experience	quad	x_0	0
Experience unit size	quad	$\bar{X}$	300

The emission cap is computed as $z_T = (1 - \zeta)Z_{max}$, where Z_{max} equals the cumulative emissions when they are not constrained (which naturally leads to zero utilization of the new technology). The cost parameters γ and $\bar{X}$ were selected to achieve $c_n(x(T)) \approx 30$, a 60% reduction in unit cost by the end of the modeling period (but still more expensive than the old technology with its unit cost of 20). The demand scale is calibrated to match current output level: $q^o(0) + q^n(0) = 60$.

A Basic Model. Our model takes into account only the effect of learning by doing and selects the energy output schedules to maximize the net discounted welfare without exceeding the emission cap. The model is an optimal control problem. Energy output amounts $q^o(t), q^n(t)$ are the controls, and the state variables are

[1] British thermal unit (BTU) is the unit of energy used in power industry. It is equal to 1,055 joules.

the experience level $x(t)$ and the accumulated emissions $z(t)$:

$$\underset{\{q^o,q^n,x,z\}(t)}{\text{maximize}} \quad \int_0^T e^{-rt}\left[\tilde{S}(q^o(t)+q^n(t),t)-c_o q^o(t)-c_n(x(t))q^n(t)\right]dt \tag{2.6a}$$

$$\text{subject to} \quad \dot{x}(t)=q^n(t), \qquad x(0)=x_0=0 \tag{2.6b}$$

$$\dot{z}(t)=e^{-at}\big(b_o q^o(t)+b_n q^n(t)\big), \qquad z(0)=z_0=0 \tag{2.6c}$$

$$z(T)\le z_T \tag{2.6d}$$

$$q^o(t)\ge 0, \qquad q^n(t)\ge 0. \tag{2.6e}$$

The objective (2.6a) is the discounted net welfare, computed as the difference between consumer surplus and production cost. Constraint (2.6b) defines the cumulative output $x(t)$ and is a transformation of (2.3) into a differential equation; this transformation has the advantage that the discretized equations are sparse, allowing us to use the large-scale nonlinear programming (NLP) solvers. Constraints (2.6c) and (2.6d) represent the cumulative emission cap (2.5); (2.6c) is again a differential constraint that defines $z(t)$ to be cumulative emissions at time t, and (2.6d) imposes the cap. Constraint (2.6e) requires that output amounts be nonnegative. We do not need to impose a nonnegativity constraint on $x(t)$ because (2.6b) and (2.6e) guarantee it.

We can derive more complex models that include adjustment costs and capital investment. In all these cases, we can discretize the control problem using on a finite mesh of time points, $t_0 < t_1 < \ldots, < t_p$, and obtain an approximate nonlinear optimization model.

2.3.2 Other Control Applications

A wide range of important DOE applications can be cast as optimal control problems. Application challenges requiring the solution of optimal control problems featured prominently at recent DOE workshops in the role of extreme-scale computing. Examples include the optimal control of the power grid [130, 248, 286, 429] to ensure overall reliability . The power grid is arguably one of the most challenging optimal control applications, consisting of a large network of roughly 100,000 nodes and hundreds of control centers. Optimal control challenges also arise in the transition to renewable energies such as the control of variable-speed wind turbines [313, 330, 364, 409, 449], the design and control of smart grids and distributed generation [303, 335], and the integration of plug-in hybrid-electric vehicles (PHEVs) into the grid [237, 306, 358]. Other network control problems [97] arise in off-ramp metering and route guidance for motorway networks [199, 282] and gas and water networks [106, 126, 170, 338]. Materials science [202] applications are concerned with the design and engineering of materials for extreme conditions and the control of chemical reactions for combustion and catalysis [218]. Specific challenges for optimal control in material science include the design of nanoscale materials by controlled self-assembly [217, 369], the manufacturing of nanoscale tailored materials [302, 317], and the design of nanophotonic devices [226, 452]. Challenges in biology [169] include the control of population dynamics [38, 421]. Scientists are also interested in the control of systems governed by systems of partial differential equations (PDEs). DOE applications include fusion [416], in particular the control of tokamaks [144, 269, 438], and complex flow control and shape optimization for example of aircraft wings and nozzle exits [24, 334, 406]. The control of PDEs is a challenging problem beyond the scope of this proposal, but it could become an extension of our work.

2.4 Design of Complex Systems

The dramatic growth in computing power increasingly enables scientists and engineers to move beyond the simulation of complex systems to consider higher-level problems including optimal design and control

of these systems. This paradigm shift enables a diverse set of new, nonlinear optimization applications. The *operational management* of Department of Energy (DOE) facilities (such as the Advanced Photon Source, Facility for Rare Isotope Beams, and Stanford Linear Accelerator) requires the optimization of reliability and cost-effective operation [415]. Many *environmental engineering* applications, such as the remediation of contaminated sites and carbon sequestration [203], minimize costs subject to subsurface flow constraints [98]. Achieving *energy efficiency targets* benefits from the computational discovery of new nanomaterials for ultra-efficient solar cells [98, 203] and the optimal placement and control of wind turbines [201]. Nonlinearities arise in these problems as a result of the underlying dynamics, multiscale couplings, and/or interactions between design and state variables. The transition from simulation to design requires the development of robust and scalable numerical methods that can meet the challenges posed by these applications.

Many emerging design problems involve discrete design parameters in addition to continuous variables and nonlinear relationships. Such problems are commonly modeled as mixed-integer nonlinear programming problems (MINLPs). MINLP applications within DOE include design of water distribution networks [94, 268]; operational reloading of nuclear reactors [367, 368]; optimal responses to catastrophic events such as the recent Deepwater Horizon oil spill in the Gulf of Mexico [453, 456]; design of nanophotonic devices [323, 331]; electricity transmission expansion [204, 374] and switching [40, 241]; and optimal response to a cyber attack [16].

2.5 Exercises

2.1. Install AMPL and solvers such as Minotaur, IPOPT, and Bonmin.

2.2. Create an AMPL model of problem (2.6) by discretizing the integral and the ODE. For example, you can consider the approximations:

$$\int_{t=a}^{b} f(t)dt \simeq \sum_{i=0}^{p-1} f(t_i)h_i, \quad \text{where} a = t_0 < t_1 < \ldots < t_p = b.$$

where the step-size is $h_i = t_{i+1} - t_i$ (try a uniform step-size first). Similarly, we can approximate the ODE $\dot{x}(t) = f(x(t))$ by an implicit Euler method (other schemes may be better!):

$$x_{i+1} - x_i = h_i f(x_{i+1}),$$

where $x_i \simeq x(t_i)$.

Part II

Unconstrained and Bound-Constrained Optimization

Chapter 3

Methods for Unconstrained Optimization

We start by deriving optimality conditions for unconstrained optimization problems, and then describe the structure of some simple methods that form the basis of more sophisticated techniques in subsequent chapters. Finally, we generalize the techniques to bound-constrained optimization problems.

3.1 Optimality Conditions for Unconstrained Optimization

We start by considering the following (smooth) unconstrained optimization problem,

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x), \tag{3.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable.

We will derive first-and second-order optimality conditions. The first derivative of $f(x)$ with respect to the unknowns x is the vector of partial derivatives of f, or gradient,

$$\nabla f(x) := \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^T,$$

which exists for all $x \in \mathbb{R}^n$ due to our smoothness assumption. The second derivative of $f(x)$ is called the Hessian matrix (neither named after the rugs, nor after the Germanic region), and it is defined as the $n \times n$ symmetric matrix,

$$\nabla^2 f(x) := \left[\frac{\partial^2 f}{\partial x_i \partial x_j} \right]_{i=1,\ldots,n, j=1,\ldots,n}.$$

As an example, consider two special functions, namely an (affine) linear function, $l(x)$, and a quadratic function, $q(x)$, given by:

$$l(x) = a^T x + b, \quad \text{and} \quad q(x) = \frac{1}{2} x^T G x + b^T x + c,$$

respectively. Then it follows that the gradient and Hessian of $l(x)$ and $q(x)$ are given by:

$$\nabla l(x) = a, \; \nabla^2 l(x) = 0, \quad \text{and} \quad \nabla q(x) = Gx + b, \; \nabla^2 q(x) = G, \tag{3.2}$$

respectively.

3.1.1 Lines and Restrictions along Lines

It is useful to consider restrictions of a nonlinear function along a line. A line through the point $x' \in \mathbb{R}^n$ in the direction s is defined as

$$\{x \in \mathbb{R}^n : x = x(\alpha) = x' + \alpha s, \ \forall \alpha \in \mathbb{R}\}$$

where α is the steplength. Given this definition of a line, we can define the restriction of a function $f(x)$ along the line as

$$f(\alpha) := f(x(\alpha)) = f(x' + \alpha s).$$

See Figure 3.1 for an illustration of a nonlinear function, and its restriction along a line. The left image shows an elevation plot and the contours. The line s is illustrated by the red line. The right image shows the restriction of $f(x)$ along the line s.

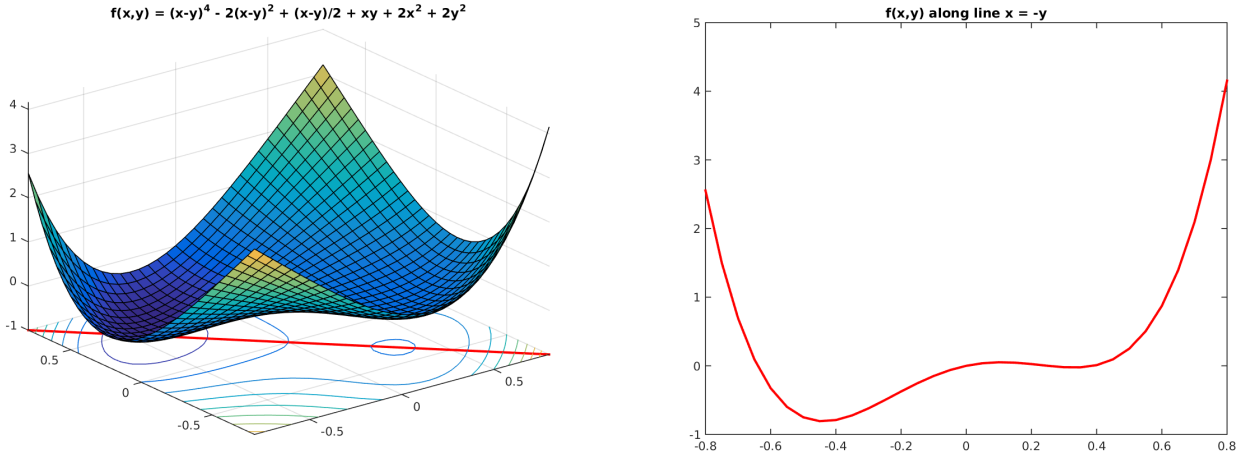


Figure 3.1: Plot and contours of $f(x, y) = (x - y)^4 - 2(x - y)^2 + (x - y)/2 + xy + 2x^2 + 2y^2$, and its restriction along $x = -y$.

Using the restriction of the objective function $f(x)$ along a line, we can derive optimality conditions using traditional calculus. Recall, that the sufficient conditions for a local minimum of a one-dimensional function $f(\alpha)$ are that

$$\frac{df}{d\alpha} = 0, \quad \text{and} \quad \frac{d^2 f}{d\alpha^2} > 0, \tag{3.3}$$

while the first-order necessary condition is just the first equation (which also holds at saddle points and local maximizers). Using the chain rule (for a line $x = x' + \alpha s$), we can derive the operator corresponding to a line gradient:

$$\frac{d}{dx} = \sum_{I=1}^{n} \frac{dx_i}{d\alpha} \frac{\partial}{\partial x_i} = \sum_{I=1}^{n} s_i \frac{\partial}{\partial x_i} = s^T \nabla.$$

Therefore, it follows for $f(\alpha) = f(x' + \alpha s)$ that the slope and curvature of f along s are given by

$$\frac{df}{dx} = s^T \nabla f(x') =: s^T g(x') \quad \text{and} \quad \frac{d^2 f}{dx^2} = \frac{d}{d\alpha} s^T g(x') = s^T \nabla g(x')^T s =: s^T H(x') s. \tag{3.4}$$

We denote the gradient and the Hessian of of $f(x)$ as

$$g(x) := \nabla f(x), \quad \text{and} \quad H(x) := \nabla^2 f(x), \tag{3.5}$$

respectively. Then, it follows that

$$f(x' + \alpha s) = f(x') + \alpha s^T g(x') + \frac{1}{2}\alpha^2 s^T H(x') s + \ldots, \tag{3.6}$$

where we have ignored higher-order terms, that behave like $|\alpha|^3$.

3.1.2 Local and Global Minimizers

In general, a nonlinear function can be unbounded, have one or more local minimizers or maximizers. The following definition formalizes the different notions of minimizers that we will study.

Definition 3.1.1 *Let $x^* \in \mathbb{R}^n$, and let $B(x^*, \epsilon) := \{x : \|x - x^*\| \leq \epsilon\}$ be a ball of radius $\epsilon > 0$ around x^*.*

1. *x^* is called a* global minimizer *of $f(x)$, iff $f(x^*) \leq f(x)$ for all $x \in \mathbb{R}^n$.*
2. *x^* is called a* local minimizer *of $f(x)$, iff $f(x^*) \leq f(x)$ for all $x \in B(x^*, \epsilon)$.*
3. *x^* is called a* strict local minimizer *of $f(x)$, iff $f(x^*) < f(x)$ for all $x \in B(x^*, \epsilon)$ with $x \neq x^*$.*

Clearly, every global minimizer is also a local minimizer. We note, that the minimum may not exist:

- The function $f(x) = x^3$ is unbounded below, and the minimizer does not exist.
- The function $f(x) = \exp(-x)$ is bounded below, but the minimizer still does not exist.

In practice, we can detect these two situations easily (by monitoring the sequence of iterates). On the other hand, finding (and verifying) a global minimizer is much harder, unless the function has some special structure. In the first two parts of this course, we will only consider local minimizers (and count ourselves lucky if we detect a global minimum).

It follows from (3.3) and (3.4) that at a local minimizer, x^*, the slope of $f(x)$ along any line s must be zero, i.e. $s^T g(x^*) = 0$, and that the curvature of $f(x)$ along each line must be nonnegative, i.e. $s^T H(x^*) s \geq 0$. This observation motivates the following theorem.

Theorem 3.1.1 (Necessary Conditions for a Local Minimizer) *Let x^* be a local minimizer. Then it follows that*

$$g(x^*) := \nabla f(x^*) = 0, \quad \textit{and} \quad H(x^*) := \nabla^2 f(x^*) \succeq 0,$$

where $A \succeq 0$ means that the symmetric matrix A is positive semi-definite (e.g. all its eigenvalues are nonnegative).

Proof. See Exercise 3.1.

A sufficient condition for an isolated local minimizer is obtained by strengthening the condition on the Hessian matrix.

Theorem 3.1.2 (Sufficient Conditions for a Local Minimizer) *Assume that the following two conditions hold,*

$$g(x^*) := \nabla f(x^*) = 0, \quad \textit{and} \quad H(x^*) := \nabla^2 f(x^*) \succ 0,$$

then it follows that x^* is an isolated local minimizer of $f(x)$.

Here, $A \succ 0$ means that the symmetric matrix A is positive definite. A matrix is positive definite, if all its eigenvalues are positive, or $A = L^T DL$ factors exist with L a lower triangular matrix with $L_{ii} = 1$ and D a diagonal matrix with positive entries, or the Cholesky factors, $A = L^T L$, exist with $L_{ii} > 0$, or $s^T As > 0$ for all $s \in \mathbb{R}^n, s \neq 0$.

It is clear, that there exists a gap between the necessary and the sufficient conditions. Next we define what we mean by a critical or stationary point.

Definition 3.1.2 *We call x^* a stationary point of $f(x)$, iff $g(x^*) = 0$. This condition is also known as the first-order condition.*

Given a stationary point, we can now classify these points:

Local Minimizer: If $H(x^*) \succ 0$ then x^* is a local minimizer, see Theorem 3.1.2.

Local Maximizer: If $H(x^*) \prec 0$ then x^* is a local maximizer, see Theorem 3.1.2.

Unknown: If $H(x^*) \succeq 0$ then we cannot classify the point.

Saddle Point: If $H(x^*)$ is indefinite (i.e. it has both positive as well as negative eigenvalues), then x^* is a saddle point, see Theorem 3.1.1.

3.2 Iterative Methods for Unconstrained Optimization

It is clear from the simple example (1.2) that we cannot expect to solve optimization problems analytically. In practice, we therefor use iterative methods to solve optimization problems. An iterative method is a solution approach that starts with an initial guess (or iterate) of the solution, $x^{(0)}$, and then creates a sequence of (improving) solution estimates (or iterates), $x^{(k)}$, for $k = 1, 2, \ldots$ that hopefully converge to a solution of the optimization problem. This course is concerned with the study of such iterative methods for a broad range of different classes of optimization problems. In particular, we will study the convergence of methods to stationary points, x^*, i.e. we are interested in methods for which $\|x^{(k)} - x^*\| \to 0$ holds.

We start by giving a simple optimization method for the (smooth) unconstrained optimization problem,

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(x). \tag{3.7}$$

In particular, we define a simple method that (surprisingly) still plays a role in modern optimization, namely steepest descend. This method uses the concept of a search direction, and a line search.

3.2.1 General Structure of Line-Search Methods for Unconstrained Optimization

The main idea behind line-search methods is to find a direction, $s^{(k)}$, such that we can reduce the objective function by moving along this direction. A direction $s^{(k)}$ is a descend direction, iff

$$s^{(k)^T} g(x^{(k)}) < 0, \tag{3.8}$$

which means that the slope of $f(x)$ at $x^{(k)}$ in the direction $s^{(k)}$ is negative. The general format of a line-search method is defined in Algorithm 6.1.

Algorithm 6.1 is a very rough and general outline of a broad class of methods, and may actually fail (see Exercise 3.3).

Remark 3.2.1 *The following remarks apply for the general descend method:*

General Line-Search Method
Given $x^{(0)}$, set $k = 0$.
repeat
  Determine a search direction $s^{(k)}$ such that (3.8) holds.
  Compute a steplength α_k such that $f(x^{(k)} + \alpha_k s^{(k)}) < f(x^{(k)})$.
  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 3.1: General Line-Search Method.

- *The descend condition,* (3.8) *alone is not sufficient to ensure convergence, see Exercise 3.3*
- *The determination of effective search directions has led to a number of line-search methods. We briefly introduce two simple methods in the remainder of this chapter.*
- *Ideally, the step-length should be determined by minimizing the nonlinear function along the direction $s^{(k)}$. However, the exact minimization is typically impractical, and we present an alternative technique below.*
- *The simple descend condition, $f(x^{(k)} + \alpha_k s^{(k)}) < f(x^{(k)})$, does not in itself guarantee convergence to a stationary point.*

3.2.2 Steepest Descend and Armijo Line Search

The steepest descend method takes the direction that (locally) maximizes the descend, namely the negative gradient, i.e. we choose

$$s^{(k)} := -g(x^{(k)}) \quad \text{steepest descend direction.}$$

It follows that this direction satisfies the descend property, (3.8), because

$$s^{(k)^T} g(x^{(k)}) = -g^{(k)^T} g^{(k)} = -\|g^{(k)}\|_2^2 < 0,$$

where we have used the convention that $g^{(k)} := g(x^{(k)})$. In particular, the normalized direction $g^{(k)}/\|g^{(k)}\|$ is the direction that obtains the most negative slope. To prove this fact, we let θ denote the angle between the search direction, s, and the gradient g. then it follows that

$$s^T g = \|s\| \cdot \|g\| \cdot \cos(\theta),$$

and this term is minimized when $\cos(\theta) = -1$, or $\theta = \pi$, i.e. $s = -g$.

Next, we define a simple line-search technique that works well in practice, namely the Armijo line-search. This method does not possess very strong theoretical properties, and, in particular, it cannot be made to be arbitrarily accurate. More accurate line-search techniques are based on interpolation techniques and enforce conditions both on descend and the gradient that allow stronger convergence results to be established.

Algorithm 3.2 is a backtracking line-search, typically starting at $t = 1$. The term $g(x)^T s$ is the *predicted reduction* from a linear model of the objective, and the term $f(x) - f(x + \alpha s)$ is the *actual reduction*. The goal of the comparison in the while-statement is to ensure that the step that is chosen achieves at least a small proportion (a factor $\sigma < 1$) of the predicted reduction. We will see later that this devise ensures convergence to stationary points. Typical values for the constants are $\beta = \frac{1}{2}$ and $\sigma = 0.1$

Armijo Line-Search Method at x in Direction s
$\alpha =$ **function** Armijo$(f(x), x, s)$
Let $t > 0$, $0 < \beta < 1$, and $0 < \sigma < 1$ be fixed constants. Set $\alpha_0 := t$, and $j := 0$.
while $f(x) - f(x + \alpha s) < -\alpha\sigma g(x)^T s$ **do**
  Set $\alpha_{j+1} := \beta\alpha_j$ and $j := j + 1$.
end

Algorithm 3.2: Armijo Line-Search Method.

Steeped Descend Armijo Line-Search Method
Given $x^{(0)}$, set $k = 0$.
repeat
  Compute the steepest descend direction $s^{(k)} := -g(x^{(k)})$.
  Compute the steplength $\alpha_k :=$ Armijo$(f(x), x^{(k)}, s^{(k)})$ using Algorithm 3.2.
  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 3.3: Steeped Descend Armijo Line-Search Method.

With this line-search, we can now define our first practical optimization algorithm, which is the steepest-descend method with an Armijo line-search.

Provided that $f(x)$ is bounded below, we can show that this algorithm converges to a stationary point (if it did not, then we would take an infinite number of steps on which the objective would be reduced). Unfortunately, Algorithm 3.3 can be very inefficient in practice as we will see in the exercises.

3.3 Exercises

3.1. Prove Theorem 3.1.1.

3.2. Consider $f(x) = 2x_1^3 - 3x_1^2 - 6x_1x_2(x_1 - x_2 - 1)$. Plot the function in the domain $[-1, 1]^2$ using Matlab. Find its gradient and Hessian matrix, and hence find and classify all its stationary points.

3.3. Consider the objective function $f(x) = x^2$, which has a global minimum at $x = 0$. Show that the series of iterates, $x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)}$, defined by $s^{(k)} = (-1)^{k+1}$ and $\alpha_k = 2 + 3/2^{k+1}$ starting from $x^{(0)} = 2$ satisfy the two descend properties of Algorithm 6.1. Plot the iterates using Matlab, hence, or otherwise show that the algorithm does not converge to a stationary point.

Chapter 4

Newton and Quasi-Newton Methods

We have seen in the previous chapter that while the steepest descend methods is easy to implement it may not be the most efficient solution approach (though it has made somewhat of a revival over the last decade mainly due to applications in machine-learning and big-data). In this chapter, we consider classes of methods that enjoys faster local convergence properties at the expense of a somewhat higher per-iteration cost. In particular, we study Newton's method and its variants, called quasi-Newton methods.

4.1 Quadratic Models and Newton's Method

The main idea behind Newton methods is the observation, that, at least locally, a quadratic function approximates a nonlinear function well. At the same time, minimizing a quadratic function is rather easy, because the first-order condition of a quadratic reduce to a linear system of equations, see (3.2). Newton's method uses a truncated Taylor series of $f(x)$:

$$f(x^{(k)} + d) = f^{(k)} + g^{(k)^T} d + \frac{1}{2} d^T H^{(k)} d + o(\|d\|^2), \tag{4.1}$$

where we have used the convention that nonlinear functions evaluated at $x^{(k)}$ are identified by its subscript, e.g. $f^{(k)}$ etc, and $o(\|d\|^2) \to 0$ as $\|d\|^2 \to 0$.

Newton's method defines the quadratic approximation around $x^{(k)}$ of $f(x)$ as

$$q^{(k)}(d) := f^{(k)} + g^{(k)^T} d + \frac{1}{2} d^T H^{(k)} d,$$

and takes a step to the minimum of $q^{(k)}(d)$. Provided that $H^{(k)}$ is positive definite, we can find the minimizer of $q^{(k)}(d)$ by solving the following linear system:

$$\min_d \; q^{(k)}(d) \quad \Leftrightarrow \quad \nabla q^{(k)}(d) = 0 \quad \Leftrightarrow \quad \nabla H^{(k)} d = -g^{(k)}.$$

Newton's method then sets $x^{(k+1)} := x^{(k)} + d$ (or possibly performs a line search along d). We formally define this algorithm in Algorithm 4.1, but note that we will have to be careful, if the Hessian matrix, $H^{(k)}$, is not positive definite (we come back to this point later).

The following lemma shows that this method is a descend method as long as $H^{(k)}$ is positive definite.

Lemma 4.1.1 *Assume that $H^{(k)}$ is positive definite. Then it follows that the Newton direction obtained by solving the linear system, $H^{(k)} s^{(k)} := -g(x^{(k)})$, is a descend direction.*

Simple Newton Line-Search Method
Given $x^{(0)}$, set $k = 0$.
repeat
  Compute the Newton direction by solving $H^{(k)}s^{(k)} := -g(x^{(k)})$.
  Compute the steplength $\alpha_k :=$ Armijo$(f(x), x^{(k)}, s^{(k)})$ using Algorithm 3.2.
  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 4.1: Simple Newton Line-Search Method.

Proof. We drop superscripts (k) for simplicity. Because H is positive definite, its inverse exists and is also positive definite. Thus, it follows that $g^T s = g^T H^{-1}(-g) < 0$, and s is a descend direction. □

It can be shown that Newton's method exhibits second-order convergence near a local minimum, which is in contrast to the steepest-descend method which converges only linearly.

Theorem 4.1.1 *Assume that $f(x)$ is twice continuously differentiable and that its Hessian $H(x)$ satisfies a Lipschitz condition, $\|H(x) - H(y)\| \leq L\|x - y\|$ in a neighborhood of a local minimum with Lipschitz constant $L > 0$. If $x^{(k)}$ is sufficiently close to the minimum, x^*, for some k, and if H^* is positive definite, then Newton's method is well defined, and converges with $\alpha_k = 1$ at second-order rate.*

Proof. See Exercise 4.2. □

Figure 4.1 illustrates Newton's method with unit step-length for $f(x) = x_1^4 + x - 1x_2 + (1 + x_2)^2)$. The first three plots clockwise from top left show the first three iterations of Newton's method. The contours of the function are the dashed lines, and the contours of the quadratic approximation are the solid lines.

We provide a number of remarks regarding Newton's method.

1. The full step of Newton's method may fail to reduce the function, which is the reason for introducing the line search. As an example, consider the objective function in Figure 4.2,

$$\underset{x}{\text{minimize}} \;\; f(x) = x^2 - \frac{1}{4}x^4.$$

   Starting at $x^{(0)} = \sqrt{2/5}$ Newton's method with unit step size creates a sequence of iterates that alternates between $-\sqrt{2/5}$ and $\sqrt{2/5}$.

   Remedy: This behavior is remedied by using a line search in Algorithm 4.1.

2. The Hessian matrix, $H^{(k)}$, may not be positive definite (or even singular). In this case the Newton direction may not be defined. Even with a line-search, Newton's method may fail, as the following example due to Powell shows:

$$\underset{x}{\text{minimize}} \;\; f(x) = x_1^4 + x_1x_2 + (1 + x_2)^2.$$

   Starting at $x = 0$, we compute the gradient, Hessian, and Newton step as:

$$x^{(0)} = 0, \quad g^{(0)} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad H^{(0)} = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}, \quad \Rightarrow s^{(0)} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}$$

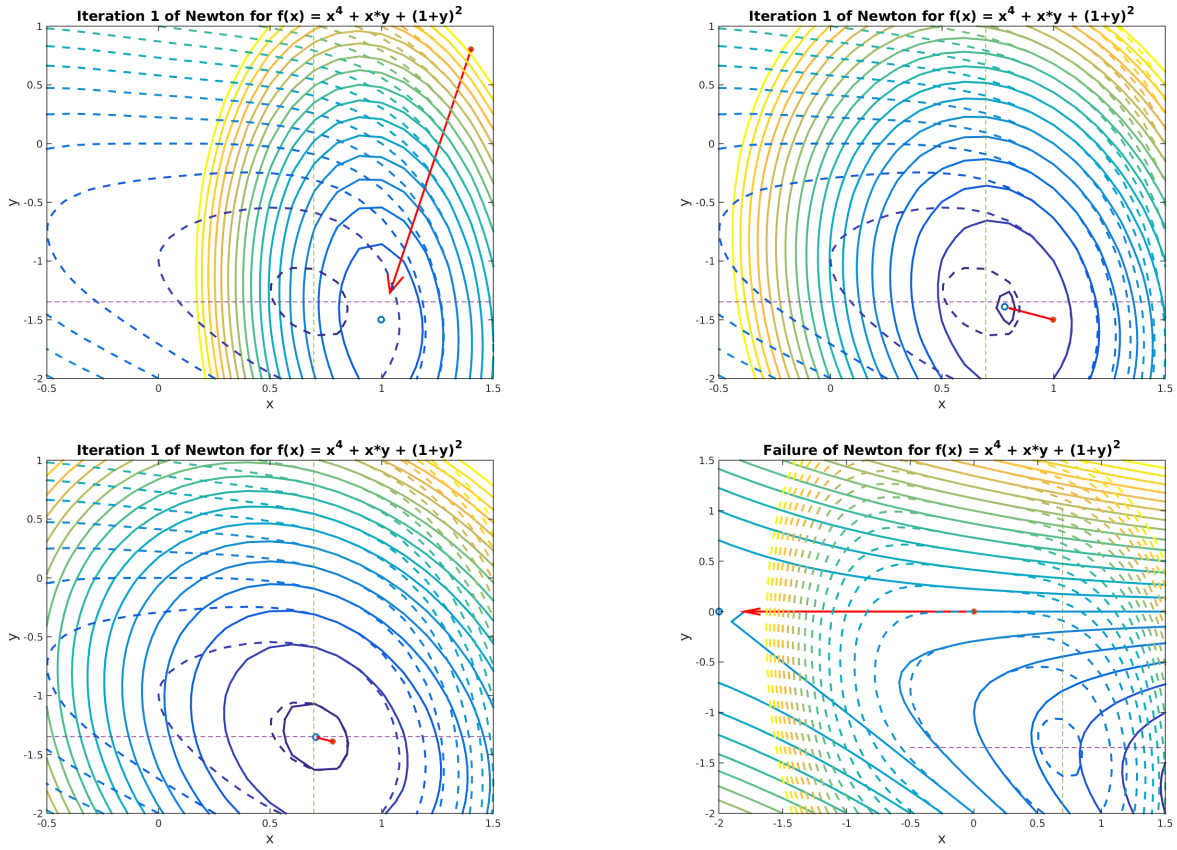


Figure 4.1: Example of Newton's method: the first three plots clockwise from top left show the first three iterations, and the fourth plot shows how Newton's method can fail.

A search along the first component, x_1, cannot decrease the function, and hence, $\alpha_0 = 0$, and Newton's method stalls, even though the steepest descend method would reduce f. This behavior is illustrated in Figure 4.1.

Remedy: We can modify the Hessian matrix to ensure that it is positive definite, see below.

3. Newton's method requires the solution of a linear system of equations at every iteration. This can become computationally too expensive, for problems with millions of unknowns.

   Remedy: We can apply iterative solvers such as the conjugate-gradient method to solve the linear systems. We discuss this method in Section 6.2.

4. Newton's method requires both first and second derivatives. They can be computed using finite differences, which is computationally expensive and error-prone. Derivatives can also be obtained using automatic differentiation: We can compute the full gradient of a function of an arbitrary number of variables at a computational cost that is proportional to evaluating the function, making gradient computations feasible. Unfortunately, the same is not true for the Hessian, but it does hold for Hessian-vector products, such as $H^{(k)}v$.

   Remedy: Make sure you have efficient gradients, or consider iterative solvers discussed in Section 6.2.

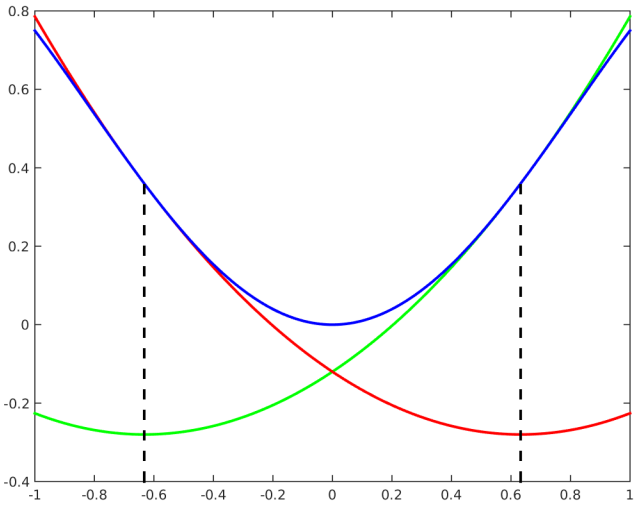


Figure 4.2: Example that shows that Newton's method may fail with a a unit step, even for strictly convex problems.

4.1.1 Modifying the Hessian to Ensure Descend

We saw that one way that Newton's method can fail is if the Hessian matrix, $H^{(k)}$, is not positive definite at an iterate. In this case, the Newton direction may fail to exist, or point up hill, in which case any line-search would fail. If the Hessian $H^{(k)}$ is indefinite, then we can modify it to make it positive definite and obtain a descend direction by virtue of Lemma 4.1.1.

One way to modify the Hessian is to estimate its smallest eigenvalue, $\lambda_{\min}(H^{(k)})$, and then define a modification,

$$M_k := \max\left(0, \epsilon - \lambda_{\min}(H^{(k)})\right) I, \tag{4.2}$$

where $\epsilon > 0$ is a small constant, and $I \in \mathbb{R}^{n\times n}$ is the identity matrix. We then use the modified Hessian, $H^{(k)} + M_k$, which is positive definite, in the modified Newton method, see Algorithm 4.2. Note, that if $\lambda_{\min}(H^{(k)}) > \epsilon$, then the modification is set to zero, because the Hessian is already positive definite.

Modified Newton Line-Search Method
Given $x^{(0)}$, set $k = 0$.
repeat
  Form a modification, M_k, using (4.2) or (4.3).
  Compute the modified Newton direction by solving $\left(H^{(k)} + M_k\right) s^{(k)} := -g(x^{(k)})$.
  Compute the steplength $\alpha_k := \text{Armijo}(f(x), x^{(k)}, s^{(k)})$ using Algorithm 3.2.
  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 4.2: Modified Newton Line-Search Method.

The Hessian modification (4.2) can be interpreted as biasing the step towards the steepest descend di-

rection. Letting $\mu = \lambda_{\min}(H^{(k)})^{-1}$, the modified Newton system becomes

$$\lambda_{\min}(H^{(k)}) \left(\mu H^{(k)} + I\right) s^{(k)} := -g(x^{(k)}),$$

ad if we let $\mu \to 0$, then we recover the steepest-descend direction. We will see below, that this modification is also related to a trust-region.

An alternative modification to the Hessian can be obtained by computing so-called modified Cholesky factors of $H^{(k)}$, such that

$$H^{(k)} + M_k = L_k L_k^T, \tag{4.3}$$

where L_k is a lower triangular matrix with positive diagonal entries (so, $L_k L_k^T$ is positive definite). The modification M_k is zero, if $H^{(k)}$ is sufficiently positive definite, and "not unreasonably large" if $H^{(k)}$ is not positive definite. There exists a Matlab implementation due to Nick Higham that computes related factors, $L_k D_k L_k^T$, with L_k lower triangular with ones along the diagonal. In practice, this modification is performed during the solve of the Newton system.

4.2 Quasi-Newton Methods

In this section, we present an alternative approach that avoids some of the pitfalls of Newton's method, such as:

1. The failure of Newton's method, if $H^{(k)}$ is not positive definite; and
2. the requirement to provide second derivatives; and
3. the need to solve a linear system at every iteration.

Quasi-Newton methods and their modern computational cousins, the limited-memory quasi-Newton methods, overcome these difficulties, whilst almost maintaining the fast convergence of Newton's method.

Quasi-Newton methods work like Newton's method, except that instead of computing the Hessian, and solving a linear system with $H^{(k)}$, we update an approximation of the inverse Hessian, $B^{(k)} \simeq H^{(k)^{-1}}$. The approximation is computed and updated using first-order information at every iteration. Given such an approximation, $B^{(k)}$, the Newton step is computed as

$$s^{(k)} = -B^{(k)} g^{(k)},$$

and the search direction is a descend direction as long as we keep $B^{(k)}$ positive definite.

We typically choose the initial approximation, $B^{(0)} = \nu I$, as a multiple of the identity matrix. Defining

$$\begin{array}{rcll} \gamma^{(k)} & := & g^{(k+1)} - g^{(k)} & \text{gradient difference} \\ \delta^{(k)} & := & x^{(k+1)} - x^{(k)} & \text{iterate difference,} \end{array}$$

then for a quadratic function, $q(x) := q_0 + g^T x + \frac{1}{2} x^T H x$, it follows that

$$\gamma^{(k)} = H\delta^{(k)} \;\Leftrightarrow\; \delta^{(k)} = H^{-1}\gamma^{(k)}, \tag{4.4}$$

see Exercise 4.4.

Because $B^{(k)} \simeq H^{(k)^{-1}}$, we would ideally like that the update satisfies, $B^{(k)}\gamma^{(k)} = \delta^{(k)}$. Unfortunately, we require $B^{(k)}$ in order to compute $x^{(k+1)}$, and hence $\delta^{(k)}$. So instead, we ask that the update satisfies the following condition:

$$B^{(k+1)}\gamma^{(k)} = \delta^{(k)} \qquad \text{quasi-Newton condition.} \tag{4.5}$$

There exist various ways to achieve this condition, and many update formulas have been derived over the years. Here, we consider only two: the rank-one update, and the so-called BFGS update.

4.2.1 The Rank-One Quasi-Newton Update.

We can express a symmetric rank-one matrix as the outer product,

$$uu^T = [u_1 u, \ldots, u_n u],$$

for some vector, $u \in \mathbb{R}^n$. Note, that this matrix is clearly symmetric. Now we set

$$B^{(k+1)} = B^{(k)} + auu^T,$$

and then choose the scalar a and the vector u such that the update, $B^{(k+1)}$, satisfies the quasi-Newton condition, (4.5),

$$\delta^{(k)} = B^{(k+1)}\gamma^{(k)} = B^{(k)}\gamma^{(k)} + auu^T\gamma^{(k)}.$$

This last condition implies that

$$u = \left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right) / \left(au^T\gamma^{(k)}\right),$$

provided that $au^T\gamma^{(k)} \neq 0$. Thus, u must be a multiple of $\delta^{(k)} - B^{(k)}\gamma^{(k)}$. We choose,

$$u = \delta^{(k)} - B^{(k)}\gamma^{(k)} \quad \text{and set} \quad a = \frac{1}{u^T\gamma^{(k)}} = \frac{1}{\left(\delta^{(k)} - B^{(k)}\gamma^{(k)}\right)^T\gamma^{(k)}}.$$

Omitting the superscript $^{(k)}$ we obtain the rank-one update:

$$B^{(k+1)} = B + \frac{(\delta - B\gamma)(\delta - B\gamma)^T}{(\delta - B\gamma)^T\gamma}. \tag{4.6}$$

The next theorem shows that if the directions, $\delta^{(k)}$ are linearly independent, then the rank-one update terminates after $n+1$ iterations for a quadratic.

Theorem 4.2.1 (Quadratic Termination of the Rank-One Formula) *If the rank-one formula, (4.6), is well defined, and if $\delta^{(1)}, \ldots, \delta^{(n)}$ are linearly independent, then the rank-one method terminates in at most $n+1$ steps with $B^{(n+1)} = H^{-1}$ for a quadratic function with positive definite Hessian.*

Proof. See Exercise 4.5. □

We note, that this theorem makes no assumptions about any line search. The proof exploits the fact that for a quadratic, we have that $\gamma^{(k)} = H\delta^{(k)}$.

Remark 4.2.1 (Disadvantages of Rank-One Formula) *The rank-one formula has a number of disadvantages:*

1. *The rank-one formula does not always maintain positive definiteness of $B^{(k)}$, and hence its steps may not be descend direction.*

2. *The rank-one method break down, if the denominator becomes zero or small.*

4.2.2 The BFGS Quasi-Newton Update.

The BFGS update is a rank-two update, which is more flexible than the rank-one update, and allows us to avoid some of the pitfalls of the rank-one update, namely the zero denominator, and the possible loss of positive definiteness. The BFGS update is given by

$$B_{\text{BFGS}}^{(k+1)} = B - \left(\frac{\delta\gamma^T B + B\gamma\delta^T}{\delta^T\gamma}\right) + \left(1 + \frac{\gamma^T B\gamma}{\delta^T\gamma}\right)\frac{\delta\delta^T}{\delta^T\gamma}. \tag{4.7}$$

The BFGS update is the method-of-choice in optimization. It works much better in practice than other rank-two methods, especially for low-accuracy line-searches. The next theorem shows that the BFGS update is positive definite as long as $\delta^T\gamma > 0$.

Theorem 4.2.2 *If $\delta^T\gamma > 0$, then the BFGS update preserves positive definiteness.*

Convergence properties of the BFGS method have only recently been established. Of particular interest has been the long-standing questions:

Question 4.2.1 (Convergence of BFGS Method with Wolfe Line Search) *Does the BFGS method converge for nonconvex functions with the Wolfe line-search?*

The Wolfe line search finds a step size α that satisfies:

$$f(x^{(k)} + \alpha_k s^{(k)}) - f^{(k)} \leq \delta\alpha_k {g^{(k)}}^T s^{(k)} \quad \text{and} \quad g(x^{(k)} + \alpha_k {s^{(k)}}^T s^{(k)} \geq \sigma {g^{(k)}}^T s^{(k)}. \tag{4.8}$$

Question 4.2.1 was answered in the negative by Dai in 2013. He constructs a "perfect 4D example" for the BFGS method as follows:

- The steps, $s^{(k)}$, and the gradients, $g^{(k)}$, satisfy the recurrence relation

$$s^{(k)} = \begin{bmatrix} R_1 & 0 \\ 0 & \tau R_2 \end{bmatrix} s^{(k-1)} \quad \text{and} \quad g^{(k)} = \begin{bmatrix} \tau R_1 & 0 \\ 0 & R_2 \end{bmatrix} g^{(k-1)},$$

  where τ is a decay parameter, and R_1, R_2 are rotation matrices

$$R_1 = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} R_2 = \begin{bmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{bmatrix}$$

With this set-up, it can be shown that step size $\alpha_k = 1$ can be satisfied with the Wolfe or Armijo line search. The function $f(x)$ is a polynomial of degree 38 that is strongly convex along each search direction. Finally, the iterates converge to a circle around the vertices of a regular octagon that are *not stationary points.*

4.2.3 Limited-Memory Quasi-Newton Methods

One of the disadvantages of quasi-Newton methods is their storage and computational requirement. In particular, quasi-Newton matrices are typically dense (though there exists sparse quasi-Newton updates that can be computed at a significantly higher cost). Thus, quasi-Newton methods require $\mathcal{O}(n^2)$ storage and computations to update the quasi-Newton matrix, and compute the search direction. If n is large, say 100s of millions, then these methods would be prohibitive.

Limited-memory methods are a clever way to re-rite the quasi-Newton update, and compute search directions at a cost of $\mathcal{O}(nm)$, where m is a small number. Typically, m is between 5 and 30, *independent of* n. We start by re-writing the BFGS update, (4.7), as

$$B_{\text{BFGS}}^{(k+1)} = V_k^T B V_k + \rho_k \delta\delta,$$

where

$$\rho_k = \delta^T \gamma, \quad \text{and} \quad V_k = I - \rho_k \gamma \delta^T.$$

We can now recur the update back to an initial quasi-Newton matrix, $B^{(0)} \succ 0$. In particular, the idea is to apply $m \ll n$ quasi-Newton updates at iteration k that correspond to difference pairs, (δ_i, γ_i) for $i = k-m, \ldots, k-1$:

$$\begin{aligned} B^{(k)} \quad = \quad & \left[V_{k-1}^T \cdots V_{k-m}^T\right] B^{(0)} \left[V_{k-1} \cdots V_{k-m}\right] \\ & + \rho_{k-m} \left[V_{k-1}^T \cdots V_{k-m+1}^T\right] B^{(0)} \left[V_{k-1} \cdots V_{k-m+1}\right] \\ & + \ldots \\ & + \rho_{k-1} \delta^{(k-1)} \delta^{(k-1)^T} \end{aligned} \tag{4.9}$$

This expression is equivalent to applying m BFGS updates to $B^{(0)}$. We can derive a recursive procedure that computes the BFGS direction, s:

Limited Memory BFGS Search Direction Computation
Given initial BFGS matrix $B^{(0)}$, and memory size m. Set gradient, $q = \nabla f(x^{(k)})$.
for $i = k-1, \ldots, k-m$ **do**
  Set $\alpha_i = \rho_i \delta^{(i)^T} \gamma^{(i)}$
  Update gradient: $q = q - \alpha_i \gamma^{(i)}$
end
Apply initial quasi-Newton matrix: $r = H^{(0)} q$
for $i = k-1, \ldots, k-m$ **do**
  Set $\beta = \rho_i \gamma^{(i)^T} r$
  Update search direction: $r = r + \delta^{(i)}(\alpha_i - \beta)$
end
Return the quasi-Newton search direction: $s^{(k)} := r \left(= H^{(k)} g^{(k)}\right)$

Algorithm 4.3: Limited Memory BFGS Search Direction Computation.

Provided that $H^{(0)}$ is a diagonal matrix, the cost of this recursion is $\mathcal{O}(4nm)$, which is much less than the $\mathcal{O}(n^2)$ cost of th full BFGS update.

Putting it all together, we can now define our quasi-Newton algorithm in a general form:

General Quasi-Newton Line-Search Method
Given $x^{(0)}$, set $k = 0$.
repeat
  Compute a quasi-Newton direction, $s^{(k)} = H^{(k)} g^{(k)}$, using (4.6), (4.7), or Algorithm 4.3.
  Compute the steplength $\alpha_k :=$ Armijo$(f(x), x^{(k)}, s^{(k)})$ using Algorithm 3.2.
  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k+1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 4.4: General Quasi-Newton Line-Search Method.

4.3 Exercises

4.1. Show that Newton's method oscillates for the example, $\min\ f(x) = x^2 - x^4/4$.

4.2. Prove Theorem 4.1.1.

4.3. Program the modified Newton method with Armijo line-search in Matlab, and run it on a few examples.

4.4. Show that (4.4) holds for a quadratic function.

4.5. Show that the rank-one formula terminates for a quadratic:

- Show by induction that $H^{(k+1)}\gamma^{(j)} = \delta^{(j)}$ for all $j = 1, \ldots, k$.
- Hence conclude that the method terminates after $n + 1$ iterations.

4.6. Program the limited-memory BFGS method in Matlab.

4.7. Apply Newton's method to the nonlinear least-squares problem,

$$\underset{x}{\text{minimize}}\ f(x) = \sum_{i=1}^{m} r_i(x)^2 = r(x)^T r(x) = \|r(x)\|_2^2.$$

What do you observe, if the $r_i(x)$ are linear? Can you propose a strategy for handling the case, where $\nabla^2 r_i(x)$ are bounded, and $r_i(x) \to 0$?

Chapter 5

Conjugate Gradient Methods

We have seen in the previous chapter that Newton-like methods are a powerful tool for solving unconstrained optimization problems. Here, we present a class of problems that is based on conjugate directions. These methods can be shown to be related to limited-memory quasi-Newton methods.

5.1 Conjugate Direction Methods

Conjugate direction methods are also related to quadratic models of the function $f(x)$. We start be defining conjugacy.

Definition 5.1.1 *A set of $m \leq n$ nonzero vectors, $s^{(1)}, \ldots, s^{(m)} \in \mathbb{R}^n$ is* conjugate with respect to the positive definite Hessian *G, iff $s^{(i)^T} G s^{(j)} = 0$ for all $i \neq j$. A* conjugate direction method *is a method that generates conjugate directions when applied to a quadratic function with positive definite Hessian.*

We can interpret conjugacy as orthogonality across the positive definite Hessian, G, and note that for $G = I$, we recover orthogonality. The following theorem shows that (not surprisingly), conjugate directions are linearly independent.

Theorem 5.1.1 (Linear Independence of Conjugate Directions) *A set of m conjugate directions is linearly independent.*

Proof. Let $s^{(1)}, \ldots, s^{(m)} \in \mathbb{R}^n$ be the conjugate directions, and consider

$$\sum_{i=1}^{m} a_i s^{(i)} = 0.$$

To show that the directions $s^{(i)}$ are linearly independent we need to show that $a_i = 0$ is the only solution to this system of equations. Since G is positive definite, it is also nonsingular. Hence, the system is equivalent to

$$G\left(\sum_{i=1}^{m} a_i s^{(i)}\right) = 0.$$

Now, we pre-multiply the system with $s^{(j)}$ and exploit the conjugacy condition:

$$s^{(j)^T} G\left(\sum_{i=1}^{m} a_i s^{(i)}\right) = 0 \quad \Leftrightarrow \quad a_j s^{(j)^T} G s^{(j)} = 0 \quad \Leftrightarrow \quad a_j = 0,$$

because G is positive definite. □

As a consequence of this theorem, we can show that any conjugate direction method terminates after at most n iterations for a quadratic with a positive definite Hessian.

Theorem 5.1.2 (Quadratic Termination of Conjugate Direction Methods) *A conjugate direction method terminates for a quadratic function with positive definite Hessian, G, in at most n exact line-searches, and each iterate, $x^{(k+1)}$ is reached by $k \leq n$ descend steps along the conjugate directions $s^{(1)}, \ldots, s^{(k)} \in \mathbb{R}^n$.*

Proof. We let the quadratic function be defined as

$$q(x) = \frac{1}{2} x^T G x + b^T x.$$

Applying a general Newton-like method with conjugate search direction, $s^{(k)}$, we can write th $k+1$ iterate as

$$x^{(k+1)} = x^{(k)} + \alpha_k s^{(k)} = \ldots = x^{(1)} + \sum_{j=1}^{k} \alpha_j s^{(j)} = x^{(i+1)} + \sum_{j=i+1}^{k} \alpha_j s^{(j)}.$$

Now observe, that the gradient of the quadratic can be written as

$$g^{(k+1)} = G x^{(k+1)} + b = G \left(x^{(i+1)} + \sum_{j=i+1}^{k} \alpha_j s^{(j)} \right) + b = g^{(i+1)} + \sum_{j=i+1}^{k} \alpha_j G s^{(j)}.$$

Premultiplying by $s^{(i)}$ we see that

$$s^{(i)^T} g^{(k+1)} = s^{(i)^T} g^{(i+1)} + \sum_{j=i+1}^{k} \alpha_j s^{(i)^T} G s^{(j)} = 0, \quad \forall i = 1, \ldots, k-1,$$

where the first term is zero because we have used an exact line-search, and the second term is zero due to the conjugacy condition. We also observe due to the exact line-search that

$$s^{(k)^T} g^{(k+1)} \quad \text{and hence} \quad s^{(i)^T} g^{(k+1)} = 0, \ \forall i = 1, \ldots, k.$$

Now, let $k = n$, then it follows that

$$s^{(i)^T} g^{(n+1)} = 0, \ \forall i = 1, \ldots, n,$$

and thus, $g^{(n+1)}$ is orthogonal to n linearly independent vectors in $\mathbb{R}^n$, see Theorem 5.1.1, which implies that $g^{(n+1)} = 0$, and the method terminates in n steps. □

This theorem holds for all conjugate direction methods. The difference between these methods lies in how the search directions are generated *without explicit knowledge of the Hessian matrix, G.* We now state a general conjugate direction method, leaving details for the subsequent sections.

Conjugate Direction Line-Search Method
Given $x^{(0)}$, set $k = 0$.
repeat
  Compute the conjugate direction $s^{(k)}$ using (5.1), or ...
  Compute the steplength $\alpha_k := \text{Armijo}(f(x), x^{(k)}, s^{(k)})$ using Algorithm 3.2.
  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 5.1: Conjugate Direction Line-Search Method.

5.2 Classical Conjugate Gradient Method

The main idea behind the conjugate gradient method is to modify the steepest descend method such that th directions that are generated are conjugate. We start by deriving the conjugate gradient method for quadratic functions, and then show how it can be generalized to other functions.

We start by setting $s^{(0)} = -g^{(0)}$, the steepest descend direction, and we would like to choose $s^{(1)}$ to be the component of $-g^{(1)}$ that is conjugate to $s^{(0)}$. We define $s^{(1)}$ as

$$s^{(1)} = -g^{(1)} + \beta_0 s^{(0)} \qquad \text{the component of } -g^{(1)} \text{ conjugate to } s^{(0)}$$

and we seek a formula for β_0 such that conjugacy holds, i.e.

$$0 = {s^{(0)}}^T G s^{(1)} = {s^{(0)}}^T G \left(-g^{(1)} + \beta_0 s^{(0)}\right).$$

Thus, if we solve for β_0, we get

$$\beta_0 = \frac{{s^{(0)}}^T G g^{(1)}}{{s^{(0)}}^T G s^{(0)}},$$

where the denominator is nonzero, because G is positive definite, and $s^{(0)} \neq 0$. Now recall, that

$$x^{(1)} = x^{(0)} + \alpha_1 s^{(0)} \Leftrightarrow s^{(0)} = \left(x^{(1)} - x^{(0)}\right)/\alpha_1,$$

where $\alpha_1 \neq 0$, because of the steepest descend condition. Thus, using the fact that, $G\delta = \gamma$, the Hessian maps differences in x into differences in the gradient, g, we can write β_0 as

$$\beta_0 = \frac{\left(x^{(1)} - x^{(0)}\right)^T G g^{(1)}}{\left(x^{(1)} - x^{(0)}\right)^T G s^{(0)}} = \frac{\left(g^{(1)} - g^{(0)}\right)^T g^{(1)}}{\left(g^{(1)} - g^{(0)}\right)^T s^{(0)}}$$

The exact line-search assumption implies that $0 = {g^{(1)}}^T s^{(0)} = -{g^{(1)}}^T g^{(0)}$, and it follows that

$$\beta_0 = \frac{{g^{(1)}}^T g^{(1)}}{{g^{(0)}}^T g^{(0)}}.$$

Now, consider a general step, k, and set

$$s^{(k)} = \text{the component of } -g^{(k)} \text{ conjugate to } s^{(0)}, \ldots, s^{(k-1)}.$$

We observe that since the desired conjugacy,

$${s^{(k)}}^T G s^{(j)} = 0, \ \forall j < k \quad \Leftrightarrow \quad {s^{(k)}}^T \gamma^{(j)} = 0, \ \forall j < k,$$

we can use the Gram-Schmidt orthogonalization procedure to express

$$s^{(k)} = -g^{(k)} + \sum_{j=0}^{k-1} \beta_j s^{(j)}.$$

In general, it seems hopeless to derive a short recurrence formula in this general case. However, for a quadratic, it turns out that $\beta_j = 0,\ \forall j < k$, and hence, we arrive at the formula

$$s^{(k)} = g^{(k)} + \beta_{k-1} s^{(k-1)} \quad \text{where} \quad \beta_k = \begin{cases} 0 & \text{if } k = -1 \\ \dfrac{g^{(k)^T} g^{(k)}}{g^{(k-1)^T} g^{(k-1)}} & \text{otherwise} \end{cases} \tag{5.1}$$

This update is known as the Fletcher-Reeves conjugate gradient method.

The next theorem shows that the conjugate gradient method with the Fletcher-Reeves update converges for a quadratic function.

Theorem 5.2.1 (Convergence of Fletcher-Reeves for Convex Quadratics) *The Fletcher-Reeves method, (5.1), with exact line-search terminates at a stationary point, $x^{(m)}$ after $m \leq n$ iterations for a quadratic with positive definite Hessian. Moreover, for $0 \leq i \leq m-1$, we have that:*

1. *The search directions are conjugate: $s^{(i)^T} G s^{(j)} = 0\ \forall i \neq j,\ j = 1, \ldots, i-1$.*
2. *The gradients are orthogonal: $g^{(i)^T} g^{(j)} = 0\ \forall i \neq j,\ j = 1, \ldots, i-1$.*
3. *The descend property holds: $s^{(i)^T} g^{(i)} = -g^{(i)^T} g^{(i)} < 0\ \forall i \neq j$.*

Proof. Let the quadratic be

$$f(x) = \frac{1}{2} x^T G x + b^T x + c \quad \Rightarrow\ \nabla f = Gx + b.$$

For $m = 0$, there is nothing to show. Now, let $m \geq 1$, and show 1. to 3. of Theorem 5.2.1 by induction over i. For $i = 0$, we observe that

$$s^{(0)} = -g^{(0)} \ \Rightarrow\ s^{(0)^T} g^{(0)} = -g^{(0)^T} g^{(0)}.$$

Hence, 3. holds for $i = 0$, and there is nothing to show for 1. and 2.

Now assume that 1.-3. hold for i, and show that they also hold for $i+1$. Because $f(x)$ is a quadratic, it follows that

$$g^{(i+1)} = Gx^{(i+1)} + b = G\left(x^{(i)} + \alpha_i s^{(i)}\right) + b = g^{(i)} + \alpha_i G s^{(i)}$$

Because α_i is determined by an exact line search, it follows that

$$\alpha_i = \frac{-g^{(i)^T} s^{(i)}}{s^{(i)^T} G s^{(i)}} = \frac{g^{(i)^T} g^{(i)}}{s^{(i)^T} G s^{(i)}}, \tag{5.2}$$

where the last equation follows from 3. by induction.

Now, we consider Part 2:

$$g^{(i+1)^T} g^{(j)} = g^{(i)^T} g^{(j)} + \alpha_i s^{(i)^T} G g^{(j)} = g^{(i)^T} g^{(j)} + \alpha_i s^{(i)^T} G\left(-s^{(j)} + \beta_{j-1} s^{(j-1)}\right),$$

where we have used the definition of $s^{(j)} = -g^{(j)} + \beta_{j-1}s^{(j-1)}$, re-arranging it get an expression for $g^{(j)}$. Thus, we obtain

$$g^{(i+1)^T}g^{(j)} = g^{(i)^T}g^{(j)} - \alpha_i s^{(i)^T}Gs^{(j)} + \alpha_i\beta_{j-1}s^{(i)^T}Gs^{(j-1)}.$$

Now observe, that for $i = j$, the sum of the first two expressions is zero by (5.2) and the exact line search, while the last expression is zero by Part 1. and the induction assumption. For $j < i$ it follows that the first expression is zero by Part 2., while both the second and third expression are zero by Part 1. and the induction assumption. Therefore it follows that $g^{(i+1)^T}g^{(j)} = 0$ for $j = 1, \dots, j$ which proves Part 2.

Now consider Part 1. Using $s^{(i+1)} = -g^{(i+1)} + \beta_i s^{(i)}$, it follows that

$$s^{(i+1)^T}Gs^{(j)} = -g^{(i+1)^T}Gs^{(j)} + \beta_i s^{(i)^T}Gs^{(j)} = \alpha_j^{-1}g^{(i+1)^T}\left(g^{(j)} - g^{(j+1)}\right) + \beta_i s^{(i)^T}Gs^{(j)},$$

where we have used the fact that $Gs^{(j)} = \alpha_j^{-1}G\left(x^{(j)} - x^{(j+1)}\right) = \alpha_j^{-1}G\left(g^{(j)} - g^{(j+1)}\right)$. Looking at the last display equation, we observe for $j < i$ that the first component is zero due to Part 2. and that the second component is zero die to Part 1. and the induction assumption. For $j = i$, we re-write this expression as

$$s^{(j+1)^T}Gs^{(j)} = \alpha_j^{-1}g^{(j+1)^T}g^{(j)} - \alpha_j^{-1}g^{(j+1)^T}g^{(j+1)} + \beta_j s^{(j+1)^T}Gs^{(j)}.$$

It follows that the first component is zero by Part 2. Next, we use the exact line-search, (5.2), to replace α_j and observe that the second component can be written as

$$-\alpha_j^{-1}g^{(j+1)^T}g^{(j+1)} + \beta_j s^{(j+1)^T}Gs^{(j)} = -s^{(j+1)^T}Gs^{(j)}\frac{g^{(j+1)^T}g^{(j+1)}}{g^{(j)^T}g^{(j)}} + \beta_j s^{(j+1)^T}Gs^{(j)} = 0,$$

from the formula for β_j. Hence, it follows that $s^{(i+1)^T}Gs^{(j)} = 0$ for all $j = 1, \dots, i$, which proves Part 1. Part 3. follows in a similar fashion.

Finally, the quadratic termination follows from Part 1., and the conjugacy of the directions, $s^{(1)}, \dots, s^{(m)}$. □

Conjugate Gradient Methods for Non-Quadratic Functions. If the function, $f(x)$, is non-quadratic, then we cannot expect to perform an exact line-search. Instead, we will rely on some approximate line-search. Moreover, we can no longer expect that the conjugate gradient method terminates after n steps. One possible remedy is to restart the conjugate gradient method with $s^{(n+1)} = -g^{(n+1)}$ the steepest descend step. An alternative is to restore orthogonality using some form of limited-memory with re-orthogonalization.

There exist numerous other conjugate gradient schemes. The best-known are the Polak-Ribiere and the Dai-Yuan formulas:

$$\beta_k^{PR} = \begin{cases} 0 & \text{if } k = -1 \\ \dfrac{\left(g^{(k+1)} - g^{(k)}\right)^T g^{(k)}}{g^{(k-1)^T}g^{(k-1)}} & \text{otherwise} \end{cases} \quad \text{and} \quad \beta_k^{DY} = \begin{cases} 0 & \text{if } k = -1 \\ \dfrac{g^{(k)^T}g^{(k)}}{s^{(k-1)^T}g^{(k-1)}} & \text{otherwise.} \end{cases} \tag{5.3}$$

In general, the Polak-Ribiere formula is superior to the Fletcher-Reeves formula, and the Dai-Yuan method has superior theoretical properties compared to both methods. In particular, it can be shown that Dai-Yuan satisfies a descend property and enjoys global convergence properties with a weak Wolfe line search, see (4.8). On the other hand, the Fletcher-Reeves method can converge to a non-stationary point, and the Polak-Ribiere method may generate uphill directions.

5.3 The Barzilai-Borwein Method

Recently, there has been renewed interest in a simpler two-step gradient method, known as the Barzilai-Borwein method. The method can be interpreted as satisfying a quasi-Newton condition in the least-squares sense.

Barzilai-Borwein Method
Given $x^{(0)}$, set $k = 0$.
repeat
  Set the step-size α_k using (5.4), (5.5), or (5.6).
  Set $x^{(k+1)} := x^{(k)} - \alpha_k g^{(k)}$ and $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 5.2: Barzilai-Borwein Method.

Surprisingly, the Barzilai-Borwein Algorithm 5.2 does not contain a line-search, and in fact, its success is contingent on a non-monotone behavior, i.e. some iterations will increase the objective function.

Popular formulas for the step size in Algorithm 5.2 are

$$\alpha_k^{BB} = \frac{\delta^{(k-1)}\delta^{(k-1)}}{\delta^{(k-1)}\gamma^{(k-1)}} \tag{5.4}$$

$$\alpha_k^{BBs} = \frac{\delta^{(k-1)}\gamma^{(k-1)}}{\gamma^{(k-1)}\gamma^{(k-1)}} \tag{5.5}$$

$$\alpha_k^{aBB} = \begin{cases} \alpha_k^{BB} & \text{for odd } k \\ \alpha_k^{BBs} & \text{for even } k \end{cases} \tag{5.6}$$

Some method occasionally reset the step length to the steepest-descend length. These methods have been generalized to bound constrained optimization, using a projection operation.

5.4 Exercises

5.1. Program the conjugate direction method in Matlab.

5.2. Implement the Barzilai-Borwein method in Matlab, and compare the different options with the conjugate-gradient method.

Chapter 6

Global Convergence Techniques

We have seen that even good methods can fail to converge to a stationary point. In many cases the failure is caused by steps that are too large, but failure can also occur, if we restrict the steps too much. To enforce convergence, we have two mechanisms that restrict the steps that we take: line-search methods, introduced in Section 3.2.1, and trust-region methods discussed below. Both methods have in common that they derive their convergence proofs from the fact that as the step is restricted, it resembles the steepest descend direction. While line-search methods must assume that the search directions are not too orthogonal to the steepest-descend direction, trust-region methods obtain this result automatically. In this chapter, we review line-search and trust-region methods, and discuss their convergence behavior.

6.1 Line-Search Methods

The general form of a line-search method should be clear by now, given that we have seen it in various guises in the previous two chapters. We state a general line-search framework for the sake of completeness in Algorithm 6.1

General Line-Search Method
Let $\sigma > 0$ be a constant. Given $x^{(0)}$, set $k = 0$.
repeat
  Obtain a search direction $s^{(k)}$ such that ${s^{(k)}}^T g(x^{(k)} < 0$.
  Compute a steplength α_k such that the Wolfe condition (4.8) holds.
  Set $x^{(k+1)} := x^{(k)} + \alpha_k s^{(k)}$ and $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 6.1: General Line-Search Method.

This generic algorithm can be shown to converge in the following sense:

Theorem 6.1.1 (Convergence of Generic Line-Search Method) *Assume that $f(x)$ is continuously differentiable and that the gradient, $g(x) = \nabla f(x)$ is Lipschitz continuous on $\mathbb{R}^n$. Then, one of the following three outcomes apply to the iterates of Algorithm 6.1:*

1. *finite termination: $g^{(k)} = 0$ for some $k > 0$, or*
2. *unbounded iterates: $\lim_{k\to\infty} f^{(k)} = -\infty$, or*

3. *directional convergence:*

$$\lim_{k\to\infty} \min\left(\left|s^{(k)^T} g^{(k)}\right|, \frac{\left|s^{(k)^T} g^{(k)}\right|}{\left\|s^{(k)}\right\|}\right) = 0.$$

The third outcome is somewhat successful. It states, that in the limit there is no descend along $s^{(k)}$.

We can strengthen the descend-step condition to avoid the third (unsuccessful) outcome in Theorem 6.1.1 to the condition, $s^{(k)^T} g(x^{(k)}) < -\sigma\|g(x^{(k)}\|$, which ensures that the search direction has at least a component proportional to σ in the steepest descend direction. This condition also links the step size to the stationarity condition (if $x^{(k)}$ is optimal, then $g^{(k)} = 0$). We have seen that there exist various ways to define a search direction. We note, that there are examples, where line-search methods can fail, such as the example in Figure 4.1.

A corollary of Theorem 6.1.1 is that that the steepest descend method with Armijo line search, see Algorithm 3.3 converges to a stationary point:

Corollary 6.1.1 (Convergence of Steepest Descend Line-Search Method) *Assume that $f(x)$ is continuously differentiable and that the gradient, $g(x) = \nabla f(x)$ is Lipschitz continuous on $\mathbb{R}^n$. Then, one of the following three outcomes apply to the iterates of the steepest-descend Algorithm 3.3:*

1. *finite termination: $g^{(k)} = 0$ for some $k > 0$, or*
2. *unbounded iterates: $\lim_{k\to\infty} f^{(k)} = -\infty$, or*
3. *convergence to a stationary point: $\lim_{k\to\infty} g^{(k)} = 0$,*

6.2 Trust-Region Methods

Trust-region methods are more conservative than line-search methods in the sense that the computation of the search direction is performed inside a trust-region. They can be computationally more expensive per iteration, but also enjoy stronger convergence properties for a wider range of methods.

The motivation for trust-region methods is that the Taylor model around $x^{(k)}$, (4.1), that we use to define Newton's method is only accurate in a neighborhood of $x^{(k)}$. Hence, it would make sense to only minimize this model inside such a neighborhood, in which we "trust our model" sufficiently. In general, it is not clear how to define a suitable neighborhood: it depends on the particular function and iterate; it's shape may vary in different regions and with different functions; and it also depends on how far we are from a solution. Hence, rather than trying to find this "optimal" neighborhood, we use a simple trust-region based on the norm-distance from the current iterate:

$$\|x - x^{(k)}\| \le \Delta_k \qquad \text{trust-region,} \tag{6.1}$$

where we have deliberately left open the definition of the norm. In this chapter, we will use the ℓ_2 norm, but in later chapters, we will see that the ℓ_∞ maximum norm is also effective. The parameter, $\Delta_k > 0$ is the trust-region radius, and will be adapted as we make progress towards the solution.

The basic idea of a trust-region method is to minimize a model of our objective function inside the trust-region, (6.1), and move to a new point if we make progress, or reduce the radius, Δ_k, if we fail to make progress. There are two main models for trust-region methods (though variants are also possible). A linear model is defined as

$$l_k(s) = f^{(k)} + s^T g^{(k)} \qquad \text{linear model} \tag{6.2}$$

and a quadratic model is defined as

$$q_k(s) = f^{(k)} + s^T g^{(k)} + \frac{1}{2} s^T B^{(k)} s \qquad \text{quadratic model} \tag{6.3}$$

where $f^{(k)} = f(x^{(k)}), g^{(k)} = \nabla f(x^{(k)})$, and $B^{(k)}$ is the Hessian of $f(x)$ or an approximation of it. Figure 6.1 shows the contours of a nonlinear function and its linear and quadratic trust-region model. We can see that for large steps, s, the agreement between the model and the function can be quite poor.

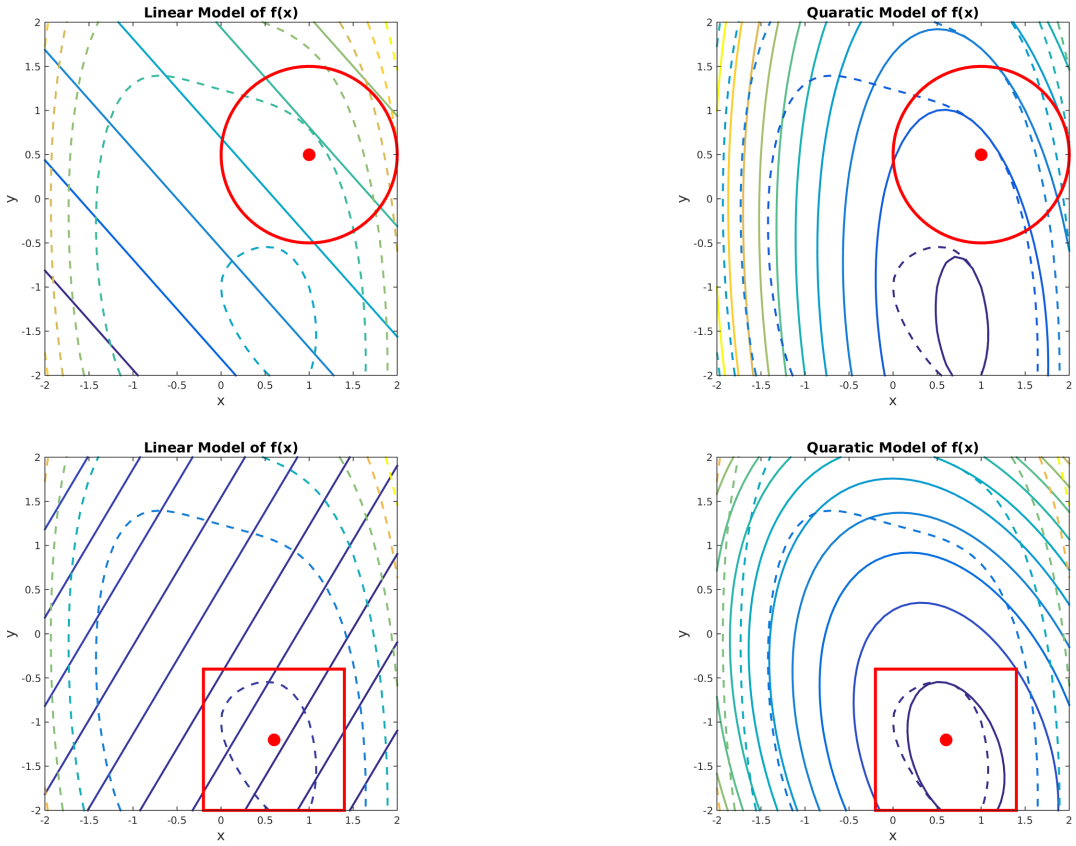


Figure 6.1: Illustration of trust-region and models around two different points. The left column shows linear models with an ℓ_2 (top) and ℓ_∞ trust region (bottom), the right column shows quadratic models. The trust regions are indicated by the red circles/boxes.

Putting it all together, in the quadratic case, we arrive at the *trust-region subproblem*:

$$\textit{(approximately)} \ \underset{s}{\text{minimize}} \ q_k(s) = f^{(k)} + s^T g^{(k)} \frac{1}{2} s^T B^{(k)} s \quad \text{subject to } \|s\|_2 \leq \Delta_k. \tag{6.4}$$

Note, that we have now chosen the trust-region to be the ℓ_2 norm, which is a natural choice for unconstrained optimization. An alternative trust region would be to use an M-norm for a positive definite matrix, M, defined as

$$\|x - x^{(k)}\|_M := \sqrt{\left(x - x^{(k)}\right)^T M \left(x - x^{(k)}\right)} \leq \Delta_k \qquad M\text{-norm trust-region.} \tag{6.5}$$

The advantage of such a norm is that it can be used to mitigate poor scaling between variables. We shall also see, that if M is chosen judiciously (though impractical), then the solution of the trust-region subproblem is

greatly simplified. Thus, we can view M as a preconditioner for the solution of the trust-region subproblem. In this section, we concentrate on the Euclidean norm.

A key concept in trust-region methods concerns the adjustment of the trust-region radius, Δ_k. Typically, the adjustment is based on a measure of agreement between the actual reduction from the step $s^{(k)}$ and the predicted reduction. Formally, we define

$$r_k := \frac{\text{actual reduction}}{\text{predicted reduction}} := \frac{f^{(k)} - f(x^{(k)} + s^{(k)})}{f^{(k)} - q_k(s^{(k)})} \tag{6.6}$$

and observe, that if the model, $q_k(s)$ closely resembles the function $f(x)$, then the ratio, r_k, will be close to one. On the other hand, if $r_k < 0$, then we observed an increase over the step, because $f^{(k)} - q_k(s^{(k)}) > 0$ due to the fact that $s^{(k)}$ solve the trust-region subproblem, (6.4). Thus, we will accept iterates for which the agreement between the function and the model is good, indicated by r_k being sufficiently positive. In this case, we may increase the trust-region radius, to encourage larger steps to reach the solution faster. On the other hand, if $r_k < 0$, then we will reject the step and decrease the trust-region radius, Δ_k, to encourage better agreement on the next iteration. The basic trust-region method can now be defined in Algorithm 6.2.

General Trust-Region Method

Let $0 < \eta_s < \eta_v$ and $0 < \gamma_d < 1 < \gamma_i$ be constants. Given $x^{(0)}$, set $k = 0$, initialize $\Delta_0 > 0$.

repeat

- Approximately solve the trust-region subproblem, (6.4).
- Compute $r_k = \frac{f^{(k)} - f(x^{(k)} + s^{(k)})}{f^{(k)} - q_k(s^{(k)})}$.
- **if** $r_k \geq \eta_v$ *very successful step* **then**
  - Accept the step $x^{(k+1)} := x^{(k)} + s^{(k)}$.
  - Increase the trust-region radius, $\Delta_{k+1} := \gamma_i \Delta_k$.
- **else if** $r_k \geq \eta_v$ *successful step* **then**
  - Accept the step $x^{(k+1)} := x^{(k)} + s^{(k)}$.
  - Keep the trust-region radius unchanged, $\Delta_{k+1} := \Delta_k$.
- **else if** $r_k < \eta_v$ *unsuccessful step* **then**
  - Reject the step $x^{(k+1)} := x^{(k)}$.
  - Decrease the trust-region radius, $\Delta_{k+1} := \gamma_d \Delta_k$.
- **end**
- Set $k = k + 1$.

until $x^{(k)}$ *is (local) optimum*

Algorithm 6.2: General Trust-Region Method.

Reasonable values for the parameters in Algorithm 6.2 are $\eta_v = 0.9$ or 0.99, $\eta_s = 0.1$ or 0.01, and $\gamma_i = 2, \gamma_d = 1/2$. In practice, we do not increase the trust-region radius, unless the step is to the boundary of the trust region.

The trust-region algorithm appears to be very simple, because all the computational difficulty is hidden in the subproblem solve. We next describe a minimalist condition that will allow us to establish convergence, and then outline the convergence proof.

6.2.1 The Cauchy Point

We have seen that the steepest descend method has very powerful theoretical convergence properties. Hence, we "borrow" the main idea behind the steepest descend method to derive minimalist conditions on the trust-

region subproblem solves. In particular, we define the *Cauchy point* as the minimizer of our model in the steepest descend direction. Formally, the Cauchy point, $s_c^{(k)} = -\alpha_C g^{(k)}$ is defined as

$$\begin{aligned} \alpha_c &:= \underset{\alpha}{\text{argmin}}\ q_k(-\alpha g^{(k)}) \text{ subject to } \alpha\|g^{(k)}\| \leq \Delta_k \\ &= \underset{\alpha}{\text{argmin}}\ q_k(-\alpha g^{(k)}) \text{ subject to } 0 \leq \alpha \leq \frac{\Delta_k}{\|g^{(k)}\|}. \end{aligned} \tag{6.7}$$

We note, that the Cauchy point can be easily computed, see Exercise 6.1. Our minimalist assumption on the solution of the trust-region is then that the (approximate) solution $s^{(k)}$ of (6.4) satisfies the following Cauchy decrease condition:

$$q_k(s^{(k)}) \leq q_k(s_c^{(k)}) \quad \text{and} \quad \|s^{(k)}\| \leq \Delta_k. \tag{6.8}$$

We note, that in particular the Cauchy point itself satisfies this condition. Hence, this condition is quite weak, and we typically hope for a better solution.

6.2.2 Outline of Convergence Proof of Trust-Region Methods

We can now outline the convergence proof of our trust-region algorithm. First, we can show that the Cauchy point produces a predicted reduction that is bounded from below by the trust-region radius, and the norm of the gradient:

$$\text{predicted reduction:} \quad f^{(k)} - q_k(s_c^{(k)}) \geq \frac{1}{2}\|g^{(k)}\|_2 \min\left(\frac{\|g^{(k)}\|_2}{1+\|B^{(k)}\|}, \kappa\Delta_k\right).$$

As a corollary of this result and the Cauchy-step condition, (6.8), it follows that our solution of (6.4), $s^{(k)}$, also satisfies this inequality. Next, we can establish a result that shows how well our quadratic model agrees with the objective function using simple Taylor analysis:

$$\left|f(x^{(k)} + s^{(k)}) - m_k(s^{(k)})\right| \leq \kappa\Delta_k^2,$$

for some constant $\kappa > 0$ that depends only on bounds on the Hessian matrix and its approximation. With these two inequalities, we can now establish a crucial result for trust-region methods: namely that it is always possible to find a trust-region radius such that we can make progress from a non-critical point with $g^{(k)} \neq 0$. In particular, as long as

$$\Delta_k \leq \|g^{(k)}\|_2 \kappa(1-\eta_s),$$

then iteration k is very successful and $\Delta_{k+1} \geq \Delta_k$. Here, $\kappa(1-\eta_s)$ is a constant that again depends on the Hessian bounds, and also on the threshold for a successful iterate, η_s. This result is at the heart of trust-region methods, and is very intuitive: as we shrink the trust-region radius, our quadratic model looks more and more like the true nonlinear function, and hence we expect to make progress with $r_k \simeq 1$. Next, it can be shown that if the gradient norm is bounded away from zero, i.e. $\|g^{(k)}\| \geq \epsilon > 0$, then the trust-region radius is also bounded away from zero:

$$\|g^{(k)}\| \geq \epsilon > 0 \ \Rightarrow\ \Delta_k \geq \epsilon\kappa(1-\eta_v).$$

Thus, if there is only a finite number of iterates, then the final iterate must be first-order optimal. Finally, we can combine these observations into the following convergence theorem.

Theorem 6.2.1 (Convergence of Trust-Region Method with Cauchy Condition) *Assume that $f(x)$ is twice continuously differentiable and that the Hessian matrices $B^{(k)}$ and $H^{(k)}$ are bounded. Then, one of the following three outcomes apply to the iterates of the trust-region method Algorithm 6.2 with the Cauchy condition (6.8):*

1. *finite termination:* $g^{(k)} = 0$ *for some* $k > 0$, *or*
2. *unbounded iterates:* $\lim_{k\to\infty} f^{(k)} = -\infty$, *or*
3. *convergence to a stationary point:* $\lim_{k\to\infty} g^{(k)} = 0$,

6.2.3 Solving the Trust-Region Subproblem

We can show that if we use the ℓ_2-norm trust region, then the trust-region subproblem can be solved to global optimality. The next theorem provides the basis for this claim.

Theorem 6.2.2 *Any global minimizer,* s^*, *of the trust-region subproblem,*

$$\underset{s}{\text{minimize}}\ q(s) := f + g^T s + \frac{1}{2} s^T B s \quad \text{subject to } \|s\|_2 \leq \Delta \tag{6.9}$$

satisfies

$$(B + \lambda^* I)s^* = -g,$$

where $B + \lambda^* I$ *is positive definite,* $\lambda^* \geq 0$, *and* $\lambda^*(\|s^*\|_2 - \Delta) = 0$. *Moreover, if* $B + \lambda^* I$ *is positive definite, then* s^* *is unique.*

This is a a remarkable result in the sense that it provides necessary and sufficient conditions for a global minimizer of a potentially nonconvex problem (we will see more on the challenges of nonconvex problems later). These optimality conditions are exactly the KKT conditions of the trust-region subproblem, see Chapter 8. Most importantly, however, these optimality conditions also suggest a way to solve the trust-region subproblem, which we briefly outline next.

We can divide the solution of the conditions in Theorem 6.2.2 into two cases. First, if B is positive definite, and if the solution of the linear system, $Bs = -g$, satisfies $\|s\| \leq \Delta$, then this is the global solution that we seek. This condition can be checked, for example by computing a Cholesky factorization of B. The second case is more involved. If either B is not positive definite, or if $\|s\| > \Delta$, then the conditions of Theorem 6.2.2 say that (s^*, λ^*) must satisfy the following system

$$(B + \lambda I)s = -g \quad \text{and} \quad s^T s = \Delta^2,$$

which is a set of $(n + 1)$ linear/quadratic equations in $(n + 1)$ unknowns. Methods for solving this set of equations essentially reduce to computing Cholesky factors of $B + \lambda I$, and then using these factors to eliminate s from the quadratic equation, which can then be solved for λ. Care has to be taken in certain difficult cases.

6.2.4 Solving Large-Scale Trust-Region Subproblems.

If n is large, then it is not computationally practical to obtain Cholesky factors. Thus, we consider using iterative methods for solving the trust-region subproblem. The conjugate-gradient method is an obvious candidate, because its first search direction is the steepest descend direction, which is consistent with our requirement that we make at least as much progress as the Cauchy step.

Considering the trust-region subproblem, (6.9), we can define the conjugate-gradient method for this problem in Algorithm 6.3, where we have deliberately been vague about the "breakdown" that might occur. This vagueness is because there are two issues that we need to resolve before we can apply the conjugate

Trust-Region Subproblem Conjugate-Gradient Method
Set $s^{(0)} = 0$, $g^{(0)} = g$, $d^{(0)} = -g$, and $i = 0$.
repeat
  Line search: $\alpha_i = \|g^{(i)}\|^2/(d^{(i)^T} B d^{(i)})$
  New iterate: $s^{(i+1)} = s^{(i)} + \alpha_i d^{(i)}$
  Gradient update: $g^{(i+1)} = g^{(i)} + \alpha_i B d^{(i)}$
  Fletcher-Reeves: $\beta_i = \|g^{(i+1)}\|^2/\|g^{(i)}\|^2$
  New search direction: $d^{(i+1)} = -g^{(i+1)} + \beta_i d^{(i)}$
  Set $i = i + 1$.
until *Breakdown or small* $\|g^{(i)}\|$ *found*

Algorithm 6.3: Trust-Region Subproblem Conjugate-Gradient Method.

gradient method: (1) What is the interaction between the iterates and the trust region; and (2) what do we do, if B is indefinite? We will address these issues below.

The first question regarding the conjugate gradient method can be answered with the following theorem, which shows that the approximate solutions that are generated by the conjugate-gradient method increase in norm as the iteration proceeds.

Theorem 6.2.3 *Consider the conjugate-gradient Algorithm 6.3 applied to the trust-region subproblem, (6.9). Assume that* $d^{(i)^T} B d^{(i)} > 0$ *fro all* $0 \leq i \leq k$. *Then it follows that all iterates satisfy:*

$$\|s^{(i)}\|_2 \leq \|s^{(i+1)}\|_2 \quad \forall\, 0 \leq i \leq k.$$

Thus, if there is an iteration, i, on which we observe that $\|s^{(i)}\| > \Delta$, then all subsequent iterates will also lie outside the trust region. This situation then suggests that the optimal solution satisfies $\|s^*\| = \Delta$. Thus, we can now specify what we mean by our termination condition in Algorithm 6.3. We terminate the conjugate-gradient method if:

1. We observe non-positive curvature: $d^{(i)^T} B d^{(i)} \leq 0$, which implies that $q(s)$ is unbounded along $d^{(i)}$.
2. We generate an iterate that lies outside the trust region, which implies that all subsequent iterates lie outside the trust region. If $\|s^{(i+1)}\| > \Delta$, then we can then compute the step to the boundary as the positive root of the quadratic equation

$$\|s^{(i)} + \alpha^B d^{(i)}\|_2^2 = \Delta.$$

This approach works reasonably well in the convex case, but can perform poorly in the nonconvex case. In this case a more elaborate method based on the Lanczos method is generally preferred.

6.3 Exercises

6.1. Give a formula for the computation of the Cauchy step, (6.7), for the quadratic model, $q(d) = f + g^T d + \frac{1}{2} d^T B d$.

6.2. Implementation of preconditioned conjugate-gradient methods in octave/Matlab; example of line-search failure.

Chapter 7

Methods for Bound Constraints

Many practical problems involve variables that must satisfy bounds. Many modern algorithms for more complex optimization problems also use bound-constrained optimization as a subproblem. Hence, it is of interest to study this class of problems more closely. In this chapter, we introduce one class of methods, called gradient-projection methods that have proven to be computationally efficient at solving large-scale instances of bound constrained optimization problems. This chapter also previews some important concepts from constrained optimization that will be considered in more detail in the next part.

7.1 Optimality Conditions for Bound-Constrained Optimization

Here, we consider the following optimization problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x) \quad \text{subject to } l \leq x \leq u, \tag{7.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, and the bounds $l, u \in \mathbb{R}^n$ can be infinite.

We can derive optimality conditions by considering every component of x in turn. If x_i lies strictly between its two bounds, $l_i < x_i < u_i$, then stationarity requires that $\frac{\partial f}{\partial x_i} = 0$. If the lower bound is active, i.e. $x_i = l_i$, then the slope of f in the direction e_i (where e_i is the i^{th} unit vector) should be nonnegative (otherwise we could reduce f by moving away from l_i). Hence, we require that $\frac{\partial f}{\partial x_i} \geq 0$ and $x_i = l_i$. If on the other hand, $x_i = u_i$, then the slope of f pointing in the direction $-e_i$ should be nonnegative, which is equivalent to saying that $\frac{\partial f}{\partial x_i} \leq 0$ and $x_i = u_i$. Putting it all together, we arrive at the following first-order optimality conditions.

Theorem 7.1.1 (Optimality Conditions for Bound-Constrained Optimization) *Consider* (7.5)*, and let* $f(x)$ *be continuously differentiable on an open set that contains the interval* $[l, u]$*. If* x^* *is a local minimizer, then it follows that*

$$\frac{\partial f}{\partial x_i}(x^*) \begin{cases} \geq & 0, \quad \text{if } x_i^* = l_i \\ = & 0, \quad \text{if } l_i < x_i^* < u_i \\ \leq & 0, \quad \text{if } x_i^* = u_i. \end{cases} \tag{7.2}$$

We will see in Chapter 8 that this sign condition is related to the Lagrange multipliers corresponding to the bound constraints. Next, we define the projection operator that projects an arbitrary point, x, into the feasible box, $[l, u]$, namely $P_{[l,u]}(x)$, which we define componentwise as:

$$\big[P_{[l,u]}(x)\big]_i := \begin{cases} l_i, & \text{if } x_i \leq l_i \\ x_i, & \text{if } l_i < x_i^* < u_i \\ u_i, & \text{if } x_i \geq u_i. \end{cases} \tag{7.3}$$

With this projection operator, we can re-state the first-order optimality conditions equivalently as follows.

Corollary 7.1.1 *Consider (7.5), an let $f(x)$ be continuously differentiable on an set that contains the interval $[l, u]$. If x^* is a local minimizer, then it follows that*

$$x^* = P_{[l,u]}\left(x^* - \nabla f(x^*)\right). \tag{7.4}$$

Proof. See Exercise 7.3.

Next, we introduce the concept of an active set that also plays an important role in general constrained optimization.

Definition 7.1.1 *The set of* active constraints *is the set of constraints that hold with equality at a point, $\hat{x}$. Formally, this active set, $\mathcal{A}(\hat{x})$ is defined as*

$$\mathcal{A}(\hat{x}) := \{i : l_i = \hat{x}_i\} \cup \{-i : u_i = \hat{x}_i\},$$

where we have used the convention of identifying lower bounds with a positive index, and upper bounds with their negative index.

The sign convention is not needed, if at most one of each pair of bounds, (l_i, u_i), is finite. The sign convention mimics the sign of the gradient at a stationary point (and also the sign convention for the Lagrange multipliers introduced in Chapter 8). Next, we derive an algorithm that exploits the projection operator and the active-set concept to find a stationary point. We first consider the case where the objective is a quadratic, and then the general case, using the concept of a Cauchy point, see Section 6.2.1, in the preceding chapter.

7.2 Bound-Constrained Quadratic Optimization

In this section, we consider the bound constrained optimization problem, (7.5), when the objective is a quadratic. Our goal is to derive the structure of a simple algorithm. In particular, we consider the problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ q(x) = c + b^T x + \frac{1}{2}x^T G x \quad \text{subject to } l \le x \le u, \tag{7.5}$$

where c is a constant ($c = 0$ wlog), $b \in \mathbb{R}^n$, and $G \in \mathbb{R}^{n\times n}$ is a symmetric matrix. We do not assume that G is positive definite, but instead, we assume that the bounds are finite, $l > -\infty$ and $u < \infty$, to ensure the existence of a stationary point.

The main idea of our algorithm is to alternate between a projected-gradient step and local optimization on a face of the feasible hyper cube. The projected-gradient step follows the steepest descend path to the first minimum of the objective. At this point, we obtain a provisional active set, and then explore the corresponding subspace, or face, of the hyper cube by minimizing the quadratic on this face.

7.2.1 Projected-Gradient Step

We start by describing the projected-gradient step. Given a feasible point, x, and a gradient, $g = Gx + b$, we consider the piecewise linear path parameterized in t:

$$x(t) := P_{[l,u]}\left(x - tg\right), \tag{7.6}$$

which is illustrated in Figure 7.1. Our goal is to find the first minimizers of the objective function along this path, which is equivalent to finding the first minimizer of $q(x(t))$. We start by showing that the piecewise

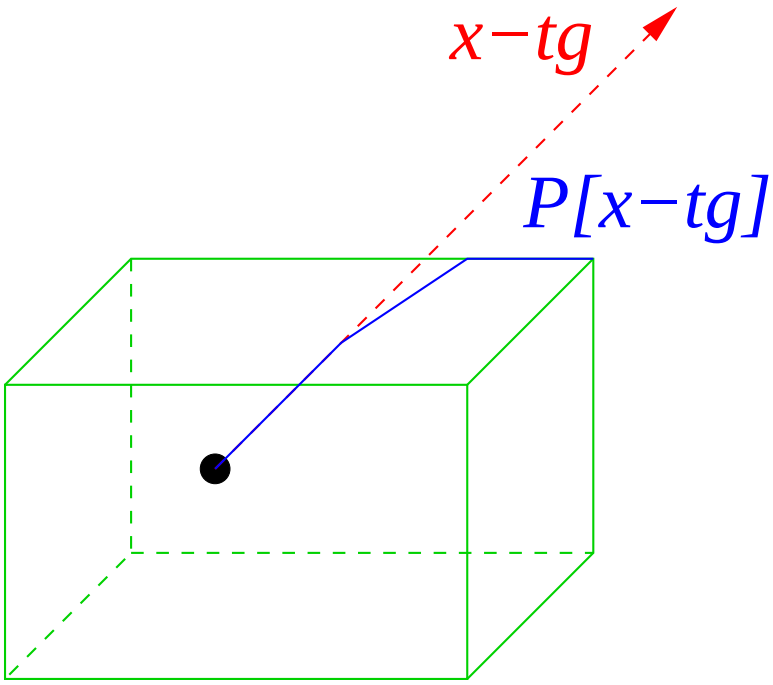


Figure 7.1: Projected gradient path.

linear path, $x(t)$, has a simple analytic description, and then show how to find the first minimizer along this path.

We first find the values of t for which each component reaches a bound in the steepest-descend direction $-g$. These values, $\hat{t}_i$, are given by:

$$\hat{t}_i = \begin{cases} (x_i - u_i)/g_i & \text{if } g_i < 0, \text{ and } u_i < \infty \\ (x_i - l_i)/g_i & \text{if } g_i > 0, \text{ and } l_i > -\infty \\ \infty & \text{otherwise.} \end{cases} \tag{7.7}$$

We note, that if $g_i = 0$, then the corresponding component of x_i does not change in the direction g, and hence $\hat{t}_i = \infty$. To fully describe the path $x(t)$, we must give a description of the components of the path $x(t)$, and identify the breakpoints along the path. The components of $x(t)$ are:

$$x_i(t) = \begin{cases} x_i - tg_i & \text{if } t \le \hat{t}_i \\ x_i - \hat{t}_i g_i & \text{if } t \ge \hat{t}_i, \end{cases}$$

i.e. once a component hits its bound at $\hat{t}_i$ it does not change anymore. The breakpoints of $x(t)$ can be identified by ordering the $\hat{t}_i$ in increasing size, and removing duplicate and zero values. This gives rise to a new sequence, $0 < t_1 < t_2 < t_3 \ldots$ Each interval, $[0, t_1], [t_1, t_2], [t_2, t_3], \ldots$ corresponds to a segment along the path with segments ordered in distance from the initial point, x.

We now find an expression of the j^{th} segment, $[t_{j-1}, t_j]$, and the quadratic along that segment. In $[t_{j-1}, t_j]$, we can write

$$x(t) = x(t_{j-1}) + \delta s^{(j-1)},$$

where the stepsize δ and the direction $s^{(j-1)}$ are defined as

$$\delta = t - t_{j-1}, \quad \delta \in [0, t_j - t_{j-1}], \qquad p_i^{(j-1)} = \begin{cases} -g_i & \text{if } t_{j-1} \le \hat{t}_i \\ 0 & \text{otherwise.} \end{cases}$$

We can now obtain an explicit expression for the quadratic in the segment $[t_{j-1}, t_j]$, and use this expression to find its minimum in the segment. For $t \in [t_{j-1}, t_j]$, we have that

$$q(x(t)) = c + b^T \left(x(t_{j-1}) + \delta s^{j-1}\right) + \frac{1}{2}\left(x(t_{j-1}) + \delta s^{j-1}\right)^T G \left(x(t_{j-1}) + \delta s^{j-1}\right),$$

which can be written as

$$q(\delta) = q(x(t)) = f_{j-1} + f'_{j-1}\delta + \frac{1}{2}\delta^2 f''_{j-1}, \quad \text{for } \delta \in [0, t_j - t_{j-1}],$$

with coefficients given by

$$\begin{aligned} f_{j-1} &= c + b^T x(t_{j-1}) + \frac{1}{2} x(t_{j-1})^T G x(t_{j-1}) \\ f'_{j-1} &= b^T s^{(j-1)} + x(t_{j-1})^T G s^{(j-1)} \\ f''_{j-1} &= s^{(j-1)^T} G s^{(j-1)}. \end{aligned}$$

To find the minimum of $q(x(t)) = q(\delta)$ in $[0, t_j - t_{j-1}]$, we differentiate, and set the gradient to zero. The minimizer then depends on the signs of f'_{j-1} and f''_{j-1} as described in Table 7.1.

Table 7.1: Details of minimizer of $q(\delta)$ for different sign cases.

	$f'_{j-1} < 0$	$f'_{j-1} = 0$	$f'_{j-1} > 0$
$f''_{j-1} < 0$	$\delta = t_j - t_{j-1}$	$\delta = t_j - t_{j-1}$	$\delta = 0$
$f''_{j-1} = 0$	$\delta = t_j - t_{j-1}$	$\delta = t_j - t_{j-1}$	$\delta = 0$
$f''_{j-1} > 0$	$\delta = \min\left(\frac{-f'_{j-1}}{f''_{j-1}}, t_j - t_{j-1}\right)$	$\delta = 0$	$\delta = 0$

We note from Table 7.1 that the optimal δ is either on the boundary of the interval, $[0, t_j - t_{j-1}]$, or in the interior. The algorithm for finding the first minimizer of $q(x(t))$ thus proceeds as follows. We examine the intervals in order, and stop on the first interval, j, where the optimum, $\delta^* < t_j - t_{j-1}$. In this case, the corresponding $t^* = t_{j-1} + \delta^*$, and the Cauchy point is $x_C = x(t^*)$, see Algorithm 7.1

First Minimizer Along Projected Gradient Path
Given initial point, x, and direction, g.
Compute $\hat{t}_i$ from (7.7), and set $j = 1$.
Obtain $t_0 := 0 < t_1 < t_2 < \ldots$ by ordering $\hat{t}_i$, and removing duplicates and zeros.
repeat
  Compute f'_{j-1}, f''_{j-1}, and find δ^* from Table 7.1.
  if $\delta^* < t_j - t_{j-1}$ **then**
    Set $t^* = t_{j-1} + \delta^*$ found.
  end
  Set $j = j + 1$.
until t^* *found*
Return t^* and $x(t^*)$.

Algorithm 7.1: First Minimizer Along Projected Gradient Path.

7.2.2 Subspace Optimization

After performing a projected-gradient search, we would like to explore the subspace or face corresponding to the current active set. We can do this by fixing the active variables at their current bounds, and using any of the unconstrained optimization techniques of the previous chapters with one modification. While we are minimizing our objective, $q(x)$, in the remaining (inactive) variables, we must make sure that we remain feasible. This can be achieved by a simple modification of the unconstrained algorithm by stopping it when we cross the boundary of our feasible box.

Denoting the active set at the Cauchy point by $\mathcal{A}(x_C)$, we can formally state the subproblem:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & q(x) = \frac{1}{2}x^T G x + b^T x + c \\ \text{subject to} & x_i = l_i, \ \forall \ i \in \mathcal{A}(x_C) \qquad x_i = u_i, \ \forall \ -i \in \mathcal{A}(x_C) \\ & l_i \leq x_i \leq u_i, \ \forall \ \pm i \notin \mathcal{A}(x_C). \end{array} \tag{7.8}$$

We can extend the global convergence analysis of Section 6.2 to show that we do not need to solve this problem to global optimality. As long as we do at least as well as the Cauchy point, we are guaranteed to find a stationary point. An attractive subproblem solver is Algorithm 6.3, which can be modified to ensure that the iterates remain feasible, and by applying a masking operation to the matrix-vector products that blanks out the active variables.

7.2.3 Overall Algorithm for Bound-Constrained Quadratic Optimization

We can now formally state a projected-gradient subspace optimization method for bound-constrained quadratic optimization.

Quadratic Projected-Gradient Subspace Optimization
Given $l \leq x^{(0)} \leq u$, set $k = 0$.
repeat
  Define the path $x^{(k)}(t) := P_{[l,u]}\left(x^{(k)} - tg^{(k)}\right)$.
  Obtain a Cauchy point, $x_C^{(k)}$, by finding the first minimizer of $q(x^{(k)}(t))$
  Define the active set, $\mathcal{A}(x_C^{(k)})$, and set up the subspace optimization problem.
  Approximately solve the subspace problem, (7.8) for $l \leq x^{(k+1)} \leq u$.
  Set $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 7.2: Quadratic Projected-Gradient Subspace Optimization.

The algorithm requires the starting point to be feasible, i.e. $l \leq x^{(0)} \leq u$. If we have an infeasible starting point, then we can simply obtain a new starting point by projecting into the box $[l, u]$. A suitable algorithm for approximately solving the subspace optimization problem, (7.8), is Algorithm 6.3 where we relax the trust-region by setting $\Delta_k = \infty$.

It can be shown that for problems for which strict complementarity holds at the solution x^*, i.e.

$$x_i^* = l_i \ \Rightarrow \ \frac{\partial f}{\partial x_i}(x^*) > 0 \quad \textit{and} \quad x_i^* = u_i \ \Rightarrow \ \frac{\partial f}{\partial x_i}(x^*) < 0$$

we will find the optimal active set, $\mathcal{A}(x^*)$, after a finite number of projected gradient steps, and hence, in this case, the algorithm is finite.

7.3 Bound-Constrained Nonlinear Optimization

How can we generalize Algorithm 7.2 to nonlinear functions, $f(x)$? From our previous chapters, it should be clear that we will apply the usual Cauchy-point search, followed by a subspace optimization of a quadratic model in which we measure progress with respect to the original objective function. We now formally state a possible framework for general bound-constrained optimization, though we note, that this framework leaves some important implementation decision unspecified.

General Projected-Gradient Subspace Optimization
Given $l \leq x^{(0)} \leq u$, set $\Delta_0 = 1$, and $k = 0$.
repeat
  Scale the steepest-descend direction to lie inside the trust region: $\hat{g}^{(k)} = g^{(k)} \frac{\|g^{(k)}\|_2}{\Delta_k}$.
  Define the path $x^{(k)}(t) := P_{[l,u]}\left(x^{(k)} - t\hat{g}^{(k)}\right)$.
  Form the quadratic model, $q_k(s)$, as in (6.3).
  Obtain a Cauchy point, $x_C^{(k)}$, by finding the first minimizer of $q_k(s^{(k)}(t))$
  Define the active set, $\mathcal{A}(x_C^{(k)})$, and set up the subspace optimization problem.
  Approximately minimize $q_k(s)$ over the inactive variables such that $l \leq x^{(k)} + s \leq u$ using Algorithm 6.3.
  Compute $r_k = \frac{f^{(k)} - f(x^{(k)} + s^{(k)})}{f^{(k)} - q_k(s^{(k)})}$.
  if $r_k \geq \eta_v$ *very successful step* **then**
    Accept the step $x^{(k+1)} := x^{(k)} + s^{(k)}$.
    Increase the trust-region radius, $\Delta_{k+1} := \gamma_i \Delta_k$.
  else if $r_k \geq \eta_v$ *successful step* **then**
    Accept the step $x^{(k+1)} := x^{(k)} + s^{(k)}$.
    Keep the trust-region radius unchanged, $\Delta_{k+1} := \Delta_k$.
  else if $r_k < \eta_v$ *unsuccessful step* **then**
    Reject the step $x^{(k+1)} := x^{(k)}$.
    Decrease the trust-region radius, $\Delta_{k+1} := \gamma_d \Delta_k$.
  end
  Set $k = k + 1$.
until $x^{(k)}$ *is (local) optimum*

Algorithm 7.3: General Projected-Gradient Subspace Optimization.

Essentially, we had to add a trust-region to Algorithm 7.3 to account for the fact that our function may not agree with the quadratic model. We are using the ℓ_2-norm trust region, because it fits more naturally with the subspace optimization. We could also have used an ℓ_∞-norm trust region, which make it easier to intersect the trust-region with our feasible box (the result is another, smaller, box).

7.4 Exercises

7.1. Modeling obstacle problems as bound-constrained optimization problems

7.2. implementation of methods based on unconstrained solvers???

7.3. Prove Corollary 7.1.1.

Part III

General Constrained Optimization

Chapter 8

Optimality Conditions

In this chapter, we present both necessary and sufficient conditions for a local minimizer of a general nonlinear optimization problem. The optimality conditions presented here extend the unconstrained and bound-constrained optimality conditions of the previous part to general constraints, and form the basis for the algorithmic developments that follow in subsequent chapters.

8.1 Preliminaries: Definitions and Notation

In this chapter we present optimality conditions for a (local) minimizer of a general nonlinear optimization problem of the form

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & c_i(x) = 0, & i \in \mathcal{E} \\ & l_j \leq c_i(x) \leq u_j & i \in \mathcal{I} \\ & l_i \leq x_i \leq u_i & i = 1, \ldots, n \end{array} \tag{8.1}$$

where we assume that the functions $f(x)$ and $c_i(x)$ are twice continuously differentiable. $\mathcal{E}$ indexes the equality constraints, and $\mathcal{I}$ indexes the inequality constraints. The bounds l_j, u_j, l_i, u_i can be finite or infinite. Often, there exists additional structure, such as linear constraints, $l \leq A^T x \leq u$, or network constraints that can be exploited within a solver. We will also refer to problem (8.1) as a *nonlinear program (NLP)*.

In order to simplify the notation in this part, we will assume that the NLP (8.1) is presented to us in the following format:

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & c_i(x) = 0 & i \in \mathcal{E} \\ & c_i(x) \geq 0 & i \in \mathcal{I}. \end{array} \tag{8.2}$$

We will also use the notation $c_{\mathcal{E}}(x) = 0$ and $c_{\mathcal{I}}(x) \geq 0$ for the equality and inequality constraints respectively. We will see that more general problems can always be formulated in this format (though in practice such reformulations may be inefficient, and both modeling languages and solvers tackle (8.1) directly.

Definition 8.1.1 (Feasible Set and Minimizers) *The feasible set of* (8.2) *is the set*

$$\mathcal{F} := \{x | c_{\mathcal{E}}(x) = 0, \text{ and } c_{\mathcal{I}}(x) \geq 0\}.$$

A point $x^* \in \mathcal{F}$ is called a global minimizer, iff $f(x^*) \leq f(x)$ for all $x \in \mathcal{F}$. A point $x^* \in \mathcal{F}$ is called a local minimizer, iff there exists a neighborhood $\mathcal{N}(x^*)$ of x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{F} \cup \mathcal{N}(x^*)$.

In this part, we will be only concerned with local minimizers.

Notation. We denote the gradient of $f(x)$ by $g(x) = \nabla f(x)$, and the Jacobian of the constraints by $A(x) = \nabla c(x)$.

Limitations and Importance of Optimality Conditions. The optimality conditions we present below have some severe limitations that are almost impossible to overcome. First, they only provide results for local optima, rather than the global solution of the NLP (8.2). Second, they are limited to smooth finite-dimensional problems, (8.1), though they can be extended to certain classes of nonsmooth or infinite-dimensional problems. Optimality conditions are important for three reasons: First, they allow us to guarantee that a candidate solution is indeed a local optimum; these are the sufficient conditions. Second, they indicate when a point is not optimal, which are the necessary conditions. And most importantly, third, they guide the development of optimization methods and their convergence proofs.

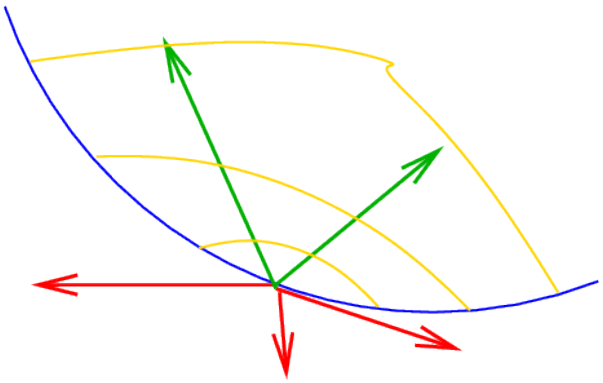


Figure 8.1: Illustration of feasible directions (green) and infeasible directions (red).

8.2 First-Order Conditions

We start by recalling the first-order conditions from unconstrained optimization, see Theorem 3.1.1. In particular, if x^* is an unconstrained local minimizer, then it follows that $g^* = 0$. We can state this condition equivalently as follows:

$$g^* = 0 \quad \Leftrightarrow \quad s^T g^* = 0, \ \forall s \quad \Leftrightarrow \quad \left\{ s \mid s^T g^* < 0 \right\} = \emptyset,$$

where the last condition states that there are no strict descend directions at x^*. We will now derive a similar condition for the constraint problem, (8.2). In particular, we are interested in conditions to classify feasible feasible directions, illustrated by the green direction in Figure 8.1, and a more practical condition to replace the condition that there are no feasible descend directions. Loosely speaking, our goal is to find conditions that show that the set,

$$\left\{ s \mid s^T g^* < 0, \ \forall s \text{ feasible directions} \right\} = \emptyset,$$

i.e. there exist no feasible descend directions. We distinguish two cases initially, depending on whether all constraints are equations or not.

8.2.1 Equality Constrained Nonlinear Programs

We start by considering equality constraints only, that is, we consider

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & c_{\mathcal{E}}(x) = 0. \end{array}$$

Consider an infinitesimal step δ from x^*. A Taylor series expansion shows that

$$c_i(x^* + \delta) = c_i(x^*) + \delta^T a_i^* + o(\|\delta\|) = \delta^T a_i^* + o(\|\delta\|),$$

because $c_i(x^*) = 0$, where $a_i^* = \nabla c_i(x^*)$, and $a = o(h)$ means that $\frac{a}{h} \to 0$ as $h \to 0$. So, in order for $x^* + \delta$ to be feasible, we need that $\delta^T a_i^* + o(\|\delta\|) = 0$, which implies that δ lies in a direction s given as:

$$\text{feasible directions} \qquad s^T a_i^* = 0,$$

i.e. s is orthogonal to the constraint normal, a_i^*. In order to derive stationarity conditions, we need a regularity assumption that ensures that the "linearized feasible set", obtained from the directions, s, above locally resembles the nonlinear feasible set. One such regularity assumption is linear independence.

Assumption 8.2.1 (Linear Independence of Constraint Normals) *Assume that the constraint normals, $a_i^* = \nabla c_i(x^*)$, for $i = 1, \ldots, m_e$, are linearly independent.*

The necessary condition that we are looking for is that, provided that under Assumption 8.2.1 it holds that

$$x^* \text{ is a local minimizer } \Rightarrow \ \left\{ s \mid s^T g^* < 0, \ s^T a_i^* = 0, \ \forall i \in \mathcal{E} \right\} = \emptyset$$

Unfortunately, this condition is rather difficult to check. The following lemma provides a practical way to check this condition.

Lemma 8.2.1 *Assume that Assumption 8.2.1 holds, and that x^* is a local minimizer, then the following two conditions are equivalent:*

1. $\left\{ s \mid s^T g^* < 0, \ s^T a_i^* = 0, \ \forall i \in \mathcal{E} \right\} = \emptyset$

2. *There exist so-called Lagrange multipliers, y_i^*, for $i \in \mathcal{E}$ such that*

$$g^* = \sum_{i \in \mathcal{E}} y_i^* a_i^* = A^* y.$$

Graphically, Lemma 8.2.1 means that th objective gradient, g^*, can be expressed as a linear combination of the constraint gradients, a_i^*. We illustrate this interpretation in Figure 8.2.

Under Assumption 8.2.1, it follows that rank$(A^*) = m_e$, i.e. A^* has full rank, and the generalized inverse, A^+, is well-defined. Hence, we can uniquely determine y^* as

$$y^* = A^{*^+} g^*, \quad \text{where } A^{*^+} = \left(A^{*^T} A^*\right)^{-1} A^{*^T},$$

which is also the unique solution of the least-squares problem, $\min \ \|A^* y - g^*\|_2^2$.

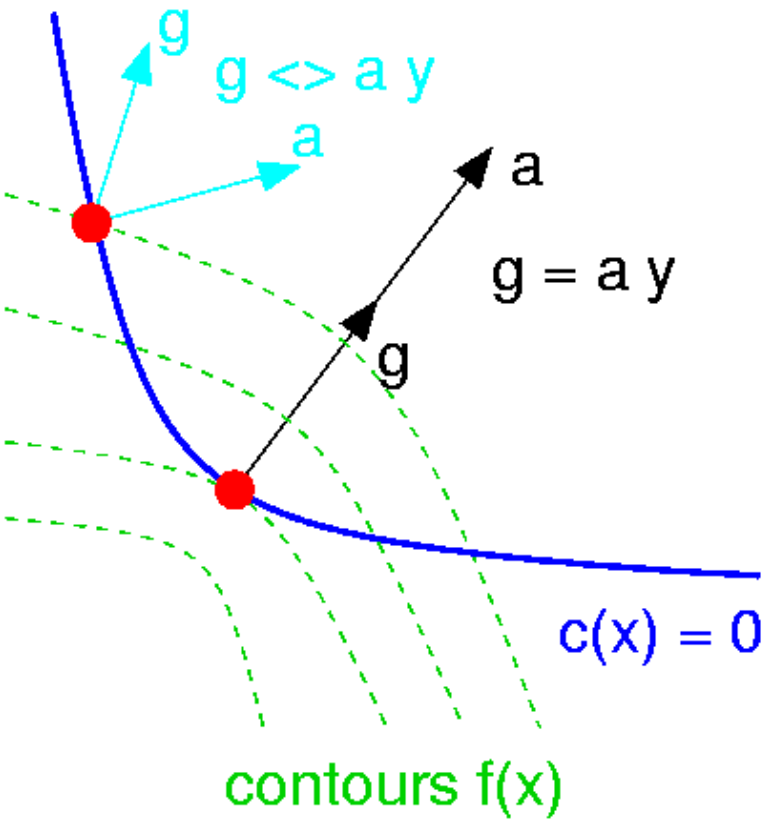


Figure 8.2: Illustration of optimality conditions. At a stationary point, we can express the gradient of the objective as a linear combination of the gradients of the constraints.

Method of Lagrange Multipliers. We can restate the optimality conditions in Lemma 8.2.1 as a system of nonlinear equations in (x, y):

$$\begin{array}{rcll} g(x) = & A(x)y & \text{necessary condition} \\ c(x) = & 0 & \text{feasibility.} \end{array} \tag{8.3}$$

Defining the Lagrangian function, $\mathcal{L}(x, y) := f(x) - y^T c(x)$, we can equivalently state the conditions (8.3) as

$$\nabla_x \mathcal{L}(x, y) = 0, \quad \text{and} \quad \nabla_y \mathcal{L}(x, y) = 0. \tag{8.4}$$

Hence, finding a stationary point of the Lagrangian is equivalent to finding a stationary point of the equality constrained NLP.

An interesting consequence of the optimality conditions in Lemma 8.2.1 is that we can express the effect of a perturbation in the constraints, $c_i(x) = \epsilon$ on the optimal objective function value. We let $x(\epsilon)$ and $y(\epsilon)$ denote the optimal values of (x, y) after this perturbation and consider the perturbed Lagrangian,

$$f(x(\epsilon)) = \mathcal{L}(x(\epsilon), y(\epsilon)) = f(x(\epsilon)) + y(\epsilon)^T \left(c(x) - \epsilon\right)$$

The chain rule implies that

$$\frac{df}{d\epsilon_i} = \frac{d\mathcal{L}}{d\epsilon_i} = \frac{\partial x^T}{\partial \epsilon_i} \nabla_x \mathcal{L} + \frac{\partial y^T}{\partial \epsilon_i} \nabla_y \mathcal{L} + \frac{\mathcal{L}}{\partial \epsilon_i}$$

Now observe, that we have $\nabla_x \mathcal{L}(x, y) = 0$ and $\nabla_y \mathcal{L}(x, y) = 0$, and hence

$$\frac{\mathcal{L}}{\partial \epsilon_i} = y_i \;\Rightarrow\; \frac{df}{d\epsilon_i} = y_i.$$

So, the multiplier, y_i, gives the rate of change in the objective if the constraints are perturbed. This observation is important for sensitivity analysis, which aims to quantify how solutions of optimization problems change upon changes to the problem definition.

8.2.2 Inequality Constrained Nonlinear Programs

Next, we consider problems that also include inequality constraints, i.e. (8.2). We note, that we only need to consider the active constraints, denoted by

$$\mathcal{A}^* := \mathcal{A}(x^*) := \{i \in \mathcal{E} \cup \mathcal{I} \mid c_i(x^*) = 0\} \qquad \text{active set.} \tag{8.5}$$

We observe that the active set automatically includes all equality constraints. Again, we are looking for feasible directions, and we let δ be a small incremental step for some active inequality, $i \in \mathcal{I} \cap \mathcal{A}^*$. As before, we get

$$c_i(x^* + \delta) = c_i(x^*) + \delta^T a_i^* + o(\|\delta\|) = \delta^T a_i^* + o(\|\delta\|).$$

However, now we require that the step remains feasible only with respect to one side of the constraint, and we get

$$c_i(x^* + \delta) \geq 0 \;\Leftrightarrow\; \delta^T a_i^* + o(\|\delta\|)$$

and hence, we require that δ lies in a direction s given as:

$$\text{feasible directions} \qquad s^T a_i^* \geq 0, \;\forall i \in \mathcal{I} \cap \mathcal{A}^*, \quad s^T a_i^* = 0, \;\forall i \in \mathcal{E}.$$

As before, we need to make a regularity assumption to ensure that our linearized analysis captures the geometry of the actual feasible set. Such regularity assumptions are called constraint qualifications, and we present two such conditions in the next definitions (there exist others, but these are the most common ones).

Assumption 8.2.2 (Linear Independence Constraint Qualification) *We say that the linear-independence constraint qualification (LICQ) holds at x^* for the NLP (8.2), iff the constraint normals, $a_i^* = \nabla c_i(x^*)$, for $i \in \mathcal{A}^*$, are linearly independent.*

The next assumption is slightly weaker, and implies the LICQ.

Assumption 8.2.3 (Mangasarian-Fromowitz Constraint Qualification) *We say that the Mangasarian-Fromowitz constraint qualification (MFCQ) holds at x^* for the NLP (8.2), iff the constraint normals of the equality constraints, $a_i^* = \nabla c_i(x^*)$, for $i \in \mathcal{E}$, are linearly independent, and there exists $s \neq 0$ such that*

$$s^T a_i^* > 0, \ \forall i \in \mathcal{I} \cap \mathcal{A}^*.$$

We note, that MFCQ is slightly stronger than the condition that we require for a stationary point, which is that

$$\left\{ s | s^T g^*, \ s^T a_i^* = 0, \ \forall i \in \mathcal{E}, \ s^T a_i^* \geq 0, \ \forall i \in \mathcal{I} \cap \mathcal{A}^* \right\} = \emptyset$$

This condition is again difficult to prove, so we use the following lemma instead.

Lemma 8.2.2 *Assume that Assumption 8.2.2 or 8.2.3 hold, and that x^* is a local minimizer, then the following two conditions are equivalent:*

1. $$\left\{ s | s^T g^* < 0, \ s^T a_i^* = 0, \ \forall i \in \mathcal{E}, \ s^T a_i^* \geq 0, \ \forall i \in \mathcal{I} \cap \mathcal{A}^* \right\} = \emptyset$$

2. *There exist so-called Lagrange multipliers, y_i^*, for $i \in \mathcal{A}^*$ such that*

$$g^* = \sum_{i \in \mathcal{A}^*} y_i^* a_i^* = A^* y \quad \text{where } y_i^* \geq 0, \ \forall i \in \mathcal{I} \cap \mathcal{A}^*.$$

Remark 8.2.1 *Assume that we are at some non-stationary point, and assume that we have found some multipliers (e.g. by solving the corresponding least-squares problems), and that we have $\lambda_q < 0$ for some $q \in \mathcal{I}$, and that we have $s^T a_q = 1$. Then it follows that we can reduce the objective by taking a step in this feasible direction s. This observation will form the basis of the active-set methods discussed in Chapter 9.*

8.2.3 The Karush-Kuhn-Tucker Conditions

The results of the previous sections can be combined into the following theorem that states first-order conditions for the NLP (8.2):

Theorem 8.2.1 (Karush-Kuhn-Tucker (KKT) Conditions) *Let x^* be a local minimizer of (8.2) and assume that a regularity assumption such as Assumption 8.2.2 or 8.2.3 holds at x^*. Then it follows that there exist Lagrange multipliers, y^* such that*

$$\begin{aligned}
\nabla_x \mathcal{L}(x^*, y^*) &= 0 && \textit{first order condition} && (8.6)\\
c_{\mathcal{E}}(x^*) &= 0 && \textit{feasibility} && (8.7)\\
c_{\mathcal{I}}(x^*) &\geq 0 && \textit{feasibility} && (8.8)\\
y_{\mathcal{I}}^* &\geq 0 && \textit{dual feasibility} && (8.9)\\
y_i^* c_i(x^*) &= 0 && \textit{complementary slackness.} && (8.10)
\end{aligned}$$

It is instructive to observe, that these first-order conditions are also the first-order conditions of the linearized problem,

$$\begin{aligned}
\underset{d}{\text{minimize}} \quad & f(x^*) + d^T \nabla f(x^*) \\
\text{subject to} \quad & c_i(x^*) + d^T \nabla c_i(x^*) = 0 \qquad i \in \mathcal{E} \\
& c_i(x^*) + d^T \nabla c_i(x^*) \geq 0 \qquad i \in \mathcal{I},
\end{aligned} \tag{8.11}$$

see Exercise 8.2. This observation motivates algorithmic approaches, where we sequentially linearize the NLP, solve an LP (possibly inside a trust region) for a step, and repeat. We will study such methods in subsequent chapters.

8.3 Second-Order Conditions

The KKT conditions are first-order necessary conditions. In this section, we will expand the second-order conditions from the unconstrained case. Throughout this section we assume that the functions $f(x)$ and $c_i(x)$ are twice continuously differentiable. We note, that it is important to include second-order effects from the constraints, i.e. $\nabla^2 c_i(x)$, and not just $\nabla^2 f(x)$. We give an example below that shows this fact.

As with the first-order conditions, it is convenient to distinguish equality and inequality constraints.

8.3.1 Second-Order Conditions for Equality Constraints

if x^* is a KKT point, and if the constraint normals, a_i^* for $i \in \mathcal{E}$ are linearly independent, then it follows that we can take an incremental step along any feasible direction, s. Letting δ be such an incremental step, we observe that

$$\begin{aligned} f(x^* + \delta) &= \mathcal{L}(x^* + \delta, y^*) \\ &= \mathcal{L}(x^*, y^*) + \delta^T \nabla_x \mathcal{L}(x^*, y^*) + \tfrac{1}{2}\delta^T W^* \delta + o(\|\delta\|^2) \\ &= f(x^*) + \tfrac{1}{2}\delta^T W^* \delta + o(\|\delta\|^2), \end{aligned}$$

where

$$W^* = \nabla^2 \mathcal{L}(x^*, y^*) = \nabla^2 f(x^*) + \sum_{i \in \mathcal{E}} y_i^* \nabla^2 c_i(x^*)$$

is the Hessian of the Lagrangian. The optimality of x^* then implies that

$$s^T W^* s \geq 0, \ \forall s : s^T \nabla a_i^* = 0.$$

Or, in other words, the Lagrangian must have nonnegative curvature for all feasible directions at x^*. This condition is known as a necessary condition for a local minimum.

Proposition 8.3.1 (Second-Order Necessary Condition for Local Minimum) *If x^* is a local minimizer, and if a constraint qualification holds, then it follows that*

$$s^T \nabla^2 \mathcal{L}(x^*, y^*) s \geq 0 \ \forall s : s^T \nabla a_i^* = 0.$$

We can also state a sufficient condition for a local minimizer.

Proposition 8.3.2 (Second-Order Sufficient Condition for Local Minimum) *If $\nabla_x \mathcal{L}(x^*, y^*) = 0$, if $c(x^*) = 0$, and if*

$$s^T \nabla^2 \mathcal{L}(x^*, y^*) s > 0, \ \forall s \neq 0 : s^T \nabla a_i^* = 0,$$

then it follows that x^* is a local minimizer.

We note, that as in the unconstrained case, there is a gap between the necessary and the sufficient conditions.

8.3.2 Second-Order Conditions for Inequality Constraints

One way to derive second-order conditions is to only consider the active constraints, $c_i(x), i \in \mathcal{A}^*$, and then observe that the NLP (8.2) is equivalent to an equality constrained problem as long as $y_i^* > 0, \ \forall i \in \mathcal{I} \cap \mathcal{A}^*$, that is the inequality constraints satisfy strict complementarity. We can then derive the following sufficient condition.

Proposition 8.3.3 (Second-Order Sufficient Condition for Local Minimum) *If* $\nabla_x \mathcal{L}(x^*, y^*) = 0$, *if* $c(x^*) = 0$, *if strict complementarity hold, i.e.* $y_i^* > 0, \ \forall i \in \mathcal{I} \cap \mathcal{A}^*$, *and if*

$$s^T \nabla^2 \mathcal{L}(x^*, y^*) s > 0, \ \forall s \neq 0 : s^T \nabla a_i^* = 0, \ \forall i \in \mathcal{A}^*,$$

then it follows that x^* *is a local minimizer.*

A more rigorous treatment that does not require strict complementarity is possible, but the resulting conditions then require that the Hessian of the Lagrangian is positive definite over a cone, which is more difficult to prove then being positive definite over a set of directions.

We can check the sufficient conditions by finding the inertia the so-called KKT matrix,

$$\begin{bmatrix} W^* & A^* \\ A^{*^T} & 0 \end{bmatrix}.$$

If the inertia of this matrix is $[n - m, 0, m]$, then the matrix is called second-order sufficient and the second order conditions are satisfied, where $m = |\mathcal{A}^*|$. The inertia of a matrix is the triple of positive, zero, and negative eigenvalues.

8.4 Exercises

8.1. Reformulations with slacks, duplicate constraints etc. Special case: equality constraints ...

8.2. Show that the KKT conditions of (8.11) are equivalent to the KKT conditions of (8.2) at x^*.

8.3. Consider the NLP

$$\underset{x}{\text{minimize}} \ \frac{1}{2}\left((x_1 - 1)^2 + x_2^2\right) \quad \text{subject to} \ -x_1 + \beta x_2^2 = 0.$$

For what values of β is $x^* = 0$ a local minimizer of this problem?

8.4. Derive the KKT conditions for the bound constrained optimization problem

$$\underset{x}{\text{minimize}} \ f(x) \quad \text{subject to } l \leq x \leq u$$

and show that they match the conditions in Theorem 7.2.

8.5. Show that the conditions in Theorem 6.2.2 are a consequence of the KKT conditions.

Chapter 9

Linear and Quadratic Programming

Linear and quadratic programming refers to optimization problems with linear constraints, and a linear and quadratic objective function respectively. In this sense, these two classes of problems are the easiest nonlinear programs. However, these two classes of problems also form the basic building blocks of classes of modern nonlinear optimization solvers. In this chapter, we will briefly review active-set methods for linear and quadratic programming, which are equivalent to classical pivoting algorithms such as the Simplex method, but more intuitive. We discuss implementation issues for large-scale solvers.

9.1 Active-Set Method for Linear Programming

In this section, we review some basic properties and present an active-set method for linear programs (LPs) of the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & a_i^T x = b_i \quad i \in \mathcal{E} \\ & a_i^T x \geq b_i \quad i \in \mathcal{I}, \end{array} \tag{9.1}$$

where $\mathcal{E}, \mathcal{I}$ are the sets of equality and inequality constraints, and $x \in \mathbb{R}^n$. Practical implementations of LP solvers allow both lower and upper bounds, and treat variable bounds (and other special structures) in a special way. We chose the notation in (9.1) for ease of presentation.

We start by recalling some basic facts about (9.1):

- The feasible set of (9.1) may be empty. However, in this case, we can use so-called phase-I methods to obtain an initial feasible point, see Section 9.1.1.
- If the feasible set is unbounded, (9.1) may be unbounded. We will show below that we can detect this situation during the line-search, and gracefully terminate the active-set method.
- The feasible set of (9.1) is a polyhedron (which may be empty or unbounded). Each vertex of the polyhedron is described by at least n constraints (there may be more constraints passing through a vertex, in which case we call the vertex *degenerate*).
- If a solution exists, then there exists a solution at a vertex of the feasible set.

The last bullet points towards a practical algorithm. If we knew the constraints that define the optimal vertex, then we could just solve a linear system and obtain the solution. Our approach will therefore be to find this feasible vertex, by moving from one vertex candidate to another (hopefully without enumerating the possibly exponential number of vertices).

Each iterate, $x^{(k)}$ of our algorithm is a vertex of the feasible set, defined by

$$a_i^T x = b_i, \quad i \in \mathcal{W} \quad \Leftrightarrow \quad A_k^T x = b_k,$$

where $\mathcal{W} \subset \mathcal{A}(x)$ is the working set (it is equal to the active set, if we assume that there are no degenerate vertices. We have also introduced the notation for the Jacobian and right-hand-side:

$$A_k := [a_i]_{i \in \mathcal{W}} \in \mathbb{R}^{n \times n} \quad \text{and} \quad b_k^T := (b_i)_{i \in \mathcal{W}} \in \mathbb{R}^n$$

Each iteration of the active-set method consists of a move from one vertex to another along a common edge, reducing the objective function along the way. At $x^{(k)}$, the Lagrange multipliers are defined as

$$y^{(k)} = A_k c.$$

Hence, the optimality test is:

$$y_i^{(k)} \geq 0, \forall i \in \mathcal{I} \cap \mathcal{W} \quad \Rightarrow \; x^{(k)} \text{ optimal.}$$

If we define the feasible edges as

$$A_k^{-T} := [s_i]_{i \in \mathcal{W}} \in \mathbb{R}^{n \times n},$$

then it follows that the slope of the objective along the edge s_i is $y_i^{(k)} = s_i^T c$.

If $x^{(k)}$ is not optimal, then there exists $y_q^{(k)} < 0$, and s_q is a feasible descend direction. One possibly choice for q is to select the most negative multiplier, i.e.

$$y_q := \min_{i \in \mathcal{I} \cap \mathcal{W}} y_i.$$

In practice, other choices that take the scaling of the search directions into account are preferred. Given, y_q, we perform a search along the edge s_q, moving away from the constraint q, which is dropped from the working set, $\mathcal{W}$, along the line

$$x = x^{(k)} + \alpha s_q.$$

During this move, we must consider the effect on the inactive constraints, $i \in \mathcal{I} : i \notin \mathcal{W}$, which we can compute as:

$$r_i := a_i^T x - b_i = a_i^T x^{(k)} + \alpha a_i^T s_q - b_i =: r_i^{(k)} + \alpha a_i^T s_q.$$

This inactive constraint can only become active, if $a_i^T s_q < 0$, in which case we reach it for a stepsize, α, given by:

$$0 = r_i = r_i^{(k)} + \alpha a_i^T s_q \quad \Leftrightarrow \quad \alpha = \frac{r_i^{(k)}}{-a_i^T s_q}.$$

Our goal is to remain feasible with respect to all constraints, and hence we search for the first constraint that becomes active, which corresponds to the smallest such α:

$$\alpha = \min_{i \in \mathcal{I} : i \notin \mathcal{W}, a_i^T s_q < 0} \frac{r_i^{(k)}}{-a_i^T s_q}. \tag{9.2}$$

We note, that if there exists not $i \in \mathcal{I} : i \notin \mathcal{W}$ such that $a_i^T s_q < 0$, then $\alpha = \infty$, which means that we have found an unbounded ray, and we can reduce the objective to $-\infty$ along this ray, and the LP is unbounded.

Typically, we have $\alpha < \infty$, and there exists a constraint p that becomes active. So we exchange p and q in the working set, move to our new vertex, $x^{(k+1)}$ and repeat. Formally, we state the active-set method in Algorithm 9.1.

Active-Set Method for Linear Programming

Given an initial feasible vertex, $x^{(0)}$, and corresponding working set $\mathcal{W}^{(0)}$, set $k = 0$.

repeat

  Optimality Test:

    Let $A_k := [a_i]_{i \in \mathcal{W}^{(k)}}$ and compute multipliers $y^{(k)} = A_k^{-1} c$.

    Find $y_q := \min \{ y_i : i \in \mathcal{W}^{(k)} \cap \mathcal{I} \}$.

  if $y_q \geq 0$ **then**

    $x^{(k)}$ **optimal** solution.

  else

    Line-Search / Ratio Test:

    Let s_q be the column of A^{-T} corresponding to y_q and define

$$\alpha = \min_{i \in \mathcal{I}: i \notin \mathcal{W}, a_i^T s_q < 0} \frac{b_i - a_i^T x^{(k)}}{-a_i^T s_q} =: \frac{b_p - a_p^T x^{(k)}}{-a_p^T s_q}$$

    if $a_i^T s_q \geq 0, \ \forall i \in \mathcal{I} : i \notin \mathcal{W}$ **then**

      LP is **unbounded** along s_q.

    else

      Update / Pivot:

        Exchange p and q in $\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} - \{q\} \cup \{p\}$.

        Set $x^{(k+1)} = x^{(k)} + \alpha s_q$ and $k = k + 1$.

    end

  end

until $x^{(k)}$ *is optimal or LP unbounded*

Algorithm 9.1: Active-Set Method for Linear Programming.

Modern LP solvers are more sophisticated than Algorithm 9.1. In particular, our algorithm might break down (cycle) if it encountered a degenerate vertex (where we could enter an infinite loop exchanging the same sets of pivots over and over). It also uses an unsophisticated way to choose the leaving constraint.

It might seem that the presence of the inverse A^{-1} might be both inefficient and numerically unstable. Modern LP solvers do not work with the inverse, but rather with factors of the active-set matrix $A_k = L_k U_k$, where L_k is a lower triangular and U_k is an upper triangular matrix. There exist efficient factorization techniques for sparse matrices, and we can update (rather than refactor) the active-set matrix A_k after removing a_q and adding q_p. These updates can be done in a numerically stable way.

Finally, for large problems, we clearly require a large number of pivots (though finite), so active-set methods may bot be efficient. In addition, there unfortunately exist examples that show that the Simplex method may require an exponential number of pivots (though practical solvers are very efficient). In Chapter 10 we will consider interior-point methods as an alternative, which have better polynomial complexity bounds, and are more competitive for large-scale problems.

9.1.1 Obtaining an Initial Feasible Point for LPs

If we do not have an initial feasible vertex, then we can introduce surplus variables that measure the amount by which we violate the feasibility and either obtain a feasible point x, or an indication that (9.1) is infeasible

by solving the following phase-I problem:

$$\begin{array}{ll} \underset{x,s}{\text{minimize}} & \sum_{i\in\mathcal{E}} \left(s_i^+ + s_i^-\right) + \sum_{i\in\mathcal{I}} s_i \\ \text{subject to} & a_i^T x - b_i = s_i^+ - s_i^- \qquad i \in \mathcal{E} \\ & a_i^T x - b_i \geq -s_i \qquad\quad i \in \mathcal{I} \\ & s^+ \geq 0,\ s^- \geq 0,\ s \geq 0. \end{array} \tag{9.3}$$

For any given x, an initial feasible point for this problem is readily obtained by setting

$$s_i := \min\left(0, b_i - a_i^T x\right), \quad s_i^- := \min\left(0, b_i - a_i^T x\right), \quad s_i^+ := \min\left(0, -b_i + a_i^T x\right),$$

and we can now use the active-set method described above to either find a feasible point (at which point, the objective of (9.3) will be zero ($s = 0, s^+ = 0, s^- = 0$), or we find a solution with non-zero objective, which provides a proof that there exists no feasible point.

Problem (9.3) is just one possible way to obtain an initial feasible solution. It is equivalent to minimizing the ℓ_1 norm of the constraint violation. Other set-ups are possible, and are generally preferred in practice. In particular, we may prefer an algorithm that removes surplus variables as soon as they become zero, reducing the linear algebra overhead, and ensuring that constraints that become feasible remain feasible.

9.2 Active-Set Method for Quadratic Programming

Like an LP, a quadratic program (QP) can be solved in a finite number of steps. A QP is characterized by having a quadratic objective function and linear constraints. It is a very important class of problems, because Newton's method applied to an NLP gives rise to a sequence of QPs. Here, we study QPs of the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2} x^T G x + g^T x \\ \text{subject to} & a_i^T x = b_i \quad i \in \mathcal{E} \\ & a_i^T x \geq b_i \quad i \in \mathcal{I}, \end{array} \tag{9.4}$$

where $G \in \mathbb{R}^{n\times n}$ is a symmetric matrix (we can always reformulate a problem to have a symmetric Hessian).

We first consider problems with equality constraints only, and then consider general QPs.

9.2.1 Equality-Constrained QPs

Similar to the LP case, we can either have no solution, if the feasible set is empty, or an unbounded solution. Both cases are readily detected, so we will assume for now, that an optimal solution, x^* exists. Unlike LPs, QPs can haven meaningful solutions even if there are only equality constraints. It can be shown that if G is positive semi-definite, then x^* is the global solution, and if G is positive definite, the x^* is unique. In this section, we consider

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2} x^T G x + g^T x \\ \text{subject to} & A^T x = b, \end{array} \tag{9.5}$$

where the columns of the matrix $A \in \mathbb{R}^{n\times m}$ are the vectors a_i for $i \in \mathcal{E}$. We assume that $m \leq n$ and that A has full rank (which implies that unique multipliers exist).

Because A has full rank, we can partition the unknowns x, the matrix, A, and the Hessian G, as

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix},$$

where $x_1 \in \mathbb{R}^m$, $A_1 \in \mathbb{R}^{m\times m}$ is nonsingular. Then it follows that

$$A^T x = b \quad \Leftrightarrow \quad A_1^T x_1 + A_2^T x_2 = b.$$

Because A has full rank, it follows that A_1^{-T} exists, and we can eliminate

$$x_1 = A_1^{-T}\left(b - A_2 x_2\right).$$

In practice, we would factorize A_1, and in this way discover whether or not (9.5) is feasible or not (if the linear system is inconsistent, then the QP has no solution). Partitioning the Hessian, G, and g similarly,

$$g = \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} \quad G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix},$$

we can eliminate x_1 and arrive at a reduced unconstrained QP:

$$\underset{x_2}{\text{minimize}} \ \frac{1}{2} x_2^T \tilde{G} x_2 + \tilde{g}^T x_2, \tag{9.6}$$

where the expressions for $\tilde{G}$ and $\tilde{g}$ are complex, but readily obtained, see Exercise 9.1.

The reduced problem (9.6) has a unique solution, if the reduced Hessian, $\tilde{G}$, is positive definite, and we can obtain this solution by solving the linear system

$$\tilde{G} x_x = -\tilde{g}$$

As before, we can apply Cholesky factors, or the conjugate gradient method. This approach has the advantage that we would also discover whether or not the (9.5) is unbounded. If $\tilde{G}$ has a negative eigenvalue, then we can drive the objective to $-\infty$ and conclude that the problem is unbounded. Having obtained x_2, we can readily obtain x_1, and calculate multipliers using the factors of A_1 by solving the system $A_1 y = g_1$. This elimination technique can be generalized to use other forms of factorization.

9.2.2 General Quadratic Programs

The active-set method for general QPs builds on our ability to solve equality-constrained QPs (EQPs). In particular, we start from an initial feasible point, $x^{(k)}$, with corresponding active (working) set, $\mathcal{W}^{(k)}$, and regard the inequality constraints in $\mathcal{W}^{(k)}$ temporarily as equality constraints. The idea is then to solve the EQP, and either conclude that $x^{(k)}$ is optimal, or find a descend direction, and change the active set, until we identify the optimal active set, $\mathcal{W}^{(k)}$.

In this approach, it is possible to have anywhere between none and n active constraints in the working set. The resulting EQP,

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2} x^T G x + g^T x \\ \text{subject to} & a_i^T x = b_i \quad i \in \mathcal{W}^{(k)}, \end{array} \tag{9.7}$$

can now be solved with any method available for solving EQPs. We need to answer two questions in order to specify our active-set method: (1) When is the solution of (9.7) an optimal solution of (9.4); and, (2) if the solution of (9.7) is not optimal, how do we identify a descend direction?

To answer the first question, we let the solution of (9.7) by $\hat{x}^{(k)}$. If the solution of the EQP $\hat{x}^{(k)}$ satisfies the currently inactive inequality constraints, then we can check whether it is also an optimal solution of (9.4), by considering the multipliers of the active inequality constraints. In particular, if

$$y_i^{(k)} \geq 0, \ \forall i \in \mathcal{I} \cap \mathcal{W}^{(k)} \quad \text{optimality test},$$

then $\hat{x}^{(k)}$ is optimal. Otherwise, there exists some $y_q < 0$, e.g. $y_q := \min\{y_i \ : \ i \in \mathcal{I} \cap \mathcal{W}^{(k)}\}$, and we can move away from constraint q, reducing the objective function. A search direction, s, is obtained by solving the new EQP with $\mathcal{W}^{(k+1)} := \mathcal{W}^{(k)} - \{q\}$.

As we move along the direction s, we can either find another constraint that becomes active, or move to the solution of the EQP (which may be unbounded, of course). We formally state the active-set method for QPs in Algorithm 9.2.

Active-Set Method for Quadratic Programming

Given an initial feasible point, $x^{(0)}$, and corresponding working set, $\mathcal{W}^{(0)}$, set $k = 0$.

repeat

  if $x^{(k)}$ *does not solve the EQP for* $\mathcal{W}^{(k)}$ **then**

    Solve an EQP:

      Solve the EQP (9.5) for the current working set, $\mathcal{W}^{(k)}$.

      Let the solution be $\hat{x}$ and set $s^{(k)} := \hat{x} - x^{(k)}$.

      Line-Search / Ratio Test: $\alpha = \min_{i \in \mathcal{I}: i \notin \mathcal{W}^{(k)}, a_i^T s_q < 0} \left\{ 1, \dfrac{b_i - a_i^T x^{(k)}}{-a_i^T s_q} \right\}$

    if $\alpha < 1$ **then**

      Update Working Set:

      Add constraint p (which attains the min) to $\mathcal{W}^{(k+1)} = \mathcal{W}^{(k)} \cup \{p\}$.

        Set $x^{(k+1)} = x^{(k)} + \alpha s^{(k)}$ and $k = k + 1$.

    else

      Move to solution of EQP, set $x^{(k+1)} = x^{(k)} + s^{(k)}$ and $k = k + 1$.

    end

  else

    Optimality Test:

      Compute Lagrange multipliers, y, and find $y_q := \min \left\{ y_i : i \in \mathcal{W}^{(k)} \cap \mathcal{I} \right\}$.

    if $y_q \geq 0$ **then**

      $x^{(k)}$ optimal solution.

    else

      Update Working Set:

        Remove q from $\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} - \{q\}$.

        Set $x^{(k+1)} = x^{(k)}$ and $k = k + 1$.

    end

  end

until $x^{(k)}$ *is optimal or QP unbounded*

Algorithm 9.2: Active-Set Method for Quadratic Programming.

Algorithm 9.2 can be implemented in a stable and efficient way, by updating factors of the constraint matrix, A, and the reduced Hessian matrix, which also allows us to check for unbounded solutions. An initial feasible point can be obtained with the phase-I method from Section 9.1.1. The active-set method in Algorithm 9.2 is a primal active-set method, because all iterates remain primal feasible. There exists a so-called dual active-set method, that maintains dual feasibility, i.e. the iterates of the multipliers satisfy $y_i^{(k)} \geq 0$ for all $i \in \mathcal{I}$, but where the primal iterates are not feasible.

9.3 Exercises

9.1. Obtain expressions for $\tilde{G}$ and $\tilde{g}$ in (9.6).

9.2. Solve some simple LPs and QPs, sensitivity analysis.

9.3. Examples in AMPL ... multipliers etc.

Chapter 10

Nonlinear Programming Methods

In this chapter, we discuss methods for solving general nonlinearly constrained optimization problems. In particular, we

10.1 Introduction

Nonlinearly constrained optimization problems, also known as nonlinear programming (NLP) problems are an important class of problems with a broad range of engineering, scientific, and operational applications. For ease of presentation, we consider NLPs of the form

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) = 0 \\ & x \geq 0, \end{array} \tag{10.1}$$

where the objective function, $f : \mathbb{R}^n \to \mathbb{R}$, and the constraint functions, $c : \mathbb{R}^n \to \mathbb{R}^m$, are twice continuously differentiable. We denote the multipliers corresponding to the equality constraints, $c(x) = 0$, by y and the multipliers of the inequality constraints, $x \geq 0$, by $z \geq 0$. An NLP may also have unbounded variables, upper bounds, or general range constraints of the form $l_i \leq c_i(x) \leq u_i$, which we omit for the sake of simplicity.

In general, one cannot solve (10.1) directly or explicitly. Instead, an iterative method is used that solves a sequence of simpler, approximate subproblems to generate a sequence of approximate solutions, $\{x^{(k)}\}$, starting from an initial guess, $x^{(0)}$. Every subproblem may in turn be solved by an iterative process. These inner iterations are also referred to as minor iterations. A simplified algorithmic framework for solving (10.1) is as follows.

Given initial estimate $(x^{(0)}, y^{(0)}, z^{(0)}) \in \mathbb{R}^{n+m+n}$, set $k = 0$;
while $x^{(k)}$ *is not optimal* **do**
  repeat
    Approximately solve and refine an approximate subproblem of (10.1) around $x^{(k)}$.
  until *an improved solution estimate* $x^{(k+1)}$ *is found*;
  Check whether $x^{(k+1)}$ is optimal; set $k = k + 1$.
end

Algorithm 10.1: Framework for Nonlinear Optimization Methods

In this chapter, we review the basic components of methods for solving NLPs. In particular, we review the four fundamental components of Algorithm 10.1: the *convergence test* that checks for optimal solutions

or detects failure; the *approximate subproblem* that computes an improved new iterate; the *globalization strategy* that ensures convergence from remote starting points, by indicating whether a new solution estimate is better than the current estimate; and the *globalization mechanism* that truncates the step computed by the local model to enforce the globalization strategy, effectively refining the local model. Our treatment generalizes the methods presented in Part II.

Algorithms for NLPs are categorized by the choice they implement for each of these fundamental components. In the next section, we review the fundamental building blocks of methods for nonlinearly constrained optimization. Our presentation is implementation-oriented and emphasizes the common components of different classes of methods.

Notation. Throughout this chapter, we denote iterates by $x^{(k)}, k = 1, 2, \ldots$, and we use subscripts to indicate functions evaluated at an iterate, for example, $f^{(k)} = f(x^{(k)})$ and $c^{(k)} = c(x^{(k)})$. We also denote the gradients by $g^{(k)} = \nabla f(x^{(k)})$ and the Jacobian by $A^{(k)} = \nabla c(x^{(k)})$. The Hessian of the Lagrangian is denoted by $H^{(k)}$.

10.2 Convergence Test and Termination Conditions

We start by describing the convergence test, a common component among all NLP algorithms. The convergence test also provides the motivation for many local models that are described next. The convergence analysis of NLP algorithms typically provides convergence only to KKT points. A suitable approximate convergence test, derived from the KKT conditions, (8.2.1), is thus given by

$$\|c^{(k)}\| \leq \epsilon \;\text{ and }\; \|g^{(k)} - A^{(k)}y^{(k)} - z^{(k)}\| \leq \epsilon \;\text{ and }\; \|\min(x^{(k)}, z^{(k)})\| \leq \epsilon, \tag{10.1}$$

where $\epsilon > 0$ is the tolerance and the $\min$ in the last expression corresponding to complementary slackness is taken componentwise. See Exercise 10.2 on the equivalence between complementary slackness and the $\min$ condition.

In practice, it may not be possible to ensure convergence to an approximate KKT point, for example, if the constraints fail to satisfy a constraint qualification [320, Ch. 7]. In that case, we replace the second condition by

$$\|A^{(k)}y^{(k)} + z^{(k)}\| \leq \epsilon,$$

which corresponds to an approximate Fritz-John point.

10.2.1 Infeasible Stationary Points.

Unless the NLP is convex or some restrictive assumptions are made, methods cannot guarantee convergence even to a feasible point. Moreover, an NLP may not even have a feasible point, and we are interested in a (local) certificate of infeasibility. In this case, neither the local model nor the convergence test is adequate to achieve and detect convergence. A more appropriate convergence test and local model can be based on the following feasibility problem:

$$\underset{x}{\text{minimize}} \;\; \|c(x)\| \quad \text{subject to } x \geq 0, \tag{10.2}$$

which can be formulated as a smooth optimization problem by introducing slack variables. Algorithms for solving (10.2) are analogous to algorithms for NLPs, because the feasibility problem can be reformulated as a smooth NLP by introducing additional variables. In general, we can replace this objective by any weighted norm. A suitable convergence test is then

$$\|A^{(k)}y^{(k)} - z^{(k)}\| \leq \epsilon \;\text{ and }\; \|\min(x^{(k)}, z^{(k)})\| \leq \epsilon,$$

where $y^{(k)}$ are the multipliers or weights corresponding to the norm used in the objective of (10.2). For example, if we use the ℓ_1 norm, then if $[c^{(k)}]_i < 0$, $[y^{(k)}]_i = -1$, if $[c^{(k)}]_i > 0$, $[y^{(k)}]_i = 1$, and $-1 \leq [y^{(k)}]_i \leq 1$ otherwise. The multipliers are readily computed as a by-product of solving the local model.

10.3 Approximate Subproblem: Improving a Solution Estimate

One key difference among nonlinear optimization methods is how the approximate subproblem is constructed. The goal of the approximate subproblem is to provide a *computable* step that improves on the current iterate. We distinguish three broad classes of approximate subproblems: sequential linear models, sequential quadratic models, and interior-point models. Methods that are based on the augmented Lagrangian method are more suitably described in the context of globalization strategies in Section 10.4.

10.3.1 Sequential Quadratic Programming for Equality Constraints

We start by describing one of the main method for solving NLPs by considering the equality constrained NLP:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) = 0. \end{array} \tag{10.1}$$

Sequential quadratic programming is most easily explained as applying Newton's method to the stationarity conditions of the Lagrangian (8.4). Using the Lagrangian, $\mathcal{L}(x, y) = f(x) - y^T c(x)$, we apply Newton's method to the system

$$\begin{pmatrix} \nabla_x \mathcal{L}(x, y) \\ \nabla_y \mathcal{L}(x, y) \end{pmatrix} = 0 \quad \Leftrightarrow \quad \begin{pmatrix} g(x) - A(x)y \\ -c(x) \end{pmatrix} = 0.$$

Recall Newton's method for solving a nonlinear system of equations, $r(z) = 0$, where, $z \in \mathbb{R}^n$ and $r(z) \in \mathbb{R}^n$ is a smooth function. Newton's method computes steps by solving the linearized system around an iterate $z^{(k)}$, given by

$$z^{(k+1)} = z^{(k)} - \nabla r^{(k)^{-T}} r^{(k)}.$$

Applying Newton's method around $(x^{(k)}, y^{(k)})$ to this system gives

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}^{(k)} & \nabla^2_{xy}\mathcal{L}^{(k)} \\ \nabla^2_{yx}\mathcal{L}^{(k)} & \nabla^2_{yy}\mathcal{L}^{(k)} \end{bmatrix} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = - \begin{pmatrix} g^{(k)} - A^{(k)}y^{(k)} \\ -c^{(k)} \end{pmatrix}.$$

We can simplify this system by observing that the Lagrangian is linear in y (hence it follows that $\nabla^2_{yy}\mathcal{L}^{(k)}$), and by observing that $\nabla^2_{xy}\mathcal{L}^{(k)} = A^{(k)}$. Hence, we obtain

$$\begin{bmatrix} H^{(k)} & -A^{(k)} \\ -A^{(k)^T} & 0 \end{bmatrix} \begin{pmatrix} d_x \\ d_y \end{pmatrix} = - \begin{pmatrix} g^{(k)} - A^{(k)}y^{(k)} \\ -c^{(k)} \end{pmatrix}, \tag{10.2}$$

where $H^{(k)} = \nabla^2_{xx}\mathcal{L}^{(k)}$. Provided that we take unit steps, we can rearrange this system by noting that,

$$x^{(k+1)} = x^{(k)} + d_x \quad \text{and} \quad y^{(k+1)} = y^{(k)} + d_y,$$

implies that (10.2) is equivalent to

$$\begin{bmatrix} H^{(k)} & -A^{(k)} \\ -A^{(k)^T} & 0 \end{bmatrix} \begin{pmatrix} d_x \\ y^{(k+1)} \end{pmatrix} = \begin{pmatrix} -g^{(k)} \\ c^{(k)} \end{pmatrix}.$$

This system can be interpreted as the first-order conditions of the following quadratic program:

$$\begin{array}{ll} \underset{d}{\text{minimize}} & q^{(k)}(d) = \frac{1}{2} d^T H^{(k)} d + g^{(k)^T} d + f^{(k)} \\ \text{subject to} & c^{(k)} + a^{(k)^T} d = 0. \end{array} \tag{10.3}$$

An outline of the SQP algorithm is given in Algorithm 10.2.

Sequential Quadratic Programming Method
Given an initial solution estimate $(x^{(0)}, y^{(0)})$, set $k = 0$.
repeat
  Solve the QP subproblem (10.3), and let the solution be $(d_x, y^{(k+1)})$.
  Set $x^{(k+1)} = x^{(k)} + d_x$ and $k = k + 1$.
until $x^{(k)}, y^{(k)})$ *optimal*

Algorithm 10.2: Sequential Quadratic Programming Method.

In general, this algorithm does not converge from an arbitrary starting point, as the Example in Figure 4.2 shows. However, we can show that this SQP method converges quadratically near a second-order sufficient point.

Theorem 10.3.1 (Quadratic Convergence of SQP) *Assume that x^* is a second-order sufficient point, satisfying the assumptions in Proposition 8.3.2, and assume that the KKT matrix in (10.2) is nonsingular at x^*. If $x^{(0)}$ is sufficiently close to x^*, then it follows that the sequence generated by Algorithm 10.2 converges quadratically to x^*.*

As before in the unconstrained case, we can use quasi-Newton approximations of the Hessian, $H^{(k)}$, by defining the gradient difference as

$$\gamma^{(k)} = \nabla \mathcal{L}(x^{(k+1)}, y^{(k+1)}) - \nabla \mathcal{L}(x^{(k)}, y^{(k+1)}).$$

One caveat is that we may not require $H^{(k)}$ to be positive definite, and hence other updates such as the symmetric rank-one update may be preferable. An alternative is to update the reduced Hessian matrix.

10.3.2 Sequential Linear and Quadratic Programming

We can generalize the SQP method to inequality constraints, if we replace the QP by an inequality constrained QP. This approach gives rise to sequential linear and quadratic programming methods for a general NLP. These methods construct a linear or quadratic approximation of (10.1) and solve a sequence of such approximations, converging to a stationary point.

Sequential Quadratic Programming (SQP) Methods. SQP methods successively minimize a quadratic model, $m^{(k)}(d)$, subject to a linearization of the constraints about $x^{(k)}$ [83, 236, 360] to obtain a displacement $d := x - x^{(k)}$.

$$\begin{array}{ll} \underset{d}{\text{minimize}} & m^{(k)}(d) := g^{(k)^T} d + \frac{1}{2} d^T H^{(k)} d \\ \text{subject to} & c^{(k)} + A^{(k)^T} d = 0 \\ & x^{(k)} + d \geq 0, \end{array} \tag{10.4}$$

where $H^{(k)} \simeq \nabla^2 \mathcal{L}(x^{(k)}, y^{(k)})$ approximates the Hessian of the Lagrangian and $y^{(k)}$ is the multiplier estimate at iteration k. The new iterate is $x^{(k+1)} = x^{(k)} + d$, together with the multipliers $y^{(k+1)}$ of the linearized constraints of (10.4). If $H^{(k)}$ is not positive definite on the null-space of the active constraint normals, then the QP is nonconvex, and SQP methods seek a local minimum of (10.4). The solution of the QP subproblem can become computationally expensive for large-scale problems because the null-space method for solving QPs requires the factorization of a dense reduced-Hessian matrix. This bottleneck has led to the development of other methods that use LP solves in the approximate subproblem, and these approaches are described next.

Sequential Linear Programming (SLP) Methods. SLP methods construct a linear approximation to (10.1). In general, this LP will be unbounded, and SLP methods require the addition of a trust region (discussed in more detail in the next section):

$$\begin{array}{ll} \underset{d}{\text{minimize}} & m^{(k)}(d) = g^{(k)^T} d \\ \text{subject to} & c^{(k)} + A^{(k)^T} d = 0, \\ & x^{(k)} + d \geq 0, \ \text{ and } \ \|d\|_\infty \leq \Delta_k, \end{array} \tag{10.5}$$

where $\Delta_k > 0$ is the trust-region radius. Griffith and Stewart [225] used this method without a trust region but with the assumption that the variables are bounded. In general, $\Delta_k \to 0$ must converge to zero to ensure convergence. SLP methods can be viewed as generalizations of the steepest descent method from Chapter 3, and typically converge only linearly. If, however there are exactly n active and linearly independent constraint normals at the solution, then SLP reduces to Newton's method for solving a square system of nonlinear equations and converges superlinearly. We also note, that at x^*, this LP corresponds to the KKT conditions at the solution, x^*.

Sequential Linear/Quadratic Programming (SLQP) Methods. SLQP methods combine the advantages of the SLP method (fast solution of the LP) and SQP methods (fast local convergence) by adding an equality-constrained QP to the SLP method [110, 119, 184]. SLQP methods thus solve two subproblems: first, an LP is solved to obtain a step for the next iteration and also an estimate of the active set $\mathcal{A}^{(k)} := \left\{ i : [x^{(k)}]_i + \hat{d}_i = 0 \right\}$ from a solution $\hat{d}$ of (10.5). This estimate of the active set is then used to construct an equality-constrained QP (EQP), on the active constraints,

$$\begin{array}{ll} \underset{d}{\text{minimize}} & q^{(k)}(d) = g^{(k)^T} d + \frac{1}{2} d^T H^{(k)} d \\ \text{subject to} & c^{(k)} + A^{(k)^T} d = 0, \\ & [x^{(k)}]_i + d_i = 0, \ \forall i \in \mathcal{A}^{(k)}, \end{array} \tag{10.6}$$

which generalizes the projected-gradient method of Chapter 3. If $H^{(k)}$ is second-order sufficient (i.e., positive-definite on the null-space of the constraints), then the solution of (10.6) is equivalent to the following linear system obtained by applying the KKT conditions to the EQP:

$$\begin{bmatrix} H^{(k)} & -A^{(k)} & -I^{(k)} \\ A^{(k)^T} & & \\ I^{(k)^T} & & \end{bmatrix} \begin{pmatrix} x \\ y \\ z_{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} -g^{(k)} + H^{(k)} x^{(k)} \\ -c^{(k)} \\ 0 \end{pmatrix},$$

where $I^{(k)} = [e_i]_{i \in \mathcal{A}^{(k)}}$ are the normals of the active inequality constraints, and $z_{\mathcal{A}}$ are the multipliers of the active inequalities. By taking a suitable basis from the LP simplex solve, SLQP methods can ensure that $[A^{(k)} : I^{(k)}]$ has full rank. Linear solvers such as MA57 can also detect the inertia; and if $H^{(k)}$ is not second-order sufficient, a multiple of the identity can be added to $H^{(k)}$ to ensure descent of the EQP step.

Sequential Quadratic/Quadratic Programming (SQQP) Methods. SQQP methods have recently been proposed as SQP types of methods [219, 220]. First, a convex QP model constructed by using a positive-definite Hessian approximation is solved. The solution of this convex QP is followed by a reduced inequality constrained model or an EQP with the exact second derivative of the Lagrangian.

Theory of Sequential Linear/Quadratic Programming Methods. If $H^{(k)}$ is the exact Hessian of the Lagrangian and if the Jacobian of the active constraints has full rank, then SQP methods converge quadratically near a minimizer that satisfies a constraint qualification and a second-order sufficient condition [83]. It can also be shown that, under the additional assumption of strict complementarity, all four methods identify the optimal active set in a finite number of iterations.

The methods described in this section are also often referred to as *active-set methods*, because the solution of each LP or QP provides not only a suitable new iterate but also an estimate of the active set at the solution.

10.3.3 Interior-Point Methods

Interior-point methods (IPMs) are an alternative approach to active-set methods. Interior-point methods are a class of perturbed Newton methods that postpone the decision of which constraints are active until the end of the iterative process. The most successful IPMs are primal-dual IPMs, which can be viewed as Newton's method applied to the perturbed first-order conditions of (10.1):

$$0 = F_\mu(x, y, z) = \begin{pmatrix} \nabla f(x) - \nabla c(x)^T y - z \\ c(x) \\ Xz - \mu e \end{pmatrix}, \tag{10.7}$$

where $\mu > 0$ is the barrier parameter, $X = \text{diag}(x)$ is a diagonal matrix with x along its diagonal, and $e = (1, \ldots, 1)$ is the vector of all ones. Note that, for $\mu = 0$, these conditions are equivalent to the first-order conditions except for the absence of the nonnegativity constraints $x, z \geq 0$.

Interior-point methods start at an "interior" iterate $x^{(0)}, z^{(0)} > 0$ and generate a sequence of interior iterates $x^{(k)}, z^{(k)} > 0$ by approximately solving the first-order conditions (10.7) for a decreasing sequence of barrier parameters. Interior-point methods can be shown to be polynomial-time algorithms for convex NLPs; see, for example, [347].

Newton's method applied to the primal-dual system (10.7) around $x^{(k)}$ gives rise to the approximate subproblem,

$$\begin{bmatrix} H^{(k)} & -A^{(k)} & -I \\ A^{(k)^T} & 0 & 0 \\ Z^{(k)} & 0 & X^{(k)} \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = -F_\mu(x^{(k)}, y^{(k)}, z^{(k)}), \tag{10.8}$$

where $H^{(k)}$ approximates the Hessian of the Lagrangian, $\nabla^2 \mathcal{L}^{(k)}$, and the step $(x^{(k+1)}, y^{(k+1)}, z^{(k+1)}) = (x^{(k)}, y^{(k)}, z^{(k)}) + (\alpha_x \Delta_x, \alpha_y \Delta_y, \alpha_z \Delta_z)$ is safeguarded to ensure that $x^{(k+1)}, z^{(k+1)} > 0$ remain strictly positive. A sketch of IPM is given in Algorithm 10.3.

Given an initial solution estimate $(x^{(0)}, y^{(0)}, z^{(0)})$, such that $(x^{(0)}, z^{(0)}) > 0$.
Choose a barrier parameter μ_0, $0 < \sigma < 1$, and a sequence $\epsilon_k \searrow 0$.
repeat
  Set $(x^{(k,0)}, y^{(k,0)}, z^{(k,0)}) = (x^{(k)}, y^{(k)}, z^{(k)})$, $l = 0$.
  repeat
    Approximately solve the Newton system (10.8) for a new iterate $(x^{(k,l+1)}, y^{(k,l+1)}, z^{(k,l+1)})$.
    Set $l = l + 1$.
  until $\|F_{\mu_k}(x^{(k,l)}, y^{(k,l)}, z^{(k,l)})\| \leq \epsilon_k$;
  Reduce the barrier parameter $\mu_{k+1} = \sigma\mu_k$, and set $k = k + 1$.
until $x^{(k)}, y^{(k)}, z^{(k)}$ *optimal*;

Algorithm 10.3: Primal-Dual Interior-Point Method.

Relationship to Barrier Methods. Primal-dual interior-point methods are related to earlier barrier methods [178]. These methods were given much attention in the 1960s but soon lost favor because of the ill-conditioning of the Hessian. They regained attention in the 1980s after it was shown that these methods can provide polynomial-time algorithms for linear programming problems. See the surveys [194, 346, 448] for further material. Barrier methods approximately solve a sequence of barrier problems,

$$\underset{x}{\text{minimize}} \;\; f(x) - \mu \sum_{i=1}^{n} \log(x_i) \quad \text{subject to } c(x) = 0, \tag{10.9}$$

for a decreasing sequence of barrier parameters $\mu > 0$. The first-order conditions of (10.9) are given by

$$\nabla f(x) - \mu X^{-1} e - A(x) y = 0 \;\text{ and }\; c(x) = 0. \tag{10.10}$$

Applying Newton's method to this system of equations results in the following linear system:

$$\begin{bmatrix} H^{(k)} + \mu X^{(k)^{-2}} & -A^{(k)} \\ A^{(k)^T} & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} g^{(k)} - \mu X^{(k)^{-1}} e - A^{(k)} y^{(k)} \\ c^{(k)} \end{pmatrix}.$$

Introducing first-order multiplier estimates $Z(x^{(k)}) := \mu X^{(k)^{-1}}$, which can be written as $Z(x^{(k)})X^{(k)} = \mu e$, we obtain the system

$$\begin{bmatrix} H^{(k)} + Z(x^{(k)})X^{(k)^{-1}} & -A^{(k)} \\ A^{(k)^T} & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = - \begin{pmatrix} g^{(k)} - \mu X^{(k)^{-1}} e - A^{(k)} y^{(k)} \\ c^{(k)} \end{pmatrix},$$

which is equivalent to the primal-dual Newton system (10.8), where we have eliminated

$$\Delta z = -X^{-1} Z \Delta x - Ze - \mu X^{-1} e.$$

Thus, the main difference between classical barrier methods and the primal-dual IPMs is that $Z^{(k)}$ is not free for barrier methods but is chosen as the primal multiplier $Z(x^{(k)}) = \mu X^{(k)^{-1}}$. This freedom in the primal-dual method avoids some difficulties with ill-conditioning of the barrier Hessian.

Convergence of Barrier Methods. If there exists a compact set of isolated local minimizers of (10.1) with at least one point in the closure of the strictly feasible set, then it follows that barrier methods converge to a local minimum [448]. Figure 10.1 illustrates the convergence of barrier methods. we plot the contours of the barrier problem for decreasing values of the barrier parameter, μ.

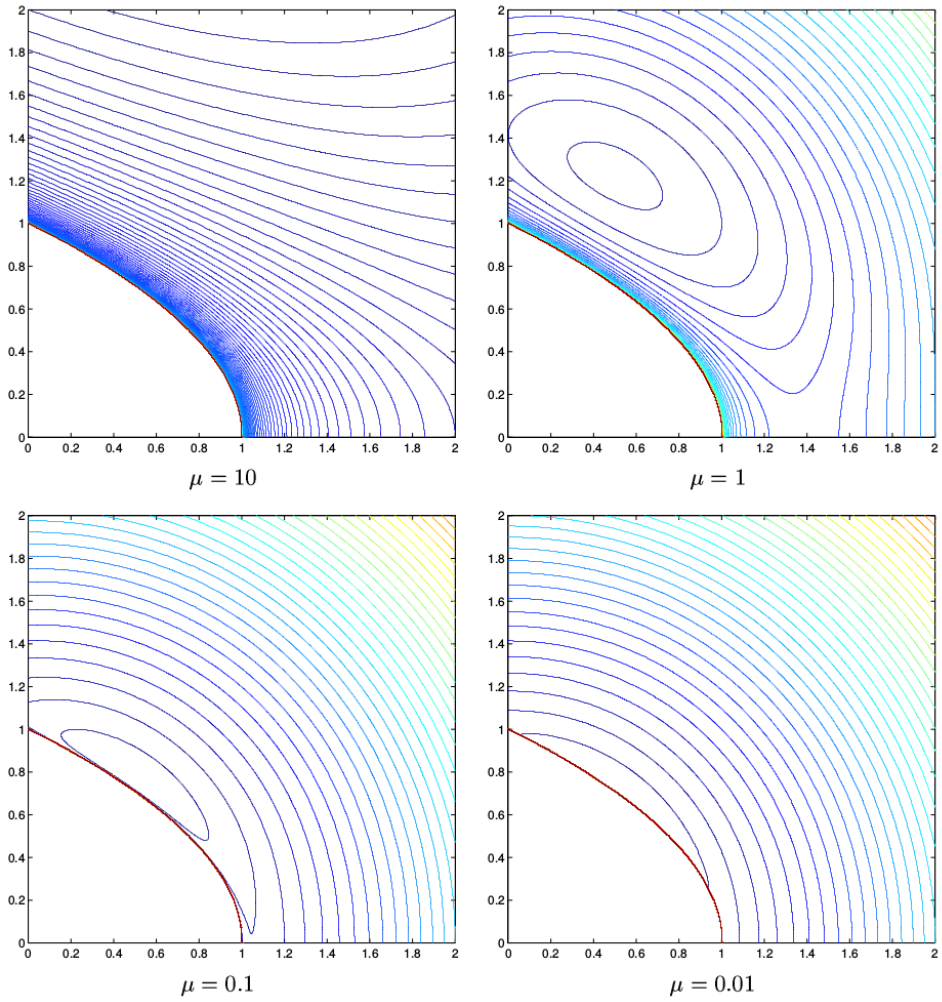


Figure 10.1: Contours of the barrier subproblem for decreasing values of the barrier parameter, μ.

10.4 Globalization Strategy: Convergence from Remote Starting Points

The approximate subproblems of the preceding section guarantee convergence only in a small neighborhood of a regular solution. Globalization strategies are concerned with ensuring convergence from remote starting points to stationary points (and should not be confused with global optimization). To ensure convergence from remote starting points, we must monitor the progress of the iterates generated by the approximate subproblem. Monitoring is easily done in unconstrained optimization, where we can measure progress by comparing objective values. In constrained optimization, however, we must take the constraint violation into account. Three broad classes of strategies exist: augmented Lagrangian methods, penalty and merit-function methods, and filter and funnel methods.

10.4.1 Penalty and Merit Function Methods

Penalty and merit functions combine the objective function and a measure of the constraint violation into a single function whose local minimizers correspond to local minimizers of the original problem (10.1). Convergence from remote starting points can then be ensured by forcing descent of the penalty or merit

function, using using either a line search or a trust region, discussed in the next section.

Exact penalty functions are an attractive alternative to augmented Lagrangians and are defined as

$$p_\rho(x) = f(x) + \rho\|c(x)\|,$$

where $\rho > 0$ is the penalty parameter. Most approaches use the ℓ_1 norm to define the penalty function. It can be shown that a local minimizer, x^*, of $p_\rho(x)$ is a local minimizer of problem (10.1) if $\rho > \|y^*\|_D$, where y^* are the corresponding Lagrange multipliers and $\|\cdot\|_D$ is the dual norm of $\|\cdot\|$ (i.e., the ℓ_∞-norm in the case of the ℓ_1 exact-penalty function); see, for example, [183, Chapter 12.3]. Classical approaches using $p_\rho(x)$ have solved a sequence of penalty problems for an increasing sequence of penalty parameters. Modern approaches attempt to steer the penalty parameter by comparing the predicted decrease in the constraint violation to the actual decrease over a step.

A number of other merit functions also exist. The oldest, the quadratic penalty function, $f(x) + \rho\|c(x)\|_2^2$, converges only if the penalty parameter diverges to infinity. Augmented Lagrangian functions and Lagrangian penalty functions such as $f(x) + y^T c(x) + \rho\|c(x)\|$ have also been used to promote global convergence. A key ingredient in any convergence analysis is to connect the approximate subproblem to the merit function that is being used in a way that ensures a descent property of the merit function; see Section 10.5.1.

10.4.2 Filter and Funnel Methods

Filter and funnel methods provide an alternative to penalty methods that does not rely on the use of a penalty parameter. Both methods use step acceptance strategies that are closer to the original problem, by separating the constraints and the objective function.

Filter Methods. Filter methods keep a record of the constraint violation, $h_l := \|c(x^{(l)})\|$, and objective function value, $f^{(l)} := f(x^{(l)})$, for some previous iterates, $x^{(l)}, l \in \mathcal{F}^{(k)}$ [187]. A new point is acceptable if it improves either the objective function or the constraint violation compared to all previous iterates. That is, $\hat{x}$ is acceptable if

$$f(\hat{x}) \le f^{(l)} - \gamma h_l \quad \text{or} \quad h(\hat{x}) \le \beta h_l,\ \forall l \in \mathcal{F}^{(k)},$$

where $\gamma > 0$ and $0 < \beta < 1$, are constants that ensure that iterates cannot accumulate at infeasible limit points. A typical filter is shown in Figure 11.1 (left), where the straight lines correspond to the region in the (h, f)-plane that is dominated by previous iterations and the dashed lines correspond to the envelope defined by γ, β.

The filter provides convergence only to a feasible limit because any infinite sequence of iterates must converge to a point, where $h(x) = 0$, provided that $f(x)$ is bounded below. To ensure convergence to a local minimum, filter methods use a standard sufficient reduction condition from unconstrained optimization,

$$f(x^{(k)}) - f(x^{(k)} + d) \ge -\sigma m^{(k)}(d), \tag{10.1}$$

where $\sigma > 0$ is the fraction of predicted decrease and $m^{(k)}(d)$ is the model reduction from the approximate subproblem. It makes sense to enforce this condition only if the model predicts a decrease in the objective function. Thus, filter methods use the switching condition $m^{(k)}(d) \ge \gamma h_k^2$ to decide when (10.1) should be enforced. A new iterate that satisfies both conditions is called an f-type iterate, and an iterate for which the switching condition fails is called an h-type iterate to indicate that it mostly reduces the constraint violation. If a new point is accepted, then it is added to the current iterate to the filter, $\mathcal{F}^{(k)}$, if $h_k > 0$ or if it corresponds to an h-type iterations (which automatically satisfy $h_k > 0$).

Funnel Methods. The method of Gould and Toint [221] can be viewed as a filter method with just a single filter entry, corresponding to an upper bound on the constraint violation. Thus, the filter contains only a single entry, $(U_k, -\infty)$. The upper bound is reduced during h-type iterations, to force the iterates toward feasibility; it is left unchanged during f-type iterations. Thus, it is possible to converge without reducing U_k to zero (consistent with the observation that SQP methods converge locally). A schematic interpretation of the funnel is given in Figure 11.1 (right).

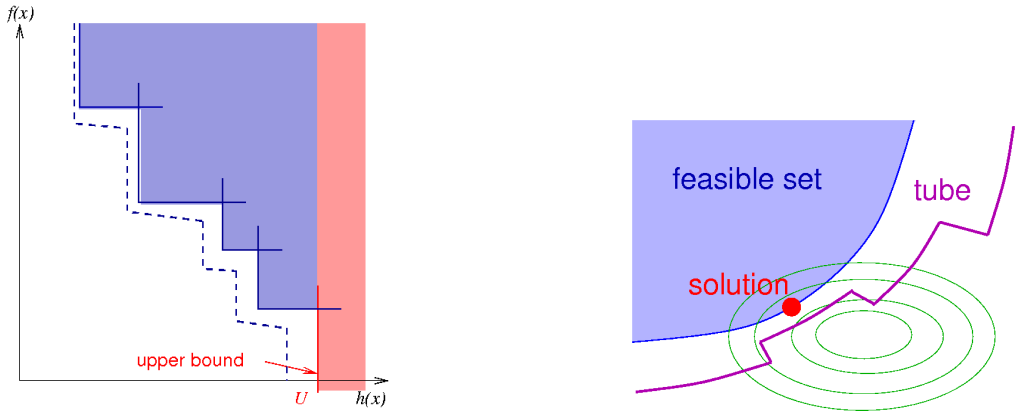


Figure 10.2: The left figure shows a filter where the blue/red area corresponds to the points that are rejected by the filter. The right figure shows a funnel around the feasible set.

10.4.3 Maratos Effect and Loss of Fast Convergence

One can construct simple examples showing that arbitrarily close to an isolated strict local minimizer, the Newton step will be rejected by the exact penalty function [322], resulting in slow convergence. This phenomenon is known as the Maratos effect. It can be mitigated by computing a second-order correction step, which is a Newton step that uses the same linear system with an updated right-hand side [183, 348]. An alternative method to avoid the Maratos effect is the use of nonmonotone techniques that require descent over only the last M iterates, where $M > 1$ is a constant.

10.5 Globalization Mechanisms

In this section, we review two mechanisms to reduce the step that is computed by the approximate subproblem: line-search methods and trust-region methods. Both mechanisms can be used in conjunction with any of the approximate subproblems and any of the global convergence strategies, giving rise to a broad family of algorithms. Below, we describe how these components are used in software for NLPs.

10.5.1 Line-Search Methods

Line-search methods enforce convergence with a backtracking line search along the direction s. For interior-point methods, the search direction, $s = (\Delta_x, \Delta_y, \Delta_z)$, is obtained by solving the primal-dual system (10.8). For SQP methods, the search direction is the solution of the QP (10.4), $s = d$. It is important to ensure that the model produces a descent direction, e.g., $\nabla\Phi(x^{(k)})^T s < 0$ for a merit or penalty function $\Phi(x)$; otherwise, the line search may not terminate. A popular line search is the Armijo search [348], described

in Algorithm 10.4 for a merit function $\Phi(x)$. The algorithm can be shown to converge to a stationary point, detect unboundedness, or converge to a point where there are no directions of descent.

Given initial estimate $x^{(0)} \in \mathbb{R}^n$, let $0 < \sigma < 1$, and set $k = 0$;
while $x^{(k)}$ *is not optimal* **do**
  Approximately solve an approximate subproblem of (10.1) around $x^{(k)}$ for a search direction s.
  Make sure that s is a descent direction, e.g. $\nabla\Phi(x^{(k)})^T s < 0$.
  Set $\alpha^0 = 1$ and $l = 0$.
  repeat
    Set $\alpha^{l+1} = \alpha^l/2$ and evaluate $\Phi(x^{(k)} + \alpha^{l+1}s)$. Set $l = l + 1$.
  until $\Phi(x^{(k)} + \alpha^l s) \leq f^{(k)} + \alpha^l \sigma s^T \nabla\Phi^{(k)}$;
  set $k = k + 1$.
end

Algorithm 10.4: (Armijo) Line-Search Method for Nonlinear Optimization

Line-search methods for filters can be defined in a similar way. Instead of checking descent in the merit function, a filter method is used to check acceptance to a filter. Unlike merit functions, filter methods do not have a simple definition of descent; hence, the line search is terminated unsuccessfully once the step size α^l becomes smaller than a constant. In this case, filter methods switch to a restoration step, obtained by solving a local approximation of (10.2).

10.5.2 Trust-Region Methods

Trust-region methods explicitly restrict the step that is computed by the approximate subproblem, by adding a trust-region constraint of the form $\|d\| \leq \Delta_k$ to the approximate subproblem. Most methods use an ℓ_∞-norm trust region, which can be represented by bounds on the variables. The trust-region radius, $\Delta_k > 0$, is adjusted at every iteration depending on how well the approximate subproblem agrees with the NLP, (10.1).

Given initial estimate $x^{(0)} \in \mathbb{R}^n$, choose $\Delta_0 \geq \underline{\Delta} > 0$, and set $k = 0$;
repeat
  Reset $\Delta_{k,l} := \Delta^{(k)} \geq \underline{\Delta} > 0$; set `success = false`, and $l = 0$.
  repeat
    Solve an approximate subproblem in a *trust-region*, e.g. (10.4) with $\|d\| \leq \Delta_{k,l}$.
    if $x^{(k)} + d$ *is sufficiently better than* $x^{(k)}$ **then**
      Accept the step: $x^{(k+1)} = x^{(k)} + d$; possibly increase $\Delta_{k,l+1}$; set `success = true`.
    else
      Reject the step and decrease the trust-region radius, e.g. $\Delta_{k,l+1} = \Delta_{k,l}/2$.
    end
  until `success = true`;
  Set $k = k + 1$.
until $x^{(k)}$ *is optimal*;

Algorithm 10.5: Trust-Region Methods for Nonlinear Optimization

Step acceptance in this algorithm can be based either on filter methods, or on sufficient decrease in a merit or penalty function, see Algorithm 10.4. Trust-region methods are related to regularization techniques, which add a multiple of the identity matrix, $\sigma^{(k)}I$, to the Hessian, $H^{(k)}$. Locally, the solution of the regularized problem is equivalent to the solution of a trust-region problem with an ℓ_2 trust-region. One disadvantage of trust-region methods is the fact that the subproblem may become inconsistent as $\Delta_{k,l} \to 0$. This situation can be dealt with in three different ways: (1) a penalty function approach, (2) a restoration phase in which the algorithm minimizes the constraint violation [189], or (3) a composite step approach.

10.6 Nonlinear Optimization Software: Summary

Software for nonlinearly constrained optimization can be applied to problems that are more general than (10.1). In particular, solvers take advantage of linear constraints or simple bounds. Thus, a more appropriate problem is of the form

$$\begin{cases} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & l_c \le c(x) \le u_c \\ & l_A \le A^T x \le u_A \\ & l_x \le x \le u_x, \end{cases} \tag{10.1}$$

where the objective function, $f : \mathbb{R}^n \to \mathbb{R}$, and the constraint functions, $c_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, m$, are twice continuously differentiable. The bounds, $l_c, l_A, l_x, u_c, u_A, u_x$, can be either finite or infinite Equality constraints are modeled by setting $l_j = u_j$ for some index j. Maximization problems can be solved by multiplying the objective by -1 (most solvers handle this transformation internally).

NLP solvers are typically designed to work well for a range of other optimization problems such as solving a system of nonlinear equations (most methods reduce to Newton's method in this case), bound-constrained problems, and LP or QP problems. In this survey, we concentrate on solvers that can handle general NLP problems possibly involving nonconvex functions.

Methods for solving (10.1) are iterative and contain the following four basic components: an *approximate subproblem or model* that computes an improved iterate of (10.1), a *global convergence strategy* to promote convergence from remote starting points, a *global convergence mechanism* to force improvement in the global convergence strategy, and a *convergence test* to detect the type of limit point that is reached (see Section 10.2). Solvers for NLP are differentiated by how each of these key ingredients is implemented. In addition there are a number of secondary distinguishing factors such as licensing (open-source versus commercial or academic), API and interfaces to modeling languages, sparse or dense linear algebra, programming language (Fortran/C/MATLAB), and compute platforms on which a solver can run.

main characteristics. A short overview of NLP solvers can be found in Table 10.1. We distinguish solvers mainly by the definition of their local model.

10.7 Exercises

10.1. Open-source solvers in AMPL/JuMP; implementation of primal-dual methods for convex quadratic programming in octave/Matlab.

10.2. Show the following equivalence for vectors, $x, z \in \mathbb{R}^n$:

$$\min(x, y) = 0 \quad \Leftrightarrow \quad 0 \le x \ \perp \ y \ge 0 \quad \Leftrightarrow \quad x \ge 0, y \ge 0, x^T y \le 0.$$

Table 10.1: NLP Software Overview.

Name	Model	Global Method	Interfaces	Language
ALGENCAN	Aug. Lag.	augmented Lagrangian	AMPL, C/C++, CUTEr, Java, MATLAB, Octave, Python, R	f77
CONOPT	GRG/SLQP	line-search	AIMMS, GAMS	Fortran
CVXOPT	IPM	*only convex*	Python	Python
FilterSQP	SQP	filter/trust region	AMPL, CUTEr, f77	Fortran77
GALAHAD	Aug. Lag.	nonmonotone/ augmented Lagrangian	CUTEr, Fortran	Fortran95
IPOPT	IPM	filter/line search	AMPL, CUTEr, C, C++, f77	C++
KNITRO	IPM	penalty-barrier/ trust region	AIMMS, AMPL, GAMS, Mathematica, MATLAB, MPL, C, C++, f77, Java, Excel	C++
KNITRO	SLQP	penalty/trust region	s.a.	C++
LANCELOT	Aug. Lag.	augmented Lagrangian/ trust region	SIF, AMPL, f77	Fortran77
LINDO	GRG/SLP	*only convex*	C, MATLAB, LINGO	
LOQO	IPM	line search	AMPL, C, MATLAB	C
LRAMBO	SQP	ℓ_1 exact penalty/ line search	C	C/C++
MINOS	Aug. Lag.	augmented Lagrangian	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	Fortran77
NLPQLP	SQP	augmented Lagrangian/ line-search	C, f77, MATLAB	Fortran77
NPSOL	SQP	penalty Lagrangian/ line search	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	Fortran77
PATH	LCP	line search	AMPL	C
PENNON	Aug. Lag.	line search	AMPL, MATLAB	C
SNOPT	SQP	penalty Lagrangian/ line search	AIMMS, AMPL, GAMS, MATLAB, C, C++, f77	Fortran77
SQPlab	SQP	penalty Lagrangian/ line search	MATLAB	MATLAB

Chapter 11

Augmented Lagrangian Methods

Here, we show how the augmented Lagrangian method can be used to develop two classes of methods, namely linearly constraint augmented Lagrangian, and bound constrained Lagrangian methods. We also present a method for large-scale methods for quadratic programming that is more suitable for modern high-performance architectures.

11.1 Augmented Lagrangian Methods

The augmented Lagrangian of (10.1) is given by

$$\mathcal{L}(x, y, \rho) = f(x) - y^T c(x) + \frac{\rho}{2}\|c(x)\|_2^2, \tag{11.1}$$

where $\rho > 0$ is the penalty parameter. The augmented Lagrangian is used in two modes to develop algorithms for solving (10.1): by defining a linearly constrained problem or by defining a bound constrained problem.

11.1.1 Linearly Constrained Lagrangian Methods.

These methods successively minimize a shifted augmented Lagrangian subject to a linearization of the constraints. The shifted augmented Lagrangian is defined as

$$\overline{\mathcal{L}}(x, y, \rho) = f(x) - y^T p^{(k)}(x) + \frac{\rho}{2}\|p^{(k)}(x)\|_2^2, \tag{11.2}$$

where $p^{(k)}(x)$ are the higher-order nonlinear terms at the current iterate $x^{(k)}$, that is,

$$p^{(k)}(x) = c(x) - c^{(k)} - A^{(k)^T}(x - x^{(k)}). \tag{11.3}$$

This approach results in the following approximate subproblem:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \overline{\mathcal{L}}(x, y^{(k)}, \rho_k) \\ \text{subject to} & c^{(k)} + A^{(k)^T}(x - x^{(k)}) = 0, \\ & x \geq 0. \end{array} \tag{11.4}$$

We note that if $c^{(k)} + A_k^T(x - x^{(k)}) = 0$, then minimizing the shifted augmented Lagrangian is equivalent to minimizing the Lagrangian over these constraints. Linearly constrained, augmented Lagrangian methods

solve a sequence of problems (11.4) for a fixed penalty parameter. Multipliers are updated by using a first-order multiplier update rule,

$$y^{(k+1)} = y^{(k)} - \rho_k c(x^{(k+1)}), \tag{11.5}$$

where $x^{(k+1)}$ solves (11.4). We observe, that augmented Lagrangian methods iterate on the dual variables, unlike the active-set, or interior-point methods of the previous sections.

11.1.2 Bound-Constrained Lagrangian (BCL) Methods.

These methods approximately minimize the augmented Lagrangian,

$$\underset{x}{\text{minimize}} \ \mathcal{L}(x, y^{(k)}, \rho_k) \quad \text{subject to} x \geq 0. \tag{11.6}$$

The advantage of this approach is that efficient methods for bound-constrained optimization can readily be applied, such as the gradient-projection conjugate-gradient approach [337], which can be interpreted as an approximate Newton method on the active inequality constraints.

Global convergence is promoted by defining two forcing sequences, $\omega_k \searrow 0$, controlling the accuracy with which every bound-constrained problems is solved, and $\eta_k \searrow 0$, controlling progress toward feasibility of the nonlinear constraints. A typical bound-constrained Lagrangian method can then be stated as follows:

Given an initial solution estimate $(x^{(0)}, y^{(0)})$, and an initial penalty parameter ρ_0.
repeat
  Set $x^{(k,0)} = x^{(k)}$, $\rho_{k,0} = \rho_k$, $l = 0$, and `success = false`.
  repeat
    Find an ω_k-optimal solution $x^{(k,l+1)}$ of minimize$_x$ $\mathcal{L}(x, y^{(k)}, \rho_{k,l})$ s.t.$x \geq 0$
    if $\|c(x^{(k,l+1)})\| \leq \eta_k$ **then**
      Perform a first-order multiplier update: $y^{(k+1)} = y^{(k)} - \rho_{k,l} c(x^{(k,l+1)})$.
      Set $\rho_{k+1} = \rho_{k,l}$, and `success = true`.
    else
      Increase penalty: $\rho_{k,l+1} = 10\rho_{k,l}$; set $l = l + 1$.
    end
  until `success = true`;
  Set $x^{(k+1)} + x^{(k,l+1)}$, and $k = k + 1$.
until $x^{(k)}, y^{(k)}$ *is optimal*;

Algorithm 11.1: Bound-Constrained Augmented Lagrangian Method.

We note that the inner loop in Algorithm 11.1 updates the penalty parameter until it is sufficiently large to force progress towards feasibility, at which point the multipliers are updated. Each bound constrained optimization problem (11.6) is solved using a trust-region projected-gradient method with conjugate gradient accelerations, [337]. Each minimization can be started from the previous iterate, $x^{(k,l)}$.

11.1.3 Theory of Augmented Lagrangian Methods.

Conn et al. [129] show that a bound-constrained Lagrangian method can globally converge if the sequence $\{x^{(k)}\}$ of iterates is bounded and if the Jacobian of the constraints at all limit points of $\{x^{(k)}\}$ has column rank no smaller than m. Conn et al. [129] also show that if some additional conditions are met, then their algorithm is R-linearly convergent. Bertsekas and Tsitsiklis [67] shows that the method converges Q-linearly if $\{\rho_k\}$ is bounded, and superlinearly otherwise. Linearly constrained augmented Lagrangian methods can be made globally convergent by adding slack variables to deal with infeasible subproblems.

[67] provide proof of convergence and the rate of convergence when this method is used iteratively on (P). In particular, they show that if:

1. $y^{(k)}$ is updated as $y^{(k+1)} = y^{(k)} + \rho_k c(x^{(k)})$,
2. $\{\rho^{(k)}\}$ is any sequence such that $\rho_{k+1} \geq \rho_k \; \forall \; k > 0$,
3. x^* is a strict local minimum and a regular point of (P) and y^* is the corresponding multiplier,
4. $w' \left[\nabla^2 f(x^*) + \sum_i y^* \nabla^2 c_i(x^*)\right] w > 0$ for all $w \neq 0$ with $\nabla c(x^*)' w = 0$,

then the method converges to (x^*, y^*) Q-linearly if $\{\rho_k\}$ is bounded and superlinearly otherwise.

[373] shows that if only the constraints are linearized, i.e., at each iteration, we solve the nonlinear problem

$$(L - NLP(x^{(k)}, \mathcal{A}^{(k)})) \begin{cases} \underset{x}{\text{minimize}} & m^{(k)}(x) = f(x) + y^{(k)^T} c(x) \\ \text{subject to} & c^{(k)} + A^{(k)^T}(x - x^{(k)}) = 0, \\ & x \geq 0, \end{cases}$$

and select a KKT point of (L-NLP) as the next iterate $x^{(k+1)}$, we get R-quadratic convergence. However, one needs to solve a linearly constrained NLP at each iteration.

11.2 Towards Parallel Active-Set Methods for Quadratic Programming

One of the disadvantages of active-set methods such as Algorithm 9.2 is the fact that they only exchange one active constraint per iteration. In this section, we present an alternative approach that is amenable to parallel implementation, and allows for more active constraints to be changed at each iteration.

We consider general QPs of the form

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & g^T x + \frac{1}{2} x^T G x \\ \text{subject to} & A^T x = b, \\ & x \geq 0, \end{array} \tag{11.7}$$

where b and g are m- and n-vectors, G is an $n \times n$ symmetric (and possibly indefinite) matrix, and A^T is an $m \times n$ matrix. Typically, $n \gg m$. QPs with more general upper and lower bounds are easily accommodated by our method.

We define the *augmented Lagrangian* corresponding to (11.7) as

$$\mathcal{L}_\rho(x, y) = g^T x + \frac{1}{2} x^T G x - y^T (A^T x - b) + \frac{1}{2} \rho \left\| A^T x - b \right\|^2,$$

where x and the m-vector y are independent variables and $\rho > 0$. The usual Lagrangian function is then $\mathcal{L}_0(x, y)$. When $y^{(k)}$ and ρ_k are fixed, we often use the shorthand notation $\mathcal{L}^{(k)}(x) := \mathcal{L}_{\rho_k}(x, y^{(k)})$. Define the *first-order multiplier estimate* by

$$\tilde{y}_\rho(x, y) = y - \rho(Ax - b). \tag{11.8}$$

The derivatives of $\mathcal{L}_\rho$ with respect to x may be written as follows:

$$\nabla_x \mathcal{L}_\rho(x, y) = c + Hx - A^T \tilde{y}_\rho(x, y), \tag{11.9a}$$

$$\nabla^2_{xx} \mathcal{L}_\rho(x, y) = H + \rho A^T A. \tag{11.9b}$$

We assume that (GQP) is feasible and has at least one point (x^*, y^*) that satisfies the first-order KKT conditions.

Definition 11.2.1 (first-order KKT conditions) *A pair (x^*, y^*) is a first-order KKT point for (GQP) if*

$$\min\{x^*, \nabla_x \mathcal{L}_0(x^*, y^*)\} = 0, \tag{11.10a}$$

$$Ax^* = b. \tag{11.10b}$$

The vector of $z^* := \nabla_x \mathcal{L}_0(x^*, y^*)$ is the set of Lagrange multipliers that corresponds to the bounds $x \geq 0$. Our method remains feasible with respect to the simple bounds, and we define the *active* and *inactive* bound constraints at x by the index sets

$$\mathcal{A}(x) = \{j \in 1, \ldots, n \mid x_j = 0\} \qquad \text{and} \qquad \mathcal{I}(x) = \{j \in 1, \ldots, n \mid x_j > 0\}.$$

The symbol x^* may denote a (primal) solution of (GQP) and may also be used to denote a limit point of the sequence $\{x^{(k)}\}$. Let $\mathcal{A}^* := \mathcal{A}(x^*)$ and $\mathcal{I}^* := \mathcal{I}(x^*)$. For $j \in \mathcal{I}^{(k)}$, let H_k be the submatrix formed from the jth rows and columns of H. Similarly, let A_k and A_* be the submatrices formed from the columns of A indexed by $\mathcal{I}^{(k)}$ and $\mathcal{I}^*$, respectively.

A vital component of our algorithm is the concept of a filter [187], which we use to determine the required subproblem optimality and to test acceptance during the line search procedure. The filter is defined by a collection of tuples together with a rule that must be enforced among all entries maintained in the filter. We denote the filter at the kth iteration by $\mathcal{F}^{(k)}$; it is fully defined in Section 11.2.2.

11.2.1 Outline of the Algorithm

Our algorithm differs from classical BCL method in three important ways: First, the main role the augmented Lagrangian minimization in this algorithm is to provide an estimate of the optimal active set, which is used to define an equality-constrained QP that is subsequently solved for a second-order step. The second-order step improves both the reliability and the convergence rate of BCL methods. Second, we use a filter to control various aspects of the algorithm related to global convergence. The filter allows us to dispense with two forcing sequences commonly used in BCL methods (the subproblem tolerance, and the accept/reject threshold for updating the Lagrange multipliers). It also provides a non-monotone globalization strategy that is more likely to accept steps computed by inexact solutions. Third, we exploit the special structure of the QP problem to obtain estimates of the required penalty parameter. These estimates are more adaptive than traditional penalty update schemes, which may overestimate the penalty parameter. Algorithm 11.2 outlines the main steps of our approach.

Outline of QP Filter Method (QPFIL)

Given an initial $x^{(0)}$, set $k \leftarrow 0$, initialize $\mathcal{F}^0$

while *not optimal* **do**

1. Approximately minimize $\mathcal{L}^{(k)}(x)$ to find an $\widetilde{x}^{(k)}$ acceptable to $\mathcal{F}^{(k)}$.
2. Identify an active set $\mathcal{A}^{(k)}$ and update the penalty parameter ρ_{k+1}.
3. Update the multiplier estimate: $\tilde{y}^{(k)} \leftarrow y^{(k)} - \rho_k(A\widetilde{x}^{(k)} - b)$.
4. Solve an equality-constrained QP for a second-order step $(\Delta x, \Delta y)$.
5. Line search: find α such that $(\widetilde{x}^{(k)} + \alpha\Delta x, \tilde{y}_k + \alpha\Delta y)$ is acceptable to $\mathcal{F}^{(k)}$.
6. Update iterates: $(x^{k+1}, y^{k+1}) \leftarrow (\widetilde{x}^{(k)} + \alpha\Delta x, \tilde{y}^{(k)} + \alpha\Delta y)$.
7. Update filter $\mathcal{F}^{k+1}$.
8. $k \leftarrow k + 1$.

end

Algorithm 11.2: Outline of QP Filter Method (QPFIL)

A crucial feature of QPFIL is its suitability for high-performance computing. The two computational kernels of the algorithm are the bound-constrained minimization of the augmented Lagrangian function (step 1) and the solution of an equality-constrained QP (step 4). Scalable tools that perform well on high-performance architectures exist for both steps. For example, TAO [59] and PETSc [33, 34] are suitable, respectively, for the bound-constrained subproblem and the equality-constrained QP. In the remainder of this section we give details of each step of the QPFIL algorithm.

11.2.2 An Augmented Lagrangian Filter

The iterations of a BCL method for nonconvex optimization typically are controlled by two fundamental forcing sequences that ensure convergence to a solution. A decreasing sequence, $\omega^k \to 0$, determines the required optimality of each subproblem solution and controls the convergence of the dual infeasibility (see (11.10a)). The second decreasing sequence, $\eta^k \to 0$, tracks the primal infeasibility (see (11.10b)) and determines whether the penalty parameter ρ^k should be increased or left unchanged.

In the definition of our filter we use quantities that are analogous to ω^k and η^k, and define

$$\omega(x, y) = \|\min\{x, \nabla_x \mathcal{L}_0(x, y)\}\|, \tag{11.11a}$$

$$\eta(x) = \|Ax - b\|, \tag{11.11b}$$

which are based on the optimality and feasibility of a current pair (x, y). Such a choice allows us to dispense with the sequences normally found in BCL methods and instead defines these sequences implicitly. We observe that the filter will generally be less conservative than BCL methods in the acceptance of a current subproblem solution or multiplier update.

Note that $w(x, y)$ is based on the gradient of the Lagrangian function, not on the augmented Lagrangian. Thus, our decision on when to exit the minimization of the current subproblem is based on the optimality of the current subproblem iterate for the original problem, rather than being based on the optimality of the current subproblem as is usually the case in BCL methods. This approach ensures that the subproblem iterations (defined below) always generate solutions that are acceptable to the filter. Another advantage of this definition is that the filter is, in effect, independent of the penalty parameter ρ^k and hence does not need to be updated if ρ^k is increased.

In the remainder of the paper we use the abbreviations

$$\omega^k := \omega(x^k, y^k) \text{and} \eta^k := \eta(x^k).$$

Definition 11.2.2 (augmented Lagrangian filter) *The following rules define an augmented Lagrangian filter:*

1. *A pair (ω', η') dominates another pair (ω, η) if $\omega' \leq \omega$ and $\eta' \leq \eta$, and at least one inequality holds strictly.*

2. *A filter $\mathcal{F}$ is a list of pairs (ω, η) such that no pair dominates another.*

3. *A filter $\mathcal{F}$ contains an entry (called the upper bound)*

$$(\bar{\omega}, \bar{\eta}) = (U, 0), \tag{11.12}$$

   where U is a positive constant.

4. *A pair (x', y') is* acceptable to the filter *$\mathcal{F}$ if and only if*

$$\omega' \leq \beta\omega^\ell or \eta' \leq \beta\eta^\ell - \gamma\omega', \tag{11.13}$$

   for each $(\omega^\ell, \eta^\ell) \in \mathcal{F}$, where $\beta, \gamma \in (0, 1)$ are constants.

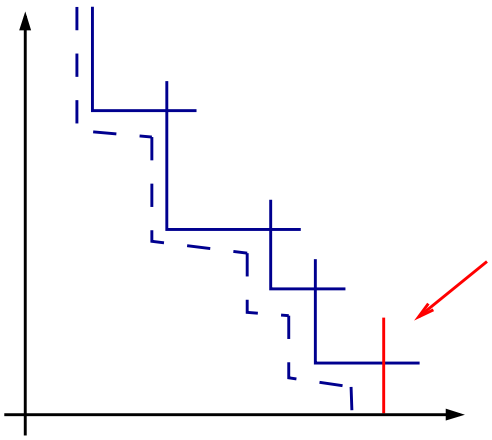


Figure 11.1: A typical filter. All pairs (ω, η) that are below and to the left of the envelope (dashed line) are acceptable to the filter (cf. (11.13)).

We use the shorthand notation $\ell \in \mathcal{F}$ to imply that $(\omega^\ell, \eta^\ell) \in \mathcal{F}$.

A typical filter is illustrated in Figure 11.1. Typical values for the envelope constants are $\beta = 0.999$, $\gamma = 0.001$. A suitable choice for the upper bound U in (11.12) is $U = \delta \max\{1, \omega^0\}$, with $\delta = 1.25$. Filter methods are typically insensitive to the choice of these parameters, and most importantly, these parameters are not problem dependent unlike penalty parameters which must be chosen with more care. We note that (11.13) creates a sloping envelope around the filter. Together with (11.12), this implies that a sequence $\{(\omega^k, \eta^k)\}$ of pairs each acceptable to $\mathcal{F}^k$ must satisfy $\omega^k \to \omega^* = 0$. If the second condition in (11.13) were weakened to $\eta^{k+1} \leq \beta\eta^\ell$, then the sequence of pairs acceptable to $\mathcal{F}^k$ could accumulate to points where $\eta^k \to \eta^* = 0$, but are nonstationary because $\omega^k \to \omega^* > 0$.

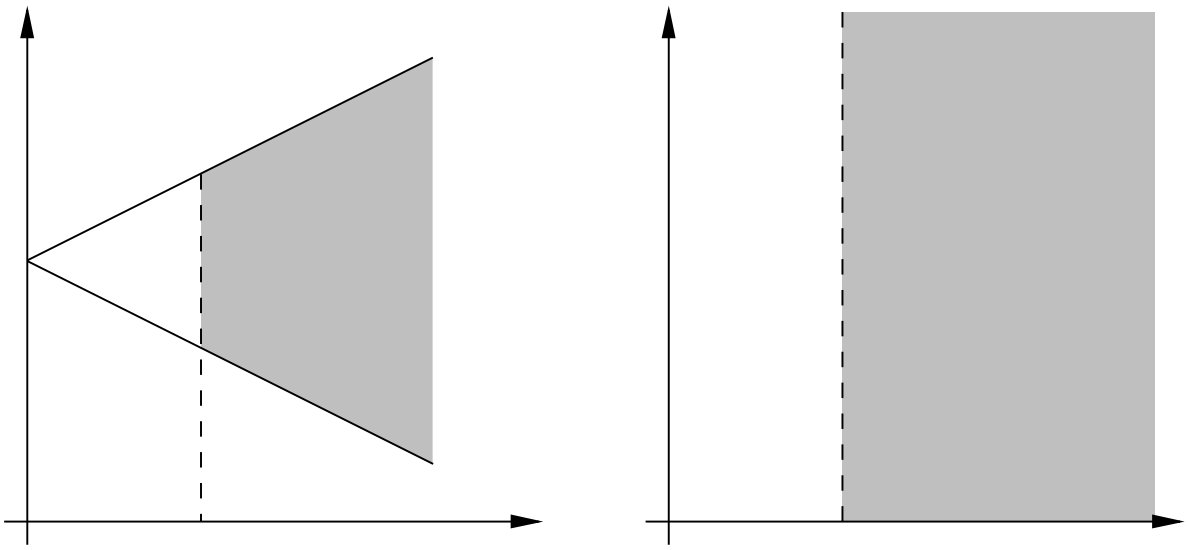
A consequence of $\eta(x) \geq 0$ and the sloping envelope is that the upper bound $(U, 0)$ is theoretically unnecessary—the sloping envelope implies an upper bound $U = \eta_{\min}/\gamma$, where $\eta_{\min}$ is the least η^ℓ for all $\ell \in \mathcal{F}$. In practice, however, we impose the upper bound U in order to avoid generating entries with excessively large values ω^k.

We remark that the axes in the augmented Lagrangian filter appear to be the reverse of the usual definition: feasibility is on the vertical axis instead of the horizontal axis, as it typically appears in the literature. This reflects the dual view of the augmented Lagrangian: it can be shown that $Ax - b$ is a steepest descent direction at x for the augmented Lagrangian [66, §2.2], and that $\omega(x, y)$ is the dual feasibility error. This definition of the filter is similar to the one used in [222]. The gradient of the Lagrangian has also been used in the filter by Ulbrich et al. [428], together with a centrality measure, in the context of interior-point methods.

11.2.3 Active-Set Prediction and Second-Order Steps

Let $\widetilde{x}^k$ be an approximate minimizer of the augmented Lagrangian $\mathcal{L}^k$ at iteration k. We use this solution to derive an active-set estimate $\mathcal{A}^k := \mathcal{A}(\widetilde{x}^k)$, which in turn is used to define an equality-constrained QP (EQP) in the free variables, which are indexed by $\mathcal{I}^k := \mathcal{I}(\widetilde{x}^k)$. The variables $\mathcal{A}^k$ are held fixed at the active bounds.

A second-order correction to $\widetilde{x}^k$ in the space of free variables may be found by solving the following



(a) Nonlinear constraints

(b) Linear constraints

Figure 11.2: The sets $\mathcal{D}$ illustrate the required penalty parameter for the BCL method when the constraints are either nonlinear or linear.

EQP for $\Delta x = (\Delta x_{\mathcal{A}}, \Delta x_{\mathcal{I}})$:

$$\begin{array}{ll} \text{minimize } \Delta x & c^T(\widetilde{x}^k + \Delta x) + \frac{1}{2}(\widetilde{x}^k + \Delta x)^T H(\widetilde{x}^k + \Delta x) \\ \text{subject to} & A(\widetilde{x}^k + \Delta x) = b, \quad \Delta x_{\mathcal{A}^k} = 0. \end{array} \tag{EQP$_k$}$$

Equivalently, a second-order search direction from the current point $(\widetilde{x}^k, \tilde{y}^k)$ is generated from the (first-order) optimality conditions of (EQP$_k$):

$$\begin{pmatrix} -H_k & A_k^T \\ A_k & \end{pmatrix} \begin{pmatrix} \Delta x_{\mathcal{I}} \\ \Delta y \end{pmatrix} = \begin{pmatrix} [c + H\widetilde{x}^k]_{\mathcal{I}^k} - A_k^T \tilde{y}^k \\ b - A\widetilde{x}^k \end{pmatrix}. \tag{11.14}$$

A projected search in the full space is then based on the vector $(\Delta x, \Delta y)$.

Note that step 1 of Algorithm 11.2 requires that the approximate augmented Lagrangian minimizer $\widetilde{x}^k$ be acceptable to the filter. We can show that the first-order multiplier estimate $\tilde{y}^k$ must also be acceptable to the filter. These two properties ensure that even if a line search along $(\Delta x, \Delta y)$ fails to obtain a positive steplength α such that $(\widetilde{x}^k + \alpha\Delta x, \tilde{y}^k + \alpha\Delta y)$ is acceptable to the filter, the algorithm can still make progress with the first-order step alone. In this case, $\alpha = 0$, and the algorithm relies on the progress of the standard BCL iterations.

11.2.4 Estimating the Penalty Parameter

It is well known that BCL methods, under standard assumptions, converge for all large-enough values of the penalty parameter ρ^k. The threshold value $\rho_{\min}$ is never computed explicitly; instead, BCL methods attempt to discover the threshold value by increasing ρ^k in stages. Typically the norm of the constraint violation is used to guide the decisions on when to increase the penalty parameter: a linear decrease (as anticipated by the BCL local convergence theory) signals that the penalty parameter may be held constant; less than linear convergence—or a large increase in constraint violations—indicates that a larger ρ^k is needed.

When the constraints are nonlinear, the penalty-parameter threshold and the initial Lagrange multiplier estimates are closely coupled. Poor estimates y^k of y^* imply that a larger ρ^k is needed to induce convergence. This coupling is fully described by Bertsekas [65, Proposition 2.4]. When the constraints are linear, however, the Lagrange multipliers do not appear in (11.9b), and we see that y^k and ρ^k are essentially decoupled—the curvature of $\mathcal{L}^k$ can be influenced by changing ρ^k alone. This observation is illustrated in Figure 11.2, in which the left figure corresponds to nonlinear constraints and the right figure to linear constraints. The regions in the penalty/multiplier plane for which BCL methods converge are indicated by the shaded regions $\mathcal{D}$. The result below provides an explicit threshold value $\rho_{\min}$ needed to ensure that the Hessian of the augmented Lagrangian is positive definite. (A positive multiple of $\rho_{\min}$ is enough to induce convergence.) Let $\lambda_{\min}(\cdot)$ and $\sigma_{\min}(\cdot)$, respectively, denote the leftmost eigenvalue, and the smallest singular value, of a matrix.

Lemma 11.2.1 *Suppose that $p^T H p > 0$ for all nonzero p such that $Ap = 0$ and A has full row rank. Then $H + \rho A^T A$ is positive definite if and only if*

$$\rho > \rho_{\min} := \lambda_{\min}\Big(A\big(H + \gamma A^T A\big)^{-1} A^T\Big)^{-1}\Big) - \gamma I, \tag{11.15}$$

for any $\gamma \geq 0$ such that $H + \gamma A^T A$ is nonsingular.

The bound provided by Lemma 11.2.1 is sharp: it is both necessary and sufficient. However, the formula on the right-hand side of (11.15) is unsuitable for large-scale computation. The following lemma develops a lower bound for the required ρ that is more easily computable.

Lemma 11.2.2 *Under the conditions of Lemma 11.2.1,*

$$\rho_{\min} < \frac{\max\{0, -\lambda_{\min}(H)\}}{\sigma_{\min}(A)^2}. \tag{11.16}$$

For a given active set $\mathcal{A}^k$, Lemma 11.2.2 implies that ρ^k larger than

$$\rho_{\min}(\mathcal{A}^k) := \frac{\max\{0, -\lambda_{\min}(H_k)\}}{\sigma_{\min}(A_k)^2} \tag{11.17}$$

is sufficient at iteration k to ensure that $\mathcal{L}^k$ is convex on that subspace. Note that this lower bound tends to infinity as the smallest singular value of A_k tends to zero. This property is consistent with (11.15), where we see that if A_k is rank deficient, then the required bound in Lemma 11.2.1 does not exist. We can show that for a given optimal active set, a multiple of this bound is required to induce convergence to an optimal solution in our method.

We are not entirely satisfied with (11.17) because it requires an estimate (or at least a lower bound) of the smallest singular value of the current A_k which can be relatively expensive to compute. One possibility for estimating this value is to use a Lanczos bidiagonalization procedure, as implemented in PROPACK [287].

Ideally, we would compute the penalty value according to (11.15) or (11.16). However, this approach would be prohibitive in terms of computational effort for the size of problems of interest. In our numerical experiments we have instead used the quantity

$$\rho_{\min}(\mathcal{A}^k) = \max\left\{1,\ \frac{\|H_k\|_1}{\max\left\{\frac{1}{\sqrt{|\mathcal{I}^k|}}\|A_k\|_\infty\ ,\ \frac{1}{\sqrt{m}}\|A_k\|_1\right\}}\right\},$$

where $|\mathcal{I}^k|$ is the number of free variables and m is the number of general equality constraints, as a simple approximation to (11.17). We note that the penalty parameter only appears within the subproblem minimization (step 1 of Algorithm 11.2), and not in the definition of the filter. If only a rough approximation to

(11.17) is available, than a multiple of the approximation might be used so as to increase the likelihood that a large enough quantity is obtained. In the remainder of the paper, we assume that $\rho_{\min}(\mathcal{A}^k)$ is given by (11.17).

11.2.5 Minimizing the Augmented Lagrangian Subproblem

Like classical BCL methods, our method generates a sequence of approximate minimizers of the bound-constrained subproblem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{L}^k(x) \\ \text{subject to} & x \geq 0. \end{array} \tag{11.18}$$

Instead of optimizing the subproblem to a prescribed tolerance, however, each iteration of the inner algorithm approximately optimizes it in stages (i.e., a few iterations of some minimization procedure are applied), so that at each iteration j of the inner algorithm, the current iterate x^j satisfies the approximate optimality conditions

$$\left\|\min\{x, \nabla_x \mathcal{L}_{\rho^j}(x, \bar{y})\}\right\|_\infty \leq \epsilon^j, \tag{11.19}$$

where $\bar{y}$ is the latest multiplier estimate y^k. The only requirement on the sequence of approximate minimizations is that they eventually solve the subproblem in the limit, and hence, that $\epsilon^j \to 0$. The iterate x^j and the implied first-order multiplier (11.8) are tested for acceptability against the current filter. The inner-minimization algorithm is described in Algorithm 11.3.

The penalty parameter ρ^j is checked at each inner iteration to ensure that it satisfies the bound implied by Lemma 11.2.1. (See steps 5 and 7.) If the current submatrix A_j is rank deficient (i.e., $\sigma_{\min}(A_j) = 0$), then there does not exist a finite ρ that makes the reduced Hessian positive definite. In that case, we are not assured that reducing the augmented Lagrangian brings the next iterate any closer to optimality of the original subproblem. Instead, we make progress towards feasibility of the iterates by approximately solving the minimum infeasibility problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|Ax - b\|^2 \\ \text{subject to} & x \geq 0, \end{array} \tag{11.20}$$

and we thus require that x^j satisfies the approximate necessary and sufficient condition

$$\left\|\min\{x, A^T(Ax - b)\}\right\|_\infty \leq \epsilon^j. \tag{11.21}$$

The point $x^j \geq 0$ solves the minimum infeasibility problem if $A_j^T(A_j x^j_{\mathcal{I}^j} - b) = 0$, which can be satisfied at infeasible points if A_j rank deficient.

An alternative to step 6 is to increase ρ^j by a fixed multiple. A similar strategy is used in the method suggested in [160], where ρ^j is increased if the current iterate is not "extended regular". With this update, it can be shown that if $\rho^j \to \infty$, then every limit point x^* of x^j is either a KKT point of (GQP), or a solution of (11.20). The analysis given in [160] shows that x^* continues to be a solution of the original QP, but this conclusion depends crucially on the strict convexity of (GQP)—an assumption that we do not make here.

In classical BCL methods, the gradient of the augmented Lagrangian at the latest iterate x^j and the latest multiplier estimate $\bar{y}$ is used to test termination of the inner iterations. The test in step 12 of Algorithm 11.3 is based on the norm of the (usual) Lagrangian function at x^j, but in contrast uses the first-order multiplier estimate $y^j = \bar{y} - \rho^j(Ax^j - b)$. We note that the identity $\nabla_x \mathcal{L}_{\rho^j}(x^j, \bar{y}) = \nabla_x \mathcal{L}_0(x^j, y^j)$ implies that the quantities used to test termination in Algorithm 11.3 and in classical BCL methods are in fact identical. Algorithm 11.3 additionally uses the current primal infeasibility η^j as a criterion. The inner minimization

Bound-Constrained Lagrangian Filter (BCLFIL)
Inputs: x^0, $\bar{y}$, ρ^1, $\mathcal{F}$
Outputs: $\widetilde{x}$, $\widetilde{y}$, $\widetilde{\rho}$

Set $\alpha \in [0, 1)$, $j \leftarrow 0$
repeat
1 $j \leftarrow j + 1$
2 Choose $\epsilon^j > 0$ such that $\lim_{j\to\infty} \epsilon^j = 0$
3 Find a point x^j that satisfies (11.19) [approximately solve (11.18)]
4 $\mathcal{A}^j \leftarrow \mathcal{A}(x^j)$ [update active set]
5 **if** $\sigma_{\min}(A_j) = 0$ **then**
6 Find a point x^j that satisfies (11.21) [feasibility restoration]
7 **else if** $\rho^j < 2\rho_{\min}(\mathcal{A}^j)$ **then**
8 $\rho^{j+1} \leftarrow 2\rho_{\min}(\mathcal{A}^j)$ [increase penalty parameter]
9 **else**
10 $y^j \leftarrow \bar{y} - \rho^j(Ax^j - b)$ [provisional multiplier update]
11 $(\omega^j, \eta^j) \leftarrow \big(\omega(x^j, y^j), \eta(x^j)\big)$ [update primal-dual infeasibility]
12 **if** (ω^j, η^j) *is acceptable to* $\mathcal{F}$ **then**
return $\widetilde{x} \leftarrow x^j$, $\tilde{y} \leftarrow y^j$, $\widetilde{\rho} \leftarrow \rho^j$
end
13 $\rho^{j+1} \leftarrow \rho^j$ [keep penalty parameter]
end
until *Converged*

Algorithm 11.3: Bound-Constrained Lagrangian Filter (BCLFIL)

terminates when the current iterates are acceptable to the filter and the penalty parameter is large enough for the current active set.

To establish that our algorithm finitely identifies the optimal active set, we assume that each approximate minimization reduces the objective by at least as much as does a Cauchy point of a projected-gradient method (see, e.g., [348, §16.6]). This is a mild assumption that is satisfied by most globally convergent bound-constrained solvers. In practice, we perform one or two steps of a bound-constrained optimization algorithm and then test the acceptability of (ω^j, η^j) to the filter. This requirement is often weaker than traditional augmented Lagrangian methods, which at each outer iteration must reduce the projected gradient beyond a specified tolerance that goes to zero; in contrast, here the sequence of inner-iteration tolerances are independent across outer iterations.

11.2.6 Detailed Algorithm Statement

The proposed algorithm is structured around *outer* and *inner* iterations. The outer iterations handle management of the filter, the solution of (EQP$_k$), and the subsequent linesearch. The inner iterations minimize the augmented Lagrangian function, update the multipliers and the penalty parameter, and identify a candidate set of active constraints used to define (EQP$_k$) for the outer iteration. Thus, the inner iteration performs steps 1–3 of Algorithm 11.2.

In step 6 of Algorithm 11.4 we perform a filter linesearch by trying a sequence of steps $\alpha = \gamma^i$, $i = 0, 1, 2, \ldots$, for some constant $\gamma \in (0, 1)$ until an acceptable point is found, or until $\alpha < \alpha_{\min}$, where $\alpha_{\min} > 0$ is a constant parameter. The parameter $\alpha_{\min}$ is needed because the first-order point $(\widetilde{x}^k, \tilde{y}^k)$

could lie in a corner of the filter with the second-order step pointing into the filter. In that case there exists no $\alpha > 0$ that yields an acceptable step. Other choices for deciding when to terminate the linesearch are possible, based, for example, on requiring that the new filter area induced by the linesearch step be larger than the new filter area induced by the first-order step.

The filter update in step 12 of Algorithm 11.4 removes redundant entries that are dominated by a new entry. The upper bound $(U, 0)$ also allows us to manage the number of filter entries that we wish to store. If this number is exceeded, then we can reset the upper bound as $U = \max_\ell\{\omega^\ell \mid \omega^\ell \in \mathcal{F}^k\}$ and subsequently delete dominated entries from $\mathcal{F}^k$, thus reducing the number of filter entries.

We can show that the Cauchy points converge to a stationary point (and hence as long as we do at least as well as the Cauchy point so does the main sequence).

Theorem 11.2.1 (Global Convergence of Algorithm 11.4) *Consider a version of Algorithm 11.4 that skips steps 5–10. Assume that the algorithm generates a sequence of Cauchy points $\{(x^k, y^k)\}$, and that x^* is the* single *limit point of $\{x^k\}$. Then $y^k \to y^*$, where $y^* := \widehat{y}(x^*)$, and (x^*, y^*) is a KKT point of (GQP).*

The algorithm also has an attractive active-set identification property:

Theorem 11.2.2 *Assume that the inner minimization performs a gradient projection that ensures at least Cauchy decrease on the augmented Lagrangian, that (GQP) satisfies strict complementarity, and that $(x^k, y^k) \to (x^*, y^*)$, which is a local minimizer of (GQP). Then Algorithm 11.4 identifies the correct active set in a finite number of iterations.*

11.3 Exercises

11.1. Experiments with qpfil ...

QP Filter Method (QPFIL)
Inputs: x^0, y^0
Outputs: x^*, y^*

Set penalty parameter $\rho^0 > 0$ and positive filter envelope parameters $\beta, \gamma < 1$.
Set filter upper bound $U \leftarrow \gamma \max\left\{1, \|Ax^0 - b\|\right\}$, and add $(U, 0)$ to filter $\mathcal{F}^0$.
Set minimum steplength $\alpha_{\min} > 0$.
Compute infeasibilities $\omega^0 \leftarrow \omega(x^0, y^0)$ and $\eta^0 \leftarrow \eta(x^0)$.
$k \leftarrow 0$
if $\omega^0 > 0$ *and* $\eta^0 > 0$ **then** add (ω^0, η^0) to $\mathcal{F}^0$
.
while *not optimal* **do**
  $k \leftarrow k + 1$
  $(\widetilde{x}^k, \tilde{y}^k, \rho^k) \leftarrow$ **BCLFIL**$(x^{k-1}, y^{k-1}, \rho^{k-1}, \mathcal{F}^{k-1})$
  $\mathcal{A}^k \leftarrow \mathcal{A}(\widetilde{x}^k)$
  Find $(\Delta x^k, \Delta y^k)$ that solves (11.14)
  Find $\alpha^k \in [\alpha_{\min}, 1]$ such that $(\widetilde{x}^k + \alpha \Delta x^k, \tilde{y}^k + \alpha \Delta y^k)$ is acceptable to $\mathcal{F}^k$
  if *linesearch failed* **then**
    $(x^k, y^k) \leftarrow (\widetilde{x}^k, \tilde{y}^k)$ [keep first-order iterates]
  else
    $(x^k, y^k) \leftarrow (\widetilde{x}^k + \alpha^k \Delta x^k, \tilde{y}^k + \alpha \Delta y^k)$ [second-order update]
  end
  $(\omega^k, \eta^k) \leftarrow \big(\omega(x^k, y^k), \eta(x^k)\big)$ [compute infeasibilities]
  if $\omega^k > 0$ **then**
    $\mathcal{F}^k \leftarrow \mathcal{F}^{k-1} \cup \{(\omega^k, \eta^k)\}$
    Remove redundant entries from $\mathcal{F}^k$
    if $\eta^k = 0$ **then** update upper bound U
  end
end
$x^* \leftarrow x^k$, $y^* \leftarrow y^k$

Algorithm 11.4: QP Filter Method (QPFIL)

Chapter 12

Mathematical Programs with Equilibrium Constraints

Mathematical programs with equilibrium constraints (MPECs) are a class of problems that generalize nonlinear programs. Until recently, it had been assumed that these problems were too difficult for standard nonlinear solvers. However, as long as we take care with the problem formulation and the algorithm design, modern NLP solvers can tackle these problems. We present an outline of why these problems are important, how nonlinear solvers can tackle these problems, and finally outline an algorithm that takes the structure of these problems directly into account.

12.1 Introduction and Applications

MPECs arise in a variety of applications, see the surveys [176, 316, 357], and the test problem libraries [154, 298]. The most famous set of applications are leader-follower games (also known as Stackelberg games Stackelberg [410]). In these problems, a leader solves a decision problem that includes the optimal decision problem of a follower in its constraints. The assumption is that the leader's market power means that he can anticipate the follower's reactions. The optimality conditions of the follower are typically expressed using the first-order conditions, introduced in Chapter 8. In particular, these optimality conditions include the complementary slackness condition, which gives rise to so called complementarity constraints. In general, MPECs are a class of nonlinear optimization problems that, in addition to nonlinear constraints, also involve complementarity constraints. Problems of this kind can be conveniently expressed as

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & c_i(x) = 0, & i \in \mathcal{E} \\ & c_i(x) \geq 0 & i \in \mathcal{I} \\ & 0 \leq x_1 \perp x_2 \geq 0, & \end{array} \tag{12.1}$$

where $x = (x_0, x_1, x_2)$ is a decomposition of the problem variables into controls $x_0 \in \mathbb{R}^n$ and states $(x_1, x_2) \in \mathbb{R}^{2p}$. The objective function, $f(x)$, and the constraint functions, $c_i(x) = 0$, are twice continuously differentiable.

The complementarity constraint,

$$0 \leq x_1 \perp x_2 \geq 0$$

means that for all $i = 1, \ldots, p$, $x_{1i} \geq 0$ and $x_{2i} \geq 0$ and that $x_{1i} = 0$ or $x_{2i} = 0$, i.e. x_{1i} and x_{2i} cannot both be non-zero. We note the similarity with the complementarity slackness condition. Clearly, an MPEC

with a more general complementarity condition such as

$$0 \leq G(x) \ \perp \ H(x) \geq 0 \tag{12.2}$$

can be written in the form (12.1) by introducing slack variables. One can easily show that the reformulated MPEC has the same properties (such as constraint qualifications or second-order conditions) as the original MPEC. In this sense, nothing is lost by introducing slack variables.

The modeling language AMPL (and also GAMS) allow users to express complementarity constraints in AMPL. The keyword that AMPL uses is complements, and general format of a complementarity constraint is

```
subject to LLow <= LExpr <= LUpp complements RLow <= RExpr <= RUpp;
```

Here, exactly two of the bounds, `LLow, LUpp, RLow, RUpp`, must be finite, and `LExpr, RExpr` are standard AMPL expressions. Examples of AMPL complementarity constraints can be found in the test problem library [298]. It is straightforward to reformulate more general complementarity constraints into the simpler form used in (12.1), see Exercise 12.1.

Formulation of MPECs as NLPs. One attractive way of solving (12.1) is to replace the complementarity condition by a set of nonlinear inequalities, such as $X_1 x_2 \leq 0$, and then solve the equivalent nonlinear program (NLP),

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to} & c_i(x) = 0, & i \in \mathcal{E} \\ & c_i(x) \geq 0 & i \in \mathcal{I} \\ & x_1, \ x_2 \geq 0, \ X_1 x_2 \leq 0, & \end{array} \tag{12.3}$$

where $X_1 = \text{diag}(x_1)$. An alternative formulation as an NLP is obtained by replacing $X_1 x_2 \leq 0$ by $x_1^T x_2 \leq 0$.

Unfortunately, it has been shown [388] that (12.3) violates the Mangasarian-Fromowitz constraint qualification (MFCQ) at *any* feasible point. This failure of MFCQ implies that the multiplier set is unbounded, the central path fails to exist, the active constraint normals are linearly dependent, and linearizations of (12.3) can become inconsistent *arbitrarily close* to a solution. In addition, early numerical experience with this approach has been disappointing [39]. Bard [39] reports failure on 50–70% of some bilevel problems for a gradient projection method. Ferris and Pang [176] attribute certain failures of lancelot to the fact that the problem contains a complementarity constraint. As a consequence, solving MPECs via NLPs such as (12.3) has been commonly regarded as numerically unsafe.

The failure of MFCQ in (12.3) can be traced to the formulation of the complementarity constraint as $X_1 x_2 \leq 0$. Consequently, algorithmic approaches have focused on avoiding this formulation. Instead, researchers have developed special purpose algorithms for MPECs, such as branch-and-bound methods Bard [39], implicit nonsmooth approaches Outrata et al. [353], piecewise SQP methods [315], and perturbation and penalization approaches Dirkse et al. [155] analyzed by Scholtes [390]. All of these techniques, however, require significantly more work than a standard NLP approach to (12.3).

Recently, exciting new developments have demonstrated that the gloomy prognosis about the use of (12.3) may have been premature. Standard NLP solvers have been used to solve a large class of MPECs, written as NLPs, reliably and efficiently. In this chapter, we review these novel developments and present some other reliable approaches for MPECs.

Remark 12.1.1 (On the Importance of the Correct Complementarity Formulation) *We note that we used the inequality $X_1 x_2 \leq 0$, rather than the more common equation form, $X_1 x_2 = 0$. This is a deliberate choice that has a profound impact on the convergence of NLP solvers. If we had used $X_1 x_2 = 0$ instead,*

we would not be able to provide the same strong convergence results. Similarly, the success of NLP solvers also relies on the fact that we only have complementarity between variables, and not more general complementarity constraints. We will give examples below that show that this formulation is critical to the success of NLP methods.

Notation. We denote the Jacobian of the $c(x)$ constraints as $A(x) := \nabla(c_{\mathcal{E}}^T(x), c_{\mathcal{I}}^T(x))^T$, where $c_{\mathcal{E}}(x)$ are the equality and $c_{\mathcal{I}}(x)$ are the inequality constraints. We will treat the Jacobian of the constraints $X_1 x_2$ separately.

12.2 Optimality Conditions and Regularization

This section reviews stationarity concepts for MPECs in the form (12.1) and introduces a second-order condition. It follows loosely the development of Scheel and Scholtes [388], although the presentation is slightly different.

Given two index sets $\mathcal{X}_1,\ \mathcal{X}_2 \subset \{1, \ldots, p\}$ with

$$\mathcal{X}_1 \cup\ \mathcal{X}_2\ =\ \{1, \ldots, p\}\,, \tag{12.4}$$

we denote their respective complements in $\{1, \ldots, p\}$ by $\mathcal{X}_1^\perp$ and $\mathcal{X}_2^\perp$. For any such pair of index sets, we define the *relaxed NLP corresponding to the MPEC* (12.1) as

$$\begin{array}{lll} \underset{x}{\text{minimize}} & f(x) & \\ \text{subject to} & c_i(x) = 0 & i \in \mathcal{E} \\ & c_i(x) \geq 0 & i \in \mathcal{I} \\ & x_{1j} = 0 & \forall j \in \mathcal{X}_2^\perp \\ & x_{2j} = 0 & \forall j \in \mathcal{X}_1^\perp \\ & x_{1j} \geq 0 & \forall j \in \mathcal{X}_2 \\ & x_{2j} \geq 0 & \forall j \in \mathcal{X}_1. \end{array} \tag{12.5}$$

Concepts such as constraint qualifications, stationarity, and a second-order condition for MPECs will be defined in terms of the relaxed NLPs. The term "relaxed NLP" stems from the observation that if x^* is a local solution of a relaxed NLP (12.5) *and* satisfies complementarity ${x_1^*}^T x_2^* = 0$, then x^* is also a local solution of the original MPEC (12.1). One can naturally associate with every feasible point $\hat{x} = (\hat{x}_0, \hat{x}_1, \hat{x}_2)$ of the MPEC a relaxed NLP (12.5) by choosing $\mathcal{X}_1$ and $\mathcal{X}_2$ to contain the indices of the vanishing components of $\hat{x}_1$ and $\hat{x}_2$, respectively. In contrast to [388], our definition of the relaxed NLP is independent of a specific point; however, it will occasionally be convenient to identify the above sets of vanishing components associated with a specific point $\hat{x}$, in which case we denote them by $\mathcal{X}_1(\hat{x}),\ \mathcal{X}_2(\hat{x})$ or use suitable superscripts. Note that for these sets the condition (12.4) is equivalent to $\hat{x}_1^T \hat{x}_2 = 0$.

The indices that are both in $\mathcal{X}_1$ and $\mathcal{X}_2$ are referred to as the *biactive components* (or second-level degenerate indices) and are denoted by

$$\mathcal{D} := \mathcal{X}_1 \cap \mathcal{X}_2.$$

Obviously, in view of (12.4), $(\mathcal{X}_1^\perp, \mathcal{X}_2^\perp, \mathcal{D})$ is a partition of $\{1, \ldots, p\}$. A solution x^* to the problem (12.1) is said to be *second-level nondegenerate* if $\mathcal{D}(x^*) = \emptyset$.

First, the linear independence constraint qualification (LICQ) is extended to MPECs.

Definition 12.2.1 *Let $x_1, x_2 \geq 0$, and define*

$$\mathcal{X}_| := \{i\ :\ x_{ji} = 0\}\ \ \textit{for}\ \ j = 1, 2.$$

The MPEC (12.1) *is said to satisfy an* MPEC-LICQ *at* x *if the corresponding relaxed NLP* (12.5) *satisfies an LICQ.*

In [388], four stationarity concepts are introduced for MPEC (12.1). The stationarity definition that allows the strongest conclusions is Bouligand or B-stationarity.

Definition 12.2.2 *A point* x^* *is called* Bouligand, or B-stationary *if* $d = 0$ *solves the LPEC obtained by linearizing* f *and* c *about* x^**,*

$$\begin{array}{ll} \underset{d}{\textit{minimize}} & {g^*}^T d \\ \textit{subject to} & c^*_{\mathcal{E}} + {A^*_{\mathcal{E}}}^T d = 0 \\ & c^*_{\mathcal{I}} + {A^*_{\mathcal{E}}}^T d \geq 0 \\ & 0 \leq x^*_1 + d_1 \ \perp \ x^*_2 + d_2 \geq 0. \end{array}$$

We note that B-stationarity implies feasibility because if $d = 0$ solves the above LPEC, then $c^*_{\mathcal{E}} = 0,\ c^*_{\mathcal{I}} \geq 0$, and $0 \leq x^*_1 \ \perp \ x^*_2 \geq 0$. B-stationarity is difficult to check because it involves the solution of an LPEC that is a combinatorial problem and may require the solution of an exponential number of LPs, unless *all* these LPs share a common multiplier vector. Such a common multiplier vectors exists if an MPEC-LICQ holds.

The results of this chapter relate to the following notion of strong stationarity.

Definition 12.2.3 *A point* x^* *is called* strongly stationary *if there exist multipliers* λ, $\hat{\nu}_1$ *and* $\hat{\nu}_2$ *such that*

$$\begin{aligned} g^* - \left[{A^*_{\mathcal{E}}}^T \ : \ {A^*_{\mathcal{I}}}^T \right] \lambda - \begin{pmatrix} 0 \\ \hat{\nu}_1 \\ \hat{\nu}_2 \end{pmatrix} &= 0 \\ c^*_{\mathcal{E}} &= 0 \\ c^*_{\mathcal{I}} &\geq 0 \\ x^*_1 &\geq 0 \\ x^*_2 &\geq 0 \\ x^*_{1j} = 0 \ \textit{or} \ x^*_{2j} &= 0 \\ \lambda_{\mathcal{I}} &\geq 0 \\ c^*_i \lambda_i &= 0 \\ x^*_{1j} \hat{\nu}_{1j} &= 0 \\ x^*_{2j} \hat{\nu}_{2j} &= 0 \\ \textit{if} \ x^*_{1j} = x^*_{2j} = 0 \ \textit{then} \ \hat{\nu}_{1j} \geq 0 \quad \textit{and} \quad \hat{\nu}_{2j} &\geq 0, \end{aligned} \tag{12.6}$$

where $g^* = \nabla f(x^*)$, $A^*_{\mathcal{E}} = \nabla c^T_{\mathcal{E}}(x^*)$, *and* $A^*_{\mathcal{I}} = \nabla c^T_{\mathcal{I}}(x^*)$.

Note that (12.6) are the stationarity conditions of the relaxed NLP (12.5) at x^*. B-stationarity is equivalent to strong stationarity if the MPEC-LICQ holds (e.g., Scheel and Scholtes [388]).

Alphabet Soup of Stationarity Conditions. There exists a whole alphabet soup of stationarity conditions, which are largely useless. These definitions differ from Definition 12.2.3 (strong stationarity) in that the conditions on the sign of the multipliers, $\hat{\nu_1}, \hat{\nu_2}$ are relaxed, when $x^*_{1j} = x^*_{2j} = 0$. If we define the degenerate set $\mathcal{D}^* := \mathcal{D}(x^*) := \{i \ : \ x^*_{1i} = x^*_{2i} = 0\}$, we can state these "stationarity" concepts by replacing the last condition in (12.6) by

1. x^* is called *A-stationary* if $\hat{\nu}_i \geq 0 \ \text{ or } \ \hat{\nu}_i \geq 0, \ \forall i \in \mathcal{D}^*$.

2. x^* is called *C-stationary* if $\hat{\nu}_i \hat{\nu}_i \geq 0 \; \forall i \in \mathcal{D}^*$.

3. x^* is called *M-stationary* if $\left(\hat{\nu}_i > 0 \;\text{ and }\; \hat{\nu}_i > 0\right)$ or $\hat{\nu}_i \hat{\nu}_i = 0, \; \forall i \in \mathcal{D}^*$.

In all these cases, we can easily derive first-order descend directions, which contradicts the fundamental meaning of stationarity, namely the absence of first-order descend directions. We visualize the relationship between these different conditions in Figure 12.1.

Next, a second-order sufficient condition (SOSC) for MPECs is given. Since strong stationarity is related to the relaxed NLP (12.5), it seems plausible to use the same NLP to define a second-order condition. For this purpose, let $\mathcal{A}^*$ denote the set of active constraints of (12.5) and $\mathcal{A}^*_+ \subset \mathcal{A}^*$ the set of active constraints with nonzero multipliers (some could be negative). Let A denote the matrix of active constraint normals, that is,

$$A = \begin{bmatrix} & & 0 & & 0 \\ A^*_{\mathcal{E}} \;:\; A^*_{\mathcal{I}\cap\mathcal{A}^*} \;:\; & & I^*_1 & : & 0 \\ & & 0 & & I^*_2 \end{bmatrix} =: \; [a^*_i]_{i\in\mathcal{A}^*} \, ,$$

where $A^*_{\mathcal{I}\cap\mathcal{A}^*}$ are the active inequality constraint normals and

$$I^*_1 := [e_i]_{i\in\mathcal{X}^*_1} \;\text{ and }\; I^*_2 := [e_i]_{i\in\mathcal{X}^*_2}$$

are parts of the $p \times p$ identity matrices corresponding to active bounds. Define the set of feasible directions of zero slope of the relaxed NLP (12.5) as

$$S^* = \left\{ s \mid s \neq 0 \; , \; {g^*}^T s = 0 \; , \; {a^*_i}^T s = 0 \; , \; i \in \mathcal{A}^*_+ \; , \; {a^*_i}^T s \geq 0 \; , \; i \in \mathcal{A}^* \backslash \mathcal{A}^*_+ \right\}.$$

We can now give an MPEC-SOSC. This condition is also sometimes referred to as the strong-SOSC.

Definition 12.2.4 *A strongly stationary point x^* with multipliers $(\lambda^*, \hat{\nu}^*_1, \hat{\nu}^*_2)$ satisfies the* MPEC-SOSC *if for every direction $s \in S^*$ it follows that*

$$s^T \nabla^2 \mathcal{L}^* s > 0,$$

where $\nabla^2 \mathcal{L}^*$ is the Hessian of the Lagrangian of (12.5) evaluated at $(x^*, \lambda^*, \hat{\nu}^*_1, \hat{\nu}^*_2)$.

The definitions of this section are readily extended to the case where a more general complementarity condition such as (12.2) is used. Moreover, any reformulation using slacks preserves all of these definitions. In that sense, there is no loss of generality in assuming that slacks are being used.

We visualize the relationships among these (confusing) stationarity concepts in Figure 12.1. Scheel and Scholtes [388] have shown that strong stationarity implies B-stationarity. However, the reverse is true only, if the MPEC satisfies an MPEC linear independence constraint qualification or if $\mathcal{D}^* = \emptyset$. If $\mathcal{D}^* = \emptyset$, then all stationarity concepts are equivalent. However, in the interesting case where $\mathcal{D}^* \neq \emptyset$, it follows that A-, C- and M-stationary points allow trivial descend directions, making these stationarity concepts too weak to be useful. We will study examples of these stationarity concepts in the exercise section below.

12.3 Convergence of Nonlinear Optimization Methods

Next, we briefly outline how modern NLP solvers can be made to work quite effectively for MPECs, and how we can prove some key results. This is a remarkable story, because the traditional analysis of SQP and interior-point methods relies heavily on a constraint qualification, which fails at any feasible point. As it turns out, SQP method by chance exploit the structure of the complementarity constraint, while interior-point methods can be made to work using either a penalty or regularization approach. In both cases we can prove powerful theoretical results that support the excellent computational performance (at present, there is no better general purpose MPEC solver than our good old NLP solvers).

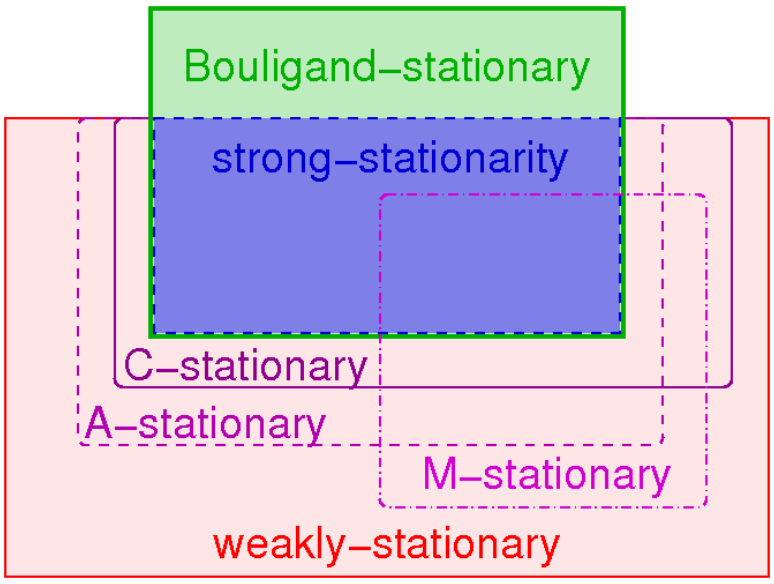


Figure 12.1: Relationships among MPEC stationarity concepts.

12.3.1 Convergence of SQP Methods

This section shows that SQP methods converge quadratically near a strongly stationary point under mild conditions. In particular, we are interested in the situation where $x^{(k)}$ is close to a strongly stationary point, x^*, but $x_1^{(k)^T} x_2^{(k)}$ is *not* necessarily zero. SQP then solves a sequence of quadratic programming approximations, given by

$$(QP^k) \left\{ \begin{array}{ll} \underset{d}{\text{minimize}} & g^{(k)^T} d + \frac{1}{2} d^T W^{(k)} d \\ \text{subject to} & c_{\mathcal{E}}^{(k)} + A_{\mathcal{E}}^{(k)^T} d = 0 \\ & c_{\mathcal{I}}^{(k)} + A_{\mathcal{I}}^{(k)^T} d \geq 0 \\ & x_1^{(k)} + d_1 \geq 0 \\ & x_2^{(k)} + d_2 \geq 0 \\ & x_1^{(k)^T} x_2^{(k)} + x_2^{(k)^T} d_1 + x_1^{(k)^T} d_2 \leq 0, \end{array} \right.$$

where $W^{(k)} = \nabla^2 \mathcal{L}(x^{(k)}, \mu^{(k)})$ is the Hessian of the Lagrangian of (12.3) and $\mu^{(k)} = (\lambda^{(k)}, \nu_1^{(k)}, \nu_2^{(k)}, \xi^{(k)})$. The last constraint of (QP^k) is the linearization of the complementarity condition $x_1^T x_2 \leq 0$.

Assumption 12.3.1 *The following assumptions are made:*

[A1] *f and c are twice Lipschitz continuously differentiable.*

[A2] *(12.1) satisfies an MPEC-LICQ (Definition 12.2.1).*

[A3] *x^* is a strongly stationary point of (12.1) with multipliers $\lambda^*, \hat{\nu}_1^*, \hat{\nu}_2^*$ (Definition 12.2.3), and x^* satisfies the MPEC-SOSC (Definition 12.2.4).*

[A4] *$\lambda_i^* \neq 0, \ \forall i \in \mathcal{E}^*, \ \lambda_i^* > 0, \ \forall i \in \mathcal{A}^* \cap \mathcal{I}$, and both $\nu_{1j}^* > 0$ and $\nu_{2j}^* > 0, \ \forall j \in \mathcal{D}^*$.*

[A5] *The QP solver always chooses a linearly independent basis.*

The most restrictive assumption is strong stationarity in **[A3]**, which follows if x^* is a local minimizer from **[A2]**. That is **[A3]** (or **[A2]**) removes the combinatorial nature of the problem. It is not clear that **[A2]** can readily be relaxed in the present context, since it allows us check B-stationarity by solving exactly one LP or QP. Without assumption **[A2]** it would not be possible to verify B-stationarity without solving several LPs (one for every possible combination of second-level degenerate indices $i \in \mathcal{D}^*$). It seems unlikely, therefore, that any method that solves only a single LP or QP per iteration can be shown to be convergent to B-stationary points for problems that violate MPEC-LICQ. Note that we do *not* assume that the MPEC (12.1) is second-level nondegenerate, in other words, we do *not* assume that $x_1^* + x_2^* > 0$. Assumption **[A5]** is a reasonable assumption in practice, as most modern SQP solvers are based on active set QP solvers that guarantee this.

Under the Assumptions 12.3.1 we can show the following result.

Theorem 12.3.1 *Let Assumptions* **[A1]–[A5]** *and* **[A7]** *hold. Then it follows that SQP applied to the NLP formulation (12.3) of the MPEC (12.1) converges quadratically near a solution* (x^*, μ^*).

The proof is rather intricate and divided into two parts. First, it is shown that if $x_1^{(k)^T} x_2^{(k)} = 0$ at some iteration k, then the SQP approximation of (12.3) about this point is equivalent to the SQP approximation of the relaxed NLP. Since the latter is a well behaved problem, superlinear convergence follows. The second part of the proof assumes that $x_1^{(k)^T} x_2^{(k)} > 0$, and it is shown that each QP basis remains bounded away from singularity. Again, convergence can be established by using standard techniques.

One undesirable assumption in [190] is that all QP approximations are consistent. This is trivially true if $x_1^{(k)^T} x_2^{(k)} = 0$ for some k, and it can be shown to hold if the lower-level problem satisfies a certain mixed-P property [315]. In practice [188], a simple heuristic is implemented that relaxes the linearization of the complementarity constraint.

12.3.2 Convergence of Interior-Point Methods

Interior-point methods require an additional reformulation of the equivalent NLP (12.3), in order to mitigate the effect of the loss of MFCQ. Two alternative approaches are penalization and regularization, which are equivalent under certain conditions.

The first category comprises relaxation approaches, where (12.3) is replaced by a family of problems in which $x_{1i}x_{2i} \leq 0$ is changed to

$$x_{1i}x_{2i} \leq \theta, \quad i = 1, \ldots, p, \tag{12.7}$$

and the relaxation parameter $\theta > 0$ is driven to zero. This type of approach has been studied from a theoretical perspective by Scholtes [390] and D. Ralph and S. J. Wright [135]. Interior methods based on the relaxation (12.7) have been proposed by Liu and Sun [310] and Raghunathan and Biegler [370]. In both studies, the parameter θ is proportional to the barrier parameter μ and is updated only at the end of each barrier problem. Raghunathan and Biegler focus on local analysis and report very good numerical results on the MacMPEC collection. Liu and Sun analyze global convergence of their algorithm and report limited numerical results. Numerical difficulties may arise when the relaxation parameter gets small, since the interior of the regularized problem shrinks toward the empty set.

The second category involves a regularization technique based on an exact-penalty reformulation of the MPEC. Here, $x_1^T x_2$ is moved to the objective function in the form of an ℓ_1-penalty term, so that the objective becomes

$$f(x) + \pi x_1^T x_2, \tag{12.8}$$

where $\pi > 0$ is a penalty parameter. If π is chosen large enough, the solution of the MPEC can be recast as the minimization of a single penalty function. The appropriate value of π is, however, unknown in advance and must be estimated during the course of the minimization.

This approach was first studied by Anitescu [20] in the context of active-set SQP methods, although it had been used before to solve engineering problems (see, e.g., [177]). It has been adopted as a heuristic to solve MPECs with interior methods in LOQO by Benson et al. [56], who present very good numerical results on the MacMPEC set. A more general class of exact penalty functions was analyzed by Hu and Ralph [257], who derive global convergence results for a sequence of penalty problems that are solved exactly. Anitescu [21] derives similar global results in the context of inexact subproblem solves.

In this section, we concentrate on the penalization approach. In Algorithm I, we provide a general interior-point penalty method for MPECs, defined by:

$$\begin{array}{ll} \text{minimize} & f(x) + \pi x_1^T x_2 \\ \text{subject to} & c_i(x) = 0, \quad i \in \mathcal{E} \\ & c_i(x) \geq 0, \quad i \in \mathcal{I} \\ & x_1 \geq 0, \; x_2 \geq 0, \end{array} \tag{12.9}$$

The following lemma shows that the penalty formulation inherits the desirable properties of the MPEC for a sufficiently large penalty parameter. The multipliers for the bound constraints $x_1 \geq 0, x_2 \geq 0$ of the penalty problem (12.9) will be denoted by $\nu_1 \geq 0, \nu_2 \geq 0$, respectively.

Lemma 12.3.1 *If Assumptions 12.3.1 hold at x^* and $\pi > \pi^*$, where*

$$\pi^* = \pi^*(x^*, \sigma_1^*, \sigma_2^*) = \max\left\{0, \max_{\{i: x_{1i}^* > 0\}} \frac{-\sigma_{2i}^*}{x_{1i}^*}, \max_{\{i: x_{2i}^* > 0\}} \frac{-\sigma_{1i}^*}{x_{2i}^*}\right\}, \tag{12.10}$$

then it follows that

1. *LICQ holds at x^* for (12.9).*
2. *x^* is a KKT point of (12.9).*
3. *Primal-dual strict complementarity holds at x^* for (12.9); that is, $\lambda_i^* \neq 0$ for all $i \in \mathcal{E} \cup \mathcal{A}_c(x^*)$ and $\nu_{ji}^* > 0$ for all $i \in \mathcal{A}_j(x^*)$, for $j = 1, 2$.*
4. *The second-order sufficiency condition holds at x^* for (12.9).*

We can now formulate the barrier problem associated to the penalized version of (12.9) is

$$\begin{array}{lrcl} \text{minimize} & f(x) + \pi x_1^T x_2 - \mu \displaystyle\sum_{i \in \mathcal{I}} \log s_i - \mu \sum_{i=1}^{p} \log x_{1i} - \mu \sum_{i=1}^{p} \log x_{2i} & & \\ \text{subject to} & c_i(x) & = & 0, \quad i \in \mathcal{E}, \\ & c_i(x) - s_i & = & 0, \quad i \in \mathcal{I}, \end{array} \tag{12.11}$$

where $\mu > 0$ is the barrier parameter and $s_i > 0, \; i \in \mathcal{I}$, are slack variables.

Under mild conditions, this algorithm can be shown to be both globally and locally superlinearly convergent.

Theorem 12.3.2 *Suppose that Algorithm* I *generates an infinite sequence of iterates $\{x^k, s^k, \lambda^k\}$ and parameters $\{\pi^k, \mu^k\}$ that satisfies conditions (12.12) and (12.13), for sequences $\{\epsilon_{pen}^k\}, \{\epsilon_{comp}^k\}, \{\mu^k\}$ converging to zero. If x^* is a limit point of the sequence $\{x^k\}$, and f and c are continuously differentiable in an open neighborhood $\mathcal{N}(x^*)$ of x^*, then x^* is feasible for the MPEC (12.1). If, in addition, MPEC-LICQ holds at x^*, then x^* is a C-stationary point of (12.1). Moreover, if $\pi_i^k x_{ji}^k \to 0$ for $j = 1, 2$ and $i \in \mathcal{A}_1(x^*) \cap \mathcal{A}_2(x^*)$, then x^* is a strongly stationary point of (12.1).*

Algorithm I: Interior-Penalty Method for MPECs

Initialization: Let x^0, s^0, λ^0 be the initial primal and dual variables. Set $k = 1$.

repeat

1. Choose a barrier parameter μ^k, stopping tolerances ϵ^k_{pen} and ϵ^k_{comp}
2. Find π^k and an approximate solution (x^k, s^k, λ^k) of problem (12.11) with parameters μ^k and π^k that satisfy $x^k_1 > 0, x^k_2 > 0, s^k > 0, \lambda^k_{\mathcal{I}} > 0$ and the following conditions:

$$\|\nabla_x \mathcal{L}_{\mu^k,\pi^k}(x^k, s^k, \lambda^k)\| \leq \epsilon^k_{pen}, \tag{12.12a}$$
$$\|S^k \lambda^k_{\mathcal{I}} - \mu^k e\| \leq \epsilon^k_{pen}, \tag{12.12b}$$
$$\|c(x^k, s^k)\| \leq \epsilon^k_{pen}, \tag{12.12c}$$

and

$$\|\min\{x^k_1, x^k_2\}\| \leq \epsilon^k_{comp} \tag{12.13}$$

3. Let $k \leftarrow k + 1$

until a stopping test for the MPEC is satisfied.

Figure 12.2: An interior-penalty method for MPECs.

The next result shows that this approach can be superlinearly convergent.

Theorem 12.3.3 *Suppose that Assumptions 12.3.1 hold at a strongly stationary point x^*. Assume that $\pi > \pi^*$, with π^* given by (12.10), and that the tolerances $\epsilon_{pen}, \epsilon_{comp}$ in Algorithm* I *are functions of μ that converge to 0 as $\mu \to 0$. Furthermore, assume that the barrier parameter and these tolerances are updated so that the following limits hold as $\mu \to 0$:*

$$\frac{(\epsilon_{pen} + \mu)^2}{\epsilon^+_{pen}} \to 0, \tag{12.14a}$$
$$\frac{(\epsilon_{pen} + \mu)^2}{\mu^+} \to 0, \tag{12.14b}$$
$$\frac{\mu^+}{\epsilon^+_{comp}} \to 0. \tag{12.14c}$$

Assume also that

$$\frac{\mu^+}{\|F_0(z;\pi)\|} \to 0 \ \ as \ \ \|F_0(z;\pi)\| \to 0. \tag{12.15}$$

Then, if μ is sufficiently small and z is sufficiently close to z^*, the following conditions hold:

1. *The stopping criteria (12.12) and (12.13), with parameters $\mu^+, \epsilon^+_{pen}, \epsilon^+_{comp}$ and π, are satisfied at z^+.*
2. $\|z^+ - z^*\| = o(\|z - z^*\|)$.

12.4 A Globally Convergent Methods: A Sequential LPEC-EQP Approach

Despite the success of NLP solvers in tackling a wide range of MPECs, there are still classes of problems where these solvers fail. In particular, problems whose stationary points are B-stationary, but not strongly-stationary, will cause NLP solvers to fail or exhibit slow convergence. Unfortunately, this behavior also occurs when other pathological situations occur, and it is not easily diagnosed of remedied. This observation motivates the development of more robust methods for MPECs.

One idea is to extend SQP methods by taking special care of the complementarity constraint. Researchers have suggested an SQPEC approach, where we minimize a quadratic approximation of the Lagrangian subject to a linearized feasible set, and a copy of the complementarity constraint. Remarkably, this approach can be shown to fail for MPECs as the following example shows:

$$\underset{x,y}{\text{minimize}}\ (x-1)^2+y^3+y^2 \quad \text{subject to } 0 \le x \ \perp\ y \ge 0. \tag{12.16}$$

The following proposition shows that SQPEC would fail for this problem.

Proposition 12.4.1 *Consider solving the MPEC (12.16) by applying SQPEC. Starting at $(x_0, y_0) = (0, t)$ for $0 < t < 1$, SQPEC generates the following sequence of iterates,*

$$(x^{(k+1)}, y^{(k+1)}) = \left(0, \frac{3y^{(k)^2}}{6y^{(k)}+2}\right),$$

which converges quadratically to the spurious M-stationary point $(0, 0)$.

Given this failure of SQPEC, how can we design robust MPEC solvers? The answer is to go back to the B-stationarity conditions, and interpret them as the optimality conditions of an LPEC. An algorithmic approach would then be to solve a sequence of LPECs inside a trust-region. In order to accelerate convergence, we could add an EQP phase, similar to the SLP-EQP method discussed in Section 10.3.2.

We start by defining the subproblems solved by our method and provide a rough outline of the SLPEC-EQP method. At each iteration, we solve an LPEC inside a trust region of radius $\Delta > 0$ around the current point x:

$$\text{LPEC}(x,\Delta)\begin{cases} \underset{d}{\text{minimize}} & g(x)^T d \\ \text{subject to} & c_{\mathcal{E}}(x) + A_{\mathcal{E}}(x)^T d = 0, \\ & c_{\mathcal{I}}(x) + A_{\mathcal{I}}(x)^T d \ge 0, \\ & 0 \le x_1 + d_{x_1} \ \perp\ x_2 + d_{x_2} \ge 0, \\ & \|d\| \le \Delta \end{cases}$$

where $g(x) = \nabla f(x)$, and $A(x) = \nabla c(x)$. Given a solution $d \neq 0$, we find the active sets that are predicted by the LPEC:

$$\mathcal{A}_c(x+d) := \left\{i : c_i(x) + a_i(x)^T d = 0\right\} \tag{12.17}$$
$$\mathcal{A}_1(x+d) := \left\{j : x_{1_j} + d_{x_j} = 0\right\} \tag{12.18}$$
$$\mathcal{A}_2(x+d) := \left\{j : x_{1_2} + d_{x_2} = 0\right\} \tag{12.19}$$

and solve the corresponding EQP:

$$\text{EQP}(x+d)\begin{cases} \underset{d}{\text{minimize}} & g(x)^T d + \frac{1}{2} d^T H(x) d \\ \text{subject to} & c_i(x) + a_i(x)^T d = 0, \quad \forall i \in \mathcal{A}_c(x+d) \\ & x_{1_j} + d_{1_j} = 0, \quad \forall j \in \mathcal{A}_1(x+d) \\ & x_{2_j} + d_{2_j} = 0, \quad \forall j \in \mathcal{A}_2(x+d) \end{cases}$$

We note that EQP$(x+d)$ can be solved as a linear system of equations. Global convergence is promoted through the use of a three-dimensional filter that separates the complementarity error and the nonlinear infeasibility. A conceptual outline of our proposed algorithm is given below.

Outline of SLPEC-EQP Algorithm

Given an initial point $x^{(0)}$, set $k = 0$, and $\Delta_0 > 0$.

while $d \neq 0$ **do**
  Solve LPEC$(x^{(k)}, \Delta_k)$ for step $d^{(k)}$
  Identify the active sets $\mathcal{A}_c(x^{(k)} + d^{(k)})$, $\mathcal{A}_1(x^{(k)} + d^{(k)})$, and $\mathcal{A}_2(x^{(k)} + d^{(k)})$.
  Solve EQP$(x+d)$ for second-order step d_{qp}.
  if $x^{(k)} + d_{qp}$ *acceptable step* **then**
    Set $x^{(k+1)} := x^{(k)} + d_{qp}$, and possibly increase $\Delta_{k+1} = 2\Delta_k$.
  else
    Set $x^{(k+1)} := x^{(k)}$, and decrease $\Delta_{k+1} = \Delta_k$ / 2.
  end
end

The algorithm outlined above leaves a number of important open questions: How should the LPEC be solved? What constitutes acceptance of a step? Most importantly, what happens if the LPEC or the EQP has no solution? In a practical implementation we might also restrict the EQP step by a trust region or a proximal-point term, and we could use the SLPEC step if the EQP step fails, or we could consider a piecewise line-search along an arc.

Our SLPEC-EQP method has one important advantage over the recent NLP approaches. The solution of the LPEC matches exactly the definition of B-stationarity (see Definition 12.2.2), and we therefore always work with the correct tangent cone. In particular, if $d = 0$ solves the LPEC for some $\Delta > 0$, then we can conclude that the current point is B-stationary. To our knowledge, this is the only algorithm that guarantees global convergence to B-stationary points.

12.5 Exercises

12.1. Reformulate the following complementarity conditions into the form used in (12.1):

$$\begin{array}{ll} \text{(a)} & l \leq F(x) \leq u \ \perp \ y \\ \text{(b)} & l \leq F(x) \ \perp \ y \geq l \\ \text{(a)} & l \leq x \leq u \ \perp \ F(y) \end{array}$$

where x, y are variables, $F()$ is a smooth function, and l, u are bounds.

12.2. Consider the two MPEC

$$\left\{ \begin{array}{ll} \underset{x}{\text{minimize}} & f_i(x) \\ \text{subject to} & 0 \leq x_2 \ \perp \ x_2 - x_1 \geq 0 \end{array} \right. \tag{12.20}$$

with $f_1(x) = (x_1 - 1)^2 + x_2^2$ and $f_2(x) = x_1^2 + (x_2 - 1)^2$. Show that the solution to both problems is $x^* = (1/2, 1/2)^T$, and draw the two sets of unbounded multipliers for the equivalent NLP.

12.3. Consider the MPEC

$$\begin{cases} \underset{x}{\text{minimize}} & x_1 + x_2 \\ \text{subject to} & x_2^2 \geq 1 \\ & 0 \leq x_1 \ \perp \ x_2 \geq 0. \end{cases} \tag{12.21}$$

Its solution is $x^* = (0,1)^T$ with NLP multipliers $\lambda^* = 0.5$ of $x_2^2 \geq 1$, $\nu_1^* = 1$ of $x_1 \geq 0$, and $\xi^* = 0$ of $x_1 x_2 \leq 0$. In particular, this solution is a strongly stationary point (see Definition 12.2.3). However, linearizing the constraints about a point that satisfies the simple bounds and is *arbitrarily close to the solution*, such as $x^{(0)} = (\epsilon, 1-\delta)^T$ (with $\epsilon, \delta > 0$), gives a QP that is *inconsistent*.

12.4. Formulation of MOOPs as MPECs.

Part IV

Mixed-Integer Nonlinear Optimization

Chapter 13

Introduction and Modeling with Integer Variables

We start our discussion of mixed-integer problems by presenting some modeling examples that involve integer variables; discuss some applications of mixed-integer optimization; and present modeling best-practices. Finally, we discuss the basic algorithmic ingredients: relaxation and separation.

13.1 Mixed-Integer Nonlinear Programming 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) 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 modeling paradigms in optimization and includes both nonlinear programming (NLP) and mixed-integer linear programming (MILP) as subproblems. MINLPs are conveniently expressed as

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X, \\ & x_i \in \mathbb{Z},\ \forall i \in I, \end{cases} \tag{13.1}$$

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 modeled by using so-called special-ordered sets of type I [43, 44].

Problem (13.1) is an NP-hard combinatorial problem, because it includes MILP [267], and its solution typically requires searching enormous search trees; see Figure 13.1. Worse, nonconvex integer optimization problems are in general undecidable [264]. 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 (13.1) is decidable, which we can achieve either by ensuring that X is compact or by assuming that the problem functions are convex.

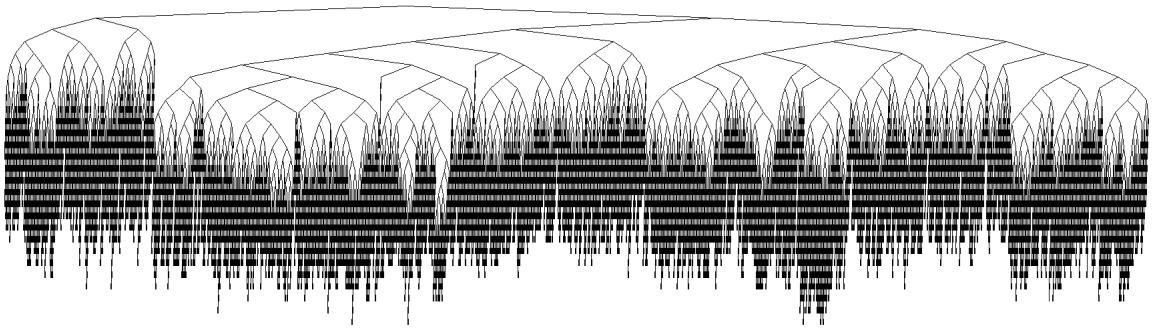


Figure 13.1: Branch-and-bound tree without presolve after 360 s CPU time has more than 10,000 nodes.

13.1.1 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 Hessian at $x^{(k)}$; for example, $\nabla f^{(k)} = \nabla f(x^{(k)})$. We use subscripts to denote components; for example, x_i is component i of x. For a set $J \subset \{1, \ldots, n\}$ we let x_J denote the components 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 (13.1) is not convex. In a slight abuse of terminology, we distinguish convex from nonconvex MINLPs.

Definition 13.1.1 *We say that (13.1) 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 nonconvex function, then we say that (13.1) is a* nonconvex MINLP.

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

Definition 13.1.2 *Given a set S, the* convex hull of S *is denoted by conv(S) and defined as*

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

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

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

The solution of the continuous relaxation is $x = (\frac{1}{2} \ldots \frac{1}{2})$, which is not an extreme point of the feasible set and, in fact, lies in the strict interior of the MINLP; see Figure 13.2. 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 (13.1) by introducing an objective variable z and a constraint $z \geq f(x)$. We obtain the following equivalent MINLP:

$$\begin{cases} \underset{z,x}{\text{minimize}} & z, \\ \text{subject to} & f(x) \leq z, \\ & c(x) \leq 0, \\ & x \in X, \\ & x_i \in \mathbb{Z},\ \forall i \in I. \end{cases} \tag{13.2}$$

The optimal solution of (13.2) always lies on the boundary of the convex hull of the feasible set and therefore allows us to use cutting-plane techniques.

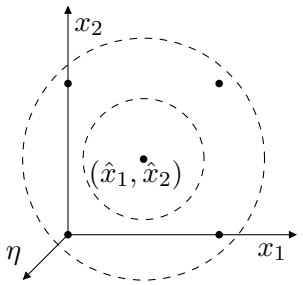


Figure 13.2: Small MINLP to illustrate the need for a linear objective function.

13.1.2 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 (13.1). 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 to 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.

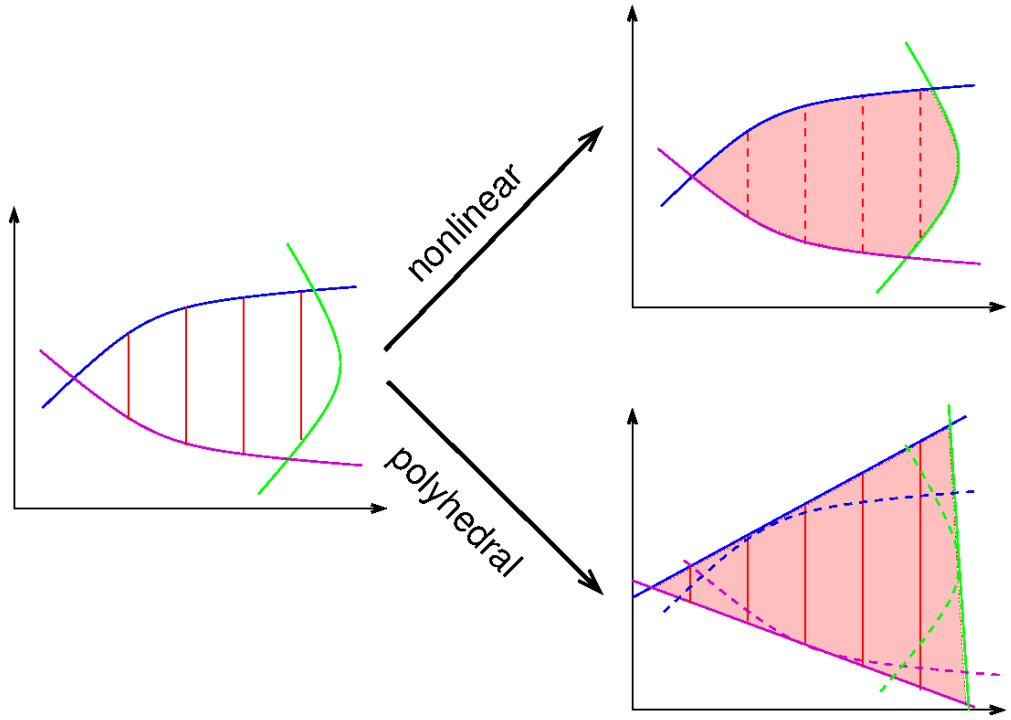


Figure 13.3: Illustration of the two classes of relaxation. The left image shows the mixed-integer feasible set, the top right image shows the nonlinear relaxation, and the bottom right shows the polyhedral relaxation.

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 14.2) and is given by

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X. \end{cases} \tag{13.3}$$

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,

$$z \geq f^{(k)} + \nabla f^{(k)^T}(x - x^{(k)}), \tag{13.4}$$
$$0 \geq c^{(k)} + \nabla c^{(k)^T}(x - x^{(k)}), \tag{13.5}$$

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 15.1.1.

3. *Relaxing nonconvex constraints.* Constraints $c(x) \leq 0$ and $f(x) \leq z$ containing nonconvex 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

$$\breve{f}(x) \leq f(x) \quad \text{and} \quad \breve{c}(x) \leq c(x), \ \forall x \in \text{conv}(X). \tag{13.6}$$

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 17.1 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 (13.2), 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 13.3 illustrates the relaxation of integrality constraints and convex nonlinear constraints. The left image shows the mixed-integer feasible set (the union of the red vertical segments), the top right image shows the nonlinear relaxation obtained by relaxing the integrality constraints (the shaded area is the NLP feasible set), and the bottom right figure shows a polyhedral relaxation (the union of the red vertical segments) as well as its LP relaxation (the shaded area). We note that an infinite number of 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 (13.2), then it also solves the MINLP. In general, however, the solution is not feasible in (13.2), and we must somehow exclude this solution from the relaxation.

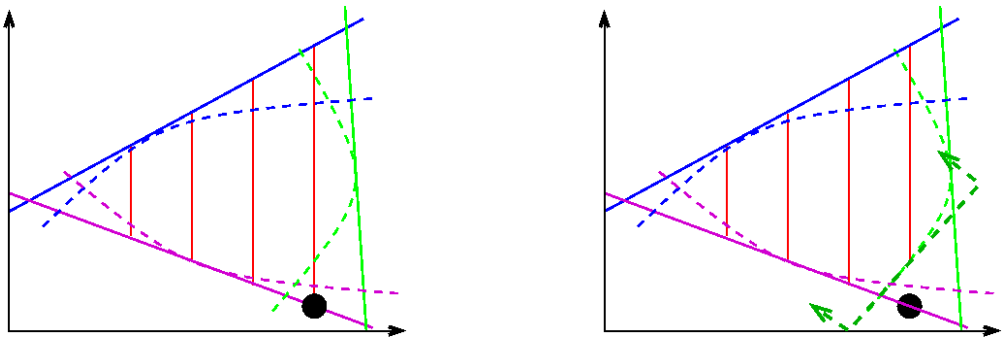


Figure 13.4: Separation of infeasible point (black dot) by adding a separating hyperplane. The dashed green line on the right shows a separating hyperplane with arrows indicating the feasible side.

Constraint enforcement. Given a point $\hat{x}$ that is feasible to a relaxation but is not feasible to 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 an infeasible relaxation solution $\hat{x}$ is no longer feasible. 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 to 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 13.4. This class of separation is used in outer approximation, Benders decomposition, and the ECP algorithm discussed in Section 15.1.1. 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 integer infeasible solution $\hat{x}$. This technique has been critical to the success of algorithms for solving mixed-integer *linear* programs.

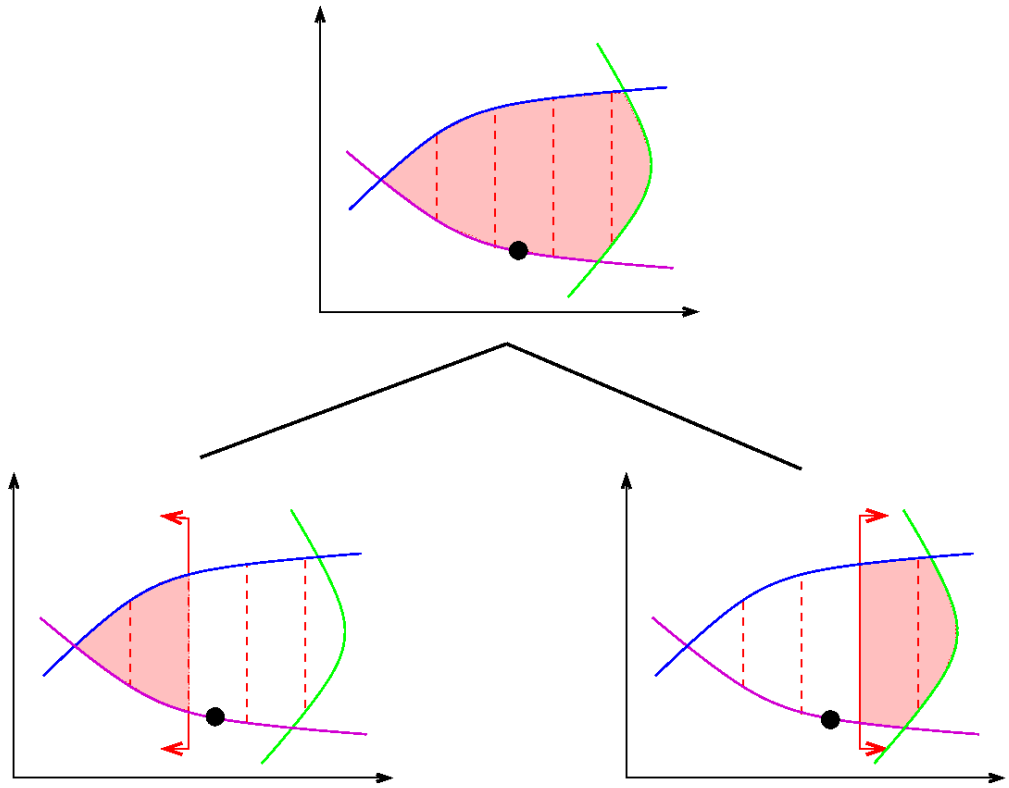


Figure 13.5: Branching on the values of an integer variable creates two new nonlinear subproblems that both exclude the infeasible point, denoted with the black dot.

The second class of constraint enforcement strategy is branching. Branching consists of 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 13.5. 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 14.2.

Constraint enforcement for relaxed nonconvex constraints involves a combination of branching and relaxation refinement. These techniques are discussed in detail in Section 17.1, but here we outline the general idea using Figure 13.6. Following the solution of the relaxation (given by the green objective on the left), we branch on a *continuous* variable and hence split its domain into two subdomains. We then compute new underestimators for use in (13.6) 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 green objective functions

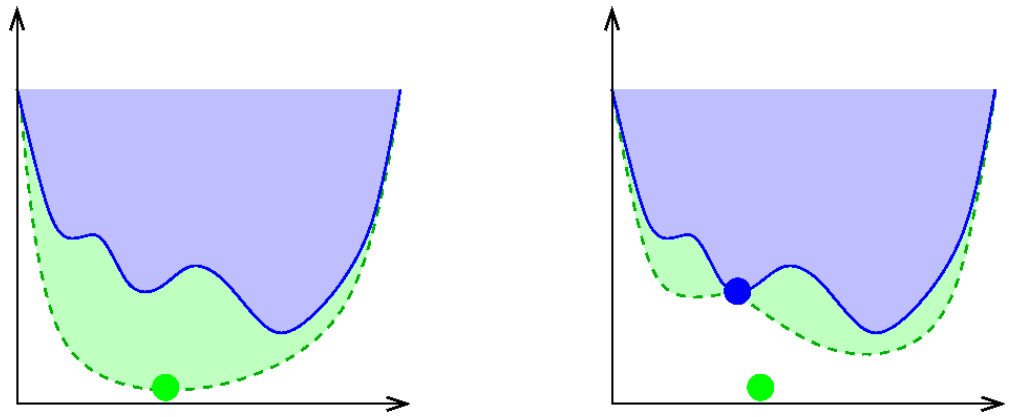


Figure 13.6: Constraint enforcement by using spatial branching for global optimization.

on the right. 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 nonconvex constraints, branching may be required on both integer and continuous decision variables.

13.1.3 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 19.3 we motivate our interest in MINLP methods by presenting some small examples, and we briefly discuss good modeling practices. In Section 14.1 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 16.1 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 17.1 we outline methods for solving *nonconvex* MINLPs. A range of heuristics to obtain good incumbent solutions quickly is discussed in Section 18.1. We review two classes of deterministic heuristics: search and improvement heuristics. In Chapter 19 we present an emerging extension of MINLP, namely mixed-integer PDE constraind optimization problem. In Section A.1 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 [183, 343, 348, 447]. 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 [258] solves integer linear programming problems in polynomial time when the number of variables in the problem is fixed. Khachiyan and Porkolab [273] extended this algorithm to integer programs with convex polynomial objective and constraints. Generalizations and improvements in the complexity bound have been made in [139, 150, 243, 251]. Using ideas from Graver bases, Hemmecke et al. [247] derive an algorithm that requires a polyno-

mial number of *augmentation steps* to solve specially structured convex minimization problems over integer sets. Baes et al. [28] 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 [153] and Dadush et al. [136, 137, 138] 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 Moran R. [152] 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 modeled through simulation.Other applications include the design of nanophotonic devices [323, 331] and the determination of loop unroll factors in empirical performance tuning [30]. Algorithms for derivative-free mixed-integer nonlinear optimization employ either global surrogates [149, 245, 246, 339, 340, 371], or mesh- and pattern-based techniques [3, 26, 197, 311].

We mention a number of surveys and monographs on MINLP, including a review of MINLP and disjunctive programming [228], a survey of MINLP algorithms [229], a survey of algorithms and software for MINLP [89], a comprehensive textbook on MINLP by Floudas [192], and a collection of articles related to a recent IMA workshop on MINLP [293].

13.2 Nonlinear Models with Integer Variables

In this section, we review a small number of models and modeling 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 modeling 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 [281, 384], which can be formulated as a convex MINLP. We then present a nonconvex MINLP that arises in the design of water distribution networks [94, 95, 166, 398, 401]. Finally, we present an time- and energy-optimal subway control example [81] that adds time-dependent integer variables and constraints to MINLP.

13.2.1 Modeling Practices for MINLP

Modeling plays a fundamental role in MILP (see the textbook by Williams [443]) and is arguably more critical in MINLP, because of the additional nonlinear relationships. The nonlinearities often allow for equivalent formulations with more favorable properties. For example, in Section 13.2.2 we present a model of a multiproduct batch plant and show that by using a nonlinear transformation, we are able to reformulate

the nonconvex problem as an equivalent convex problem, which is typically easier to solve. Here, we briefly review a number of other modeling 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

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

where the second quadratic is now convex.

Exploiting low-rank Hessians. In many applications such as model structure determination and parameter estimation problems [404], 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 17.1.1), and these can be effective for expressions that involve a small number of variables.

Linearization of $x_1 x_2$, for $x_2 \in \{0,1\}$. We can linearize the expression $x_1 x_2$, when $0 \leq x_1 \leq u$ and $x_2 \in \{0,1\}$ by observing that the product $x_1 x_2$ 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 nonconvex term $x_1 x_2$ 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

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

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 reduced the "degree of nonlinearity" of the resulting constraint set.

A common theme of these reformulation is that convex formulations are typically preferred over nonconvex ones, and linear formulations over nonlinear formulations. More useful modeling tricks for integer and binary variables are given in the classic textbook by Williams [443].

13.2.2 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 $V_j \geq S_{ij} B_i \; \forall i, j$;
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 $C_i \geq t_{ij}/N_j \; \forall i, j$.

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 nonconvex MINLP:

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

Unfortunately, this model is a nonconvex MINLP, because the objective function, the horizon-time constraint, and the constraint defining the longest stage time are nonconvex 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

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

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

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

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 modeled using a special ordered set of

type I (SOS-1) [43]. By introducing binary variables $y_{kj} \in \{0,1\}$ we can replace the last constraint in the convex model by,

$$\sum_{k=1}^{K} \ln(k) y_{kj} = n_j \quad \text{and} \quad \sum_{k=1}^{K} y_{kj} = 1, \quad \forall j \in M. \tag{13.1}$$

The resulting model is available as `batch` on MacMINLP [300]. Similar reformulations have been proposed using $\sqrt{x}$-transforms for models involving bilinear terms [239].

13.2.3 Design of Water Distribution Networks

Many MINLPs model flows on networks. Examples include the optimal design of water networks [94, 95, 105] the design of gas-networks [446] and to the optimal regulation of ventilation systems for mines [321, 450]. 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, $i \in cN$, 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 nonconvex 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: pipe diameters available to network, $k = 1, \ldots, r$;
H_s: the hydraulic head at source node $s \in \mathcal{S}$; and
H_l, H_u: are 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: hydraulic head at node $i \in \mathcal{N}$, where $h_s = H_s, \forall s \in \mathcal{S}$, and $H_l \leq h_i \leq H_u, \forall i \in \mathcal{N} - \mathcal{S}$;
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 (13.1) 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

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

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 depend as a linear set of constraints,

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

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

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

The MINLP solvers can now ether branch on individual y_{ijk} or on SOS-1 sets $(y_{ij1}, \ldots, y_{ijr})$, which is generally more efficient. See Williams [443] 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,

$$\sum_{k=1}^{r} (\pi P_k/4) y_{ijk} = a_{ij}, \; \forall (i,j) \in \mathcal{A}.$$

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

$$h_i - h_j = \frac{\operatorname{sgn}(q_{ij}) |q_{ij}|^{c_1} c_2 L_{ij} K_{ij}^{-c_1}}{d_{ij}^{c_3}}, \; \forall (i,j) \in \mathcal{A}, \tag{13.2}$$

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 nonsmooth 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 part by adding binary variables $z_{ij} \in \{0,1\}$ and the following set of constraints,

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

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 (13.2) equivalently as

$$h_i - h_j = \frac{\left[\left(q_{ij}^{+}\right)^{c_1} - \left(q_{ij}^{-}\right)^{c_1} \right] c_2 L_{ij} K_{ij}^{-c_1}}{d_{ij}^{c_3}}, \; \forall (i,j) \in \mathcal{A}.$$

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 behavior of the underlying NLP solvers.

13.2.4 A Dynamic Subway Operation Problem

As an example for a dynamic nonlinear mixed-integer problem, we present a control problem that goes back to work by Bock and Longman [81] optimizing the subway of the city of New York. The problem has been treated by, for example, Sager [379]. We are interested in minimizing the total energy consumption

$$\underset{x(\cdot),u(\cdot),v(\cdot),T}{\text{minimize}} \int_0^T L(x(t), v(t)) \, \mathrm{d}t \tag{13.3}$$

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

$$\dot{x}_{\mathrm{s}}(t) = x_{\mathrm{v}}(t) \qquad t \in [0,T], \tag{13.4}$$

$$\dot{x}_{\mathrm{v}}(t) = f_{\mathrm{v}}(x(t), u(t), v(t)) \quad t \in [0,T], \tag{13.5}$$

on a horizon $[0,T] \subset \mathbb{R}$ with free end time T. The continuous control $0 \leq u(t) \leq u_{\max}$ 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, namely

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

$$L(x(t),1) = \begin{cases} e\, p_1 & \text{if } x_{\mathrm{v}}(t) \leq v_1, \\ e\, p_2 & \text{if } v_1 < x_{\mathrm{v}}(t) \leq v_2, \\ e \sum\limits_{i=0}^{5} c_i(1)\,(0.1\gamma x_{\mathrm{v}}(t))^{-i} & \text{if } v_2 < x_{\mathrm{v}}(t), \end{cases} \tag{13.6}$$

   and the subway train's dynamics are described by

$$W_{\text{eff}}\, f_{\mathrm{v}}(t) = \begin{cases} g\, e\, a_1 & \text{if } x_{\mathrm{v}}(t) \leq v_1, \\ g\, e\, a_2 & \text{if } v_1 < x_{\mathrm{v}}(t) \leq v_2, \\ g\,(e\, F(x_{\mathrm{v}}(t),1) - R(x_{\mathrm{v}}(t))) & \text{if } v_2 < x_{\mathrm{v}}(t). \end{cases} \tag{13.7}$$

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

$$L(x(t),2) = \begin{cases} 0 & \text{if } x_{\mathrm{v}}(t) \leq v_2, \\ e\, p_3 & \text{if } v_2 < x_{\mathrm{v}}(t) \leq v_3, \\ e \sum\limits_{i=0}^{5} c_i(2)\,(0.1\gamma x_{\mathrm{v}}(t) - 1)^{-i} & \text{if } v_3 < x_{\mathrm{v}}(t), \end{cases} \tag{13.8}$$

   and the subway train's dynamics are described by

$$W_{\text{eff}}\, f_{\mathrm{v}}(t) = \begin{cases} 0 & \text{if } x_{\mathrm{v}}(t) \leq v_2, \\ g\, e\, a_3 & \text{if } v_2 < x_{\mathrm{v}}(t) \leq v_3, \\ g\,(e\, F(x_{\mathrm{v}}(t),2) - R(x_{\mathrm{v}}(t))) & \text{if } v_3 < x_{\mathrm{v}}(t). \end{cases} \tag{13.9}$$

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

$$W_{\text{eff}}\, f_{\mathrm{v}}(t) = -g\, R(x_{\mathrm{v}}(t)) - C\, W_{\text{eff}} \tag{13.10}$$

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

$$f_{\mathrm{v}}(t) = -u(t). \tag{13.11}$$

The forces occurring in these dynamics are given by

$$R(x_{\mathrm{v}}(t)) = ca\gamma^2 x_{\mathrm{v}}(t)^2 + bW\gamma x_{\mathrm{v}}(t) + \frac{1.3}{2000}W + 116, \tag{13.12}$$

$$F(x_{\mathrm{v}},1) = \sum_{i=0}^{5} b_i(1)\,(0.1\gamma x_{\mathrm{v}}(t) - 0.3)^{-i}, \tag{13.13}$$

$$F(x_{\mathrm{v}},2) = \sum_{i=0}^{5} b_i(2)\,(0.1\gamma x_{\mathrm{v}}(t) - 1)^{-i}. \tag{13.14}$$

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

$$x(0) = (0,0)^T, \quad x(T) = (S,0)^T. \tag{13.15}$$

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

$$T \leq T_{\max}. \tag{13.16}$$

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, for example, Bock and Longman [81] or Sager [379]. 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 modeled appropriately using, for example, additional integer variables, and higher-order polynomial approximations to nonlinear and possibly nonconvex system characteristics.

Classes of MINLP Problem. MINLP is a very general and broad modeling paradigm. Recently, subclasses of MINLP problems have emerged. These classes of problems often arise in important applications and exhibit structural commonalities that can be exploited in the development of algorithms. MINLPs can be subdivided further into subproblem classes.

13.2.5 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 MINLP include the efficient management of electricity transmission [27, 336], transmission expansion [204, 374], transmission switching [40, 241], and contingency analysis, and blackout prevention of electric power systems [72, 157].

MINLP also arise in many chemical engineering applications, such as the design of water [94, 268] and gas [325] distribution networks, the minimization of the environmental impact of utility plants [168], the integrated design and control of chemical processes [191], and block layout design in the manufacturing and service sectors [113], in the optimization of polymerization plants [362]. A related area is systems biology, where topics such as circadian rythms [393], and protein folding [280] are addressed using mixed-integer optimization.

Applications in communication engineering and computer science include the optimal response to a cyber attack [17, 214], wireless bandwidth allocation [68, 131, 395], selective filtering [403, 407], optical network performance optimization [171], network design with queuing delay constraints [90], and network design topology [64, 118], multi-vehicle swarm communication network optimization [2], the design of optimal paths (i.e. minimum time) for robotic arms [206], the synthesis of periodic waveforms by tripolar pulse codes [112], and the solution of MILP under uncertainty of the parameters through robust optimization [54].

Other engineering applications include the operational reloading of nuclear reactors [367], the optimal response to catastrophic oil spills such as the recent Deepwater oil spill in the Gulf of Mexico [454, 455], the design of load-bearing thermal insulation system for the Large Hadron Collider [1, 4], concrete structure design [230], and the design and operation of drinking water networks [105], gas networks [325], electric distribution networks [284], and mining networks [363]. Applications in traffic modeling and optimization are found in [200], and [408] considers a flight path optimization problem. An important financial application of MINLP arises in portfolio optimization [71, 265].

MINLP models can also be stochastic service system design problems [167], since performance metrics are often nonlinear in the decision variables.

Another developing applications area of both game theory and optimization is resource allocation for homeland security (see, e.g., Bier [73], Sandler and Arce M [382], Zhuang and Bier [461]). Simple versions of these models (involving, for example, a single target and attacker) have closed-form optimal (equilibrium) solutions (e.g., Powell [361], Sandler and Siqueira [383], Zhuang and Bier [462]). 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 [74, 75] 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 [460, 463]). 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 [172].

A special domain of MINLP are dynamic problems constrained by ordinary differential equations (ODEs) or differential-algebraic equations (DAE), often called mixed-integer optimal control problems (MIOCPs). One of the earliest practical problems was the optimal switched operation of the New York subway [81]. Recent applications include gear shifts in automotive control [209, 278], automated cruise controllers [244, 276, 422], superstructure detection in simulated moving bed processes [380], the optimization of batch distillation processes [351].

Chapter 14

Branch-and-Bound Methods

We introduce branch-and-bound methods for mixed-integer nonlinear optimization and discuss advanced algorithmic features.

14.1 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 (13.1) exist: single-tree methods and multitree methods. In this section we concentrate on methods for convex objective functions and convex constraints. See Section 17.1 for a discussion of methods for nonconvex MINLPs. Throughout this section, we make the following assumptions.

Assumption 14.1.1 *Consider Problem (13.1) 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 (13.1) satisfies a constraint qualification for every point in the convex hull of the feasible set of (13.1).*

The most restrictive assumption is the convexity assumption A2. Assumption A1 is rather mild, and A3 is technical. We do not specify the 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 (13.1), then A3 follows from A2 as a consequence of Slater's constraint qualification [227], 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.

14.2 Nonlinear Branch-and-Bound

The nonlinear branch-and-bound method for MINLPs dates back to Dakin [140]; see also [234]. The algorithm starts by solving the NLP relaxation of (13.1), 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 14.1.

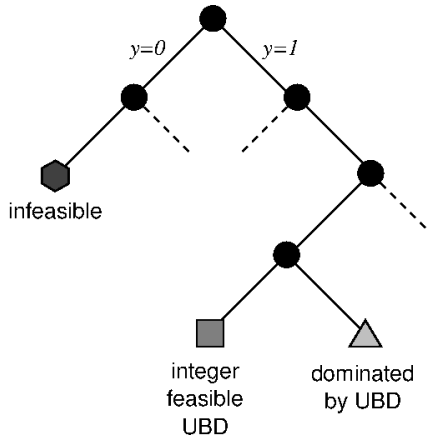


Figure 14.1: Illustration of a nonlinear branch-and-bound algorithm that traverses the tree by solving NLPs at every node of the tree.

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

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

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 nonintegral 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

$$u_i^- := \lfloor x'_i \rfloor, \ \text{ and } \ l_i^+ := \lceil x'_i \rceil. \tag{14.1}$$

The two 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 (13.1) (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 red circle in Figure 14.1).
- **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 14.1, and Proposition 14.2.1 establishes its convergence.

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: $\mathcal{H} = \mathcal{H} - \{$ 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 29
  end
end

Algorithm 14.1: Branch-and-bound for MINLP.

Proposition 14.2.1 *Consider solving (13.1) 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 (13.1) has no solution.*

Proof. The assumptions in Proposition 14.1.1 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 [391]. □

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!

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 14.2: Branch on a fractional variable.

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 [7] for the MILP case, and in [88] 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.

14.2.1 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 [443]). A simple branching rule is to select the variable with the largest integer violation for branching, in other words, choose

$$\underset{i \in I_c}{\operatorname{argmax}} \left\{ \min \left(x_i - \lfloor x_i \rfloor \ , \ \lceil x_i \rceil - x_i \right) \right\},$$

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 [7].

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:

$$\underset{i \in I_c}{\operatorname{argmax}} \left\{ s_i \right\}.$$

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 (14.1) 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 resolve 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

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

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 15.2.1), 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 [88].

Pseudocosts 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:

$$p_i^+ = \frac{f_i^+ - f}{\lceil x_i \rceil - x_i} + p_i^+, \; n_i^+ = n_i^+ + 1 \text{ or } p_i^- = \frac{f_i^- - f}{x_i - \lfloor x_i \rfloor} + p_i^-, \; n_i^- = n_i^- + 1. \tag{14.2}$$

The pseudocosts are then used to estimate D_i^+ and D_i^- whenever we need to make a branching decision as

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

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 [309]. 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 [183, 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 [43]. A more general approach is to branch on split disjunctions of the form

$$\left(a^T x_I \leq b\right) \vee \left(a^T x_I \leq b + 1\right), \tag{14.3}$$

where $a \in \mathbb{Z}^p$ and $b \in \mathbb{Z}$. Split disjunctions have been shown to produce promising results for MILP [265] but have not been used in MINLP.

14.2.2 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 [140]. 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 reuse 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 [88, 319], they generally have not been as successful in MINLP, mainly because the Jacobian and Hessian matrix 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 [285, 292].

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 $\max_p \left\{\min\left(b_p^+, b_p^-\right)\right\}$.

2. *Best estimate* chooses the node that is expected to contain the best expected integer solution within its subtree based on pseudo costs. 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 [223].

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 [5, 164, 309].

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.

14.2.3 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 [91]. 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 nonintegral solution. If convergence to nonintegral 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 to occasionally solve 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 [299]. An alternative approach is presented by Mahajan et al. [319]. 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 reusing 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 14.1 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 18.1.

14.2.4 Cutting planes for nonlinear branch-and-bound

Branch-and-bound algorithms can be enhanced by adding cutting planes, as described by Stubbs and Mehrotra [413] for convex mixed 0-1 nonlinear programs, based on Balas et al. [32] for MILP. The branch-and-cut algorithm extends the branch-and-bound Algorithm 14.1 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 14.3.

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: $\mathcal{H} = \mathcal{H} - \{$ 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 31.
    end
  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 29
  end
end

Algorithm 14.3: Branch-and-cut for MINLP.

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 satisfying $\pi^T x^{(l,u)} > \pi_0$. Various approaches to this end are discussed in Section 16.1.

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 14.4: Generate a subgradient cut by solving a separation problem.

Like branch-and-bound, a number of implementation issues are open. In their original work Stubbs and Mehrotra [413] 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.

14.3 Tutorial

Model building with integer variables: extensions to power-grid problem (transmission switching and network expansion); implementation of outer approximation in JuMP.

Chapter 15

Hybrid Methods

We discuss outer approximation and Benders decomposition approaches to MINLPs. These methods then allow us to define hybrid methods for mixed-integer optimization. We discuss presolve techniques for mixed-integer optimization, and investigate the Worst-case behavior of outer approximation, and consider disaggregation tricks tat mitigate the effect of these examples.

15.1 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 reuse 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 [86, 163, 185], generalized Benders decomposition [207, 430], and the extended cutting-plane method [441].

15.1.1 Outer approximation

We start by defining the NLP subproblem obtained by fixing the integer variables to $x_I^{(j)}$ in (13.1),

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

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

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

where $w_i > 0$ is a set of weights that can be chosen reduce to the 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, [185, 189, 222]. An optimal solution of (F($x_I^{(j)}$)) with a positive objective is a certificate that the corresponding NLP (NLP($x_I^{(j)}$)) is infeasible.

Next, we consider (13.2) and observe that the convexity of f and c implies that the linearization about the solution $x^{(j)}$ of (NLP($x_I^{(j)}$)) or (F($x_I^{(j)}$)), given by

$$\eta \geq f^{(j)} + \nabla f^{(j)^T}(x - x^{(j)}) \quad \text{and} \quad 0 \geq c^{(j)} + \nabla c^{(j)^T}(x - x^{(j)}), \tag{15.1}$$

is an outer approximation of the feasible set of (13.2). We can show that if (NLP($x_I^{(j)}$)) is infeasible, then the corresponding set of outer approximations ensures that $x_I = x_I^{(j)}$ violates (15.1).

Lemma 15.1.1 (Lemma 1 in [185]) *If (NLP($x_I^{(j)}$)) is infeasible and $x^{(j)}$ is an optimal solution of (F($x_I^{(j)}$)), then $x_I = x_I^{(j)}$ violates (15.1) for all $x \in X$.*

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

$$\mathcal{X} := \left\{ x^{(j)} \in X : x^{(j)} \text{ solves (NLP}(x_I^{(j)})) \text{ or (F}(x_I^{(j)})) \right\}. \tag{15.2}$$

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 (13.2):

$$\begin{cases} \underset{\eta,x}{\text{minimize}} & \eta, \\ \text{subject to} & \eta \geq f^{(j)} + \nabla f^{(j)^T}(x - x^{(j)}),\ \forall x^{(j)} \in \mathcal{X} \\ & 0 \geq c^{(j)} + \nabla c^{(j)^T}(x - x^{(j)}),\ \forall x^{(j)} \in \mathcal{X} \\ & x \in X, \\ & x_i \in \mathbb{Z},\ \forall i \in I. \end{cases} \tag{15.3}$$

We can show that (15.3) and (13.2) have the same optimal value and that any solution of (13.2) is an optimal solution of (15.3). However, the converse is not true, as the following example from Bonami et al. [86] shows:

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

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 15.1.1 *Assume that the assumptions in Proposition 14.1.1 hold, and let x^* solve (13.1). Then it follows that x^* also solves (15.3). Conversely, if (η^*, x^*) solves (15.3), then it follows that the optimal value of (13.1) is η^* and x_I^* is an optimal solution in both problems.*

Proof. The proof follows from the results by Bonami et al. [86] and Fletcher and Leyffer [185]. □

Of course, it is not practical to solve (15.3) because by the time it is set up, we already know the solution of (13.1). Instead, Duran and Grossmann [163] 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 (15.3) 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:

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

We note, however, that this latter constraint is not enforceable in practice, and is typically replaced by $\eta \leq U^k - \epsilon$, where $\epsilon > 0$ is a small tolerance. This upper bound ensures that once we have solved

(NLP(l,u)) and added its outer approximations (15.1) to ($M(\mathcal{X}^k)$), $x_i^{(j)}$ is not feasible in ($M(\mathcal{X}^k)$) for $k \geq j$, ensuring that outer approximation terminates finitely. Thus, the MILP master problem solved at iteration k is given by

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

A description of the outer approximation algorithm is given in Algorithm 15.1, and its convergence result is stated in Theorem 15.1.2.

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 &* $f^{(j)} < U^{k-1}$ **then**
    Update current best point: $x^* = x^{(j)}$ and $U^k = f^{(j)}$.
  else
    Set $U^k = U^{k-1}$.
  end
  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 15.1: Outer approximation.

The algorithm also detects whether (13.2) is infeasible. If $U^k = \infty$ on exit, then all integer assignments visited by the algorithm are infeasible, and hence (13.2) in 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 15.1.2 *If Assumptions 14.1.1 hold and if the number of integer points in X is finite, then Algorithm 15.1 terminates in a finite number of steps at an optimal solution of (13.1) or with an indication that (13.1) is infeasible.*

A proof of Theorem 15.1.2 was given by Fletcher and Leyffer [185]. The main argument of the proof is that the optimality of $x^{(j)}$ in (NLP($x_I^{(j)}$)) implies that $\eta \geq f^{(j)}$ for any feasible point in ($M(\mathcal{X}^k)$). The upper bound $\eta \leq f^{(j)} - \epsilon$ therefore ensures that the choice $x_I = x_I^{(j)}$ in ($M(\mathcal{X}^k)$) in 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.

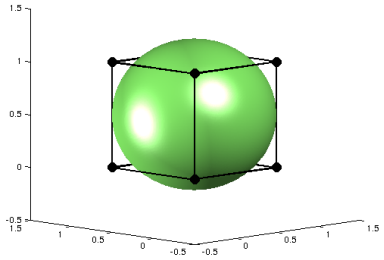


Figure 15.1: 3D plot of worst-case example for outer approximation (15.4).

Worst-case complexity of outer approximation. In practice, outer approximation often works efficiently. However, Hijazi et al. [250] provide an example where outer approximation takes an exponential number of iterations. In particular, they consider the following MINLP:

$$\underset{x}{\text{minimize}}\ 0 \quad \text{subject to} \sum_{i=1}^{n} \left(x_i - \frac{1}{2} \right)^2 \leq \frac{n-1}{4}\ x \in \{0,1\}^n. \tag{15.4}$$

Geometrically, the problem corresponds to a feasibility problem that seeks a point that lies in the intersecting the n-dimensional ℓ_2 ball with radius $\frac{n-1}{4}$ centered at $\hat{x} = (\frac{1}{2}, \ldots, \frac{1}{2})$ 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 15.1 shows a 3D plot of this example. The black lines illustrate the unit hypercube, and the green surface is the boundary of the nonlinear constraint in (15.4). The authors show that any outer approximation cut cannot 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.

15.1.2 Generalized Benders decomposition

Generalized Benders decomposition was developed before outer approximation; see [207]. Given outer approximation, however, it is straightforward to develop the Benders cuts and present Benders decomposition. We start by considering the outer approximations (15.1) and assume that the constraints X are inactive. Summing up the outer approximations (15.1) weighted with $(1, \lambda^{(j)})$, where $\lambda^{(j)} \geq 0$ are the multipliers of (NLP($x_I^{(j)}$)), we obtain the following valid inequality:

$$\eta \geq \left(f^{(j)} + \lambda^{(j)^T} c^{(j)} \right) + \left(\nabla f^{(j)} + \sum_{i=1}^{m} \lambda_i^{(j)} \nabla c_i^{(j)} \right)^T (x - x^{(j)}). \tag{15.5}$$

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 (15.5) in the integer variables only as

$$\eta \geq f^{(j)} + \left(\nabla_I f^{(j)} + \sum_{i=1}^{m} \lambda_i^{(j)} \nabla_I c_i^{(j)} \right)^T (x_I - x_I^{(j)}), \tag{15.6}$$

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

$$\eta \geq f^{(j)} + \mu_I^{(j)^T}(x_I - x_I^{(j)}). \tag{15.7}$$

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

$$0 \geq \sum_{i \in J^\perp} w_i c_i^+(x) + \mu_I^{(j)^T}(x_I - x_I^{(j)}), \tag{15.8}$$

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

15.1.3 Extended cutting-plane method

The extended cutting-plane method [439, 441] can be viewed as a variant of outer approximation that does not solve NLP subproblems such as (NLP($x_I^{(j)}$)) or (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 similar 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 [271].

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 (NLP($x_I^{(j)}$)) or (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 15.2.1; see for example [1].

15.2 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 reusing 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, (15.3), with the full set of linearizations $\mathcal{X}$ as in (15.2). 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.

15.2.1 LP/NLP-based branch-and-bound

Introduced by Quesada and Grossmann [366], 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 resolved, 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 14.1) and outer approximation (Algorithm 15.1); see also [86]. We denote by (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, (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 15.1.2.

Theorem 15.2.1 *If Assumptions 14.1.1 hold, and if the number of integer points in X is finite, then Algorithm 15.2 terminates in a finite number of steps at a solution of (13.1) or with an indication that (13.1) is infeasible.*

Figure 15.2 illustrates the progress of Algorithm 15.2. In *(i)*, 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 an *) does not give an integer feasible solution, and two new branches are introduced. The next LP in *(ii)* produces an integer feasible solution indicated by a box. The corresponding NLP subproblem is solved, and in *(iii)* 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 an *.

Algorithmic refinements of LP/NLP-BB. Abhishek et al. [1] 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 improve the performance of LP/NLP-BB dramatically. 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.

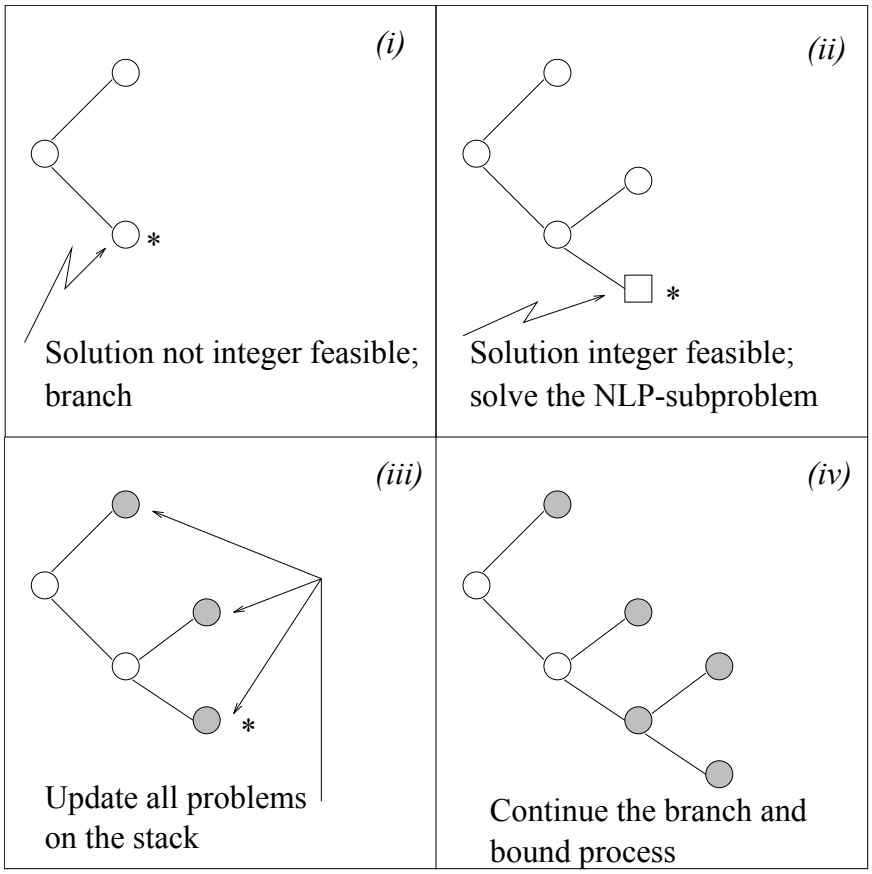


Figure 15.2: Progress of LP/NLP-based branch-and-bound.

15.2.2 Other single-tree approaches

It is straightforward to develop single-tree version of generalized Benders decomposition and the extended cutting plane method. In the case of Benders decomposition, we need only to replace the outer approximation cuts (15.1) by the Benders cut (15.7). 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 [412]) 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.

15.3 Presolve Techniques for MINLP

A key component in successful MILP software is an efficient presolve. These techniques were popularized by Savelsbergh [385], and some of these techniques can readily be 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 include 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.

15.3.1 Coefficient tightening for MINLP

We observed very large search trees when we tried to solve certain chemical engineering synthesis problem with a range of branch-and-bound solvers. For example, the search tree generated by MINOTAUR [318] for problem `Syn20M04M` grows rapidly, as shown in Figure 15.3. The left figure shows the search tree after 75 and 200 seconds CPU time (the tree after 360 seconds is shown in Figure 13.1). 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 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 [86]. In this section we show that we can improve the performance of branch-and-bound methods dramatically by extending coefficient tightening to MINLP.

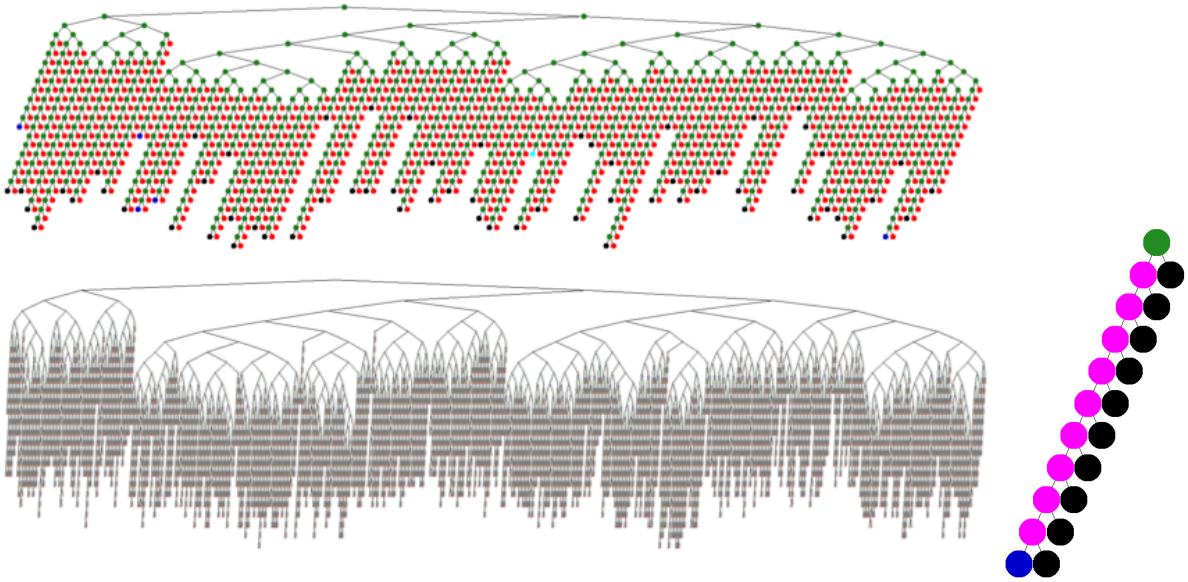


Figure 15.3: Left: branch-and-bound tree without presolve after 75 and 200 second CPU time. Right: Complete tree after presolve and coefficient tightening were applied.

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

$$x_1 + 21x_2 \leq 30,\ 0 \leq x_1 \leq 14,\ x_2 \in \{0,1\}. \tag{15.9}$$

The feasible set is the union of the two red lines in Figure 15.4. If $x_2 = 1$, then the constraint $x_1 \leq 30 - 21x_2 = 9$ is tight; but if $x_2 = 0$, then the $x_1 \leq 30 - 21x_2 = 30$ 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

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

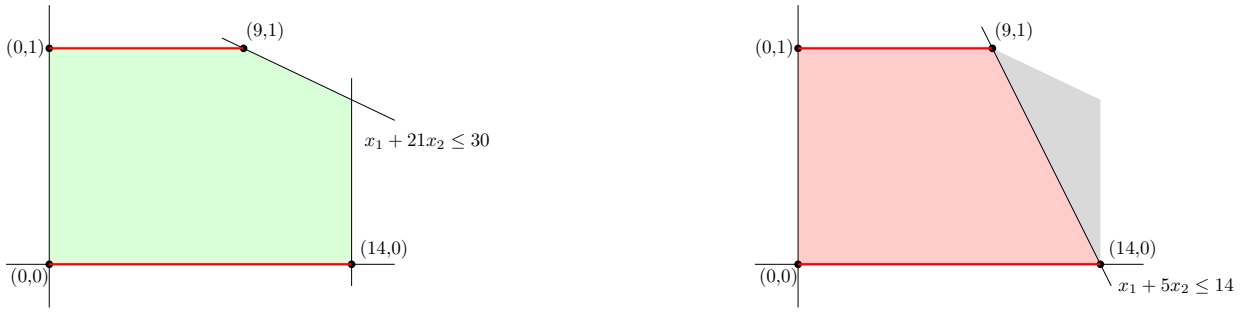


Figure 15.4: Feasible set of MILP example (15.9) (left) and feasible set after coefficient tightening (right).

is equivalent to (15.9), and corresponds to the convex hull of the two red lines (see Figure 15.4, right).

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

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

where $M > 0$ is a large constant. This form of constraints corresponds to on/off decisions that arise, for example, in process synthesis problems [427].

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:

$$\begin{cases} \underset{x}{\text{maximize}} & c(x_1, \ldots, x_k), \\ \text{subject to} & l_i \leq x_i \leq u_i, \forall i = 1, \ldots, k. \end{cases} \tag{15.10}$$

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

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

Unfortunately, this approach requires solving an NLP for every set of constraints for which we wish to tighten the formulation. Moreover, the optimization problem (15.10) is nonconvex, 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.

We can avoid the solution of this nonconvex 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:

$$c(x_1, \ldots, x_k) \leq b + M(1-y),\ y \in \{0,1\},\ \text{ and }\ 0 \leq x_i \leq u_i y, \forall i = 1, \ldots, k, \tag{15.11}$$

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 $c^u < b + M$):

$$c(x_1, \ldots, x_k) + (c^u - b)y \leq c^u,\ y \in \{0,1\},\ \text{ and }\ 0 \leq x_i \leq u_i y, \forall i = 1, \ldots, k. \tag{15.12}$$

The effect of this reformulation is dramatic as can be seen from Figure 15.3. The right tree 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 [156] in Figure 15.5 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.

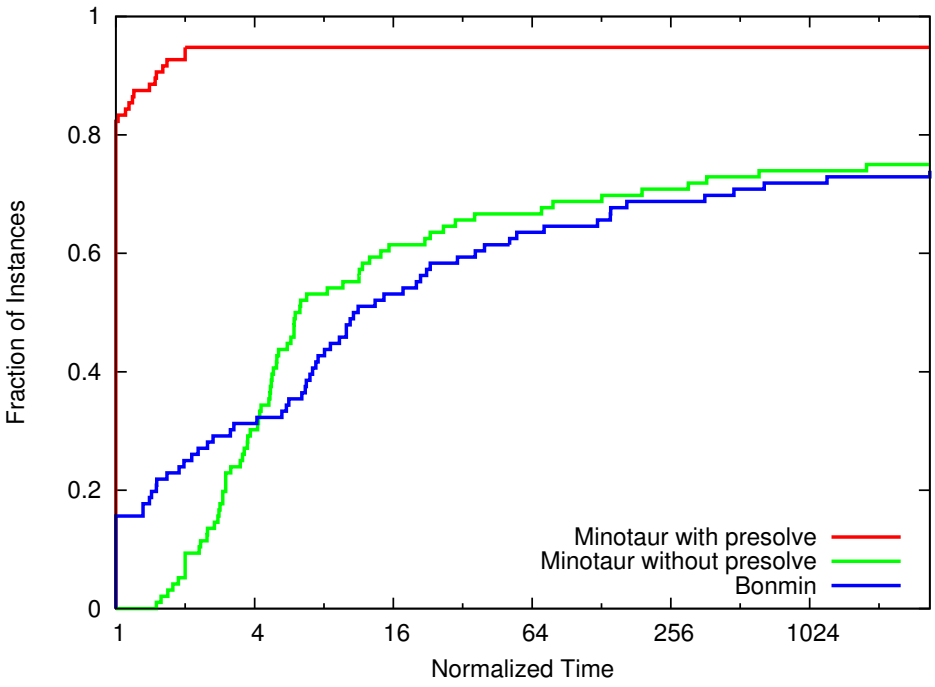


Figure 15.5: Performance profile comparing the effect of presolve on MINLP solver MINOTAUR for `Syn*` and `Rsyn*` instances.

15.3.2 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 [447]), which can be described as follows. Given a set of customers, $i = 1, \ldots, m$, and a set of facilities, $j = 1, \ldots, n$, which facilities should we open ($x_j \in \{0, 1\},\ j = 1, \ldots, n$) at cost f_j to serve the customers? The decision that facility j serves customer i is modeled with the binary variable $y_{ij} \in \{0, 1\}$. The constraints that every customer be served by exactly one facility and that only facilities that have customers assigned be open can be modeled as

$$\sum_{j=1}^{n} y_{ij} = 1,\ \forall i = 1, \ldots, m, \quad \text{and} \quad \sum_{i=1}^{m} y_{ij} \leq n x_j,\ \forall j = 1, \ldots, n, \tag{15.13}$$

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

$$\sum_{j=1}^{n} y_{ij} = 1,\ \forall i = 1, \ldots, m, \quad \text{and} \quad y_{ij} \leq x_j,\ \forall i = 1, \ldots, m,\ j = 1, \ldots, n. \tag{15.14}$$

As with (15.12), 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 (15.13) versus 2 seconds for (15.14).

Similar disaggregation tricks have been applied to nonlinear functions. Tawarmalani and Sahinidis [419] consider constraint sets of the following form:

$$S := \{x \in \mathbb{R}^n : c(x) = h(g(x)) \leq 0\}, \tag{15.15}$$

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 nondecreasing 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 [**? ? ?**].

We can derive a disaggregated version of the constraint set in (15.15) by introducing new variables $y = g(x) \in \mathbb{R}^p$, which leads to the following convex set:

$$S_d := \left\{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : h(y) \leq 0,\ y \geq g(x)\right\}. \tag{15.16}$$

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 (15.16) is preferred in any outer-approximation-based algorithm. In particular, given a set of points $\mathcal{X}^k := \left\{x^{(1)}, \ldots, x^{(k)}\right\}$ we construct the outer approximations of S and S_d as

$$S^{oa} := \left\{x : c^{(l)} + \nabla c^{(l)^T}(x - x^{(l)}) \leq 0,\ \forall x^{(l)} \in \mathcal{X}^k\right\} \tag{15.17}$$

and

$$S_d^{oa} := \left\{(x, y) : h^{(l)} + \nabla h^{(l)^T}(y - g(x^{(l)})) \leq 0,\ y \geq g^{(l)} + \nabla g^{(l)^T}(x - x^{(l)})\ \forall x^{(l)} \in \mathcal{X}^k\right\}, \tag{15.18}$$

respectively. It can be shown that the projection of S_d^{oa} onto x is contained in S^{oa}. Tawarmalani and Sahinidis [419] give an example that shows that the outer approximation (15.18) is tighter than (15.17). Moreover, for the two outer approximations to be equivalent, we may require an exponentially larger number of linearization points $x^{(l)}$ in (15.17) compared with (15.18).

Hijazi et al. [250] study a similar structure, namely, the partially separable constraint set:

$$\left\{x : c(x) := \sum_{j=1}^{q} h_j(a_j^T x + b_j) \leq 0\right\}, \tag{15.19}$$

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 $y \in \mathbb{R}^q$

$$\left\{(x, y) : \sum_{j=1}^{q} y_j \leq 0, \text{ and } y_j \geq h_j(a_j^T x + b_j)\right\}. \tag{15.20}$$

Again, one can show that the outer approximation of (15.20) is tighter than the outer approximation of (15.19). We can apply this technique to the worst-case example, (15.4), 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 (15.20) is then given by

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

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. [250] 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 $a_j^T x + b_j = x_j$.

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 $U^k = U^{k-1}$.

    end

    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($x_i^{(l,u)}, l, u, \mathcal{H}$)

  end

end

Algorithm 15.2: LP/NLP-based branch-and-bound.

Chapter 16

Branch-and-Cut Methods

In this chapter, we discuss branch-and-cut methods that can be used to enhance branch-and-bound and hybrid methods. In particular, we present cutting planes for mixed-integer problems, including standard cuts such as knapsack covers and mixed-integer rounding cuts. We also discuss perspective and disjunctive cuts, and comment on the implementation of mixed-integer solvers.

16.1 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 about 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).

16.1.1 Mixed-Integer Rounding Cuts

The LP/NLP-based branch-and-bound Algorithm 14.1 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 14.3. We start by considering mixed-integer rounding cuts. These are best introduced for the two-variable set

$$S := \{(x_1, x_2) \in \mathbb{R} \times \mathbb{Z} \mid x_2 \leq b + x_1,\ x_1 \geq 0\}, \tag{16.1}$$

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

$$x_2 \leq \lfloor b \rfloor + \frac{x_1}{1 - f_0} \tag{16.2}$$

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 16.1 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

$$X := \{(x_C^+, x_C^-, x_I) \in \mathbb{R}^2 \times \mathbb{Z}^p \mid a_I^T x_I + x_C^+ \leq b + x_C^-,\ x_C^+ \geq 0, x_C^- \geq 0,\ x_I \geq 0\}. \tag{16.3}$$

This set describes a selected constraint row of a 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

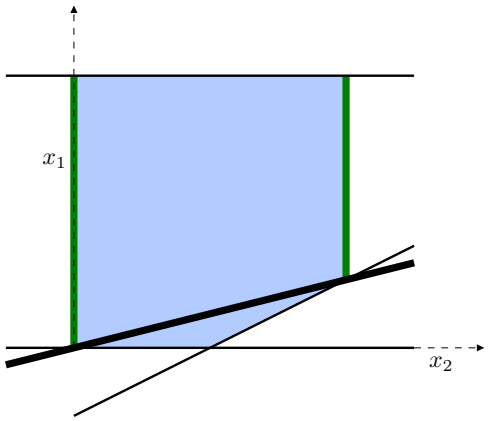


Figure 16.1: Mixed-integer rounding (MIR) cut. Feasible set of the LP relaxation (hatched), integer feasible set (bold black lines), and MIR cut (gray) $x_2 \leq 2x_1$ derived from the inequality $x_2 \leq \frac{1}{2} + x_1$.

sign of their coefficient in a_C. The extension from (16.1) is now straightforward by observing the following inequality is valid for X:

$$\sum_{i \in I} \left(\lfloor a_i \rfloor + \frac{\max\{f_i - f_0, 0\}}{1 - f_0} \right) x_i \leq \lfloor b \rfloor + \frac{x_C^-}{1 - f_0}, \tag{16.4}$$

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.

Gomory cuts were originally derived by Gomory [215, 216] and Chvátal [122] for integer linear programs. In the mixed-integer case, a Gomory cut is given by the inequality

$$\sum_{i \in I_1} f_i x_i + \sum_{i \in I_2} \frac{f_0(1 - f_i)}{f_i} x_i + x_C^+ + \frac{f_0}{1 - f_0} x_C^- \geq f_0 \tag{16.5}$$

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

$$X = \{(x_C, x_0, x_I) \in \mathbb{R}^2 \times \mathbb{Z} \times \mathbb{Z}^p \mid x_0 + a_I^T x_I + x_C^+ - x_C^- = b,\ x_C \geq 0, x_I \geq 0\}, \tag{16.6}$$

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 (13.2). Hence, it is sufficient to consider cuts for MILPs. Akrotirianakis et al. [13] report modest performance improvements using this approach for MINLP.

16.1.2 Perspective Cuts for MINLP

Many MINLP problems employ binary variables to indicate the nonpositivity 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 modeled with what is known as a *variable upper bound* constraint,

$$x_j \leq u_j x_i. \tag{16.7}$$

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 significantly improved computational performance. Frangioni and Gentile [198] 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:

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

Figure 16.2 depicts the set S, which is the union of two convex sets $S = S^0 \cup S^1$, where

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

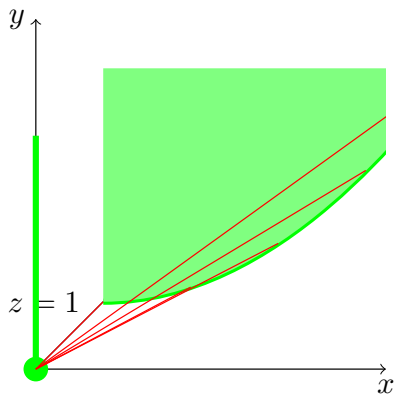


Figure 16.2: The set S.

One can observe from the geometry that the convex hull of S requires the surface defines 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

$$\text{conv}(S) = \{(x_1, x_2, x_3) \in \mathbb{R}^3 \ : \ x_2 x_3 \geq x_1^2, \ ux_3 \geq x_1 \geq 0, 1 \geq x_3 \geq 0, x_2 \geq 0\}. \tag{16.8}$$

The expression $x_2 x_3 \geq x_1^2$ in (16.8) 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

$$\mathcal{P}(x, z) := \begin{cases} 0 & \text{if } z = 0, \\ zf(x/z) & \text{if } z > 0. \end{cases} \tag{16.9}$$

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

$$z_i f(x/z_i) \leq 0, \tag{16.10}$$

results in a convex inequality that describes a significantly tighter relaxation of the feasible region. Günlük and Linderoth [233] (Lemma 3.2) slightly generalize this construction to the case where the S_0 side of the disjunction is an unbounded ray (like in Figure 16.2).

The general construction is as follows. We consider the problem

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

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

$$v \geq f(\bar{x}) + c + s^T(x - \bar{x}) + (c + f(\bar{x}) - s^T\bar{x}))(z - 1) \tag{16.11}$$

is valid for the equivalent mixed-integer program

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

Inequality (16.11), called the *perspective cut*, was introduced by Frangioni and Gentile [198] and used dynamically to build a tight formulation. [232] 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 (13.5) instead of using the nonlinear form (16.10).

16.1.3 Disjunctive Cutting Planes for MINLP

Disjunctive cuts for use in a branch-and-cut procedure were first discussed by Stubbs and Mehrotra [413] for convex mixed 0-1 nonlinear programs, based on Balas et al. [32] for MILP. Simultaneously, Ceria and Soares [115] derived the disjunctive programming arguments to be presented in a more general setting. We consider the convex mixed 0-1 nonlinear program

$$\begin{cases} \underset{x,\eta}{\text{minimize}} & \eta \\ \text{subject to} & f(x) \leq \eta, \\ & c(x) \leq 0, \\ & x \in X, \\ & x_i \in \{0,1\}\ \forall i \in I. \end{cases} \tag{16.12}$$

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,

$$\mathcal{C}_j^0 = \{x \in \mathcal{C} \mid x_j = 0,\ 0 \leq x_i \leq 1\ \forall i \in F, i \neq j\}, \tag{16.13}$$

$$\mathcal{C}_j^1 = \{x \in \mathcal{C} \mid x_j = 1,\ 0 \leq x_i \leq 1\ \forall i \in F, i \neq j\}. \tag{16.14}$$

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 [413] give the following description:

$$\tilde{M}_j(\mathcal{C}) = \left\{ (x_F, v_0, v_1, \lambda_0, \lambda_1) \;\middle|\; \begin{array}{ll} v_0 + v_1 = x_F, & v_{0j} = 0,\ v_{1j} = \lambda_1 \\ \lambda_0 + \lambda_1 = 1, & \lambda_0, \lambda_1 \geq 0 \\ \mathcal{P}_{c_i}(v_0, \lambda_0) \leq 0, & \mathcal{P}_{c_i}(v_1, \lambda_1) \leq 0,\ 1 \leq i \leq n_c \end{array} \right\}, \tag{16.15}$$

where $\mathcal{P}_f(v, \lambda)$ is the perspective function (16.9) 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 (16.15). Then, an inequality that is valid for the current node and cuts off $\bar{x}$ from the feasible set is given by

$$\pi_F^T x_F \leq \pi_F^T \hat{x}_F. \tag{16.16}$$

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

$$\left\{\begin{array}{ll} \underset{x_1,x_2}{\text{minimize}} & x_1 \\ \text{subject to} & (x_1 - \frac{1}{2})^2 + (x_2 - \frac{3}{4})^2 \leq 1 \\ & -2 \leq x_1 \leq 2 \\ & x_2 \in \{0, 1\} \end{array}\right. \tag{16.17}$$

with the relaxed optimal solution $x' = (x'_1, x'_2) = (-\frac{1}{2}, \frac{3}{4})$ shown in Figure 16.3 on the left. 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 $(x_1 - \frac{1}{2})^2 + (x_2 - \frac{3}{4})^2 \leq 1$ for x_1 given a fixed x_2,

$$\begin{aligned} \mathcal{C}^0 &= \left\{(x_1, 0) \in \mathbb{R} \times \{0, 1\} \,\middle|\, 2 - \sqrt{7} \leq 4x_1 \leq 2 + \sqrt{7}\right\}, \\ \mathcal{C}^1 &= \left\{(x_1, 1) \in \mathbb{R} \times \{0, 1\} \,\middle|\, 2 - \sqrt{15} \leq 4x_1 \leq 2 + \sqrt{15}\right\}. \end{aligned} \tag{16.18}$$

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

$$\begin{pmatrix} -0.1 & -0.03 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} -0.1 & -0.03 \end{pmatrix} \begin{pmatrix} -0.40 \\ 0.78 \end{pmatrix} \implies x_1 + 0.3x_2 \geq -0.166, \tag{16.19}$$

shown in Figure 16.3 on the right.

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 [413] observed the most favorable performance of the generated cuts when using the ∞-norm.

The cut (16.16) is in general valid for the local subtree of the branch-and-bound tree only. Stubbs and Mehrotra [413] show that a *lifting* to a globally valid cut

$$\pi^T x \leq \pi^T \hat{x}. \tag{16.20}$$

may be obtained by assigning

$$\pi_i = \min\{e_i^T H_0^T \mu_0, e_i^T H_1^T \mu_1\}, \ i \notin F \tag{16.21}$$

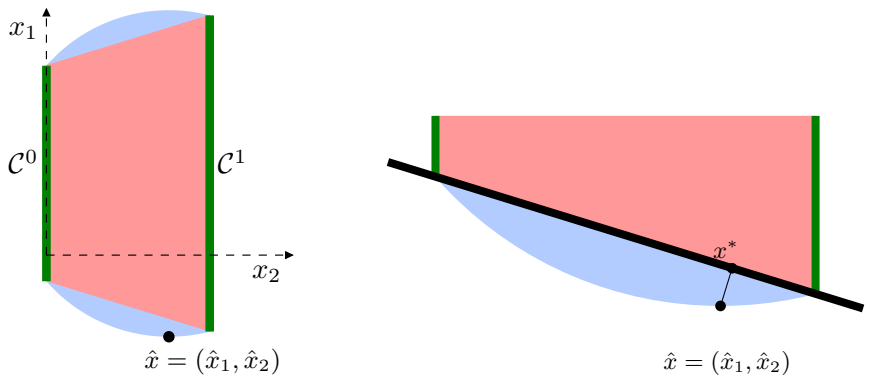


Figure 16.3: Left: NLP relaxation $\mathcal{C}$ (grey), integer feasible convex hull (hatched), and disjoint convex hulls $\mathcal{C}^0$ and $\mathcal{C}^1$ (bold black lines) for the MINLP example (16.17). Right: Solution to the minimum norm problem, and resulting disjunctive cut for the MINLP example (16.17). The next NLP solution including the disjunctive cut will produce the MINLP solution.

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 (16.15) 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 [414], 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 16.1.5.

16.1.4 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 [413] reported computational results on only four instances with no more than 30 unknowns. Addressing this issue, we mention an LP-based iterative separation procedure by Kılınç et al. [275] and Kılınç [274] that replaces (BC-SEP(x', j)), and a nonlinear separation approach by Bonami [84] that circumvents the difficulties inherent in (BC-SEP(x', j)).

Kılınç et al. [275] and Kılınç [274] 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 runtime 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

$$\mathcal{B}_j^0 = \{x \in \mathcal{B}^0 \mid x_j = 0\}, \quad \mathcal{B}_j^1 = \{x \in \mathcal{B}^0 \mid x_j = 1\}. \tag{16.22}$$

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. [275] and Kılınç [274] 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

$$\mathcal{F}_t^0 = \left\{ x \in \mathbb{R}^n \mid x_i = 0,\ g(x') + \frac{\partial g(x')}{\partial x}(x - x') \leq 0 \ \forall x \in \mathcal{K}_t^0 \right\}, \tag{16.23}$$

$$\mathcal{F}_t^1 = \left\{ x \in \mathbb{R}^n \mid x_i = 1,\ g(x') + \frac{\partial g(x')}{\partial x}(x - x') \leq 0 \ \forall x \in \mathcal{K}_t^1 \right\}. \tag{16.24}$$

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. [275] 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)) was solved. The process can be terminated early if it is observed that the cuts are not effective in 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 $\mathcal{B} = \{x \in \mathbb{R}^n \mid g(x') + \frac{\partial g}{\partial x}(x - x') \leq 0\}$ was proposed earlier by Zhu and Kuno [459] but does not necessarily converge [274, 275].

Bonami [84] 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

$$\begin{cases} \underset{v_1}{\text{maximize}} & v_{1,j} \\ \text{subject to} & c_i\left(\frac{v_1}{f_0}\right) \leq f_0 k - v_{1,j}, \\ & c_i\left(\frac{x' - v_1}{1 - f_0}\right) \leq f_0 k - v_{1,j}, \\ & v_1 \in \mathbb{R}^n. \end{cases} \tag{16.25}$$

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 (16.25), 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. [275] and the particular choice

$$\mathcal{K}_t^0 = \left\{ \frac{x' - v_1^*}{1 - f_0} \right\}, \quad \mathcal{K}_t^1 = \left\{ \frac{v_1^*}{f_0} \right\}, \tag{16.26}$$

if v_1^* denotes the optimal solution of (16.25).

16.1.5 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 $\mathcal{K} := \mathcal{K}_1 \times \ldots \times \mathcal{K}_k$, defined as

$$\mathcal{K}_j := \left\{ x_j = (x_{j0}, x_{j1}^T)^T \in \mathbb{R} \times \mathbb{R}^{n_j - 1} :\ ||x_{j1}||_2 \leq x_{j0} \right\}, \quad 1 \leq j \leq k, \tag{16.27}$$

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 [161] and Drewes and Ulbrich [162] propose a variant of the LP/NLP-based branch-and-bound, Algorithm 15.2, that solves continuous relaxations of (MISOCP) instead of NLPs for a fixed integer assignment x_I^k:

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

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 the 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

$$J_a(\bar{x}) := \{j : g_j(\bar{x}) = 0,\ \bar{x} \neq 0\}, \tag{16.28}$$

$$J_{0+}(\bar{x}, \bar{s}) := \{j : \bar{x}_j = 0,\ \bar{s}_{j0} > 0\}, \tag{16.29}$$

$$J_{00}(\bar{x}, \bar{s}) := \{j : \bar{x}_j = 0,\ \bar{s}_{j0} = 0\}, \tag{16.30}$$

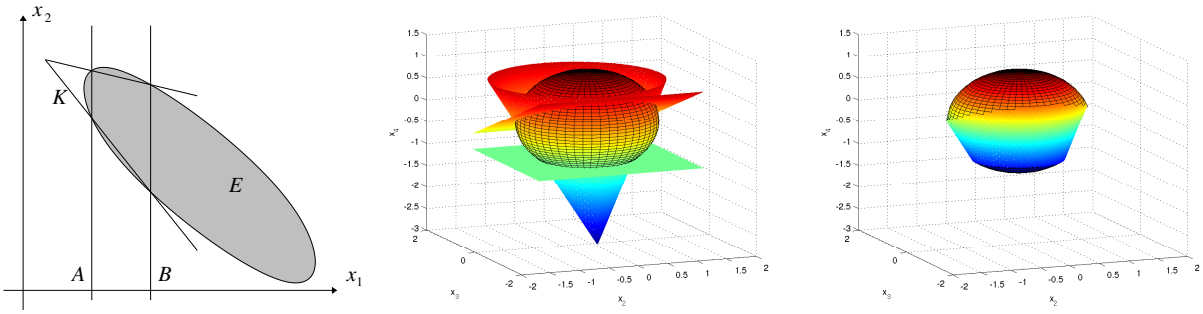
where g_i is the conic constraint function. Let $S \subset \mathbb{R}^n$ denote the set of previous primal-dual solutions $(\bar{x}, \bar{s})$ to (SOCP(x_I^k)). Then the linear outer approximation MIP for problem (MISOCP) is

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

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 [162].

Polyhedral approximations of the second order cone constraint are discussed for use in an outer approximation algorithm by Vielma et al. [434], who use the polynomial-size relaxation introduced by Ben-Tal and Nemirovski [55], while Krokhmal and Soberanis [283] generalize this to p-order cones, that is, sets of the form $\{x \in \mathbb{R}^{n+1} : x_{n+1} \geq ||x||_p\}$. 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 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 et al. [326].

Two efforts in this rapidly evolving area are the work by Dadush et al. [136], who prove that conic quadratic inequalities are necessary to represent the split closure of an ellipsoid, and the work by Belotti et al. [52], 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, $\mathcal{A} = \{x \in \mathcal{R}^n : a^T x \leq \alpha\}$, $\mathcal{B} = \{x \in \mathcal{R}^n : b^T x \leq \beta\}$; this is therefore a nonparallel disjunction, and a more general disjunction than discussed in Section 14.2.1 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



(a) (b) (c)

Figure 16.4: Disjunctive conic cuts as generated by Belotti et al. [52]. In (a), $\mathcal{K}$ is the disjunctive cone generated when intersecting the ellipsoid $\mathcal{E}$ with the intersection $\mathcal{A} \cup \mathcal{B}$. In (b), a disjunction described by two halfspaces 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).

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 16.4, is proved to be more effective than the conic MIR cut presented by Atamtürk and Narayanan [25] and is discussed below.

Gomory cuts for MISOCP. Drewes [161] describes Gomory cuts for (MISOCP) based on the work by Çezik and Iyengar [114] for pure integer conic programs. We assume here that the bounded polyhedral set X is described by

$$X = \{x \in \mathbb{R}^n \mid Ax = b,\ l \leq x \leq u\}, \tag{16.31}$$

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

Theorem 16.1.1 (Theorem 2.2.6 in Drewes [161]) *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),

$$\lceil (A_I^T(\bar{y} - \Delta y)\bar{s}_I \rceil^T s_I \geq \lceil (\bar{y} - \Delta y)^T b \rceil, \tag{16.32}$$

where Δy solves

$$\begin{pmatrix} -A_C \\ A_I \end{pmatrix} \Delta y = \begin{pmatrix} c_C \\ 0 \end{pmatrix}. \tag{16.33}$$

Furthermore, if $(\bar{y} - \Delta y)^T b \notin \mathbb{Z}$, then (16.32) *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 [161].

As an example due to Drewes [161], we consider the MISOCP

$$\begin{cases} \underset{x}{\text{minimize}} & -x_2 \\ \text{subject to} & -3x_2 + x_3 \leq 0 \\ & 2x_2 + x_3 \leq 3 \\ & 0 \leq x_1, x_2 \leq 3 \\ & x_1 \geq ||(x_2, x_3)^T||_2 \\ & x_1, x_2 \in \mathbb{Z}, \end{cases} \tag{16.34}$$

whose SOCP relaxation has the optimal solution $(3, \frac{12}{5}, -\frac{9}{5})$. The Gomory cut $x_2 \leq 2$ that can be deduced from this point is shown in Figure 16.5 and cuts off the relaxed solution.

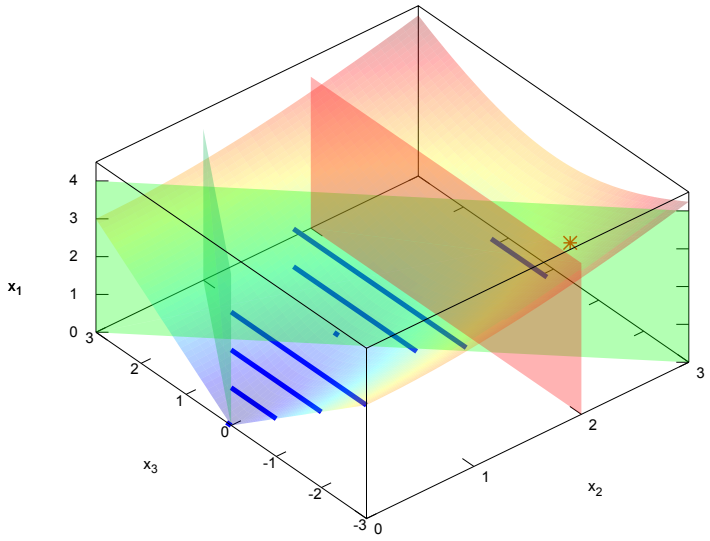


Figure 16.5: MISOCP example (16.34). Feasible cone (rainbow), linear constraints (green planes), integer feasible set (blue lines), relaxed optimal solution (red asterisk), and Gomory cut cutting off the relaxed optimal solution (red plane).

Lift and project cuts for MISOCP. Next, we consider the restricted class of mixed 0-1 second-order cone programs,

$$\begin{cases} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & x \in \mathcal{K}, \\ & Ax = b, \\ & l \leq x \leq u, \\ & x_i \in \{0, 1\} \; \forall i \in I, \end{cases} \tag{16.35}$$

and follow the notation of the lift and project procedure described for disjunctive cuts in Section 16.1.3. We are again interested in a finite conic linear description of the convex hull of the feasible set of (16.35), and we investigate the convex hull of the union of disjoint sets $\mathcal{C}_j^0$ and $\mathcal{C}_j^1$ to this end. For MISOCP, the convex

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

$$\tilde{M}_j(\mathcal{C}) := \left\{ (x, v^0, v^1, \lambda^0, \lambda^1) \;\middle|\; \begin{array}{ll} v^0 + v^1 = x, & \\ \lambda^0 + \lambda^1 = 1, & \lambda^0, \lambda^1 \geq 0 \\ Av^0 - \lambda^0 b = 0, & Av^1 - \lambda^1 b = 0 \\ v^0 \in \mathcal{K}, & v^1 \in \mathcal{K} \\ v^0_j = 0, & v^1_j = \lambda^1 \\ 0 \leq v^0_k \leq \lambda^0, & 0 \leq v^1_k \leq \lambda^1,\ k \in J, k \neq j \end{array} \right\}. \tag{16.36}$$

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 [413] for MISOCP:

Theorem 16.1.2 (Proposition 2.1.6 in Drewes [161]) *Fix a set $B \subseteq I$, let $\bar{x} \notin \tilde{M}_B(\mathcal{C})$, and let x^* be the solution of*

$$\min_{(x,v^0,v^1,\lambda^0,\lambda^1) \in \tilde{M}_B(\mathcal{C})} ||x - \bar{x}||_2. \tag{16.37}$$

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 [161] describes further lift-and-project cuts for MISOCP, based on work by Çezik and Iyengar [114] and Stubbs and Mehrotra [413] 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 MISCOP branch-and-cut procedure.

Mixed-integer rounding cuts for MISOCP. Atamtürk and Narayanan [25] describe mixed-integer rounding cuts for MISOCPs. To this end, we consider the following formulation of problem (MISOCP):

$$\left\{ \begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & ||A_j x - b_j||_2 \leq d_j^T x - h_{j0}, \quad 1 \leq j \leq k \\ & x \geq 0 \\ & x_i \in \mathbb{Z}\ \forall i \in I, \end{array} \right. \tag{16.38}$$

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 [25] 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 $A \in \mathbb{R}^{n_1 \times n}$:

$$\left\{ \begin{array}{ll} \underset{x,t}{\text{minimize}} & c^T x \\ \text{subject to} & t_l \geq |a_l^T x - b_l|, \qquad 1 \leq l \leq n_1 \\ & ||t||_2 \leq t_0 \leq d^T x - h_0 \\ & x \geq 0 \\ & x_i \in \mathbb{Z}\ \forall i \in I. \end{array} \right. \tag{16.39}$$

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

$$S_l := \{x \in \mathbb{R}^n,\ x \geq 0,\ x_i \in \mathbb{Z}\ \forall i \in I,\ t \in \mathbb{R} :\ t \geq |a_l^T x - b_l|\}. \tag{16.40}$$

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

Theorem 16.1.3 (Theorem 1 in Atamtürk and Narayanan [25]) *For any $\alpha \neq 0$ the conic mixed integer rounding (MIR) inequality*

$$\sum_{i \in I} \phi_{f_\alpha}(a_i/\alpha) x_i - \phi_{f_\alpha}(b/\alpha) \leq (t + x_C^+ + x_C^-)/|\alpha| \tag{16.41}$$

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 $0 \leq f_\alpha := b/\alpha - \lfloor b/\alpha \rfloor \leq 1$,*

$$\phi_{f_\alpha}(a) := \begin{cases} (1-2f_\alpha)p - (a-p) & \text{if } p \leq a < p + f_\alpha \\ (1-2f_\alpha)p + (a-p) - 2f_\alpha & \text{if } p + f_\alpha \leq a < p + 1 \end{cases}, \quad n \in \mathbb{Z}. \tag{16.42}$$

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 16.1.4 (Proposition 4 in Atamtürk and Narayanan [25]) *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.*

16.2 Tutorial

Introduction to MINOTAUR and other open-source solvers; implementation of cutting planes.

Chapter 17

Nonconvex Optimization

Global optimization of nonconvex MINLP problems is a hugely challenging problem. We present a basic algorithmic outline that builds on relaxation and separation, to define a nonlinear spatial branch-and-bound technique for nonconvex optimization. We also discuss piecewise linear approximation approaches, that offer an alternative to handling nonlinear constraints.

17.1 Nonconvex MINLP

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

Nonconvex MINLP is closely related to global optimization, a topic that also seeks optimal solutions to optimization problems having nonconvex functions, although the focus in global optimization has often been on problems with only continuous decision variables. Huge literature on global optimization exists, including several textbooks [193, 238, 255, 256]. It is outside 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 nonconvex MINLPs.

One approach to solving nonconvex MINLPs is to replace the nonconvex functions with piecewise linear approximations and solve the corresponding approximation by using mixed-integer linear programming solvers. We discuss this approach in Section 17.1.1. In the remaining sections we discuss components of methods for directly solving nonconvex MINLPs. In Section 17.1.2, we discuss generic strategies for obtaining convex relaxations nonconvex functions. We then describe in Section 17.1.3 how spatial branching can be used with these relaxations to obtain a convergent algorithm. Section 17.1.4 provides a sample of some techniques to obtain improved convex relaxations by exploiting particular types of nonconvex structures.

We refer the reader to the works of Tawarmalani and Sahinidis [417] and Burer and Letchford [102] for additional surveys focused on nonconvex MINLP.

17.1.1 Piecewise Linear Modeling

A common approach for approximately solving MINLPs with nonconvex 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 [146, 324].

Figure 17.1 shows an example of a nonconvex function with a corresponding piecewise linear approximation.

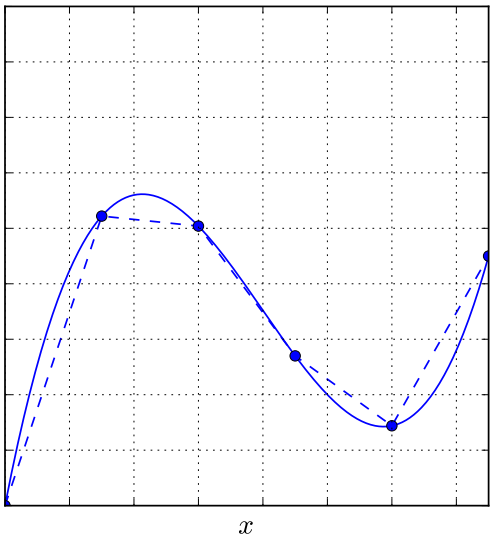


Figure 17.1: Example of a function (solid line) and a piecewise linear approximation of it (dashed line).

We focus our attention on modeling 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 17.1.1 we briefly introduce extensions of the piecewise linear approximation approach to more general multivariate functions.

Using a piecewise linear modeling technique for solving MINLPs involves two steps: obtaining piecewise linear approximations of the nonlinear functions and modeling the piecewise linear functions in a way that mixed-integer linear programming solvers can handle. We discuss these two steps in Sections 17.1.1 and 17.1.1, respectively. In Section 17.1.1 we provide a brief overview of how these modeling approaches can be extended to multivariate function.

Our treatment of this approach is necessarily brief. For more details on this topic, we refer to the reader to Geissler et al. [205], who provide a recent survey of piecewise linear modeling in MINLP, and to Vielma et al. [435], who provide a detailed review of methods for modeling piecewise linear functions using binary variables.

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

$$\hat{f}(x) = y^{k-1} + \left(\frac{y^k - y^{k-1}}{b^k - b^{k-1}}\right)(x - b^{k-1}),\ x \in [b^{k-1}, b^k],\ \forall k = 1, \ldots, d. \tag{17.1}$$

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:

$$\hat{f}(x) = a_k + m_k x,\ x \in [b^{k-1}, b^k],\ \forall k = 1, \ldots, d. \tag{17.2}$$

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 for obtaining a piecewise linear approximation is to simply 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 17.1. For a given number of break points, however, one can to 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 for 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 breakpoint 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 breakpoint exist, such as choosing the point where the error is largest or choosing the midpoint of the interval. Rote [375] analyzes 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 [208] studies methods for finding optimal piecewise linear approximations of convex or concave functions for a given number of breakpoints. Bellman [45] introduces a dynamic programming approach. Boyd and Vandenberghe [93] 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 [424] 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 [424] 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.

Modeling piecewise linear functions. This section describes techniques for modeling a piecewise linear function $\hat{f} : [l, u] \to \mathbb{R}$, as defined in (17.1). 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 the piecewise linear function $\hat{f}$ is continuous, although many of these formulations can be extended to lower semicontinuous piecewise linear functions [435]. In describing the different approaches, we follow the names used by Vielma et al. [435].

The first approach we present is the *multiple choice model*, which uses the representation of $\hat{f}(x)$ given in (17.2). In this model, a set of binary variables z_k, $k = 1, \ldots, d$, is introduced, where $z_k = 1$ indicates that x is in intervral 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:

$$\sum_{k=1}^{d} w^k = x, \qquad \sum_{k=1}^{d} (m_k w_k + a_k z_k) = y, \quad \sum_{k=1}^{d} z_k = 1, \tag{17.3a}$$

$$b^{k-1} z_k \le w_k \le b^k z_k, \quad z_k \in \{0, 1\} \quad k = 1, \ldots, d. \tag{17.3b}$$

This model was introduced by Jeroslow and Lowe [262] and has been studied by Balakrishnan and Graves [29] and Croxton et al. [132]. In computational experiments conducted by Vielma et al. [435], 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 (17.1). In this model, a set of binary variables z_k, $k = 1, \ldots, d$, 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:

$$\sum_{k=1}^{d} (\lambda_k b^{k-1} + \mu_k b^k) = x, \qquad \sum_{k=1}^{d} (\lambda_k y^{k-1} + \mu_k y^k) = y, \tag{17.4a}$$

$$\sum_{k=1}^{d} z_k = 1, \qquad \lambda_k + \mu_k = z_k, \quad k = 1, \ldots, d, \tag{17.4b}$$

$$\lambda_k \ge 0, \mu_k \ge 0, z_k \in \{0, 1\}, \quad k = 1, \ldots, d. \tag{17.4c}$$

The constraints (17.4b) 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 (17.4a) 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 in [132, 262, 263, 329, 399].

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:

$$\sum_{k=0}^{d} \lambda_k b^k = x, \qquad \sum_{k=0}^{d} \lambda_k y^k = y, \tag{17.5a}$$

$$\sum_{j=k}^{d} \lambda_j \le \sum_{j=k}^{d} z_j, \qquad \sum_{j=0}^{k-1} \lambda_j \le \sum_{j=1}^{k} z_j, \quad k = 1, \ldots, d, \tag{17.5b}$$

$$\sum_{k=0}^{d} \lambda_k = 1 \qquad \sum_{k=1}^{d} z_k = 1, \tag{17.5c}$$

$$\lambda_k \ge 0, \quad k = 0, 1, \ldots, d \qquad z_k \in \{0, 1\}, \quad k = 1, \ldots, d. \tag{17.5d}$$

The constraints (17.5b) enforce that when $z_k = 1$, $\lambda_j = 0$ for all $j \notin \{k-1, k\}$. In most presentations of the convex combination model [146, 147, 205, 263, 294, 344, 435, 444] this condition is instead formulated by using the following constraints:

$$\lambda_0 \le z_1, \quad \lambda_d \le z_d, \quad \lambda_k \le z_k + z_{k+1}, \quad k = 1, \ldots, d-1. \tag{17.6}$$

However, Padberg [356] demonstrated that the formulation using (17.6) allows more continuous solutions when the binary constraints on the z_k variables are relaxed, which makes (17.5b) 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 [324]. 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:

$$b^0 + \sum_{k=1}^{d} \delta_k (b^k - b^{k-1}) = x, \quad y^0 + \sum_{k=1}^{d} \delta_k (y^k - y^{k-1}) = y, \tag{17.7a}$$

$$\delta_{k+1} \le z_k \le \delta_k, \qquad k = 1, \ldots, d-1, \tag{17.7b}$$

$$\delta_1 \le 1, \delta_d \ge 0, z_k \in \{0, 1\}, \qquad k = 1, \ldots, d-1 \tag{17.7c}$$

If $\delta_i < 1$ for some i, then constraints (17.7b) 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 [356], the convex combination formulation (17.7) and the incremental formulation (17.7) 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) [43, 44, 270, 423]. 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 nonzero and the nonzero variables are adjacent. This is exactly the condition that the binary variables in the convex combination model (17.5) are used to enforce. Therefore, by instead explicitly stating this condition, the following

formulation is obtained:

$$\sum_{k=0}^{d} \lambda_k b^k = x, \qquad \sum_{k=0}^{d} \lambda_k y^k = y, \tag{17.8a}$$

$$\sum_{k=0}^{d} \lambda_k = 1, \tag{17.8b}$$

$$\lambda_k \geq 0, \quad k = 0, 1, \ldots, d \quad (\lambda_0, \lambda_1, \ldots, \lambda_d) \text{ is SOS2.} \tag{17.8c}$$

The condition that an ordered set of variables is SOS2 can be declared in most commercial mixed-integer linear programming solvers [175, 235, 259], 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 to 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 [270].

Vielma and Nemhauser [433] have recently developed a novel approach for modeling 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 [433] is still linear in the number of segments. The computational results of Vielma and Nemhauser [433] suggest that this approach is most beneficial when modeling multivariate piecewise linear functions.

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 modeling techniques. We 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 17.1.2 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 Lee and Wilson [294], Tomlin [423] and Vielma et al. [435]. 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 was 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 either is 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 also can directly model nonseparable multivariate piecewise linear functions using binary variables. We refer the reader to the works of Vielma et al. [435] and Geissler et al. [205] for details of these techniques. In particular, the convex combination method has been extended to multivariate functions by Lee and Wilson [294], and the incremental method has been extended by Wilson [444]. Tomlin [423] also extended the notion of special ordered sets for modeling multivariate piecewise linear functions. D'Ambrosio et al. [142] 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 requires 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.

17.1.2 Generic Relaxation Strategies

For general nonconvex 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 $\int_{x_0}^{x} h(x)dx$ whose antiderivative is unknown and *black-box* functions, whose value can be computed by running, for instance, a simulation. 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 [124]. The expression tree of the function $f(x_1, x_2) = x_1 \log(x_2) + x_2^3$ is depicted in Figure 17.2.

Relaxations of factorable functions. If the objective and the constraints of an MINLP of the form (13.1) 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 (13.1), 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).

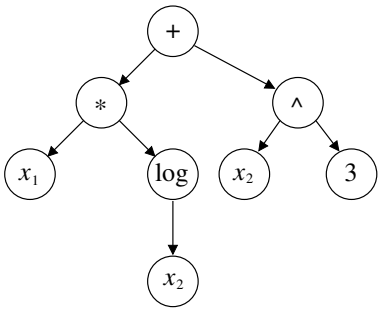


Figure 17.2: The expression tree of $f(x_1, x_2) = x_1 \log(x_2) + x_2^3$. Leaf nodes are for variables x_1 and x_2 and for the constant 3.

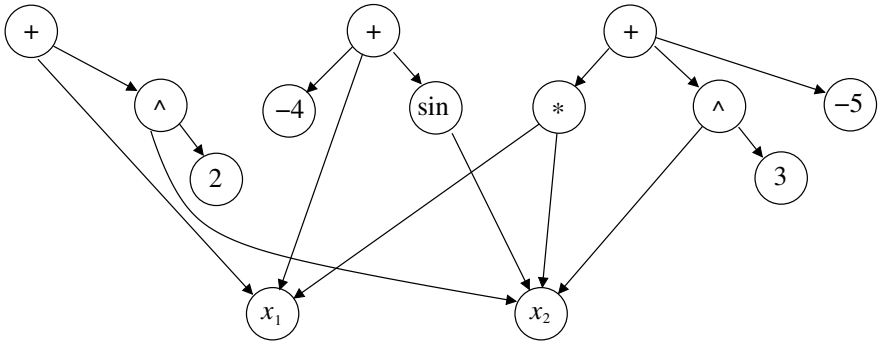


Figure 17.3: The DAG associated with the problem in (17.9). Each node without entering arcs is associated with the root of the expression tree of either a constraint or the objective function. In common with all constraints and the objective are the leaf nodes associated with variables x_1 and x_2.

Example. Consider the following MINLP:

$$\begin{array}{ll} \min & x_1 + x_2^2 \\ \text{s.t.} & x_1 + \sin x_2 \leq 4 \\ & x_1 x_2 + x_2^3 \leq 5 \\ & x_1 \in [-4, 4] \cap \mathbb{Z} \\ & x_2 \in [0, 10] \cap \mathbb{Z}. \end{array} \tag{17.9}$$

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 17.3.

Factorable problems allow a *reformulation* of the problem (13.1) [327, 405, 417]. The reformulation is another MINLP as follows:

$$\left\{ \begin{array}{lll} \underset{x}{\text{minimize}} & x_{n+q} & \\ \text{subject to} & x_k = \vartheta_k(x) & k = n+1, n+2, \ldots, n+q \\ & l_i \leq x_i \leq u_i & i = 1, 2, \ldots, n+q \\ & x \in X, & \\ & x_i \in \mathbb{Z}, \ \forall i \in I, & \end{array} \right. \tag{17.10}$$

where ϑ_k is an operator of the set $\mathcal{O}$ introduced above. The bound 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 (17.9) admits the following reformulation:

$$\begin{array}{lllll} \min & x_9 & & & \\ \text{s.t.} & x_3 = \sin x_2 & x_7 = x_5 + x_6 - 5 & 0 \le x_2 \le 10 & 0 \le x_6 \le 1000 \\ & x_4 = x_1 + x_3 - 4 & x_8 = x_2^2 & -1 \le x_3 \le 1 & -45 \le x_7 \le 0 \\ & x_5 = x_1 x_2 & x_9 = x_1 + x_8 & -9 \le x_4 \le 0 & 0 \le x_8 \le 100 \\ & x_6 = x_2^3 & -4 \le x_1 \le 4 & -40 \le x_5 \le 40 & -4 \le x_9 \le 104 \\ & x_1, x_2, x_5, x_6, x_7, x_8, x_9 \in \mathbb{Z}. & & & \end{array}$$

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 (17.10) and (13.1) are equivalent in that for any feasible (resp. optimal) solution $\bar{x} \in \mathbb{R}^{n+q}$ of (17.10) one can obtain a feasible (resp. optimal) solution $\tilde{x} \in \mathbb{R}^n$ of (13.1) and vice versa. Although still a nonconvex MINLP, (17.10) makes it easier to obtain a convex relaxation of (13.1). Consider the nonconvex sets

$$\Theta_k = \{x \in \mathbb{R}^{n+q} : x_k = \vartheta_k(x), x \in X, l \le x \le u, x_i \in \mathbb{Z}, i \in I\}, \qquad k = n+1, n+2, \ldots, n+q.$$

Note that Θ_k are, in general, nonconvex 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 (17.10) intersects $\breve{\Theta}_k$ for $k = n+1, n+2, \ldots, n+q$, and hence it is a relaxation of (13.1) (note that the integrality constraints are also relaxed):

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

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

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

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 (13.1) use the following linear relaxation to obtain a lower bound:

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

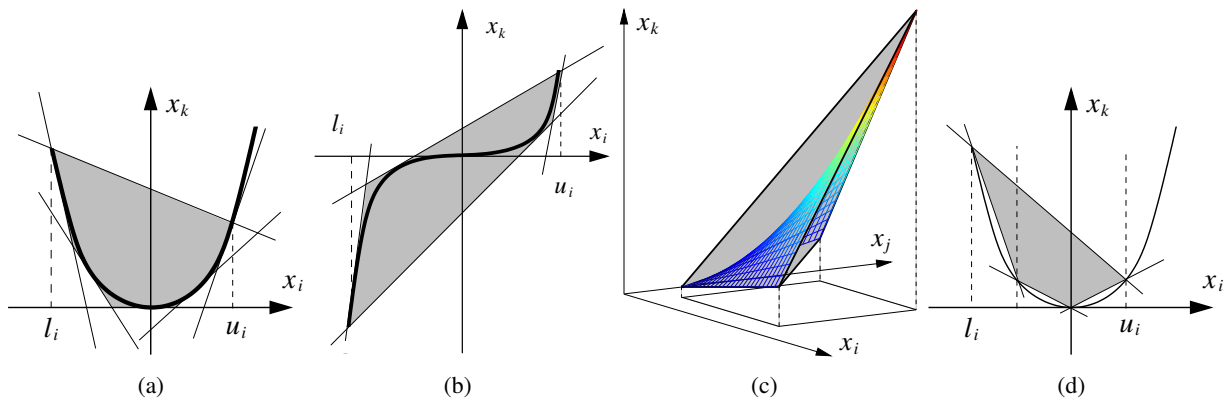


Figure 17.4: Polyhedral relaxations $\breve{\Theta}_k$ for several univariate and bivariate operators: $x_k = x_i^2$, $x_k = x_i^3$, $x_k = x_i x_j$, and $x_k = x_i^2$ with x_i integer. Note that these relaxations are *exact* at the bounds on x_i and x_j.

Although finding a polyhedral superset of Θ_k is nontrivial, 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 [304]. For products of two variables $x_k = x_i x_j$, the following four inequality proposed by McCormick [327] provide the convex hull of the set $\Theta_k = \{(x_i, x_j, x_k) : x_k = x_i x_j, (l_i, l_j, l_k) \leq (x_i, x_j, x_k) \leq (u_i, u_j, u_k)\}$, as proved by Al-Khayyal and Falk [14]:

$$\begin{array}{lll} x_k \geq l_j x_i + l_i x_j - l_i l_j & \qquad & x_k \leq l_j x_i + u_i x_j - u_i l_j \\ x_k \geq u_j x_i + u_i x_j - u_i u_j & & x_k \leq u_j x_i + l_i x_j - l_i u_j. \end{array} \tag{17.11}$$

Figure 17.4 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 17.4(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 $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ be the arguments of ϑ_k, each of them bounded by the interval $[l_{i_j}, u_{i_j}]$. Define for each $j = 1, 2, \ldots, k$ 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 17.1.3.

Disjunctive cuts. In the class of convex MINLPs, the only nonconvex constraints are represented by integer variables, and these nonconvexities are resolved by integer branching, which represents a specific class of *disjunctions*. There are classes of nonconvex MINLP whose nonlinear objective and constraint 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 16.1.3. For nonconvex, 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 et al. [266].

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 nonconvex 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 17.5. Figure 17.5(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 17.5(b); note that both relaxations exclude $(\hat{x}_i, \hat{x}_j)$. For an example with a different function $x_j = e^{x_i}$, see Figure 17.5(c). Belotti [48] creates disjunctive cuts by means of a *cut generating LP* (CGLP) [31], a linear optimization problem used to devise a disjunctive cut that maximizes the violation w.r.t. $\hat{x}$. 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 one to avoid creating two new subproblems and can be effective, each of these cuts is generated by solving very large LPs and can be computationally expensive for large MINLPs.

17.1.3 Spatial Branch-and-Bound

The best-known method for solving nonconvex 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 14.1, 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 17.1.2 to obtain a lower bound [50, 377, 378, 381, 405, 417, 418].

Other approaches employ a relaxation technique called α-*convexification* [19]. For a nonconvex 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 $\alpha = -\lambda_{\text{MIN}}(Q)$, namely, the opposite of the minimum eigenvalue of Q, to obtain a convex relaxation. Some implementation of the α-convexification adopt a quadratic approximation of the objective and the constraints and hence guarantee optimality only for quadratic problems [350]. This method can be extended to nonquadratic functions while preserving the validity of the lower bound. A generalization of this method is at the base of the MINLP solver GloMIQO [332].

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 for nonconvex MINLP follows the scheme described in Section 14.1, 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 (13.1) as follows:

$$\begin{cases} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c(x) \leq 0, \\ & x \in X \\ & l_i \leq x_i \leq u_i \ \forall i = 1, 2, \ldots, n \\ & x_i \in \mathbb{Z}, \ \forall i \in I. \end{cases} \tag{17.12}$$

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):

$$\begin{cases} \underset{x}{\text{minimize}} & x_{n+q} \\ \text{subject to} & a^k x_k + B^k x \geq d^k & k = n+1, n+2, \ldots, n+q \\ & l_i \leq x_i \leq u_i & i = 1, 2, \ldots, n+q \\ & x \in X. \end{cases} \tag{17.13}$$

Suppose an optimal solution $\hat{x}$ of LP(l, u) is found. If $\hat{x}$ is feasible for (17.12) and hence for (13.1), subproblem NLP(l, u) can be eliminated. If $\hat{x}$ is infeasible for (17.12), then at least one of the following two holds:

1. $\hat{x}$ is not integer feasible, i.e., $\exists i \in I : \hat{x}_i \notin \mathbb{Z}$.
2. At least one of the continuous nonconvex constraints of the reformulation is violated, that is,

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

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 nonempty intersection, where $x_i = \hat{x}_i$. This constitutes a strong point of departure with the subclass of pure integer nonconvex MINLPs, where all variables are integer, and with convex MINLP discussed in Section 14.1. 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 (13.1):

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

whose only difference from the feasible set of (13.1) 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 *consistent* if, at every step, subsets $\mathcal{F}(l^-, u^-)$ and $\mathcal{F}(l^+, u^+)$ either are 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 [256]

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

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 [327] and Horst and Tuy [256].

Theorem 17.1.1 (Horst and Tuy [256], McCormick [327]) *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 nonconvex 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 [327] states that convergence is still ensured as long as the bounding operation is finitely consistent.

Spatial branching. Partitioning the feasible set of a subproblem NLP(l, u) yields $h \geq 2$ new subproblems NLP(l', u'), NLP(l'', u''), …, NLP$(l^{(h)}, u^{(h)})$, whose lower bounds $\lambda_{\text{NLP}(l',u')}, \lambda_{\text{NLP}(l'',u'')}, \ldots, \lambda_{\text{NLP}(l^{(h)},u^{(h)})}$ are not smaller than that of NLP(l, u). We will assume w.l.o.g. that two new problems NLP(l^-, u^-) and NLP(l^+, u^+) are created. Most practical implementation 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 [50, 417]. 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 14.1) 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 $\lambda_{\text{NLP}(l^-,u^-)}$ and $\lambda_{\text{NLP}(l^+,u^+)}$; (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 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 bound do not coincide), 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.

Upon branching, the two generated subproblems each will obtain a lower bound by solving two tighter relaxations than that of their ancestor. A geometrical intuition is provided in Figure 17.5. Suppose the auxiliary x_j is defined as $x_j = \vartheta_j(x_i) = (x_i)^2$ and $x_i \in [l_i, u_i]$ for this subproblem. Because the LP solution $\hat{x}$ is such that $\hat{x}_j \neq (\hat{x}_i)^2$ (see Figure 17.5(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 17.5(b), which are disjoint except for the point (b, b^2) and which exclude the point $(\hat{x}_i, \hat{x}_j)$. Figure 17.5(c) provides a similar example for $x_j = \vartheta_j(x_i) = e^{x_i}$.

Tawarmalani and Sahinidis [417] 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 nonconvex 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 14.1, can be applied with little modification to nonconvex MINLP and have been implemented in nonconvex MINLP solvers [50].

The choice of branching point is also crucial and differs from integer branching in that one has the freedom of choosing 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 LP(l^-, u^-) and 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 LP(l^-, u^-) and 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 one from excluding $\hat{x}$.

Bounds tightening. Bounds tightening (also referred to as bounds reduction or domain reduction) is a class of algorithms aiming at reducing the bound intervals on the variables of (13.1). Although these algo-

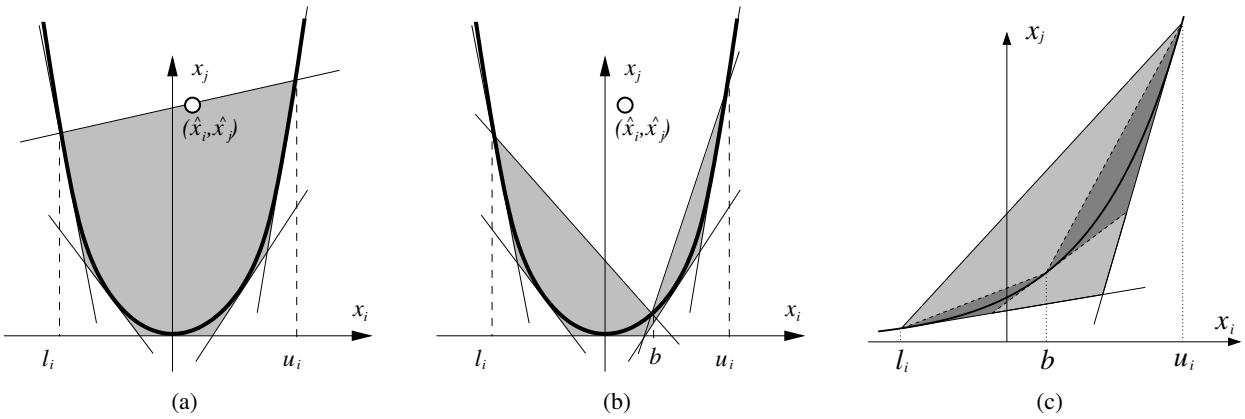


Figure 17.5: Polyhedral relaxations upon branching: In (a), the set $\breve{\Theta}_k$ is shown with the components $(\hat{x}_i, \hat{x}_j)$ of the LP solution. Branching on x_i excludes the LP solution—see (b). In (c), the LP relaxation before and after branching is shown for $x_j = e^{x_i}$ in lighter and darker shade, respectively.

rithms are optional in a BB solver, they are crucial to obtaining an optimal solution in reasonable time and therefore are implemented in the vast majority of MINLP solvers.

Their importance is directly connected to the LP relaxation (17.13): 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 [328].

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 (13.1). In these two cases, the procedure proves that the current node is infeasible and can be fathomed without the need of computing a lower bound by solving the convex relaxation.

Consider again the feasible set of a 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 (13.1) 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

$$l_i' = \min\{x_i : x \in S, f(x) \leq \check{z}\}; \qquad u_i' = \max\{x_i : x \in S, f(x) \leq \check{z}\}. \tag{17.14}$$

Solving the $2n$ optimization problems above would yield tighter bounds, but these problems can be as hard as problem (13.1) 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 [148] and is a strong component of constraint programming solvers. It is part of nonlinear optimization solvers [328] as well as of MILP solvers [18, 385].

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

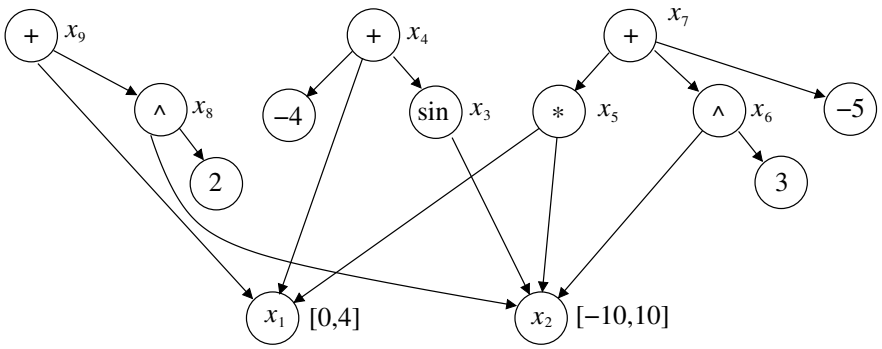


Figure 17.6: Association between auxiliary variables and the nodes of the DAG related with the problem in (17.9).

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, namely, $l'_i = \sqrt[3]{l_j}$. Another example is given by $x_k = x_i x_j$, with $(1,1,0) \leq (x_i, x_j, x_k) \leq (5,5,2)$. 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 $x_i \leq \frac{u_k}{l_j}$ and $x_j \leq \frac{u_k}{l_i}$, 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 $J^+ = \{j = 1, 2, \ldots, n : a_j > 0\}$ and $J^- = \{j = 1, 2, \ldots, n : a_j < 0\}$. Then valid bounds on x_k are

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

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

$$\begin{array}{rrcl} \forall j : a_j > 0, & l'_j & = & \frac{1}{a_j}\left(l_k - \left(a_0 + \sum_{i \in J^+ \setminus \{j\}} a_i u_i + \sum_{i \in J^-} a_i l_i\right)\right) \\ & u'_j & = & \frac{1}{a_j}\left(u_k - \left(a_0 + \sum_{i \in J^+ \setminus \{j\}} a_i l_i + \sum_{i \in J^-} a_i u_i\right)\right) \\ \forall j : a_j < 0, & l'_j & = & \frac{1}{a_j}\left(u_k - \left(a_0 + \sum_{i \in J^+} a_i l_i + \sum_{i \in J^- \setminus \{j\}} a_i u_i\right)\right) \\ & u'_j & = & \frac{1}{a_j}\left(l_k - \left(a_0 + \sum_{i \in J^+} a_i u_i + \sum_{i \in J^- \setminus \{j\}} a_i l_i\right)\right). \end{array} \tag{17.15}$$

These *implied bounds* are commonly used as a preprocessing technique [18] prior to solving MILP problems. For both MILP and MINLP problems, bound reduction can be obtained by using *pairs* of inequalities [49], specifically through the convex combination of two inequalities $a^T x \geq \alpha$ and $b^T x \geq \beta$ using a parameter $\lambda \in [0,1]$. The resulting inequality $(\lambda a + (1-\lambda)b)^T x \geq \lambda\alpha + (1-\lambda)\beta$ yields bounds on the variables similar to (17.15), 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 (17.9) 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 17.6.

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 to 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 bound is 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 $[-\frac{1}{\alpha}, \frac{1}{\alpha}]$; this will trigger a reduction of the bound interval of x_1 to $[-\frac{1}{\alpha^2}, \frac{1}{\alpha^2}]$, which in turn will propagate to yield new bounds $[-\frac{1}{\alpha^3}, \frac{1}{\alpha^3}]$ 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 (13.1) [51].

Optimality-based bounds tightening. Solving problems (17.14) is impractical because of the nonconvexity of their feasible set, which is the same feasible set of (13.1). A more practical approach considers the feasible set of a convex relaxation of (13.1), such as the one in (17.13):

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

Then the following are valid bounds on variable x_i:

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

Empirical evidence shows that this technique is effective in obtaining tight bounds [50], but it requires solving $2n$ linear programming problems. Thus, its use is limited to the root node or to the nodes of small depth.

Probing and reduced-cost bounds tightening. Consider the bounds $[l_i, u_i]$ on a variable x_i as defined in (17.13), 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 hyperrectangle renders the problem infeasible or drives the lower bound l_{n+q} on x_{n+q} above the cutoff value, then we have a 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 bounds tightening procedures. Probing is used in MILP [385] and MINLP [50, 417] on binary, integer, and continuous variables.

Reduced cost bound tightening [377], akin to the *reduced cost fixing* technique utilized in MILP [343], uses the solution of an LP relaxation of (13.1) to infer new, and possibly tighter, bounds on the variables. Suppose that an optimal solution $\hat{x}$ of (17.13) 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 (13.1) 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 $x^\star$ one has $x_i = u_i$ and a negative reduced cost ρ_i, then a valid lower bound on x_i is $l'_i = u_i + \frac{\check{z} - z_{\text{LP}}}{\rho_i}$.

17.1.4 Relaxations of Structured Nonconvex Sets

The methods described in Sections 17.1.2 and 17.1.3 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 tightest possible relaxation, the convex hull of feasible solutions, leading to an impractically large branch-and-bound search tree.

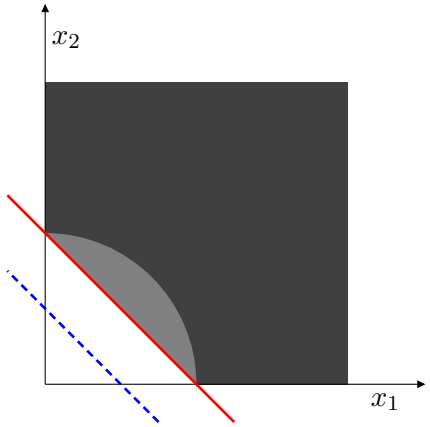


Figure 17.7: The dark shaded area is the feasible region. The convex hull is the entire shaded area, and is defined by solid line, $x_1 + x_2 \geq 1$. The dashed line corresponds to the weaker inequality $x_1 + x_2 \geq 1/2$.

As a simple example, consider the nonconvex constraint in two variables x_1 and x_2:

$$x_1^2 + x_2^2 \geq 1 \tag{17.16}$$

and suppose $x_1, x_2 \in [0, 2]$. The set defined by these constraints is the dark shaded area in Figure 17.7. 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 17.1.2. 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 (17.16) with

$$x_3 + x_4 \geq 1. \tag{17.17}$$

The nonconvex 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 (17.17), obtaining $2x_1 + 2x_2 \geq x_3 + x_4 \geq 1$, or $x_1 + x_2 \geq 1/2$ (the 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.

Nonconvex quadratic functions. One structure that appears in many MINLP problems is the presence of (nonconvex) 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:

$$\begin{cases} \underset{x}{\text{minimize}} & x^T Q_0 x + c_0^T x, \\ \text{subject to} & x^T Q_k x + c_k^T x \leq b_k, \quad k = 1, \ldots, q \\ & Ax \leq b, \\ & 0 \leq x \leq u,\ x_i \in \mathbb{Z},\ \forall i \in I, \end{cases} \tag{17.18}$$

where for each $k = 0, 1, \ldots, q$, 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 nonconvex even when the integrality constraints are relaxed. It is also possible to have nonzero lower bounds $x \geq \ell$, but we assume here simply $x \geq 0$ to simplify exposition.

Many relaxation strategies for QCQPs are based on introducing additional variables X_{ij} for all i, j pairs, and reformulating (17.18) as follows:

$$\begin{cases} \underset{x}{\text{minimize}} & Q_0 \bullet X + c_0^T x, \\ \text{subject to} & Q_k \bullet X + c_k^T x \leq b_k, \quad k = 1, \ldots, q \\ & Ax \leq b, \\ & 0 \leq x \leq u,\ x_i \in \mathbb{Z},\ \forall i \in I, \\ & X = xx^T, \end{cases} \tag{17.19}$$

where X is the $n \times n$ matrix containing all the X_{ij} variables, so that the constraint $X = xx^T$ records the nonconvex constraints $X_{ij} = x_i x_j$ for all $i, j = 1, \ldots, n$. Observe that these constraints are the only nonlinear constraints in this reformulation. Obtaining a relaxation of (17.19) 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) [9, 396, 397] 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* nonconvex 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 nonconvex constraints, based on multiplying pairs of the nonnegative 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 (17.11) 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 (17.19). 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 to this technique to general integer variables, based on Lagrange interpolating polynomials, has been proposed as well [8]. 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 nonnegative 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 nonnegative 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 (17.19) 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 thus 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 [22]:

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

SDP relaxations can be used within a branch-and-bound algorithm to solve QCQPs to optimality [99, 104].

Anstreicher [22] 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 Quadratic Polytope [355] can also be added to further strengthen the relaxation [101, 451]. Anstreicher [23] 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 [400] and further refined by Qualizza et al. [365] to provide a polyhedral approximation of the SDP constraint.

For both the RLT and SDP relaxation approach, 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 (17.19), 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 (17.19). This approach leads to a hierarchy of relaxations of improving quality [10, 288, 289]. The drawback of this approach is that the size of these formulations grows dramatically, limiting their current practical use. Indeed, even the formulation (17.19) may be significantly larger than the original formulation (17.18), leading Saxena et al. [387] to study of relaxations of QCQPs that do not use the additional variables X_{ij}.

For MINLPs with only a quadratic objective (i.e., problem (17.18) with $q = 0$), Burer [100] 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 (17.19) with certain enhancements (such as using 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 nonnegative.

We close this section by mentioning a few other approaches for solving or relaxing QCQPs. Saxena et al. [386] study techniques for deriving disjunctive cuts based on the nonconvex constraint $xx^T - X \succeq 0$, which is also implied by the constraint $X = xx^T$. Vandenbussche and Nemhauser [431, 432] 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 [308] studied a simplicial branch-and-bound algorithm for QCQP. Burer and Letchford [103] study relaxations of QCQPs with unbounded integer decision variables. The general α-BB relaxation approach Androulakis et al. [19] has also been used for solving QCQPs, although Anstreicher [23] 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 [333] study the additional use of piecewise linear and edge-concave relaxations of QCQPs. Bao et al. [37] and Luedtke et al. [314] consider related approaches for relaxing *multilinear* functions in which the decision variables are bounded, which can also be applied to QCQPs.

Bilinear covering sets. Tawarmalani et al. [420] 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 pure integer covering set defined below:

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

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

$$a^k x_i + b^k y_i \geq 1, \quad k = 1, \ldots, d \tag{17.20}$$

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 $\pi : \{1, \ldots, n\} \to \{1, \ldots, d\}$. That is, if $\pi \in \Pi$, then for each $i \in 1, \ldots, n$, $\pi(i)$ selects an inequality in the description of $\mathrm{conv}(B^{\mathrm{I}}_i)$. Then, $\mathrm{conv}(B^{\mathrm{I}})$ is characterized as follows.

Theorem 17.1.2 (Proposition 6 in Tawarmalani et al. [420]) *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*

$$\sum_{i=1}^{n} (a^{\pi(i)} x_i + b^{\pi(i)} y_i) \geq 1, \quad \forall \pi \in \Pi. \tag{17.21}$$

While there are an exponential number of inequalities in (17.21), 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 (17.20). Tawarmalani et al. [420] provide a 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 turn next to the case of a continuous bilinear covering set of the form

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

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

Theorem 17.1.3 (Proposition 9 in Tawarmalani et al. [420]) *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:*

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

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 analyze. In particular, Chung et al. [121] show that optimizing a linear function over such a set is $\mathcal{NP}$-hard. Chung et al. [121] did, however, study valid inequalities for this set that may be useful in strengthening the continuous relaxation.

Chapter 18

Heuristics for Mixed-Integer Optimization

Heuristics for mixed-integer nonlinear optimization play an important role, because they provide initial feasible points quickly. We discuss rounding-based heuristics, the feasibility-pump, and relaxation-induced neighborhood search. The combination of heuristics with modern implementations is briefly discussed.

18.1 Heuristics for Solving MINLPs

Some real-world applications cannot be solved to global optimality by using the methods described in Sections 14.1 to 17.1, 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 14.2 and in the branch-and-cut algorithm of Section 14.2.4. An incumbent solution may additionally be used for "guided dives" [145], that is, selecting a child node during a dive. Bounds tightening described in Section 17.1.3 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 [279], ant colony optimization [159], particle-swarm optimization [272], cross-entropy [376], tabu search [211, 212] and genetic algorithms [213] are some methods that fall under 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 14.1 to 17.1 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 (13.1); x' refers to a (local) solution of the continuous relaxation (13.3); $x^\diamond$ denotes the solution to a polyhedral relaxation of

(13.1); 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.

18.1.1 Search Heuristics

Several heuristics to search for a feasible solution of a 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.

MILP-based rounding. Finding a locally optimal solution to the continuous relaxation (13.3) of the MINLP (13.1) 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 usually will not produce a feasible solution. Nannicini and Belotti [341] propose to overcome this difficulty by solving an MILP. The constraints of the MILP are linear relaxations of the original MINLP obtained by methods described in Section 17.1.2. 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 $x_I^\diamond$. The process is repeated until we obtain a feasible solution or reach termination criteria (limits on time or iterations).

Nannicini and Belotti [341] 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

$$\sum_{i \in I} |x_i - x_i^\diamond| \geq 1. \tag{18.1}$$

When all integer variables are binary, the constraint (18.1) is simplified to

$$\sum_{i \in I : x_i^\diamond = 0} x_i + \sum_{i \in I : x_i^\diamond = 1} (1 - x_i) \geq 1. \tag{18.2}$$

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

Feasibility pump. The feasibility pump heuristic was introduced by Fischetti et al. [182] in the context of MILP and improved by Achterberg and Berthold [6] and Fischetti and Salvagnin [181]. Bonami et al. [87] extend 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 to the MINLP. The main difference from the rounding approach of Section 18.1.1 is the way MILP is set up. Suppose x' is a locally optimal solution of the NLP relaxation of the MINLP (13.1). Bonami et al. [87] obtain the MILP using the linearization (15.1) 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. [87] 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 finite number of iterations. They call their version of feasibility pump for convex MINLPs "enhanced feasibility pump." A simpler version of the feasibility pump [85] 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 [182].

D'Ambrosio et al. [143] note that feasibility-pump-based heuristics can be viewed as applications of the sequential projection method (SPM) or the alternating projection methods. SPM has been used extensively for solving convex feasibility problems. We refer to the survey by Bauschke and Borwein [42] 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 to even converge. D'Ambrosio et al. [143] consider two sets related to the feasible region of the MINLP (13.1):

$$A = \{x \mid c(x) \leq 0, x \in X\}, \text{and} \tag{18.3}$$
$$B = \{x \mid c_C(x) \leq 0, x \in X, x_i \in \mathbb{Z}\ \forall\ i \in I\}, \tag{18.4}$$

where c_C refers to the convex constraints in the MINLP. Set A is in general nonconvex, 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 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. [143] 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 14.1. By selecting different methods to solve these problems, one can obtain several variants of the feasibility pump.

Undercover. The Undercover heuristic [61] is specially designed for nonconvex 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 [61] 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. The authors 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

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & x_1^3 + x_2 + x_3 x_4 + x_5 \\ \text{subject to} \quad & x_4 x_5 \geq 1 \\ & x_1, x_2, x_3, x_4, x_5 \in [0, 10] \end{aligned}$$

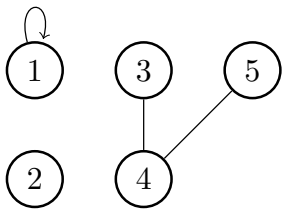


Figure 18.1: Graph (right) denoting the nonzero structure of the Hessian of the example on the left.

Lagrangian of the NLP relaxation has a nonzero entry (i, j). G may also contain loops if a diagonal entry in the Hessian is nonzero. We illustrate a graph for a simple example below. 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 [61] 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 a 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.

RENS: Relaxation Enforced Neighborhood Search. The RENS [60] 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', one has $2^{|F|}$ 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 [60] 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 a 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 the test-instances have solutions that could be obtained by rounding.

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 [85] propose to start 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 resolve 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 such as a limit on the number of iterations is met.

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 [85] 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

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

  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 nonzero 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 nonnegative, and it is rounded down otherwise. The selection favors 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 a 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 resolves of the modified NLP. To mitigate this cost, Bonami and Gonçalves [85] propose to fix $K > 1$ variables at the same time before resolving the modified NLP. Mahajan et al. [319] propose to solve QPs instead of NLPs, which speeds up the diving process by exploiting the warm starting capabilities of active set QP solvers.

18.1.2 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 neighborhood of a known solution have been adapted from MILP to MINLP. We describe them next.

Local branching. Local branching is a heuristic for MINLPs, where all integer variables are binary, that is, $x_i \in \{0, 1\}, \ \forall i \in I$. It was first introduced in the context of MILP by Fischetti and Lodi [180] and generalizes readily to convex MINLPs. We start by describing local branching for convex MINLPs and then describe an extension to nonconvex 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 (13.1), and consider the following disjunction (generalized branching) for a fixed constant $k \in \mathbb{Z}$:

$$\|x_I - x_I^*\|_1 \leq k \ \text{ (left branch) or } \ \|x_I - x_I^*\|_1 \geq k + 1 \ \text{ (right branch).} \tag{18.5}$$

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 (18.5) as two linear constraints:

$$\sum_{i \in I : x_i^* = 0} x_i + \sum_{i \in I : x_i^* = 1} (1 - x_i) \leq k \ \text{ (left) or } \sum_{i \in I : x_i^* = 0} x_i + \sum_{i \in I : x_i^* = 1} (1 - x_i) \geq k + 1 \ \text{ (right).} \tag{18.6}$$

The left branch is constructed in such a way that it is much easier to solve than (13.1), typically by choosing $k \in [10, 20]$. We start by solving the left branch using any of the methods introduced in Section 14.1 and

obtain a new incumbent. We can then either solve the right branch or again divide the right branch using a new disjunction (18.6). 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 [180] 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 nonconvex MINLPs [342]. 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 18.1.1. The local branching is used only as a no-good cut (18.1) and not viewed as a strategic (outer) branching technique. A related heuristic is RECIPE [305].

RINS: Relaxation-Induced Neighborhood Search. In the RINS [145] heuristic, one searches for better solutions in the neighborhood of an already known solution, much like local branching. However, instead of imposing a distance constraint (18.5) to determine a neighborhood, 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 [85] extend this idea from MILP to the NLP-based branch-and-bound algorithm of Section 14.2 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 15.2.1. They show that the LP/NLP-BB algorithm is much faster on the smaller problems than is the NLP-based branch-and-bound algorithm.

Chapter 19

Mixed-Integer PDE Constrained Optimization

This chapter addresses what is arguably one of the most difficult optimization problems. We consider problems that mix integer variables and patrial-differential (PDE) constraints. This is a challenging class of problems for which few, if any solvers exists. We discuss important applications, problem formulations, and discuss some preliminary solution approaches.

19.1 Introduction and Background

Many complex science and engineering applications can be formulated as optimization problems, constrained by partial differential equations (PDEs), that involve both continuous and integer variables. This new class of problems, called mixed-integer PDE-constrained optimization (MIPDECO) [301], must overcome the combinatorial challenge of integer decision variables combined with the numerical and computational complexity of PDE-constrained optimization. We briefly review the existing literature in terms of application and solution approaches to MIPDECO, and present a collection of test problems. Preliminary numerical results indicate that this is a tremendously challenging new class of problems.

MIPDECO have the potential to impact a broad range of science and engineering applications. For instance, the design of nuclear plants depends on selecting different types of core (fuel rod) configurations while controlling flow rates to maximize the heat extraction process [127]. Remediation of contaminated sites and maximizing oil recovery both involve flow through porous media to determine the number of wellbores in addition to calculating optimal flow rates [179, 354], and operational schedule [35, 36, 46]. Related applications also arise in the optimal schedule of shale-gas recovery [394]. Next-generation solar cells face complicated geometric and discrete design decisions to achieve perfect electromagnetic performance [372]. In disaster-recovery scenarios, such as oil spills [454, 455], wildfires [158], and hurricanes [295], resources need to be scheduled for mitigation purposes while predicting material properties to calibrate the underlying dynamics for accurate forecasts. Many other science and engineering examples have similar decision-making characteristics including wind farm design [458], climate science, and the design, control, and operation of gas networks [151, 165, 325, 411, 457]. The common theme of these applications is a need to address integral and continuous optimization variables in the context of large-scale multi-physics applications.

Very little work has been done in this area, partly because of the overwhelmingly large computational requirements and because, historically, the two research communities — PDE-constrained optimization and mixed-integer nonlinear programming — have developed algorithms separately. The solution of discrete optimization variables has been studied in isolation of partial differential equations and the solution of

large-scale optimization has mostly ignored integer variables. Moreover, even the underlying computational kernels of MIP and PDEs are fundamentally incompatible. For example, PDE solver employ iterative methods to solve the linear systems, which are often too large to handle with direct methods. On the other hand, MIP solvers rely on fast re-optimization and pivoting methods that require rank-one updates of the basis factors. Furthermore, the computational expense of either combinatorial or PDE-constrained optimization is sufficiently large that the combination of the two has been completely discounted. However, the continuing improvements in computational power, provides a timely opportunity to develop new mathematics, algorithms, and software capabilities to address MIPDECO. The goal of this chapter is to present a library of challenging test problems to motivate this research, and to highlight the impact of modeling choices at the interface of mixed-integer programming and PDE-constrained optimization.

PDE-Constrained Optimization Background. PDE-constrained optimization refers to the optimization of systems governed by partial differential equations. In most cases the goal is to optimize an objective function with respect to a quantity that lives in subregions or everywhere in the computational domain. The inversion for initial conditions or the reconstruction of material properties are examples of typical optimization problems. The large scale of the optimization variables dictates the use of efficient sensitivity methods (adjoints), Newton-based methods to handle the nonlinearity of the optimization formulation, coordinate globalization, and parallel matrix-vector operators to address the computational requirements [69, 70, 349]. Considerable advances have been made to accelerate the convergence of these algorithms. Special preconditioners, reduced-space approaches, full-space methods, and multigrid approaches are recent examples of the most promising developments, see [12, 41, 76, 77, 92, 242, 402].

Mixed-Integer Nonlinear Programming Background. Nonlinear MIPs are a challenging class of problems in their own right: they are in general NP-hard [267] and worse undecidable [264]. Most nonlinear MIP methods use a tree search to resolve the integrality restrictions. We distinguish three basic classes of methods: branch-and-bound or single-tree methods, multi-tree methods such as outer approximation, and hybrid techniques. Branch-and-bound [140, 234] searches a tree where each node corresponds to a nonlinear subproblem. Branching corresponds to adding integer bounds on fractional integer variables, that separate the fractional solution from the integer feasible set, creating two new nonlinear subproblems. Branch-and-bound methods can be improved by adding cutting planes [13, 116, 161, 162, 198, 231, 413] to tighten the continuous relaxations, resulting in a smaller tree that needs to be searched. Outer approximation [163], Benders decomposition [207], and the extended cutting plane method [412] are multi-tree techniques. These methods define a linear MIP master problem that can be solved more efficiently using commercial solvers. The solution of the MIP master problem typically violates the nonlinear constraints, and new linearizations obtained from the solution of a nonlinear subproblem are added to the MIP master problem, resulting in an alternating sequence of linear MIP and nonlinear optimization subproblem. Hybrid methods [1, 86, 366] combine nonlinear branch-and-bound with methods such outer approximation, and for the basis of the most efficient nonlinear MIP solvers [1, 86]. LP/NLP-based branch and bound starts by solving an initial linear MIP master problem. it. Whenever a new integer assignment is found, the linear tree-search is interrupted, and a nonlinear problem is solved obtained by fixing all integer variables to this assignment. The master problem is then updated by adding outer approximations from the solution of the nonlinear problem, see also More details can be found in the monographs [193, 417], the collection [293], and the survey papers [53, 102, 107, 228, 229].

Outline. This chapter is organized as follows. In the remainder of this section, we formally define MIPDECOs. We then briefly discuss some pertinent background concepts from PDE constrained optimization and MINLP, and present an extensible problem characterization for MIPDECOs. In Section 19.3 we present our

test problems, and describe simple discretization schemes that allow us to formulate AMPL [196] models. Efficient solution approaches, however, are still an open research question, because current solvers fail to solve this class of problems satisfactorily.

19.2 Problem Definition, Challenges, and Classification

We start by defining a generic MIPDECO problem, discuss the theoretical and computational challenges arising from this class of new problems, and then provide a problem classification.

19.2.1 Definition of MIPDECO

We formulate mixed-integer PDE-constrained optimization problems as

$$\begin{aligned} \underset{u,w}{\text{minimize}} \quad & \mathcal{F}(u,w) && \text{(19.1a)}\\ \text{subject to} \quad & \mathcal{C}(u,w) = 0, && \text{(19.1b)}\\ & \mathcal{G}(u,w) \leq 0, && \text{(19.1c)}\\ & u \in \mathcal{D}, \quad \text{and} \quad w \in \mathbb{Z}^p \ \ \text{(integers)}, && \text{(19.1d)} \end{aligned}$$

which is defined over a domain Ω. We use x, y, z to indicate spatial coordinates of the domain Ω, and t to denote time. The objective function of (19.1) is $\mathcal{F}$, $\mathcal{C}$ are the equality constraints, and $\mathcal{G}$ are inequality constraints. The equality constraints include the PDEs as well as boundary and initial conditions. We denote the continuous decision variables of the problem by $u(t, x, y, z)$, which includes the PDE states, controls, and design parameters. We denote the integer variables by $w(t, x, y, z)$, which may include design parameters that are independent of (t, x, y, z). Thus, in general, problem (19.1) is an infinite-dimensional optimization problem, because the unknowns, (u, w), are functions defined over the domain Ω. All the application problems summarized in Section 19.1 can be encapsulated in this mathematical form. We avoid a formal discussion of function spaces in this chapter, and instead concentrate on practical formulations.

19.2.2 Classification of MIPDECO

We classify MIPDECOs by the type of PDE, the class of integer variables, and the functional form of the objective and the constraints. The goal of this classification is to provide a short overview of the pertinent features of each test problem presented in Section 19.3. Before presenting our classification, we define mesh-dependent and mesh-independent integer variables, which is an important part of our classification.

Definition 19.2.1 *We say that the integer variables $w(t, x, y, z) \in \mathbb{Z}^p$ in (19.1) are* mesh-independent*, if and only if, they do not depend on (t, x, y, z), and we can thus write $w(t, x, y, z) = w$. Otherwise, the integer variables $w(t, x, y, z) \in \mathbb{Z}^p$ in (19.1) are called* mesh-dependent*.*

Figure 19.1 illustrates the difference between mesh-independent and mesh-dependent integer variables. The left two images show mesh-independent integer variables (where w is defined at the locations shown by blue dots). As we refine the mesh that discretizes the PDE, the number of integer variables is unchanged. The right two images show a mesh-dependent integer variable, which is defined at every mesh point. As we refine the mesh, we increase the number of discretized integer variables. Examples of mesh-dependent integers include topology optimization, and the inversion of material types, while examples of mesh-independent integers include the identification of source terms from a fixed number of possible locations. Finally, we

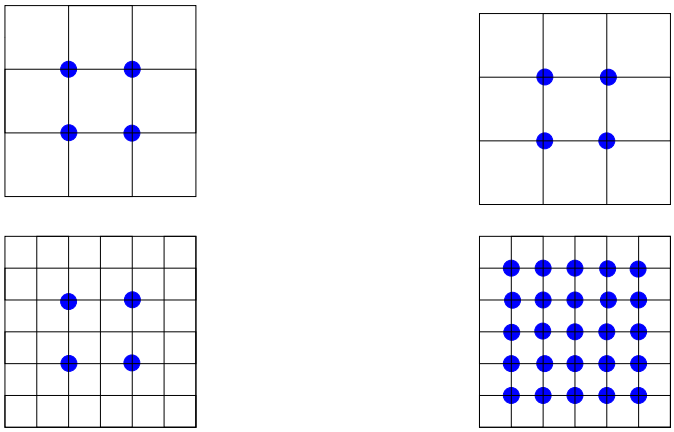


Figure 19.1: The left two images show mesh-independent integer variables (where w is defined at the locations shown by blue dots), and the right two images show a mesh-dependent integer variable.

observe that the case of time-dependent (but not mesh-dependent) integers corresponds to the special case of mixed-integer optimal control, see e.g. [379].

MIPDECO problems with mesh-dependent integers provide additional challenges, because the number of discretized integer variables grows as we refine the mesh, making it impractical to solve large instances with existing solvers.

We now introduce our classification of MIPDECOs, based on the following five features.

Type of PDE We distinguish different classes of PDEs such as elliptic, parabolic, and hyperbolic problems. We also distinguish linear and nonlinear PDEs, and the form of the boundary conditions.

Class of Integers We classify problems firstly based on whether the integer decision variables are mesh-dependent or mesh-independent. In addition, we distinguish binary variables, general integer variables, special-ordered sets, and semicontinuous variables.

Type of Objective We characterize problems according to the functional form of the objective, which could be constant, linear, quadratic, sum-of-squares, or nonlinear. In addition, we also classify problems according to their PDE objective class, which could be of tracking type, an inverse problem, and contain a regularization term.

Type of Constraints In addition to the PDE constraint, it is useful to classify problems according to other types of constraints such as bounds on the controls or states, general nonlinear constraints, linear constraints, knapsack constraints, generalized upper bound constraints, etc.

Discretization Describes the method used to discretize the PDE, such as finite difference or finite element methods, and characterizes the type of optimization problem that is obtained after discretization, using the CUTEr classification scheme [128].

Our goal is to provide a short description that captures the pertinent features of each problem, and that illustrates which parts could potentially be exploited in a solution approach. Note, that we do not include the discretization within this classification, because the discretization provides a solution approach. However, we can easily imagine situations, where different discretization schemes might lead to finite-dimensional nonlinear optimization problems with different problem characteristics. Finally, we deliberately avoided using the weak form of the PDE, because in many applications, the PDE provides a more natural way to express the problem.

19.2.3 Challenges of MIPDECO

The MIPDECO (19.1) combines two powerful modeling paradigms, namely PDEs and mixed-integer optimization. As a result, our efforts to solve problems of this kind face a number of theoretical and computational challenges:

1. The presence of mesh-dependent integer variables, $w(t, x, y, z) \in \mathbb{Z}$, raises some theoretical challenges that are beyond the scope of this chapter. In particular, it is not clear how to characterize the function space of these variables, and we conjecture that even

$$\big\{w(t, x, z, y, z) : \Omega \to \{0, 1\}\big\} \not\subset L_2(\Omega),$$

   where Ω is the domain of the underlying PDE, and $L_2(\Omega)$ is the set of square integrable functions. In particular, if we consider $w_c(x, y, z)$ to be the indicator function of the Cantor set, then it is well known that the Cantor set is not integrable, and hence it follows that $w_c \notin L_2(\Omega)$.

   In practice, we may need to add additional assumptions or regularization terms to ensure that $w \in L_2(\Omega)$. This property is important, because it allows us to construct consistent approximations on a sequence of meshes that converge to a consistent limit.

2. The possibility of coupling between integer variables and discretization raises another set of theoretical concerns. In certain PDEs, such as the wave equation, the correct discretization depends on the direction of flow (e.g. upwinding scheme for the wave equation). On the other hand, the direction of flow may depend on the binary or integer variables. One example, where this occurs are gas networks where integer variables control valves and compressors that affect the direction of flow through the pipes. Thus, we must take the value of binary variables into account when discretizing the PDE.

3. Computational challenges arise from the potentially large branch-and-bound trees, in particular for problems with mesh-dependent integers. For example, topology optimization can easily involve millions of binary variables, resulting in trees that are too large to handle. Thus, we may need to develop new search paradigms that scale to millions of binary variables.

4. MIPDECOs also pose a computational paradox. On the one hand, we need to use iterative solvers to efficiently resolve the PDE, on the other hand, these techniques are not efficient for warm-starting solves within a search tree. Developing efficient tree-search strategies that efficiently re-use information from the parent node is an important challenge of MIPDECO.

5. Finally, global solution guarantees are likely to be difficult to obtain for nonlinear PDEs. Standard global optimization techniques such as factorable programming are unlikely to scale to discretized nonlinear PDEs, because we would need to construct outer approximations of the computational graph on each element.

These challenges indicate that MIPDECO is a nontrivial class of problems. However, we believe that these challenges are not insurmountable. On the other hand, MIPDECOs have a range of important applications. In Section 19.3 we present a set of test problems to motivate further research into MIPDECO.

19.2.4 Eliminating State Variables

Many MIPDECOs have specially structured PDE constraints that can be exploited to eliminate the state variables, u. In particular, consider (19.1) without inequality constraints on the states, and where the PDE is

linear, and the binary controls, w, appear only on the right-hand side, i.e.

$$\begin{aligned} \underset{u,w}{\text{minimize}} \quad & \mathcal{F}(u,w) && \text{(19.2a)} \\ \text{subject to} \quad & \mathcal{A}u = c(w), && \text{(19.2b)} \\ & u \in \mathcal{D}, \quad \text{and} \quad w \in \{0,1\}, && \text{(19.2c)} \end{aligned}$$

which can also contain additional constraints on w.

After discretization, we obtain a finite-dimensional approximation of (19.2), which is a standard NLP:

$$\begin{aligned} \underset{\boldsymbol{u},\boldsymbol{w}}{\text{minimize}} \quad & F(\boldsymbol{u},\boldsymbol{w}) && \text{(19.3a)} \\ \text{subject to} \quad & A\boldsymbol{u} = \sum_{k,l} w_{kl}\boldsymbol{c}_{kl}, && \text{(19.3b)} \\ & w_{kl} \in \{0,1\}, && \text{(19.3c)} \end{aligned}$$

where $\boldsymbol{u} = (u_{ij})_{ij}$ and $\boldsymbol{w} = (w_{kl})_{kl}$ approximate the functions u and w, respectively. A is the discretization of the PDE operator (including boundary and initial conditions), and the vectors $\boldsymbol{c}_{kl}$ with has components $f_{kl}(ih, jh)$. Because A is nonsingular, it follows that we can eliminate the discretized states, $\boldsymbol{u}$, by noting that

$$A\boldsymbol{u} = \sum_{k,l} w_{kl}\boldsymbol{c}_{kl} \;\Leftrightarrow\; \boldsymbol{u} = A^{-1}\left(\sum_{k,l} w_{kl}\boldsymbol{c}_{kl}\right) = \sum_{k,l} w_{kl}A^{-1}\boldsymbol{c}_{kl}$$

If we denote the solution of the PDE for each right-hand-side $\boldsymbol{c}_{kl}$ as $\boldsymbol{u}^{(kl)} := A^{-1}\boldsymbol{c}_{kl}$. Then we can write $\boldsymbol{u}$ as the following sum:

$$\boldsymbol{u} = \sum_{k,l} w_{kl}\boldsymbol{u}^{(kl)}$$

and eliminate the state variables $\boldsymbol{u}$ from (19.3), resulting in the following (purely) binary knapsack problem:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & F\left(\sum_{k,l=1}^{N} w_{kl}u_{i,j}^{(k,l)}\;,\; \boldsymbol{w}\right) && \text{(19.4a)} \\ \text{subject to} \quad & w_{kl} \in \{0,1\}. && \text{(19.4b)} \end{aligned}$$

We note, that this approach is efficient, because we only need to solve at most $N \times N$ PDEs, where N is the discretization size, to precompute the solutions, but can then run the MIP solvers on a much simpler problem. The following proposition summarizes this result.

Proposition 19.2.1 *The two MINLP problems* (19.3) *and* (19.4) *are equivalent in the sense that* w_{kl}^* *solves* (19.3)*, iff it solves* (19.4)*.*

We also observe, that the two problems will have identical search trees, provided every node is solved exactly. Thus, our elimination of variables has no negative side effect, and has the potential to solve problems much faster (as long as the tree is larger than N^2 (the number of PDE solves required to precompute the solutions).

Remark 19.2.1 *Note that our elimination of the state variables only depends on the linearity of the PDE, the linear dependence of the right-hand-side on the controls,* w*, and the fact that* A *is nonsingular. Thus, our result covers both steady-state problems and time-dependent problems, and extends to the case where the controls affect the initial or boundary conditions linearly.*

To the best o our knowledge, this type of elimination of continuous variables is new to mixed-integer optimization. The elimination of state variables is related to the elimination of state variables and Lagrange multipliers in PDE-constrained optimization, see e.g. [78, 79].

19.3 MIPDECO Test Problems

We describe several families of MIPDECO test problems. Each family corresponds to a particular PDE parameterized with suitable boundary and initial conditions to provide a rich set of challenging test problems. In each case, we relate the mathematical problem to our AMPL model.

19.3.1 Laplace Source Inversion Problem

Our first model is a source inversion problem based on Laplace equation with Dirichlet boundary conditions. Binary variables select the sources from a set of of possible sources to match an observed solution, $\bar{u} = \bar{u}(x, y)$ in the domain $\Omega = [0, 1]^2$. The model is motivated by an application to determine an appropriate number of boreholes in subsurface flow [179, 354].

We present two classes of models with mesh-dependent and mesh-independent integer variables, respectively. We also consider variants that include a regularization term, and a formulation that eliminates the state variables.

Table 19.1: Problem Characteristics for Inverse Laplace Problem.

Type of PDE	Laplace equation on $[0, 1]^2$ with Dirichlet boundary conditions
Class of Integers	Mesh-dependent & mesh-independent binary variables
Type of Objective	Least-squares (inverse problem) & regularization term
Type of Constraints	Knapsack constraint on binary variables
Discretization	Five-point finite difference stencil; **SLR-AN-V-V**

Mesh-Independent Problem Variant and Discretization. We let $\mathcal{L}$ denote the *finite* set of possible source-term locations. The set of potential source term location is illustrated in Figure 19.2, and consists of 36 points in four groups of 3×3 grids of size $1/8$ centered in each quadrant of $[0, 1]^2$. The target or observed solution $\bar{u}$ is constructed from four sources defined in (19.6)

$$(x_k, y_l) \in \mathcal{L} := \left\{ (3/16, 5/16), (5/16, 13/16), (13/16, 11/16), (11/16, 3/16) \right\}.$$

By constructing the forward solution $\bar{u}$ from sources that are not exactly representable from the finite set of source locations, we ensure that the inverse problem has no exact recovery, making the inversion nontrivial.

With a slight abuse of notation, where $\mathcal{L}$ also denotes the index set of possible source location, we can

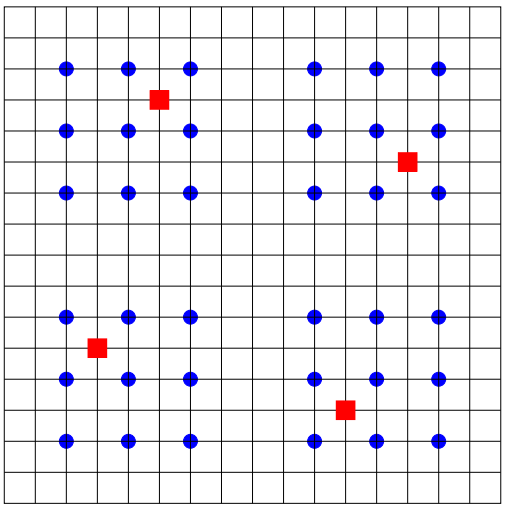


Figure 19.2: Distribution of sources (red squares) and potential source locations for inverse model (blue dots) on a 16×16 mesh.

formulate the mesh-independent source-term inversion problem as

$$\begin{aligned}
\underset{u,w}{\text{minimize}} \quad & \mathcal{J} = \int_\Omega (u - \bar{u})^2 d\Omega && \text{least-squares fit} && \text{(19.5a)}\\
\text{subject to} \quad & -\Delta u = \sum_{(k,l)\in\mathcal{L}} w_{k,l} f_{k,l} \ \text{ in } \ \Omega && \text{Poisson equation} && \text{(19.5b)}\\
& \sum_{(k,l)\in\mathcal{L}} w_{k,l} \le S \text{ and } w_{k,l} \in \{0,1\} && \text{source budget.} && \text{(19.5c)}
\end{aligned}$$

We impose Dirichlet boundary conditions, $u = 0$ on the boundary $\partial\Omega$ of $\Omega = [0,1]^2$. The number of source terms in (19.5) are limited by a budget $S = 4$ (or $S = 5$), and the binary variables $w_{k,l} \in \{0,1\}$ multiply the source terms $f_{k,l}(x,y)$, which are given by Gaussians centered at a finite set of points $(x_k, y_l) \in \Omega$:

$$f_{kl}(x,y) := f(x - x_k, y - y_l) := \exp\left(- \left\| \begin{pmatrix} x - x_k \\ y - y_l \end{pmatrix} \right\|^2 \Big/ \sigma^2 \right), \tag{19.6}$$

where $\sigma > 0$ is the fixed variance, chosen as $\sigma = 3/16$ in our examples.

We discretize the two-dimensional PDE in (19.9) using a five-point finite-difference stencil, with uniform mesh-size $h = 1/N$ for some integer N. We denote by $u_{i,j} \approx u(ih, jh)$ the approximation at the grid points, and obtain the finite-dimensional MINLP,

$$\begin{aligned}
\underset{u,w}{\text{minimize}} \quad & \mathcal{J} = \sum_{i,j=0}^{N} \left(u_{i,j} - \bar{u}_{i,j}\right)^2 && \text{(19.7a)}\\
\text{subject to} \quad & \frac{4u_{i,j} - u_{i,j-1} - u_{i,j+1} - u_{i-1,j} - u_{i+1,j}}{h^2} = \sum_{(k,l)\in\mathcal{L}} w_{k,l} f_{k,l}(ih, jh) && \text{(19.7b)}\\
& u_{0,j} = u_{N,j} = u_{i,0} = u_{i,N} = 0 && \text{(19.7c)}\\
& \sum_{(k,l)\in\mathcal{L}} w_{kl} \le S \text{ and } w_{kl} \in \{0,1\}. && \text{(19.7d)}
\end{aligned}$$

We also define a variant of the model in which we add a regularization term for the controls to the objective function of (19.5), given by

$$\beta \left\| \sum_{k,l} w_{k,l} f_{k,l} \right\|$$

for some suitable norm. In the case of the squared $L_2(\Omega)$-norm this results in the addition of the term

$$\beta \sum_{i,j} \left(\sum_{k,l} w_{k,l} f_{k,l}(ih, jh) \right)^2 . \tag{19.8}$$

This regularization does not change the characteristic of the resulting nonlinear MIP.

Mesh-Dependent Problem Variant and Discretization. The mesh-dependent variant is defined by allowing source terms to be at any location, i.e. $w(x, y) \in \{0, 1\}$. This change affects the right-hand-side in (19.5b), and the source budget constraint, (19.5c), which are no longer finite sums, but must be represented by integrals. We first define the right-hand-side of the PDE as

$$F(x, y) := \int_{(\xi,\eta)\in\Omega} w(\xi, \eta) f(x - \xi, y - \eta) d\xi d\eta,$$

which is the convolution of f with w. Given this definition, the mesh-dependent version becomes

$$\begin{aligned}
\underset{u,w}{\text{minimize}} \quad & \mathcal{J} = \int_\Omega (u - \bar{u})^2 d\Omega && \text{least-squares fit} && (19.9a)\\
\text{subject to} \quad & -\Delta u = F \text{ in } \Omega && \text{Poisson equation} && (19.9b)\\
& \int_{(x,y)\in\Omega} w(x, y) dx dy \le S \text{ and } w(x, y) \in \{0, 1\} && \text{source budget.} && (19.9c)
\end{aligned}$$

As before, we discretize the PDE using a five-point finite-difference stencil, with uniform mesh-size $h = 1/N$, and we denote by $u_{i,j} \approx u(ih, jh)$ the approximation at the grid points, to obtain the MINLP,

$$\begin{aligned}
\underset{u,w}{\text{minimize}} \quad & \mathcal{J} = \sum_{i,j=0}^{N} (u_{i,j} - \bar{u}_{i,j})^2 && (19.10a)\\
\text{subject to} \quad & \frac{4u_{i,j} - u_{i,j-1} - u_{i,j+1} - u_{i-1,j} - u_{i+1,j}}{h^2} = \sum_{k,l=1}^{N} w_{k,l} f_{k,l}(ih, jh) && (19.10b)\\
& u_{0,j} = u_{N,j} = u_{i,0} = u_{i,N} = 0 && (19.10c)\\
& \sum_{k,l=1}^{N} w_{k,l} \le S \text{ and } w_{k,l} \in \{0, 1\}. && (19.10d)
\end{aligned}$$

We define two sets of mesh-dependent instances. In the first set, all internal mesh points are potential source locations. In the second set, the meshpoints with odd indices are potential source locations. This setup allows us to explore the behavior of methods as we vary the proportion of binary variables to continuous variables.

Table 19.2: Definition of $\bar{u}$.

Instance 1		Instance 2		Instance 3	
Coord.	Weight	Coord.	Weight	Coord.	Weight
(N/8 , N/8)	1/3	(3/16 , 5/16)	1	(N/8 , N/8)	1
(N/8 ,7*N/8)	1/3	(5/16, 13/16)	1	(N/4 , N/2)	1
(N/4 , N/2)	2/3	(13/16, 11/16)	1	(3*N/8,3*N/8)	1
(3*N/8,3*N/8)	2/3	(11/16, 5/16)	1	(N/2 , N/4)	1
(3*N/8,5*N/8)	3/4			(N/2 ,3*N/4)	1
(N/2 , N/4)	3/4			(5*N/8,5*N/8)	1
(N/2 ,3*N/4)	3/4			(3*N/4, N/2)	1
(5*N/8,3*N/8)	3/4			(7*N/8, N/8)	1
(5*N/8,5*N/8)	5/8				
(3*N/4, N/2)	7/8				
(7*N/8, N/8)	5/8				
(7*N/8,7*N/8)	7/8				

In both cases, we compute the inversion target, $\bar{u}$, by fixing weights at fractional values and then solving Poisson's equation. We choose fractional weights (that sum up to S) to ensure that an exact solution to the inversion is not feasible for the integer problem. The precise form of $\bar{u}$ is shown in Table 19.2.

It is possible to define several extensions of this model. For example, we can define a measurement area, $\mathcal{M} \subset \Omega$, and compute the error between u and $\bar{u}$ in that area. This scenario corresponds to a limited set of observations. Other extensions include an L_1 regularization term, which can be modeled as a smooth function using additional variables.

19.3.2 Distributed Control with Neumann Boundary Conditions

This problem is derived from the so-called "mother problem" presented in the OPTPDE library [352], see also [426]. It is a distributed optimal control problems where the PDE is Poisson equation with Neumann boundary conditions. Our version of this problem forces the control variables to be binary. We describe versions with both mesh-dependent and mesh-independent integers.

Table 19.3: Problem Characteristics for Mother Problem.

Type of PDE	Poisson equation on $[0, 1]^2$ with Neuman boundary conditions
Class of Integers	Mesh-dependent & mesh-independent binary variables
Type of Objective	Least-squares (inverse problem), regularization, and boundary integral term
Type of Constraints	Binary controls.
Discretization	Five-point finite difference stencil; **QLR-AN-V-V**

Distributed Control with Mesh-Dependent Binary Variables. We let the domain be $\Omega = [0, 1]^2$ with boundary Γ and center $\hat{x} = (0.5, 0.5)$. The state variables are denoted by $u \in H^1(\Omega)$, and the control

variables are $w \in \{0,1\}$ (with relaxation $w \in L^2(\Omega)$, see e.g. [426]). The model is stated as follows:

$$\begin{aligned}
\underset{u,w}{\text{minimize}} \quad & \frac{1}{2}\|u-u_\Omega\|^2_{L^2(\Omega)} + \int_\Gamma e_\Gamma\, u\, ds + \frac{1}{2}\|w\|^2_{L^2(\Omega)} && \text{least squares with reg.} && (19.11a)\\
\text{subject to} \quad & -\Delta u + u = w + e_\Omega \ \text{ in } \ \Omega && \text{Poisson equation} && (19.11b)\\
& \frac{\partial u}{\partial n} = 0 \ \text{ on } \ \Gamma && \text{Neumann boundary} && (19.11c)\\
& w(x) \in \{0,1\} \ \text{ in } \ \Omega && \text{bounds on control,} && (19.11d)
\end{aligned}$$

where the desired state is $\bar{u}(x) = -\frac{142}{3} + 12\,\|x-\hat{x}\|^2$, the uncontrolled force is $e_\Omega = 1 - \mathrm{proj}_{[0,1]}(12\,\|x-\hat{x}\|^2 - \frac{1}{3})$, see Figure 19.3, and the boundary observation coefficient is $e_\Gamma = -12$.

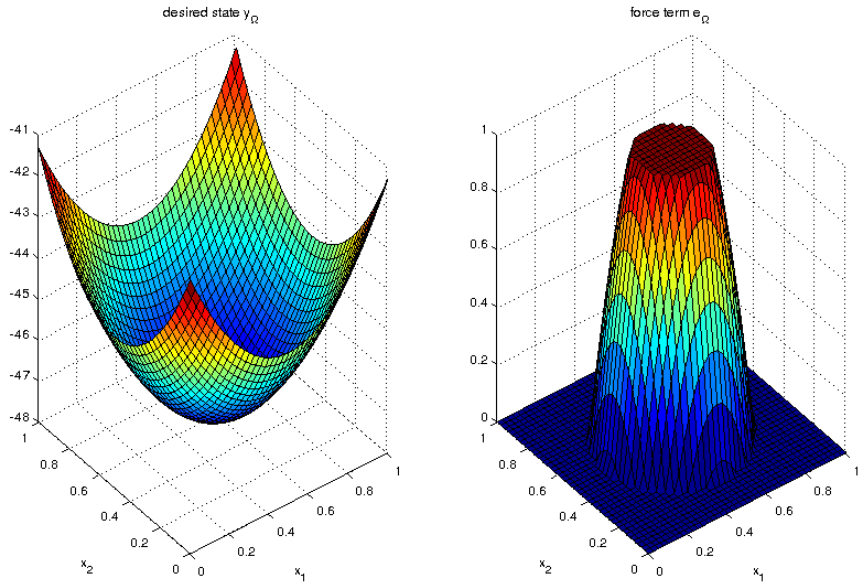


Figure 19.3: Desired state $\bar{u}$ and uncontrolled force term e_Ω on the computational doamin Ω.

An equivalent regularization is to use the L_1-norm, because $w(x)^2 = w(x)$ for any function $w(x) \in \{0,1\}$. This regularization has the advantage that its discretization is linear (rather than quadratic), which forces more binary variables to obtain discrete values, and hence dramatically reduces the size of the branch-and-bound tree. Hence, we can replace (19.11a) and arrive at the model

$$\begin{aligned}
\underset{u,w}{\text{minimize}} \quad & \frac{1}{2}\|u-u_\Omega\|^2_{L^2(\Omega)} + \int_\Gamma e_\Gamma\, u\, ds + \frac{1}{2}\|w\|_{L^1(\Omega)} \\
\text{subject to} \quad & (19.11b), (19.11c), (19.11d).
\end{aligned} \qquad (19.12)$$

We observe, that because $w(x) \geq 0$, we do not need to introduce additional variables to handle the absolute value in (19.12). We summarize these observations in the following proposition.

Proposition 19.3.1 *Problems* (19.11) *and* (19.12) *are equivalent in the sense that any optimal solution of one problem is also an optimal solution of the other problem.*

To solve problem (19.11) or (19.12), we use a five-point finite-difference stencil with a uniform mesh-size $h = 1/N$ for some integer N. We let $I = \{0, \ldots, N\}$, and denote by $u_{i,j} \approx u(ih, jh)$ the approximation at the grid point $(i, j) \in I \times I$. We implemented a symmetric discretization for the Neumann boundary conditions as follows:

$$u_{-1j} = u_{1j}, \quad u_{N+1j} = u_{N-1j}, \quad u_{j-1} = u_{j1}, \quad u_{jN+1} = u_{jN-1}, \qquad \text{for } j \in I.$$

Thus, we obtain the discretized form of (19.11) as

$$\underset{u,w}{\text{minimize}} \quad \frac{h^2}{2} \sum_{(i,j)\in I\times I} (u_{i,j} - u_{\Omega i,j})^2 - 12h \left(\sum_{i\in I} (u_{i,N} + u_{i,0}) + \sum_{i\in I} (u_{N,i} + u_{0,i}) \right) + \frac{h^2}{2} \sum_{(i,j)\in IxI} w_{i,j}^2 \tag{19.13a}$$

$$\text{subject to} \quad \frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j+1} - u_{i,j-1}}{h^2} = w_{i,j} + e_{\Omega i,j} - u_{i,j} \qquad \forall (i,j) \in I \times I \tag{19.13b}$$

$$u_{-1,j} = u_{1,j},\ u_{N+1,j} = u_{N-1,j},\ u_{j,-1} = u_{j,1},\ u_{j,N+1} = u_{j,N-1}, \qquad \forall j \in I \tag{19.13c}$$

$$w_{ij}(x) \in \{0, 1\} \qquad \forall (i,j) \in I \times I. \tag{19.13d}$$

In the discretized formulation (19.13) of the mother problem, the objective function (19.13a) is discretized on the domain and boundary. Equation (19.13a) is the five-point stencil discretization of the Poisson equation with a potential term, and the Neumann boundary conditions are discretized in (19.13c). We note, that in PDE textbooks, the indices $i, j = -1, N + 1$ are eliminated. However, we prefer the addition of extra constraints, because they are easier to implement in AMPL. If we use the L_1 regularization, then we replace the last term in th objective function of (19.13) by $\frac{h^2}{2} \sum w_{i,j}$.

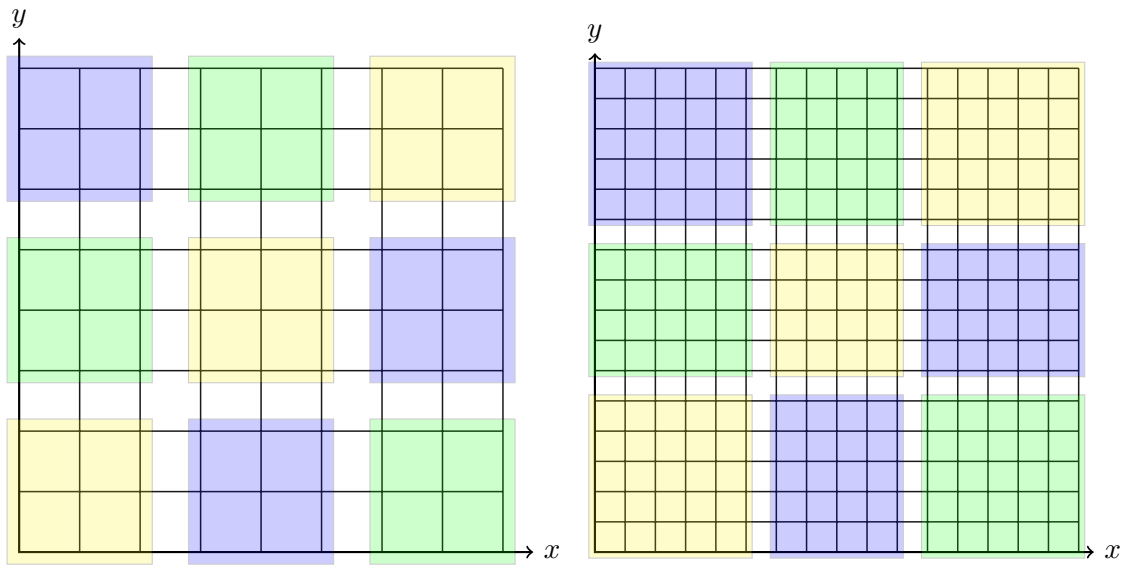


Figure 19.4: Definition of $P = 9$ patches for mother problem on 8×8 and 16×16 mesh.

Distributed Control with Mesh-Independent Binary Variables. We define a problem variant with mesh-independent integers by splitting the computational domain, Ω, into P patches, and requiring that the binary variables defined over each patch take the same value (this is equivalent to replacing $w(x) \in \{0,1\}$ by a piecewise binary function defined over each patch). The resulting problem has P binary variables independent of the mesh-size of the discretization (we assume that the discretization is finer than the P patches). Figure 19.4 shows the patches on an 8×8 and 16×16 mesh, and Figure 19.5 shows the patches on a 32×32 mesh.

We implement the mesh-independent patches by relaxing integrality of the control, w, and introducing a binary control defined on each patch, denoted by $v_k,\ \forall k = 1, \ldots, p$. Each patch then consists of the indices of mesh-points within the distinct regions regions in Figures 19.4 and 19.5. We then add the equations

$$w_{ij} = v_k,\ \forall (i,j) \in P_k,\ \forall k = 1, \ldots, P. \tag{19.14}$$

The additional controls are necessary, because AMPL does not automatically eliminate binary variables.

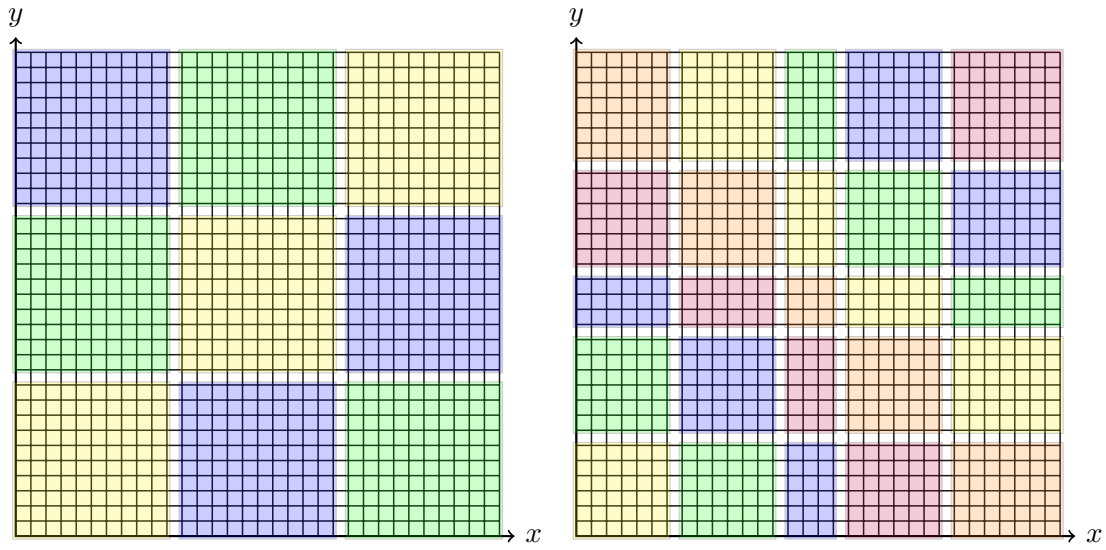


Figure 19.5: Definition of $P = 9$ and $P = 25$ patches for mother problem on 32×32 mesh.

19.3.3 Parabolic Robin Boundary Problem in Two Dimensions

This problem generalizes a 1D problem in the OPTPDE library [352] which is a classical parabolic Robin boundary control problem in one spatial dimension with control constraints, see also [425]. Our new model has two spatial dimensions. An important difference between the two models is the fact that the boundary controls in one dimension are applied everywhere, here, we consider discrete locations for the boundary controls. This also allows us to add knapsack constraints on the controls. The problem set-up is illustrated in Figure 19.6, where the computational domain is $[0, T_f] \times [0,1]^2$ for for some fixed final time T_f. The main problem characteristics are summarized in Table 19.4.

Problem Description and Discretization. We start by describing the infinite dimensional problem, then discuss variations, and finally present our discretized model. We let the state variables be denoted by

Table 19.4: Problem Characteristics for Robin Boundary Problem in Two Dimensions.

Type of PDE	Heat equation on $[0,1]^2$ with Neuman and Robin boundary conditions
Class of Integers	Mesh-dependent (t) binary variables
Type of Objective	Least-squares (inverse problem) & regularization term
Type of Constraints	Knapsack constraint on the discrete controls
Discretization	Forward-difference in time, central difference in space; **SQR-AN-V-V**

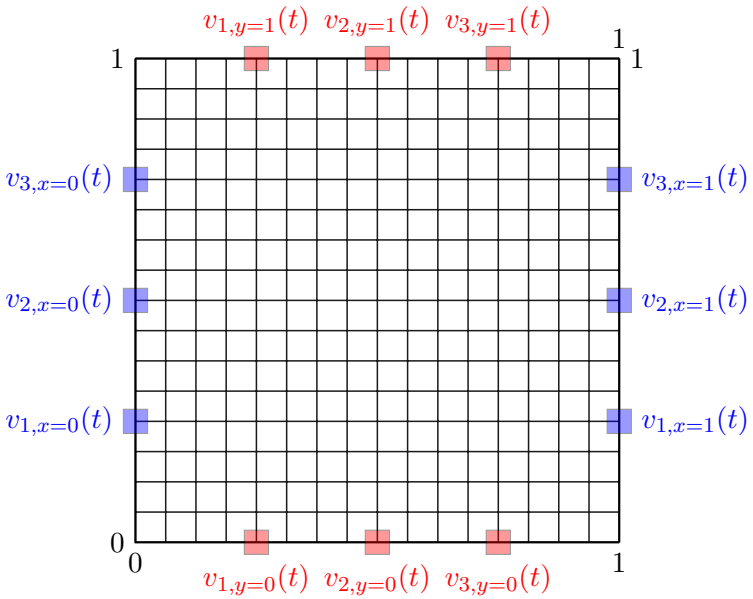


Figure 19.6: Illustration of control positions (red and blue squares) for Robin problem on 16×16 mesh.

$u(t,x,y)$, and denote the continuous controls by $v_l^k(t)$ for $l = 1,\ldots,N_c$ and $k = 0,1$ to denote the $y = 0$ and $y = 1$ boundary resectively, where $N_c > 1$ is the number of control locations. The binary controls that switch the $v_l^k(t)$ on and off are denoted by $w_l^k(t) \in \{0,1\}$ for $l = 1,\ldots,N_c$ and $k = 0,1$. We model the effect of the control using a Gaussian centered at the control boundary locations, $(x_l,0)$ and $(x_l,1)$, by defining

$$f_l(x) := \exp\left(\frac{(x-x_l)^2}{\sigma^2}\right)$$

for $l = 1, \ldots, N_c$, where $\sigma > 0$ is the variance, which we set to $\sigma = 3/256$. An infinite-dimensional description of the model is given by:

$$\underset{u,v,w}{\text{minimize}} \quad \frac{1}{2}\|u - u_d\|^2_{L^2(\Omega)} + \alpha \sum_{k=0}^{1} \sum_{l=1}^{N_c} \int_{t=0}^{T_f} \|v_l^k(t)\| dt \quad \text{least squares with reg.} \tag{19.15a}$$

$$\text{subject to} \quad \frac{\partial u}{\partial t} - \Delta u = 0 \ \text{ in } \ [0,T] \times [0,1]^2 \quad \text{Heat equation} \tag{19.15b}$$

$$u(0,x,y) = 0 \quad \text{in } \ [0,1]^2 \quad \text{Initial condition} \tag{19.15c}$$

$$\frac{\partial u}{\partial n}(t,0,y) = \frac{\partial u}{\partial n}(t,1,y) = 0 \text{ for } (t,y) \in (0,T) \times [0,1] \quad \text{Neumann boundary} \tag{19.15d}$$

$$\frac{\partial u}{\partial n}(t,x,0) = b\left(\sum_{l=1}^{N_c} v_l^0(t) f_l(x) - u(t,x,0)\right) \text{ for } (t,x) \in (0,T) \times [0,1] \quad \text{Robin boundary} \tag{19.15e}$$

$$\frac{\partial u}{\partial n}(t,x,1) = b\left(\sum_{l=1}^{N_c} v_l^1(t) f_l(x) - u(t,x,1)\right) \text{ for } (t,x) \in (0,T) \times [0,1] \quad \text{Robin boundary} \tag{19.15f}$$

$$w_l^k(t) \in \{0,1\} \in (0,T), \ \sum_{k=0}^{1} \sum_{l=1}^{N_c} w_l^k(t) \leq U \quad \text{bounds on control,} \tag{19.15g}$$

where U is an upper bound on the number of open controls in every time step.

We employ a forward difference discretization in time with time step $k := T_f/N_t$, and a five-point finite-difference stencil in space with $h = 1/M$, where $N_t, M > 0$ are the number of uniform steps in time and space. Denoting by $u_{ijt} \approx u(ih, jh, tk)$ the finite difference approximation, we arrive at the following

discretized MIPDECO:

$$\underset{u,v,w}{\text{minimize}} \qquad \frac{ph^2k}{2}\sum_{i,j=1}^{M}\sum_{t=0}^{N_t}\left(u_{ijt}-u_d(ih,jh,tk)\right)^2+\alpha k\sum_{t=0}^{N_t}\sum_{l=1}^{N_c}\left(v_{lt}^0+v_{lt}^1\right) \tag{19.16a}$$

$$\text{subject to} \qquad \frac{u_{i,j,t+1}-u_{i,j,t}}{k}-c\frac{u_{i+1,j,t+1}+u_{i-1,j,t+1}+u_{i,j+1,t+1}+u_{i,j-1,t+1}-4u_{i,j,t+1}}{h^2}=0 \tag{19.16b}$$

$$2hb\left(\sum_{l=1}^{N_c}v_{l,x=0}(tk)f_l(ih)-u_{i,0,t}\right)=u_{i,1,t}-u_{i,-1,t}\quad \forall t=0,\dots,N_t,\ i=1,\dots,M \tag{19.16c}$$

$$2hb\left(\sum_{l=1}^{N_c}v_{l,x=1}(tk)f_l(ih)-u_{i,M,t}\right)=u_{i,M-1,t}-u_{i,M+1,t}\quad \forall t=0,\dots,N_t,\ i=1,\dots,M \tag{19.16d}$$

$$2hb\left(\sum_{l=1}^{N_c}v_{l,y=0}(tk)f_l(jh)-u_{0,j,t}\right)=u_{1,j,t}-u_{-1,j,t}\quad \forall t=0,\dots,N_t,\ j=1,\dots,M \tag{19.16e}$$

$$2hb\left(\sum_{l=1}^{N_c}v_{l,y=1}(tk)f_l(jh)-u_{M,j,t}\right)=u_{M-1,j,t}-u_{M+1,j,t}\quad \forall t=0,\dots,N_t,\ j=1,\dots,M \tag{19.16f}$$

$$\sum_{l=1}^{N_c}\left(w_{l,x=0}(tk)+w_{l,x=1}(tk)w_{l,y=0}(tk)w_{l,y=1}(tk)\right)\le U\quad \forall t=0,\dots,N_t \tag{19.16g}$$

$$V_l w_{l,x=0}(tk)\le v_{l,x=0}(tk)\le V_u w_{l,x=0}(tk),\quad \forall t=0,\dots,N_t \tag{19.16h}$$

$$V_l w_{l,x=1}(tk)\le v_{l,x=1}(tk)\le V_u w_{l,x=1}(tk),\quad \forall t=0,\dots,N_t \tag{19.16i}$$

$$V_l w_{l,y=0}(tk)\le v_{l,y=0}(tk)\le V_u w_{l,y=0}(tk),\quad \forall t=0,\dots,N_t \tag{19.16j}$$

$$V_l w_{l,y=1}(tk)\le v_{l,y=1}(tk)\le V_u w_{l,y=1}(tk),\quad \forall t=0,\dots,N_t \tag{19.16k}$$

19.3.4 Actuator-Placement Problem

This problem generalizes the actuator placement and operation problem from Iftime and Demetriou [260] by considering a semilinear heat equation, and allowing nonuniform material properties. Its main caracterisrtics are shown in Table 19.5. The goal is to match a given final-time temperature in the computational domain by operating a set of actuators withibn the domain that control hot/cold inflow. The location of the actuators is shown in Figure 19.7, and we can operate only a finite number simultaneously.

Table 19.5: Problem Characteristics for Robin Boundary Problem in Two Dimensions.

Type of PDE	semilinear heat equation on $[0,1]^2$ with Neuman boundary conditions
Class of Integers	Mesh-dependent (t) binary variables
Type of Objective	Least-squares (target final state) & regularization term
Type of Constraints	Knapsack constraint on the discrete controls
Discretization	Forward-difference in time, central difference in space; **SQR-AN-V-V**

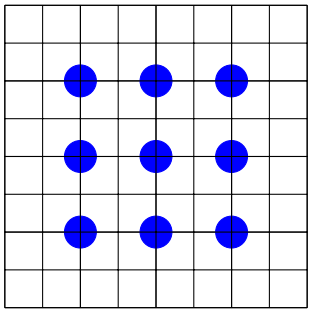


Figure 19.7: Potential Actuator Locations $(k,l) \in \mathcal{A}$ indicated by the blue dots at grid points, $1/4, 1/2, 3/4$.

Problem Description and Discretization. We first describe the infinite dimensional problem, and then discuss extensions, before presenting our discretized model. We let the state variables be denoted by $u(t,x,y)$, and denote the continuous controls by $v_{(k,l)}(t)$ for $(k,l) \in \mathcal{A}$ were the index set $\mathcal{A}$ is the set of possible actuator locations. The binary controls that switch the $v_{(k,l)}(t)$ on and off are denoted by $w_{(k,l)}(t) \in \{0,1\}$ for $(k,l) \in \mathcal{A}$. As before, each actuator is modeled as a Gaussian centered at th control location (x_k, y_l) and defined as

$$f_{(k,l)}(x,y) := \exp\left(\frac{(x-x_k)^2 + (y-y_l)^2}{\sigma^2}\right)$$

for $(k,l) \in \mathcal{A}$ and $(x,y) \in [0,1]^2$, where $\sigma > 0$ is the variance, which we set to $\sigma = 0.02$. The full model is then

$$\underset{u,v,w}{\text{minimize}} \quad \frac{1}{2}\|u - u_d\|^2_{L^2(\Omega)} \quad \text{least squares} \tag{19.17a}$$

$$\text{subject to } \frac{\partial u}{\partial t} - K\Delta u = \lambda u^2 + \sum_{(k,l)\in\mathcal{A}} v_{(k,l)}(t) f_{(k,l)} \text{ in } [0,T]\times[0,1]^2 \quad \text{semilinear heat equation} \tag{19.17b}$$

$$u(0,x,y) = 0 \quad \text{in } [0,1]^2 \quad \text{Initial condition} \tag{19.17c}$$

$$\frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega \quad \text{Neumann boundary} \tag{19.17d}$$

$$V_l w_{(k,l)}(t) \le v_{(k,l)}(t) \le V_u w_{(k,l)}(t), \quad \forall (k,l) \in \mathcal{A},\ \forall t \in [0,T] \tag{19.17e}$$

$$w_{(k,l)}(t) \in \{0,1\} \in (0,T), \sum_{(k,l)\in\mathcal{A}} w_{(k,l)}(t) \le U \quad \text{bounds on control,} \tag{19.17f}$$

where U is an upper bound on the number of open controls in every time step, and K is a heat conductivity field.

We discretize this model with a forward difference scheme in time with time step $k := T_f/N_t$, and a five-point finite-difference stencil in space with $h = 1/M$, where $N_t, M > 0$ are the number of uniform steps in time and space. Denoting by $u_{ijt} \approx u(ih, jh, tk)$ the finite difference approximation, we arrive at the following discretized MIPDECO:

$$\underset{u,v,w}{\text{minimize}} \quad \frac{h^2}{2} \sum_{i,j=1}^{M} \sum_{t=0}^{N_t} (u_{ijt} - u_d(ih, jh, tk))^2 \tag{19.18a}$$

$$\text{subject to} \quad \frac{u_{i,j,t+1} - u_{i,j,t}}{k} - K_{ij} \frac{u_{i+1,j,t+1} + u_{i-1,j,t+1} + u_{i,j+1,t+1} + u_{i,j-1,t+1} - 4u_{i,j,t+1}}{h^2} = \lambda u_{i,j,t}^2 + \sum_{(k,l)\in\mathcal{A}} v_{(k,l)}(tk) f_{(k,l)}(ih, jh) \tag{19.18b}$$

$$u_{i,j,0} = 0 \; \forall i, j = 0, M \tag{19.18c}$$

$$u_{-1,j,t} = u_{1,j,t}, \; u_{M+1,j,t} = u_{M-1,j,t}, \; u_{i,-1,t} = u_{i,1,t}, \; u_{i,M+1,t} = u_{i,M-1,t}, \; \forall t = 0, \ldots, N_t \tag{19.18d}$$

$$\sum_{l=1}^{N_c} (w_{l,x=0}(tk) + w_{l,x=1}(tk) w_{l,y=0}(tk) w_{l,y=1}(tk)) \leq U \quad \forall t = 0, \ldots, N_t \tag{19.18e}$$

$$V_l w_{(k,l)}(tk) \leq v_{(k,l)}(tk) \leq V_u w_{(k,l)}(tk), \quad \forall (k,l) \in \mathcal{A}, \; \forall t = 0, \ldots, N_t \tag{19.18f}$$

$$w_{(k,l)}(tk) \in \{0, 1\}, \; \sum_{(k,l)\in\mathcal{A}} w_{(k,l)}(tk) \leq U, \quad \forall (k,l) \in \mathcal{A}, \; \forall t = 0, \ldots, N_t \tag{19.18g}$$

where $K_{ij} = K(ih, jh)$ is the discretized conductivity field.

We deliberately chose target states with positive and negative values to create a more exciting control problem. We choose $u_d = 2\sin(2\pi x)\cos(piy)$ and $u_d = 4xy - 2$. The two non-constant target states are shown in Figure 19.8 together with the conductivity field in the last example.

19.4 Tutorial

Implementation of heuristic methods in AMPL/JuMP or MNOTAUR; modeling with PDEs (source inversion and control of heat equation); simple discretization schemes.

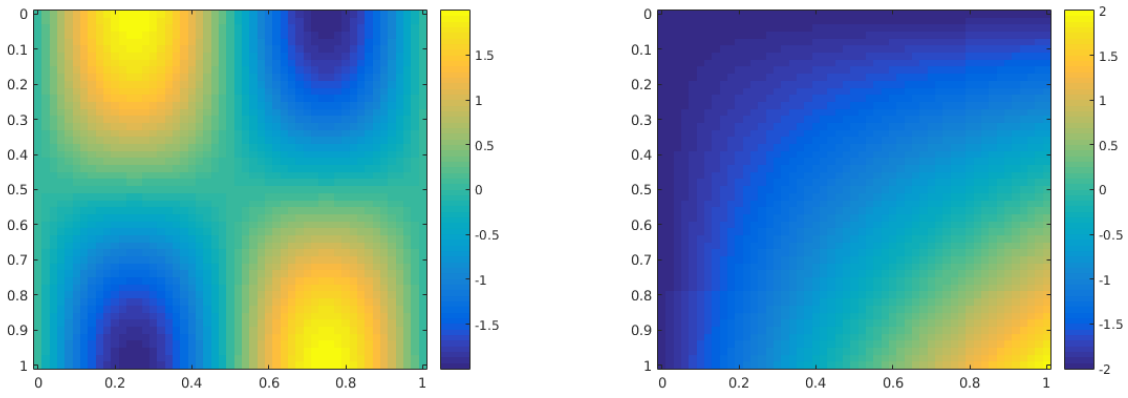


Figure 19.8: Target states for instances (b), right, and (c-d), left.

Appendix A

Online Resources

Nonlinear and mixed-integer optimization survey papers co-authored by Sven Leyffer:

1. Gould and Leyffer. An introduction to algorithms for nonlinear optimization. In Frontiers in Numerical Analysis, pp. 109-197. Springer Verlag, Berlin, 2003.

2. Leyffer and Mahajan. Foundations of Constrained Optimization. In Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Inc. 2010.

3. Leyffer and Mahajan. Software For Nonlinearly Constrained Optimization. In Wiley Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Inc. 2010.

4. Belotti, Kirches, Leyffer, Linderoth, Luedtke, and Mahajan. Mixed-Integer Nonlinear Optimization. Acta Numerica 22:1-131, 2013.

MINLP lectures: https://wiki.mcs.anl.gov/leyffer/index.php/Sven_Leyffer's_Lectures

MINOTAUR solver: https://wiki.mcs.anl.gov/minotaur/index.php/Main_Page

AMPL: http://ampl.com/andhttp://ampl.com/try-ampl/

GAMS: http://gams.com/andhttp://gams.com/download/

Julia/JuMP: https://jump.readthedocs.org/en/latest/andhttps://github.com/JuliaOpt/JuMP.jl

A.1 Software for MINLP

The availability as well as the maturity of software for modeling 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 [108] and D'Ambrosio and Lodi [141] also provide an excellent description of available MINLP solvers. We divide the solvers into those for convex and nonconvex MINLPs; little intersection exists between the two categories.

Key characteristics and features of the solvers are summarized in Tables A.1 and A.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 14.2); LP/NLP-BB for LP/NLP-based branch-and-bound (Section 15.2.1); OA for outer approximation (Section 15.1.1); Hybrid is a hybrid between OA and LP/NLP-BB; QP-Diving (Section 14.2.3); α-BB for α-branch-and-bound (Section 17.1.3); and LP-BB for LP-based branch-and-bound (Section 17.1.3).

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. We

Table A.1: MINLP Solvers (Convex). A "—" means that we do not know the language the solver is written in.

Name	Algorithm(s)	Interfaces	Language	Open-Source
α-ECP	Extended cutting-plane	Customized, GAMS	Fortran-90	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, and others	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

Table A.2: MINLP Solvers (Nonconvex). A "—" means that we do not know the language the solver is written in.

Name	Algorithm(s)	Interfaces	Language	Open-Source
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, Heursitics	AIMMS, AMPL, GAMS, MATHEMATICA	C	No
LindoGlobal	LP-BB	C++, GAMS, Lindo	—	No
SCIP	LP-BB	AMPL, C, GAMS, MATLAB, OSiL, ZIMPL	C	Yes

briefly describe it in Section A.1.3. Expressing and modeling MINLPs is significantly different from LPs or MILPs because general nonlinear functions are much more difficult to represent with data structures. Hence, good modeling tools are indispensable for MINLPs. In Section A.1.4 we describe some tools available for modeling and reformulating MINLPs.

A.1.1 Convex MINLP solvers

α-ECP is a solver written by Westerlund and Lundqvist [440] to solve convex MINLPs by using the extended-cutting plane method (Section 15.1.3). It also implements methods for MINLPs with pseudoconvex functions [442]. The solver can read MINLPs in an extended LP format and can be called through the GAMS [96] modeling system. It requires user to specify an MILP solver for solving the MILP in each iteration. It also provides a graphical interface for the MS-Windows operating system.

BONMIN stands for Basic Open-source Nonlinear Mixed Integer optimizer. It is an open-source solver available at the COIN-OR website [86]. It implements nonlinear branch-and-bound (Section 14.2), LP/NLP-based branch-and-bound (Section 15.2.1), and outer approximation (Section 15.1.1) 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 (https://projects.coin-or.org/Cbc) for performing all the MILP operations, such as management of cuts and tree-search. It can solve NLPs using IPOPT [437] or Filter-SQP [186]. Source code and documentation are available at the website (https://projects.coin-or.org/Bonmin).

DICOPT stands for Discrete and Continuous Optimizer. It implements a variant of outer approximation (Section 15.1.1), which has been generalized to tackle nonconvex MINLPs through a penalty function heuristic; see [436]. The problem is input through the GAMS modeling system. The user can specify options to select both the MILP solver and the NLP solver at each iteration of the algorithm.

FilMINT [1] implements the LP/NLP-BB algorithm (Section 15.2.1) with several practical improvements. It exploits the presolve, cutting planes, searching rules, and other MILP tools of the MINTO solver [345]. Filter-SQP [186] is used to solve NLPs. It also implements some of the disjunctive cuts described in Section 16.1.3 and the feasibility pump heuristic (Section 18.1.1).

KNITRO [111] was initially designed as an NLP solver. Two branch-and-bound (Section 14.2,15.2.1) based algorithms were recently added for solving convex MINLPs [464].

MILANO is a MATLAB-based solver for convex MINLPs. It implements the nonlinear branch-and-bound (Section 14.2) and outer approximation (Section 15.1.1). The source code of MILANO is available on the project website (http://www.pages.drexel.edu/~hvb22/milano). The main focus of this code is to develop efficient warm-starting methods for interior-point methods [57, 58] so as to make them more effective in solving MINLPs.

MINLPBB [297] is a Fortran-based nonlinear branch-and-bound solver (Section 14.2) for convex MINLPs. Filter-SQP [186] 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 nonconvex MINLPs, and it provides options for choosing different branching-rules and tree-search strategies, see Sections 14.2.1 and 14.2.2.

MINOPT is a framework for both modeling and solving MINLPs and MIOCPs Developed in 1998, MINOPT [392]. It implements generalized Benders decomposition (Section 15.1.2) and outer approximation (Section 15.1.1). MINOPT requires linking with an NLP solver and an MILP solver for which it has built-in routines for different solvers. License for MINOPT can be obtained by contacting the authors. More information, a reference manual and examples are available on the project website (http://titan.princeton.edu/MINOPT).

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 14.2) and QP-Diving (Section 14.2.3) 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 (http://wiki.mcs.anl.gov/minotaur/).

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 derviates 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 [173] to MINLPs. Exler et al. [174] provide documentation of the solver along with examples. More information is available online (http://www.ai7.uni-bayreuth.de/misqp.htm).

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 14.2). It also can handle SOS1 and SOS2 (see (17.8)) constraints. A user manual is available online (http://www.gams.com/dd/docs/solvers/sbb.eps).

A.1.2 Nonconvex MINLP solvers

Most modern MINLP solvers designed for nonconvex 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 17.1.2 is most commonly used.

α-BB is a branch-and-bound solver that uses the α-convexification [11] and its variants to obtain quadratic underestimators of nonconvex 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 (http://titan.princeton.edu/tools).

BARON is the acronym for "Branch And Reduce Optimization Navigator" [381]. It implements a branch-and-bound algorithm that computes a lower bound at each subproblem by means of a linear relaxation of (13.1), as discussed in Section 17.1.2. It includes various bound tightening techniques (Section 17.1.3) such as probing and the *violation transfer* outlined in Section 17.1.3. It is available through the GAMS and AIMMS modeling systems. More information, examples, and documentation are available at the project website (http://archimedes.cheme.cmu.edu/?q=baron).

COCONUT is an open-source environment for global optimization problems [123, 389]. Although it does not solve problems with integer variables, we include it in this section because it uses various techniques common to MINLP solvers: bounds tightening (Section 17.1.3), reformulation (Section 17.1.2), and heuristics. The source code and documentation are available at the project homepage (http://www.mat.univie.ac.at/~coconut/coconut-environment).

COUENNE or "Convex Over- and Under-ENvelopes for Nonlinear Estimation" [47] 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 17.1.2. It also implements several bound tightening procedures (Section 17.1.3) as well as a recently introduced feasibility-pump heuristic (Section 18.1.1), a separator of disjunctive cuts (Section 17.1.2), and different branching schemes including strong, pseudocost, and reliability branching (Section 17.1.3). Recently, it also introduced the linear cuts described by Qualizza et al. [365] and briefly discussed in Section 17.1.4. Source code and documentation are available online (https://projects.coin-or.org/Couenne).

GloMIQO is an evolution of α-BB with additional algorithms based on the work of Misener and Floudas [332]. 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 through the GAMS modeling system. Related publications and more information are available on the project website (http://helios.princeton.edu/GloMIQO/publications.html).

LaGO is a branch-and-bound algorithm that is guaranteed to return the global optimum for mixed-integer quadratic problems [350]. It uses α-convexification (see Section 17.1.3) 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. Source-code and documentation are available online (https://projects.coin-or.org/LaGO).

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. License for LGO can be purchased from the website (http://www.pinterconsulting.com).

LindoGlobal is the MINLP solver of the LINGO modeling suite [307]. 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 17.1.2. This solver can be purchased through LINDO Systems (http://www.lindo.com).

SCIP started as a MILP solver [5] but evolved first into a solver for MINLPs with quadratic objective function and constraints [62] and, more recently, into a solver for nonconvex MINLP [63]. Following the basic approach of BARON and COUENNE, it implements a branch-and-bound algorithm (Section 17.1.2) with linear relaxation, various heuristics, and bound-tightening procedures. Source code and documentation are available at the project website (http://scip.zib.de).

A.1.3 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 [82], and was later extended to DAE-constrained problems [296] and mixed-integer optimal control problems [379]. Kirches and Leyffer [277] propose an extension for modeling MIOCPs in the symbolic modeling language AMPL and present an interface to MUSCOD-II. More information about this solver is available from its website on NEOS (http://www.neos-server.org/neos/solvers/miocp:MUSCOD-II/AMPL.html).

A.1.4 Modeling Languages and Online Resources

Diverse modeling 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.

Modeling 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 modeling languages include automatic differentiation tools [224] 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 modeling languages are AIMMS [80], AMPL [195], GAMS [96], MOSEL [125], TomLab [253, 254] and YALMIP [312]. 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 nonconvex MINLPs using inbuilt algorithms. Recently, an open-source modeling system, 'Pyomo' [240], that enables users to express MINLPs using the PYTHON scripting language has been developed. Opti Toolbox [133] is another open-source modeling 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 (`http://wiki.mcs.anl.gov/leyffer/index.php/MacMINLP` and `http://minlp.org/`) and GAMS (`http://www.gamsworld.org/minlp/` and `http://minlp.org/`). Arguably the most important factor in making optimization solvers widely available has been the NEOS server [134]. Many of the solvers described above are now available on NEOS (`http://www.neos-server.org/neos/`. NEOS provides a collection of state-of-the-art optimization software. Optimization problems are submitted through a web interface (or from within a modeling language session) and solved remotely. The MINLP solvers available on NEOS are AlphaECP, BARON, Bonmin, Couenne, DICOPT, FilMINT, LINDOGlobal, MINLPBB, SBB, and SCIP.

Bibliography

er.linderoth:10 [1] K. Abhishek, S. Leyffer, and J. T. Linderoth. FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs. *INFORMS Journal on Computing*, 22:555–567, 2010. DOI:10.1287/ijoc.1090.0373. pages 128, 145, 146, 194, 215

Abichandani2008 [2] P. Abichandani, H. Benson, and M. Kam. Multi-vehicle path coordination under communication constraints. In *American Control Conference, 2008*, pages 650–656, june 2008. DOI:10.1109/ACC.2008.4586566. pages 128

Abramson:2009 [3] M. Abramson, C. Audet, J. Chrissis, and J. Walston. Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3:35–47, 2009. doi:10.1007/s11590-008-0089-2. pages 122

abramson:04 [4] M. A. Abramson. Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm. *Optimization and Engineering*, 5:157–177, 2004. pages 128

achterberg:04 [5] T. Achterberg. SCIP — a framework to integrate constraint and mixed integer programming. Technical Report ZIB-Report 04-19, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, Berlin, 2005. pages 137, 217

rg2007improving [6] T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007. pages 188

.koch.martin:04 [7] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33: 42–54, 2004. pages 134

adams2011use [8] W. Adams. Use of Lagrange interpolating polynomials in the RLT. *Wiley Encyclopedia of Operations Research and Management Science*, 2011. pages 182

adams1986tight [9] W. Adams and H. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986. pages 182

s2005hierarchy [10] W. Adams and H. Sherali. A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems. *Annals of Operations Research*, 140(1):21–47, 2005. pages 183

xis.floudas:98 [11] C. S. Adjiman, I. Androulakis, and C. Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs - II. implementation and computational results. *Computers & Chemical Engineering*, 22:1159–1179, 1998. pages 216

Akcelik-SC05 [12] V. Akcelik, G. Biros, A. Draganescu, O. Ghattas, J. Hill, and B. van Bloemen Waanders. Dynamic data-driven inversion for terascale simulations: Real-time identification of airborne contaminants. In *Proceedings of SC2005, Seattle, WA*, 2005. pages 194

aros.rustem:01 [13] I. Akrotirianakis, I. Maros, and B. Rustem. An outer approximation based branch-and-cut algorithm for convex 0-1 MINLP problems. *Optimization Methods and Software*, 16:21–47, 2001. pages 154, 194

hayyal.falk:83 [14] F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8:273–286, 1983. pages 174

onejotransmilp [15] N. Alguacil, A. L. Motto, and A. Conejo. Transmission expansion planning: a mixed-integer LP approach. *IEEE Transactions on Power Systems*, 18:1070–1077, 2003. pages 9

Altunay201161 [16] M. Altunay, S. Leyffer, J. T. Linderoth, and Z. Xie. Optimal response to attacks on the Open Science Grid. *Computer Networks*, 55(1):61–73, 2011. doi:10.1016/j.comnet.2010.07.012. pages 18

Altunayetal:11 [17] M. Altunay, S. Leyffer, J. T. Linderoth, and Z. Xie. Optimal security response to attacks on open science grids. *Computer Networks*, 55:61–73, 2011. pages 128

ndersenpresolv [18] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71:221–245, 1995. ISSN 0025-5610. URL http://dx.doi.org/10.1007/BF01586000. pages 178, 179

nas.floudas:95 [19] I. P. Androulakis, C. D. Maranas, and C. A. Floudas. αBB : A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7:337–363, 1995. pages 175, 184

AnitM:00b [20] M. Anitescu. On solving mathematical programs with complementarity constraints as nonlinear programs. Preprint ANL/MCS-P864-1200, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2000. pages 108

AnitM:04 [21] M. Anitescu. Global convergence of an elastic mode approach for a class of mathematical programs with complementarity constraints. Preprint ANL/MCS-P1143-0404, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2004. pages 108

anstreicher:09 [22] K. M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43: 471–484, 2009. pages 183

streicher:mp12 [23] K. M. Anstreicher. On convex relaxations for quadratically constrained quadratic programming. *Mathematical Programming*, 136:233–251, 2012. ISSN 0025-5610. doi:10.1007/s10107-012-0602-3. URL http://dx.doi.org/10.1007/s10107-012-0602-3. pages 183, 184

Exa-1 [24] S. Ashly, P. Beckman, J. chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, T. Mezzacapa, P. Moin, M. Norman, R. Rosner, V. Sarkar, A. Siegel, F. Streitz, A. White, and M. Wright. Opportunities and challenges of exascale computing. Summary report of the ascac subcommittee on exascale computing, U.S. Department of Energy, Office of Advanced Scientific Computing Research, 2010. pages 17

Atamturk2010 [25] A. Atamtürk and V. Narayanan. Conic mixed-integer rounding cuts. *Mathematical Programming A*, 122(1):1–20, 2010. pages 161, 163, 164

Audet:2000 [26] C. Audet and J. E. Dennis, Jr. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, 11(3):573–594, 2000. doi:10.1137/S1052623499352024. pages 122

Bacher [27] R. Bacher. The Optimal Power Flow (OPF) and its solution by the interior point approach. EES-UETP Madrid, Short Course, 10-12 December 1997. pages 128

baesetal:mp12 [28] M. Baes, A. Del Pia, Y. Nesterov, S. Onn, and R. Weismantel. Minimizing lipschitz-continuous strongly convex functions over integer points in polytopes. *Mathematical Programming*, 134:305–322, 2012. 10.1007/s10107-012-0545-8. pages 122

balagraves:89 [29] A. Balakrishnan and S. Graves. A composite algorithm for a concave-cost network flow problem. *Networks*, 19(2):175–202, 1989. pages 168

rakash20112136 [30] P. Balaprakash, S. M. Wild, and P. D. Hovland. Can search algorithms save large-scale automatic performance tuning? *Procedia Computer Science (ICCS 2011)*, 4:2136–2145, 2011. doi:10.1016/j.procs.2011.04.234. pages 122

BCC93 [31] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58:295–324, 1993. pages 175

.cornuejols:96 [32] E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42:1229–1246, 1996. pages 137, 156

petsc-web-page [33] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc. pages 93

petsc-user-ref [34] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Users manual. Technical Report ANL-95/11 (Revision 2.1.5), Argonne National Laboratory, 2004. pages 93

h2005autonomic [35] W. Bangerth, H. Klie, V. Matossian, M. Parashar, and M. F. Wheeler. An autonomic reservoir framework for the stochastic optimization of well placement. *Cluster Computing*, 8(4):255–269, 2005. pages 193

Wheeler:06 [36] W. Bangerth, H. Klie, M. Wheeler, P. Stoffa, and M. Sen. On optimization algorithms for the reservoir oil well placement problem. *Computational Geosciences*, 10(3):303–319, 2006. ISSN 1420-0597. doi:10.1007/s10596-006-9025-7. URL http://dx.doi.org/10.1007/s10596-006-9025-7. pages 193

ikmultiterm:09 [37] X. Bao, N. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex quadratically constrained quadratic programs. *Optimization Methods and Software*, 24:485–504, 2009. pages 184

:1021865709529 [38] V. Barbu and M. Iannelli. Optimal control of population dynamics. *Journal of Optimization Theory and Applications*, 102:1–14, 1999. ISSN 0022-3239. URL http://dx.doi.org/10.1023/A:1021865709529. 10.1023/A:1021865709529. pages 17

BarJF:88 [39] J. F. Bard. Convex two-level optimization. *Mathematical Programming*, 40(1):15–27, 1988. pages 102

holomewetal:08 [40] E. F. Bartholomew, R. P. O'Neill, and M. C. Ferris. Optimal transmission switching. *IEEE Transactions on Power Systems*, 23:1346–1355, 2008. pages 18, 128

Bartlett-CM-05 [41] R. Bartlett, M. Heinkenschloss, D. Ridzal, and B. van Bloemen Waanders. Domain decomposition methods for advection dominated linear-quadratic elliptic optimal control problems. *Computer Methods in Applied Mechanics and Engineering*, 2005. pages 194

1996projection [42] H. H. Bauschke and J. M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Review*, 38(3):367–426, 1996. pages 189

letomlin:sos70 [43] E. Beale and J. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In J. Lawrence, editor, *Proceedings of the 5th International Conference on Operations Research*, pages 447–454, Venice, Italy, 1970. pages 115, 125, 135, 169

ale.forrest:76 [44] E. M. L. Beale and J. J. H. Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10:52–69, 1976. pages 115, 169

bellman:pwl61 [45] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Commun. ACM*, 4(6):284, 1961. pages 167

llout2012joint [46] M. C. Bellout, D. E. Ciaurri, L. J. Durlofsky, B. Foss, and J. Kleppe. Joint optimization of oil well placement and controls. *Computational Geosciences*, 16(4):1061–1079, 2012. pages 193

elottiCouMan09 [47] P. Belotti. COUENNE: a user's manual. Technical report, Lehigh University, 2009. URL https://projects.coin-or.org/Couenne/browser/trunk/Couenne/doc/couenne-user-manual.pdf?format=raw. pages 216

Belotti:12 [48] P. Belotti. Disjunctive cuts for non-convex MINLP. In *Mixed Integer Nonlinear Programming*, volume 154 of *IMA Volume Series in Mathematics and its Applications*, pages 117–144. Springer, 2012. pages 175

belotti:2012 [49] P. Belotti. Bound reduction using pairs of linear inequalities. *Journal of Global Optimization*, 2012. DOI:10.1007/s10898-012-9848-9. pages 179

BelottiLLMW09 [50] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009. pages 175, 177, 180

fpfbbt [51] P. Belotti, S. Cafieri, J. Lee, and L. Liberti. Feasibility-based bounds tightening via fixed points. In W. Wu and O. Daescu, editors, *Combinatorial Optimization and Applications*, volume 6508 of *Lecture Notes in Computer Science*, pages 65–76. Springer Berlin / Heidelberg, 2010. pages 180

elotti.etal:12 [52] P. Belotti, J. Góez, I. Pólik, T. Ralphs, and T. Terlaky. A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization. Technical report, n. 12T-009, Lehigh University, Department of Industrial and Systems Engineering, 2012. http://www.optimization-online.org/DB_FILE/2012/06/3494.pdf. pages x, 160, 161

ANU:8877390 [53] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 5 2013. ISSN 1474-0508. doi:10.1017/S0962492913000032. URL http://journals.cambridge.org/article_S0962492913000032. pages 194

ben1995optimal [54] A. Ben-Tal and A. Nemirovski. Optimal design of engineering structures. *Optima*, 47:4–8, 1995. pages 128

.nemirovski:01 [55] A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26(2):193–205, 2001. pages 160

SenShanVand:03 [56] H. Benson, A. Sen, D. F. Shanno, and R. V. D. Vanderbei. Interior-point algorithms, penalty methods and equilibrium problems. Technical Report ORFE-03-02, Princeton University, Operations Research and Financial Engineering, Oct. 2003. To appear in Computational Optimization and Applications. pages 108

enson2011mixed [57] H. Y. Benson. Mixed integer nonlinear programming using interior point methods. *Optimization Methods and Software*, 26(6):911–931, 2011. pages 215

enson2012using [58] H. Y. Benson. Using interior-point methods within an outer approximation framework for mixed integer nonlinear programming. In *Mixed Integer Nonlinear Programming*, The IMA Volumes in Mathematics and its Applications, pages 225–243, 2012. pages 215

urfMoreSari:04 [59] S. J. Benson, L. C. McInnes, J. J. Moré, and J. Sarich. Scalable algorithms in optimization: Computational experiments. Technical Report ANL/MCS-P1175-0604, Mathematics and Computer Science Division, Argonne National Laboratory, 2004. pages 93

rthold2012rens [60] T. Berthold. RENS - the optimal rounding. ZIB-Report 12-17, Zuse Institut Berlin, April 2012. pages 190

2012undercover [61] T. Berthold and A. M. Gleixner. Undercover: a primal MINLP heuristic exploring a largest sub-MIP. ZIB-Report 12-07, Zuse Institut Berlin, February 2012. pages 189, 190

d2010extending [62] T. Berthold, A. Gleixner, S. Heinz, T. Koch, and S. Vigerske. Extending SCIP for solving MIQCPs. In *Proceedings of the European Workshop on Mixed Integer Nonlinear Programming*, pages 181–196, 2010. pages 217

old2012solving [63] T. Berthold, G. Gamrath, A. Gleixner, S. Heinz, T. Koch, and Y. Shinano. Solving mixed integer linear and nonlinear problems using the SCIP optimization suite. ZIB-Report 12-27, Zuse Institut Berlin, 2012. URL http://vs24.kobv.de/opus4-zib/files/1565/ZR-12-27.pdf. pages 217

as.gallager:87 [64] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Endlewood Cliffs, NJ, 1987. pages 128

Bert:1982 [65] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. New York, 1982. pages 96

Ber99 [66] D. P. Bertsekas. *Nonlinear Programming*. Belmont, MA, second edition, 1999. pages 94

Bert:96 [67] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, New York, NY, 1996. pages 90, 91

Bhatiaetal:06 [68] R. Bhatia, A. Segall, and G. Zussman. Analysis of bandwidth allocation algorithms for wireless personal area networks. *Wireless Networks*, 12:589–603, 2006. pages 128

PDECO-03 [69] L. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, editors. *Large-Scale PDE-Constrained Optimization, Lecture Notes in Computational Sience and Engineering*, volume 30. Springer-Verlag, 2001. pages 194

RTPDECO-07 [70] L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors. *Real-Time PDE-Constrained Optimization*. SIAM, 2007. pages 194

bienstock:96 [71] D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74:121–140, 1996. pages 128

tock.mattia:07 [72] D. Bienstock and S. Mattia. Using mixed-integer programming to solve power grid blackout problems. *Discrete Optimization*, 4:115–141, 2007. pages 128

Bi05 [73] V. M. Bier. Game-theoretic and reliability methods in counterterrorism and security. In A. Wilson, N. Limnios, S. Keller-McNulty, and Y. Armijo, editors, *Mathematical and Statistical Methods in Reliability, Series on Quality, Reliability and Engineering Statistics*, pages 17–28. World Scientific, Singapore, 2005. pages 129

BNA05 [74] V. M. Bier, A. Nagaraj, and V. Abhichandani. Protection of simple series and parallel systems with components of different values. *Reliability Engineering System Safety*, 87(3):315–323, 2005. pages 129

BOS07 [75] V. M. Bier, S. Oliveros, and L. Samuelson. Choosing what to protect. *J. Public Economic Theory*, 9 (4):563–587, 2007. pages 129

Biros-1 [76] G. Biros and O. Ghattas. Parallel lagrange-newton-krylov-schur methods for PDE-constrained optimization. part i: The krylov-schur solver. *SIAM Journal on Scientific Computing*, 27(2), 2005. pages 194

Biros-2 [77] G. Biros and O. Ghattas. Parallel lagrange-newton-krylov-schur methods for PDE-constrained optimization. part ii: The lagrange-newton solver, and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing*, 27(2), 2005. pages 194

OGhattas_2005a [78] G. Biros and O. Ghattas. Parallel Lagrange–Newton–Krylov–Schur Methods for PDE-Constrained Optimization. Part I: The Krylov–Schur Solver. *SIAM J. Sci. Comput.*, 27(2):687–713, 2005. doi:10.1137/S106482750241565X. pages 199

OGhattas_2005b [79] G. Biros and O. Ghattas. Parallel Lagrange–Newton–Krylov–Schur methods for PDE–constrained optimization. part II: The Lagrange–Newton solver and its application to optimal control of steady viscous flows. *SIAM J. Sci. Comput.*, 27(2):714–739, 2005. doi:10.1137/S1064827502415661. pages 199

AIMMSmanual [80] J. Bisschop and R. Entriken. *AIMMS The Modeling System*. Paragon Decision Technology, 1993. pages 218

Bock1982 [81] H. Bock and R. Longman. Computation of optimal controls on disjoint control sets for minimum energy subway operation. *Advances in the Astronautical Sciences*, 50:949–972, 1985. Proceedings of the American Astronomical Society Symposium on Engineering Science and Mechanics, Taiwan, 1982. pages 122, 126, 128, 129

Bock1984 [82] H. Bock and K. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Budapest, 1984. Pergamon Press. pages 217

BogTol:95 [83] P. Boggs and J. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–51, 1995. pages 78, 80

bonami:11 [84] P. Bonami. Lift-and-project cuts for mixed integer convex programs. In O. Günlük and G. Woeginger, editors, *Integer Programming and Combinatoral Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 52–64. Springer, Berlin, 2011. pages 158, 159

2012heuristics [85] P. Bonami and J. P. M. Gonçalves. Heuristics for convex mixed integer nonlinear programs. *Computational Optimization and Applications*, 51:729–747, 2012. pages 189, 190, 191, 192

bonami.etal:08 [86] P. Bonami, L. Biegler, A. Conn, G. Cornuéjols, I. Grossmann, C. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008. pages 141, 142, 146, 148, 194, 215

bonami.et.al:09 [87] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, 119:331–352, 2009. pages 188, 189

hterLeyffer:11 [88] P. Bonami, J. Lee, S. Leyffer, and A. Wächter. More branch-and-bound experiments in convex nonlinear integer programming. Preprint ANL/MCS-P1949-0911, Argonne National Laboratory, Mathematics and Computer Science Division, Sept. 2011. pages 134, 135, 136

2012algorithms [89] P. Bonami, M. Kılınç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. *IMA Volumes in Mathematics and its Applications*, 154:61–92, 2012. pages 122

rstyn.frank:77 [90] R. Boorstyn and H. Frank. Large-scale network topological optimization. *IEEE Transactions on Communications*, 25:29–47, 1977. pages 128

rs.mitchell:94 [91] B. Borchers and J. E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research*, 21:359–368, 1994. pages 137

_VSchulz_2009a [92] A. Borzì and V. Schulz. Multigrid methods for PDE optimization. *SIAM Review*, 51(2):361–395, 2009. doi:10.1137/060671590. pages 194

andenberghe:04 [93] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004. pages 167

galli.et.al:06 [94] C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth. An MINLP solution method for a water network problem. In *Algorithms - ESA 2006 (14th Annual European Symposium. Zurich, Switzerland, September 2006, Proceedings)*, pages 696–707. Springer, 2006. pages 18, 122, 125, 128

ragallietal:12 [95] C. Bragalli, C. D'Ambrosio, J. Lee, A. Lodi, and P. Toth. On the optimal design of water distribution networks: a practical minlp approach. *Optimization and Engineering*, 13:219–246, 2012. ISSN 1389-4420. doi:10.1007/s11081-011-9141-7. URL http://dx.doi.org/10.1007/s11081-011-9141-7. pages 122, 125

GAMSManual [96] A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. *GAMS, A User's Guide*. GAMS Development Corporation, 1992. pages 214, 218

AppliedMaths [97] D. L. Brown, J. Bell, D. Estep, W. Gropp, B. Hendrickson, S. Keller-McNulty, D. Keyes, J. T. Oden, L. Petzold, and M. Wright. Applied Mathematics at the U.S. Department of Energy: past, present and a view to the future. Report by an independent panel from the applied mathematics research community, DOE-ASCR, 2008. pages 17

Brownreport [98] D. L. Brown, J. Bell, D. Estep, W. Gropp, B. Hendrickson, S. Keller-McNulty, D. Keyes, J. T. Oden, L. Petzold, and M. Wright. Applied mathematics at the U.S. Department of Energy: Past, present and a view to the future, May 2008. URL http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Brown_report_may_08.pdf. pages 18

imwiegele:mp12 [99] C. Buchheim and A. Wiegele. Semidefinite relaxations for non-convex quadratic mixed-integer programming. *Mathematical Programming*, pages 1–18, 2012. ISSN 0025-5610. URL http://dx.doi.org/10.1007/s10107-012-0534-y. 10.1007/s10107-012-0534-y. pages 183

burer:09 [100] S. Burer. On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming*, 120:479–495, 2009. pages 183

rletchford:09 [101] S. Burer and A. Letchford. On nonconvex quadratic programming with box constraints. *SIAM Journal on Optimization*, 20(2):1073 – 89, 2009. ISSN 1052-6234. URL http://dx.doi.org/10.1137/080729529. pages 183

rletchford:12 [102] S. Burer and A. Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17:97–106, 2012. pages 165, 194

etchford:mp12 [103] S. Burer and A. Letchford. Unbounded convex sets for non-convex mixed-integer quadratic programming. *Mathematical Programming*, pages 1–26, 2012. ISSN 0025-5610. 10.1007/s10107-012-0609-9. pages 184

burervand:09 [104] S. Burer and D. Vandenbussche. Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. *Computational Optimization and Applications*, 43 (2):181 – 195, 2009. URL http://dx.doi.org/10.1007/s10589-007-9137-6. pages 183

schweiger2008 [105] J. Burgschweiger, B. Gnädig, and M. Steinbach. Optimization models for operative planning in drinking water networks. *Optimization and Engineering*, 10(1):43–73, 2008. pages 125, 128

Burgetal:09 [106] J. Burgschweiger, B. Gnaedig, and M. C. Steinbach. Nonlinear programming techniques for operative planning in large drinking water networks. *The Open Appl. Math. J.*, 3:14–28, 2009. pages 17

BussPruess:03 [107] M. R. Bussieck and A. Pruessner. Mixed-integer nonlinear programming. *SIAG/OPT Views-and-News*, 14(1):19–22, 2003. pages 194

ieck2010minlp [108] M. R. Bussieck and S. Vigerske. MINLP solver software. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, P. Jeffrey, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. Wiley, 2010. pages 213

shahid [109] M. O. Buygi, G. Balzer, H. M. Shanechi, and M. Shahidehpour. Market-based transmission expansion planning. *IEEE Transactions on Power Systems*, 19:2060–2067, 2004. pages 9

NoceWalt04:mp [110] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Mathematical Programming Series B*, 100(1):27–48, 2004. pages 79

yrd2006knitro [111] R. H. Byrd, J. Nocedal, and W. A. Richard. KNITRO: An integrated package for nonlinear optimization. In G. Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, pages 35–59. Springer, US, 2006. pages 215

10approximate [112] S. Callegari, F. Bizzarri, R. Rovatti, and G. Setti. On the approximate solution of a class of large discrete quadratic programming problems by $\Delta\Sigma$ modulation: The case of circulant quadratic forms. *IEEE Transactions on Signal Processing*, 58(12):6126–6139, 2010. pages 128

illo.et.al:05 [113] I. Castillo, J. Westerlund, S. Emet, and T. Westerlund. Optimization of block layout deisgn problems with unequal areas: A comparison of MILP and MINLP optimization methods. *Computers & Chemical Engineering*, 30:54–69, 2005. pages 128

Cezik2005 [114] M. Çezik and G. Iyengar. Cuts for mixed 0-1 conic programming. *Mathematical Programming A*, 104:179–202, 2005. pages 161, 163

ria.soares:99 [115] S. Ceria and J. Soares. Convex programming for disjunctive optimization. *Mathematical Programming*, 86:595–614, 1999. pages 156

ik.iyengar:05 [116] M. T. Çezik and G. Iyengar. Cuts for mixed 0-1 conic programming. *Mathematical Programming*, 104:179–202, 2005. pages 194

nyisoferc [117] H. Chao. NYISO reliability and economic planning process. In *FERC Workshop: Increasing Market and Planning Efficiency Through Improved Software and Hardware-Enhanced wide-area planning models*, 2010. pages 10

Chietal:08 [118] K. Chi, X. Jiang, S. Horiguchi, and M. Guo. Topology design of network-coding-based multicast networks. *IEEE Transactions on Mobile Computing*, 7(4):1–14, 2008. pages 128

ChiFle:03 [119] C. Chin and R. Fletcher. On the global convergence of an SLP-filter algorithm that takes EQP steps. *Mathematical Programming*, 96(1):161–177, 2003. pages 79

christie [120] R. D. Christie, B. F. Wollenberg, and I. Wangensteen. Transmission management in the deregulated environment. *Proceedings of the IEEE*, 88:170–195, 2000. pages 9

mohitjp:oo11 [121] K. Chung, J.-P. Richard, and M. Tawarmalani. Lifted inequalities for 0-1 mixed-integer bilinear covering sets, 2011. Available at http://www.optimization-online.org/DB_FILE/2011/03/2949.pdf. pages 185

chvatal:73-2 [122] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973. pages 154

conutbench:04 [123] Coconut. The COCONUT benchmark: A benchmark for global optimization and constraint satisfaction, 2004. http://www.mat.univie.ac.at/~neum/glopt/coconut/benchmark.html. pages 216

cohen2002 [124] J. S. Cohen. *Computer algebra and symbolic computation: elementary algorithms*. Universities Press, 2003. pages 171

ni.heipcke:02 [125] Y. Colombani and S. Heipcke. Mosel: An extensible environment for modeling and programming solutions. In N. Jussien and F. Laburthe, editors, *Proceedings of the Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR'02)*, pages 277–290, 2002. pages 218

colombo:2032 [126] R. M. Colombo, G. Guerra, M. Herty, and V. Schleper. Optimal control in networks of pipes and canals. *SIAM Journal on Control and Optimization*, 48(3):2032–2050, 2009. doi:10.1137/080716372. URL http://link.aip.org/link/?SJC/48/2032/1. pages 17

learReport-10 [127] T. Committee. Advanced fueld pellet materials and fuel rod design for water cooled reactors. Technical report, International Atomic Energy Agency, 2010. pages 193

CUTE-Charact [128] A. Conn, N. Gould, and P. Toint. The cute classification scheme. http://www.cuter.rl.ac.uk//Problems/classification.shtml, 1992. pages 196

ConGouToi:91 [129] A. R. Conn, N. I. M. Gould, and P. L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal of Numerical Analysis*, 28(2):545–572, 1991. pages 90

4663816 [130] P. Cortes, M. Kazmierkowski, R. Kennel, D. Quevedo, and J. Rodriguez. Predictive control in power electronics and drives. *Industrial Electronics, IEEE Transactions on*, 55(12):4312 –4324, dec. 2008. ISSN 0278-0046. doi:10.1109/TIE.2008.2007480. pages 17

Costaetal:07 [131] E. Costa-Montenegro, F. J. González-Castaño, P. S. Rodriguez-Hernández, and J. C. Burguillo-Rial. Nonlinear optimization of IEEE 802.11 mesh networks. In *ICCS 2007, Part IV*, pages 466–473, Springer Verlag, Berlin, 2007. pages 128

croxtonpwl:03 [132] K. Croxton, B. Gendron, and T. Magnanti. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science*, 49:1268–73, Sept. 2003. pages 168

urrie2012opti [133] J. Currie and D. I. Wilson. OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In N. Sahinidis and J. Pinto, editors, *Foundations of Computer-Aided Process Operations*, Savannah, Georgia, USA, 8–11 January 2012. pages 218

snier.more:98 [134] J. Czyzyk, M. Mesnier, and J. Moré. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998. pages 218

RalpWrig:03 [135] D. Ralph and S. J. Wright. Some properties of regularization and penalization schemes for MPECs. Technical Report 03-04, Computer Science Department, University of Wisconsin, December 2003. Revised April 2004, to appear in *Computational Optimization and Applications*. pages 107

Dadush2011 [136] D. Dadush, S. Dey, and J. P. Vielma. The split closure of a strictly convex body. *Operations Research Letters*, 39(2):121 – 126, 2011. pages 122, 160

ddvcg:ipco11 [137] D. Dadush, S. S. Dey, and J. P. Vielma. On the chvatal-gomory closure of a compact convex set. In *Lecture Notes in Computer Science*, volume 6655 LNCS, pages 130 – 142, New York, NY, 2011. Springer. pages 122

ddvcg:mor11 [138] D. Dadush, S. S. Dey, and J. P. Vielma. The chvatal-gomory closure of a strictly convex body. *Mathematics of Operations Research*, 36(2):227 – 239, 2011. pages 122

dadush:focs11 [139] D. Dadush, C. Peikert, and S. Vempala. Enumerative lattice algorithms in any norm via m-ellipsoid coverings. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 580 –589, 2011. doi:10.1109/FOCS.2011.31. pages 121

dakin:65 [140] R. J. Dakin. A tree search algorithm for mixed programming problems. *Computer Journal*, 8:250–255, 1965. pages 131, 136, 194

osio2011mixed [141] C. D'Ambrosio and A. Lodi. Mixed integer nonlinear programming tools: a practical overview. *4OR*, 9(4):329–349, 2011. pages 213

dambrosio2010 [142] C. D'Ambrosio, A. Lodi, and S. Martello. Piecewise linear approximation of functions of two variables in MILP models. *Operations Research Letters*, 38(1):39–46, 2010. pages 171

osio2012storm [143] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi. A storm of feasibility pumps for nonconvex MINLP. *Mathematical Programming*, 136:375–402, 2012. ISSN 0025-5610. doi:10.1007/s10107-012-0608-x. URL http://dx.doi.org/10.1007/s10107-012-0608-x. pages 189

Danilo2007676 [144] Danilo and Rastovic. Optimal control of tokamak and stellarator plasma behaviour. *Chaos, Solitons & Fractals*, 32(2):676 – 681, 2007. ISSN 0960-0779. doi:10.1016/j.chaos.2005.11.016. URL http://www.sciencedirect.com/science/article/pii/S0960077905011008. pages 17

erg.lepape:05 [145] E. Danna, E. Rothberg, and C. LePape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102:71–90, 2005. pages 187, 192

tzig:econ1960 [146] G. B. Dantzig. On the significance of solving linear programming problems with some integer variables. *Econometrica*, 28(1):30–44, 1960. pages 165, 169

dantzig:63 [147] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963. pages 169

Davis1987281 [148] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, 1987. pages 178

Davis:2009 [149] E. Davis and M. Ierapetritou. A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. *Journal of Global Optimization*, 43(2-3):191–205, 2009. doi:10.1007/s10898-007-9217-2. pages 122

eraetal:mor06 [150] J. A. De Loera, R. Hemmecke, M. Koppe, and R. Weismantel. Integer polynomial optimization in fixed dimension. *Mathematics of Operations Research*, 31(1):147 – 153, 2006. URL http://dx.doi.org/10.1287/moor.1050.0169. pages 121

olf.smeers:00 [151] D. De Wolf and Y. Smeers. The gas transmission problem solved by an extension of the simplex algorithm. *Management Science*, 46:1454–1465, 2000. pages 193

deymoran:mp12 [152] S. S. Dey and D. A. Moran R. Some properties of convex hulls of integer points contained in general convex sets. *Mathematical Programming*, pages 1 – 20, 2012. pages 122

dvcg:10 [153] S. S. Dey and J. P. Vielma. The chvatal-gomory closure of an ellipsoid is a polyhedron. In *Lecture Notes in Computer Science*, volume 6080 LNCS, pages 327 – 340, Lausanne, 2010. Springer. pages 122

MPECWORLD [154] S. P. Dirkse. MPEC world. Webpage, GAMS Development Corp., www.gamsworld.org/mpec/, 2001. pages 101

rkFerrMeer:02 [155] S. P. Dirkse, M. C. Ferris, and A. Meeraus. Mathematical programs with equilibrium constraints: Automatic reformulation and solution via constraint optimization. Technical Report NA-02/11, Oxford University Computing Laboratory, July 2002. pages 102

dolan.more:02 [156] E. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002. pages 149

donde05 [157] V. Donde, V. Lopez, B. Lesieutre, A. Pinar, C. Yang, and J. Meza. Identification of severe multiple contingencies in electric power networks. In *Proceedings 37th North American Power Symposium*, 2005. LBNL-57994. pages 128

Donovan_03 [158] G. Donovan and D. Rideout. An integer programming model to optimize resource allocation for wildfire containment. *Forest Science*, 61(2), 2003. pages 193

dorigo1996ant [159] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 26(1):1–13, 1996. pages 187

FrieSant:2003 [160] Z. Dostál, A. Friedlander, and S. A. Santos. Augmented Lagrangians with adaptive precision control for quadratic programming with simple bounds and equality constraints. 13(4):1120–1140, 2003. pages 97

Drewes2009 [161] S. Drewes. *Mixed Integer Second Order Cone Programming*. PhD thesis, Technische Universität Darmstadt, 2009. pages 160, 161, 162, 163, 194

Drewes2012 [162] S. Drewes and S. Ulbrich. Subgradient based outer approximation for mixed integer second order cone programming. In *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 41–59. Springer, New York, 2012. ISBN 978-1-4614-1926-6. pages 160, 194

.grossmann:86 [163] M. A. Duran and I. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986. pages 141, 142, 194

eckstein:94-2 [164] J. Eckstein. Parallel branch-and-bound algorithms for general mixed integer programming on the CM-5. *SIAM Journal on Optimization*, 4:794–814, 1994. pages 137

2005nonlinear [165] K. Ehrhardt and M. C. Steinbach. *Nonlinear optimization in gas networks*. Springer, 2005. pages 193

eigerwater:94 [166] G. Eiger, U. Shamir, and A. Ben-Tal. Optimal design of water distribution networks. *Water Resources Research*, 30(9):2637–2646, 1994. pages 122

elhedhli:06 [167] S. Elhedhli. Service System Design with Immobile Servers, Stochastic Demand, and Congestion. *Manufacturing & Service Operations Management*, 8(1):92–97, 2006. doi:10.1287/msom.1050.0094. pages 128

n.martinez:07 [168] A. M. Eliceche, S. M. Corvalán, and P. Martínez. Environmental life cycle impact as a tool for process optimisation of a utility plant. *Computers & Chemical Engineering*, 31:648–656, 2007. pages 128

Biology [169] M. Ellisman, R. Stevens, M. Colvin, T. Schlick, A. Arkin, D. Galas, E. Delong, G. Olsen, J. George, G. Karniakadis, C. Johnson, and N. Sematova. Scientific grand challenges: Opportunities in biology at the extreme scale of computing. Report from the workshop held august 17-19, 2009, U.S. Department of Energy, Office of Biological and Environmental Science and the Office of Advanced Scientific Computing Research, 2009. pages 17

Ellison:06 [170] J. Ellison. Modeling the US natural gas network. Technical report, Sandia National Laboratories, 2006. pages 17

lwalidetal:06 [171] A. Elwalid, D. Mitra, and Q. Wang. Distributed nonlinear integer optimization for data-optical internetworking. *IEEE Journal on Selected Areas in Communications*, 24(8):1502–1513, 2006. pages 128

Engelhart2012 [172] M. Engelhart, J. Funke, and S. Sager. A decomposition approach for a new test-scenario in complex problem solving. *Journal of Computational Science*, 2012. (to appear). pages 129

hittkowski:07 [173] O. Exler and K. Schittkowski. A trust region SQP algorithm for mixed-integer nonlinear programming. *Optimization Letters*, 1:269–280, 2007. pages 216

xler2012misqp [174] O. Exler, T. Lehmann, and K. Schittkowski. 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, April 2012. pages 216

xpress4 [175] FICO Xpress. *FICO Xpress Optimization Suite: Xpress-BCL Reference manual*. Fair Isaac Corporation, 2009. pages 170

FerPan:97 [176] M. C. Ferris and J. S. Pang. Engineering and economic applications of complementarity problems. *SIAM Review*, 39(4):669–713, 1997. pages 101, 102

FerrTinL:99a [177] M. C. Ferris and F. Tin-Loi. On the solution of a minimum weight elastoplastic problem involving displacement and complementarity constraints. *Computer Methods in Applied Mechanics and Engineering*, 174:107–120, 1999. pages 108

FiaccoMcco:90 [178] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Number 4 in Classics in Applied Mathematics. SIAM, 1990. Reprint of the original book published in 1968 by Wiley, New York. pages 81

Fipki_08 [179] S. Fipki and A. Celi. The use of multilateral well designs for improved recovery in heavy oil reservoirs. In *IADV/SPE Conference and Exhibition*, Orlanda, Florida, 2008. SPE. pages 193, 199

hetti.lodi:03 [180] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2003. pages 191, 192

09feasibility [181] M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Mathematical Programming Computations*, 1: 201–222, 2009. pages 188

lover.lodi:05 [182] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104:91–104, 2005. pages 188, 189

FleR:87 [183] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, Chichester, 1987. pages 83, 84, 121, 135

FleSai:89 [184] R. Fletcher and E. S. de la Maza. Nonlinear programming and nonsmooth optimization by successive linear programming. *Mathematical Programming*, 43:235–256, 1989. pages 79

er.leyffer:94 [185] R. Fletcher and S. Leyffer. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349, 1994. pages 141, 142, 143

r.leyffer:98b [186] R. Fletcher and S. Leyffer. User manual for filterSQP, 1998. University of Dundee Numerical Analysis Report NA-181. pages 215

FleLey:02 [187] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91:239–270, 2002. pages 83, 92

FletLeyf:02 [188] R. Fletcher and S. Leyffer. Numerical experience with solving MPECs as NLPs. Numerical Analysis Report NA/210, Department of Mathematics, University of Dundee, Dundee, UK, 2002. pages 107

FleLey:03 [189] R. Fletcher and S. Leyffer. Filter-type algorithms for solving systems of algebraic equations and inequalities. In G. di Pillo and A. Murli, editors, *High Performance Algorithms and Software for Nonlinear Optimization*, pages 259–278. Kluwer, Dordrecht, 2003. pages 85, 141

FLRS:02 [190] R. Fletcher, S. Leyffer, D. Ralph, and S. Scholtes. Local convergence of SQP methods for mathematical programs with equilibrium constraints. Numerical Analysis Report NA/209, Department of Mathematics, University of Dundee, Dundee, UK, May 2002. To appear in SIAM J. Optimization. pages 107

ac.biegler:07 [191] A. Flores-Tlacuahuac and L. T. Biegler. Simultaneous mixed-integer dynamic optimization for integrated design and control. *Computers & Chemical Engineering*, 31:648–656, 2007. pages 128

FlouCA:95 [192] C. Floudas. *Nonlinear and Mixed-Integer Optimization*. Topics in Chemical Engineering. Oxford University Press, New York, 1995. pages 122

floudas:00 [193] C. A. Floudas. *Deterministic Global Optimization: Theory, Algorithms and Applications*. Kluwer Academic Publishers, 2000. pages 165, 194

rsGillWrig:02 [194] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 4(4):525–597, 2002. pages 81

.kernighan:93 [195] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. The Scientific Press, 1993. pages 218

FouGayKer:03 [196] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modelling Language for Mathematical Programming*. Books/Cole—Thomson Learning, 2nd edition, 2003. pages 195

Shootout07 [197] 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. A comparison of derivative-free optimization methods for water supply and hydraulic capture community problems. *Advances in Water Resources*, 31(5):743–757, 2008. doi:10.1016/j.advwatres.2008.01.010. pages 122

ni.gentile:06 [198] A. Frangioni and C. Gentile. Perspective cuts for a class of convex 0-1 mixed integer programs. *Mathematical Programming*, 106:225–236, 2006. pages 154, 156, 194

Frieszstal:89 [199] T. L. Friesz, J. Luque, R. L. Tobin, and B.-W. Wie. Dynamic network traffic assignment considered as a continuous time optimal control problem. *Operations Research*, 37(6):893–901, 1989. pages 17

ugenschuh2006 [200] A. Fügenschuh, M. Herty, A. Klar, and A. Martin. Combinatorial and continuous models for the optimization of traffic flows on networks. *SIAM Journal on Optimization*, 16(4):1155–1176, 2006. pages 128

DOE-ASCR-4 [201] S. H. G. Bothun and S. Picataggio. Computational research needs for alternative and renewable energy. U.S. Department of Energy workshop report, DOE-ASCR, 2007. URL http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Crnare_workshop_report.pdf. pages 18

BES [202] G. Galli, T. Dunning, M. Head-Gordon, G. Kotliar, J. C. Grossman, K.-M. Ho, M.-Y. Chou, M. Dupuis, M. Asta, and C. Simmerling. Discovery in basic energy sciences: The role of computing at the extreme scale. Report from the workshop held August 13-15, 2009, U.S. Department of Energy, Office of Basic Energy Sciences and the Office of Advanced Scientific Computing Research, 2009. pages 17

DOEexaBES [203] G. Galli, T. Dunning, M. Head-Gordon, G. Kotliar, J. C. Grossman, K.-M. Ho, M.-Y. Chou, M. Dupuis, M. Asta, and C. Simmerling. Scientific grand challenges: Discovery in basic energy sciences: The role of computing at the extreme scale, August 2009. URL http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Bes_exascale_report.pdf. pages 18

Garver:97 [204] L. L. Garver. Transmission network estimation using linear programming. *IEEE Transactions on Power Apparatus Systems*, 89:1688–1697, 1997. pages 18, 128

slermartin:12 [205] B. Geissler, A. Martin, A. Morsi, and L. Schewe. Using piecewise linear functions for solving MINLPs. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 287–314. Springer New York, 2012. pages 166, 169, 171

ini.etal:2012 [206] I. Gentilini, F. Margot, and K. Shimada. The travelling salesman problem with neighbourhoods: MINLP solution. *Optimization Methods and Software*, 28(2):364–378, 2013. doi:10.1080/10556788.2011.648932. URL http://www.tandfonline.com/doi/abs/10.1080/10556788.2011.648932. pages 128

geoffrion:72 [207] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972. pages 141, 144, 194

ffrion:pwla77 [208] A. M. Geoffrion. Objective function approximations in mathematical programming. *Mathematical Programming*, 13:23–37, 1977. pages 167

Gerdts2005 [209] M. Gerdts. Solving mixed-integer optimal control problems by Branch&Bound: A case study from automobile test-driving with gear shift. *Optimal Control Applications and Methods*, 26:1–18, 2005. pages 129

Gero:00 [210] P. Geroski. Models of technology diffusion. *Research Policy*, 29:603–625, 2000. pages 14

lover1989tabu [211] F. Glover. Tabu search: part I. *ORSA Journal on Computing*, 1(3):190–206, 1989. pages 187

over1989tabu2 [212] F. Glover. Tabu search: part II. *ORSA Journal on Computing*, 2(1):4–32, 1990. pages 187

rg1989genetic [213] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Boston, 1989. pages 187

yfferSafro:11 [214] N. Goldberg, S. Leyffer, and I. Safro. Optimal response to epidemics and cyber attacks in networks. Preprint ANL/MCS-1992-0112, Argonne National Laboratory, Mathematics and Computer Science Division, Jan. 2012. URL http://wiki.mcs.anl.gov/leyffer/images/e/e3/NetworkResponse.pdf. pages 128

gomory:58 [215] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Monthly*, 64:275–278, 1958. pages 154

gomory:60 [216] R. E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Corporation, 1960. pages 154

GordonRice:97 [217] R. J. Gordon and S. A. Rice. Aactive control of the dynamics of atoms and molecules. *Annual Review of Physical Chemistry*, 48:601–641, 1997. pages 17

021/ar9701191 [218] R. J. Gordon, L. Zhu, and T. Seideman. Coherent control of chemical reactions. *Accounts of Chemical Research*, 32(12):1007–1016, 1999. doi:10.1021/ar9701191. URL http://pubs.acs.org/doi/abs/10.1021/ar9701191. pages 17

ouldRobin:08b [219] N. I. M. Gould and D. P. Robinson. A second derivative SQP method: Local convergence. Numerical Analysis Report 08/21, Oxford University Computing Laboratory, 2008. To appear in SIAM Journal on Optimization. pages 80

GouldRobin:10 [220] N. I. M. Gould and D. P. Robinson. A second derivative SQP method: Global convergence. *SIAM Journal on Optimization*, 20(4):2023–2048, 2010. pages 80

GouToi:10 [221] N. I. M. Gould and P. L. Toint. Nonlinear programming without a penalty function or a filter. *Mathematical Programming*, 122(1):155–196, 2010. pages 84

GouLeyToi:04 [222] N. I. M. Gould, S. Leyffer, and P. L. Toint. A multidimensional filter algorithm for nonlinear equations and nonlinear least squares. *SIAM Journal on Optimization*, 15(1):17–38, 2004. pages 94, 141

ux.leyffer:03 [223] J.-P. Goux and S. Leyffer. Solving large MINLPs on computational grids. *Optimization and Engineering*, 3:327–354, 2003. pages 136

Griewank:00 [224] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 2000. pages 218

GrifStew:61 [225] R. Griffith and R. Stewart. A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, 7(4):379–392, July 1961. pages 79

Grig:09 [226] I. Grigorenko and H. Rabitz. Optimal control of the local electromagnetic response of nanostructured materials: Optimal detectors and quantum disguises. *Appl. Phys. Lett.*, 94, 2009. pages 17

NLP-book [227] I. Griva, S. G. Nash, and A. Sofer. *Linear and Nonlinear Optimization*. SIAM, 2nd edition, 2009. pages 131

grossman:02 [228] I. E. Grossmann. Review of nonlinear mixed–integer and disjunctive programming techniques. *Optimization and Engineering*, 3:227–252, 2002. pages 122, 194

GroKra:97 [229] I. E. Grossmann and Z. Kravanja. Mixed-integer nonlinear programming: A survey of algorithms and applications. In A. C. L.T. Biegler, T.F. Coleman and F. Santosa, editors, *Large-Scale Optimization with Applications, Part II: Optimal Design and Control*, Springer, New York, 1997. pages 122, 194

GueNewLey:11 [230] A. Guerra, A. M. Newman, and S. Leyffer. Concrete structure design using mixed-integer nonlinear programming with complementarity constraints. *SIAM Journal on Optimization*, 21(3):833–863, 2011. pages 128

.linderoth:08 [231] O. Günlük and J. Linderoth. Perspective relaxation of mixed integer nonlinear programs with indicator variables. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *IPCO 2008: The Thirteenth Conference on Integer Programming and Combinatorial Optimization*, volume 5035, pages 1–16, 2008. pages 194

.linderoth:10 [232] O. Günlük and J. Linderoth. Perspective relaxation of mixed integer nonlinear programs with indicator variables. *Mathematical Programming Series B*, 104:186–203, 2010. pages 156

.linderoth:12 [233] O. Günlük and J. T. Linderoth. Perspective reformulation and applications. In *IMA Volumes*, volume 154, pages 61–92, 2012. pages 155

.ravindran:85 [234] O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31:1533–1546, 1985. pages 131, 194

gurobi5 [235] Gurobi. *Gurobi Optimizer Reference Manual, Version 5.0*. Gurobi Optimization, Inc., 2012. pages 170

HanSP:77 [236] S. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22(3):297–309, 1977. pages 78

5446440 [237] S. Han, S. Han, and K. Sezaki. Development of an optimal vehicle-to-grid aggregator for frequency regulation. *Smart Grid, IEEE Transactions on*, 1(1):65 –72, june 2010. ISSN 1949-3053. doi:10.1109/TSG.2010.2045163. pages 17

hansen [238] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, Inc., New York, 1992. pages 165

oski.et.al:99 [239] I. Harjunkoski, T. Westerlund, R. Pörn, and H. Skrifvars. Different transformations for solving non-convex trim loss problems by MINLP. *European Journal of Operational Research*, 105:594–603, 1998. pages 125

hart2011pyomo [240] W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computations*, 3:219–260, 2011. pages 218

Hedmanetal:08 [241] K. W. Hedman, R. P. O'Neill, E. B. Fisher, and S. S. Oren. Optimal transmission switching - sensitivity analysis and extensions. *IEEE Transactions on Power Systems*, 23:1469–1479, 2008. pages 18, 128

kenschloss-08 [242] M. Heinkenschloss and D. Ridzal. *Lecture Notes in Computational Science and Engineering*, chapter Integration of Sequential Quadratic Programming and Domain Decomposition Methods for Nonlinear Optimal Control Problems. Springer-Verlag, 2008. pages 194

heinz:05 [243] S. Heinz. Complexity of integer quasiconvex polynomial optimization. *Journal of Complexity*, 21 (4):543 – 56, 2005. ISSN 0885-064X. URL http://dx.doi.org/10.1016/j.jco.2005.04.004. pages 121

Hellstrom2009 [244] E. Hellström, M. Ivarsson, J. Aslund, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17:245–254, 2009. pages 129

HemkerPhD [245] T. Hemker. *Derivative Free Surrogate Optimization for Mixed-Integer Nonlinear Black Box Problems in Engineering*. PhD thesis, Technischen Universität Darmstadt, Darmstadt, Germany, 2008. URL http://tuprints.ulb.tu-darmstadt.de/2162/1/hemker_diss.pdf. pages 122

Hemker08 [246] T. Hemker, K. Fowler, M. Farthing, and O. von Stryk. A mixed-integer simulation-based optimization approach with surrogate functions in water resources management. *Optimization and Engineering*, 9:341–360, 2008. doi:10.1007/s11081-008-9048-0. pages 122

cke-etal:mp11 [247] R. Hemmecke, S. Onn, and R. Weismantel. A polynomial oracle-time algorithm for convex integer minimization. *Mathematical Programming*, 126:97–117, 2011. URL http://dx.doi.org/10.1007/s10107-009-0276-7. 10.1007/s10107-009-0276-7. pages 121

omplexSystems [248] B. A. Hendrickson and M. H. Wright. Mathematical research challenges inoptimization of complex systems. Report on a Department of Energy Workshop, DOE-ASCR, 2006. pages 17

pjmferc [249] S. Herling. Planning tools and challenges. In *FERC Workshop: Increasing Market and Planning Efficiency Through Improved Software and Hardware-Enhanced wide-area planning models*, 2010. pages 10

hamiOuorou:10 [250] H. Hijazi, P. Bonami, and A. Ouorou. An outer-inner approximation for separable MINLPs. Technical report, LIF, Faculté des Sciences de Luminy, Université de Marseille, 2010. pages 144, 151

hildebrand:10 [251] R. Hildebrand and M. Köppe. A new Lenstra-type algorithm for quasiconvex polynomial integer minimization with complexity $2^{O(n \log n)}$, 2010. arxiv.org/abs/1006.4661. pages 121

oereporttrans [252] E. Hirst. U.S. transmission capacity: Present status and future prospects. Technical report, U.S. Department of Energy, 2004. pages 9

TOMLAB [253] K. Holmström and M. Edvall. The tomlab optimization environment. In J. Kallrath, editor, *Modeling languages in mathematical optimization*, pages 369–378. Kluwer Academic Publishers, Boston, MA, 2004. http://tomopt.com/tomlab/. pages 218

TOMLABManual [254] K. Holmström, A. O. Göran, and M. M. Edvall. *User's guide for TOMLAB 7*. Tomlab Optimization Inc., 2010. pages 218

alos.thoai:95 [255] H. Horst, P. M. Pardalos, and V. Thoai. *Introduction to Global Optimization*. Kluwer, Dordrecht, 1995. pages 165

horst.tuy:93 [256] R. Horst and H. Tuy. *Global Optimization*. Springer-Verlag, New York, 1993. pages 165, 176, 177

HuRalph:02 [257] X. Hu and D. Ralph. Convergence of a penalty method for mathematical programming with complementarity constraints. Technical report, Judge Institute of Management Science, England, UK, 2002. To appear in JOTA. pages 108

lenstra:mor83 [258] J. H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983. pages 121

cplex12 [259] IBM Ilog CPLEX. *IBM Ilog CPLEX V12.1: User's Manual for CPLEX*. IBM Corp., 2009. pages 170

me2009optimal [260] O. V. Iftime and M. A. Demetriou. Optimal control of switched distributed parameter systems with spatially scheduled actuators. *Automatica*, 45(2):312–323, 2009. pages 208

Jens:82 [261] R. Jensen. Adoption and diffusion of an innovation of uncertain profitability. 3:182–193, 1982. pages 14

lowlowe:mps84 [262] R. Jeroslow and J. Lowe. Modelling with integer variables. *Mathematical Programming Studies*, 22: 167–84, 1984. pages 168

owlowe:jors85 [263] R. Jeroslow and J. Lowe. Experimental results on the new techniques for integer programming formulations. *Journal of the Operational Research Society*, 36(5):393–403, 1985. pages 168, 169

Jeroslow:73 [264] R. G. Jeroslow. There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research*, 21(1):221–224, 1973. pages 115, 194

obst.et.al:01 [265] N. J. Jobst, M. D. Horniman, C. A. Lucas, and G. Mitra. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quantitative Finance*, 1:489–501, 2001. pages 128, 135

judiceMPEC [266] J. J. Júdice, H. D. Sherali, I. M. Ribeiro, and A. M. Faustino. A complementarity-based partitioning and disjunctive cut algorithm for mathematical programming problems with equilibrium constraints. *Journal of Global Optimization*, 36:89–114, 2006. pages 174

annanMonma:78 [267] R. Kannan and C. Monma. On the computational complexity of integer programming problems. In R. Henn, B. Korte, and W. Oettli, editors, *Optimization and Operations Research,*, volume 157 of *Lecture Notes in Economics and Mathematical Systems*, pages 161–172. Springer, 1978. pages 115, 194

.grossmann:06 [268] R. Karuppiah and I. E. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Computers & Chemical Engineering*, 30:650–673, 2006. pages 18, 128

Katsuo:07 [269] O. Katsuro-Hopkins, J. Bialek, D. A. Maurer, and G. A. Navratil. Enhanced ITER resistive wall mode feedback performance using optimal control techniques. *Nuclear Fusion*, 41:1157–1165, 2007. pages 17

nabncpwl:2006 [270] A. B. Keha, I. R. De Farias Jr., and G. L. Nemhauser. A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. *Operations Research*, 54(5):847–858, 2006. pages 169, 170

Kelley:60 [271] J. E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703712, 1960. pages 145

y1995particle [272] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995. pages 187

anporkolab:00 [273] L. Khachiyan and L. Porkolab. Integer optimization on convex semialgebraic sets. *Discrete & Computational Geometry*, 23:207–224, 2000. URL http://dx.doi.org/10.1007/PL00009496. 10.1007/PL00009496. pages 121

kilinc:11 [274] M. Kılınç. *Disjunctive Cutting Planes and Algorithms for Convex Mixed Integer Nonlinear Programming*. PhD thesis, Department of Industrial and Systems Engineering, University of Wisconsin-Madison, 2011. pages 158, 159

th.luedtke:10 [275] M. Kılınç, J. Linderoth, and J. Luedtke. Effective separation of disjunctive cuts for convex mixed integer nonlinear programs. Technical Report 1681, Computer Sciences Department, University of Wisconsin-Madison, 2010. pages 158, 159

Kirches2011a [276] C. Kirches. Fast numerical methods for mixed-integer nonlinear model-predictive control. In H. Bock, W. Hackbusch, M. Luskin, and R. Rannacher, editors, *Advances in Numerical Mathematics*. Springer Vieweg, Wiesbaden, July 2011. ISBN 978-3-8348-1572-9. PhD thesis, Ruprecht-Karls-Universität Heidelberg. pages 129

Kirches2011c [277] C. Kirches and S. Leyffer. TACO — a toolkit for AMPL control optimization. Preprint ANL/MCS-P1948-0911, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, U.S.A., October 2011. pages 217

Kirches2010 [278] C. Kirches, S. Sager, H. Bock, and J. Schlöder. Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 31(2):137–153, March/April 2010. pages 129

3optimization [279] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983. pages 187

Klepeis2003 [280] J. L. Klepeis and C. A. Floudas. ASTRO-FOLD: a combinatorial and global optimization framework for ab initio prediction of three-dimensional structures of proteins from the amino acid sequence. *Biophysical Journal*, 85:2119–2146, 2003. pages 128

.grossmann:88 [281] G. R. Kocis and I. E. Grossmann. Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. *Industrial Engineering Chemistry Research*, 27: 1407–1421, 1988. pages 122

:0968-090X:65 [282] A. Kotsialos, M. Papageorgiou, M. Mangeas, and H. Haj-Salem. Coordinated and integrated control of motorway networks via non-linear optimal control. *Transportation Research Part C: Emerging Technologies*, 10(1):65–84, 2002. doi:doi:10.1016/S0968-090X(01)00005-5. URL http://www.ingentaconnect.com/content/els/0968090x/2002/00000010/00000001/art00005. pages 17

Krokhmal2010 [283] P. A. Krokhmal and P. Soberanis. Risk optimization with p-order conic constraints: A linear programming approach. *European Journal of Operational Research*, 201(3):653–671, 2010. ISSN 0377-2217. pages 160

Lakhera2011 [284] S. Lakhera, U. V. Shanbhag, and M. McInerney. Approximating electrical distribution networks via mixed-integer nonlinear programming. *International Journal of Electric Power and Energy Systems*, 33(2):245–257, 2011. pages 128

land.doig:60 [285] A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960. pages 136

4142914 [286] S. Larrinaga, M. Vidal, E. Oyarbide, and J. Apraiz. Predictive control strategy for dc/ac converters based on direct power control. *Industrial Electronics, IEEE Transactions on*, 54(3):1261 –1271, june 2007. ISSN 0278-0046. doi:10.1109/TIE.2007.893162. pages 17

Lars:2001 [287] R. M. Larsen. Combining implicit restart and partial reorthogonalization in Lanczos bidiagnalization, 2001. http://sun.stanford.edu/~rmunk/PROPACK/. pages 96

e-lmi-quad:00 [288] J. Lasserre. Convergent LMI relaxations for nonconvex quadratic programs. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume vol.5, pages 5041 – 6, Piscataway, NJ, USA, 2000. pages 183

re-ip-heir:01 [289] J. Lasserre. An explicit exact SDP relaxation for nonlinear 0-1 programs. In K. Aardal and A. Gerards, editors, *Integer Programming and Combinatorial Optimization 2001*, Lecture Notes in Computer Science, Vol. 2081, pages 293 – 303, Berlin, Germany, 2001. pages 183

reviewtrans [290] G. Latorre, R. D. Cruz, J. M. Areiza, and A. Villegas. Classification of publications and models on transmission expansion planning. *IEEE Transactions on Power Systems*, 18:938–946, 2003. pages 9

misoferc [291] J. Lawhorn. MidwestISO planning process and models. In *FERC Workshop: Increasing Market and Planning Efficiency Through Improved Software and Hardware-Enhanced wide-area planning models*, 2010. pages 10

ler1966branch [292] E. L. Lawler and D. E. Woods. Branch-and-bound methods: A survey. *Operations Research*, 14(4): 699–719, 1966. pages 136

LeeLeyf:11 [293] J. Lee and S. Leyffer, editors. *Mixed Integer Nonlinear Programming*, IMA Volume in Mathematics and its Applications, 2011. Springer, New York. pages 122, 194

ewilson:dam01 [294] J. Lee and D. Wilson. Polyhedral methods for piecewise-linear functions I: The lambda method. *Discrete Applied Mathematics*, 108(3):269 – 285, 2001. pages 169, 171

Legg_13 [295] M. Legg, R. Davidson, and L. Nozick. Optimization-based regional hurricane mitigation planning. *Journal of Infrastructure Systems*, 19, 2013. pages 193

ineweber2003b [296] D. Leineweber, I. Bauer, A. Schäfer, H. Bock, and J. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization (Parts I and II). *Computers & Chemical Engineering*, 27:157–174, 2003. pages 217

leyffer:98b [297] S. Leyffer. User manual for MINLP-BB, 1998. University of Dundee. pages 215

MacMPEC [298] S. Leyffer. MacMPEC: AMPL collection of MPECs. Webpage, www.mcs.anl.gov/~leyffer/MacMPEC/, 2000. pages 101, 102

LeyS:01 [299] S. Leyffer. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization & Applications*, 18:295–309, 2001. pages 137

leyffer:03 [300] S. Leyffer. MacMINLP: Test problems for mixed integer nonlinear programming, 2003. http://www.mcs.anl.gov/~leyffer/macminlp. pages 125

Leyffer:13b [301] S. Leyffer, T. Munson, S. Wild, B. van Bloemen Waanders, and D. Ridzal. Mixed-integer pde-constrained optimization. Position Paper #15 submitted in response to the ExaMath13 Call for Position Papers, August 2013. https://collab.mcs.anl.gov/download/attachments/7569466/examath13_submission_15.pdf. pages 193

li2011optimal [302] J.-S. Li, J. Ruths, T.-Y. Yu, H. Arthanari, and G. Wagner. Optimal pulse design in quantum control: A unified computational method. *PNAS*, 108(5):879–1884, February 2011. pages 17

1331480 [303] Y. Li, D. Vilathgamuwa, and P. C. Loh. Design, analysis, and real-time testing of a controller for multibus microgrid system. *Power Electronics, IEEE Transactions on*, 19(5):1195 – 1204, sept. 2004. ISSN 0885-8993. doi:10.1109/TPEL.2004.833456. pages 17

ntelides:2003 [304] L. Liberti and C. C. Pantelides. Convex envelopes of monomials of odd degree. *Journal of Global Optimization*, 25(2):157–168, 2003. pages 174

rti2011recipe [305] L. Liberti, N. Mladenović, and G. Nannicini. A recipe for finding good solutions to MINLPs. *Mathematical Programming Computations*, 3:349–390, 2011. pages 192

1255660 [306] C.-C. Lin, H. Peng, J. Grizzle, and J.-M. Kang. Power management strategy for a parallel hybrid electric truck. *Control Systems Technology, IEEE Transactions on*, 11(6):839 – 849, nov. 2003. ISSN 1063-6536. doi:10.1109/TCST.2003.815606. pages 17

indoGlobal:09 [307] Y. Lin and L. Schrage. The global solver in the LINDO API. *Optimization methods and software*, 24 (4):657–668, 2009. pages 217

inderoth:05-1 [308] J. T. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Mathematical Programming, Series B*, 103:251–282, 2005. pages 184

avelsbergh:99 [309] J. T. Linderoth and M. W. P. Savelsbergh. A computational study of search strategies in mixed integer programming. *INFORMS Journal on Computing*, 11:173–187, 1999. pages 135, 137

LiuSun:02 [310] X. Liu and J. Sun. Generalized stationary points and an interior-point method for mathematical programs with equilibrium constraints. *Mathematical Programming*, 101(1):231–261, 2004. pages 107

LLR11 [311] G. Liuzzi, S. Lucidi, and F. Rinaldi. Derivative-free methods for bound constrained mixed-integer optimization. *Computational Optimization and Applications*, pages 1–22, 2011. doi:10.1007/s10589-011-9405-3. pages 122

erg2004yalmip [312] J. Löfberg. Yalmip : a toolbox for modeling and optimization in matlab. In *IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, sept. 2004. pages 218

1159919 [313] W. Lu and B.-T. Ooi. Optimal acquisition and aggregation of offshore wind power by multiterminal voltage-source hvdc. *Power Delivery, IEEE Transactions on*, 18(1):201 – 206, jan 2003. ISSN 0885-8977. doi:10.1109/TPWRD.2002.803826. pages 17

dtkeetal:mp12 [314] J. Luedtke, M. Namazifar, and J. Linderoth. Some results on the strength of relaxations of multilinear functions. *Mathematical Programming*, pages 1–27, 2012. ISSN 0025-5610. URL http://dx.doi.org/10.1007/s10107-012-0606-z. 10.1007/s10107-012-0606-z. pages 184

LuoPanRal:96 [315] Z.-Q. Luo, J.-S. Pang, and D. Ralph. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, Cambridge, UK, 1996. pages 102, 107

uoPanRalWu:96 [316] Z.-Q. Luo, J.-S. Pang, D. Ralph, and S.-Q. Wu. Exact penalization and stationarity conditions of mathematical programs with equilibrium constraints. *Mathematical Programming*, 75(1):19–76, 1996. pages 101

Ma20021103 [317] D. L. Ma, D. K. Tafti, and R. D. Braatz. Optimal control and simulation of multidimensional crystallization processes. *Computers & Chemical Engineering*, 26(7-8):1103 – 1116, 2002. ISSN 0098-1354. doi:10.1016/S0098-1354(02)00033-9. URL http://www.sciencedirect.com/science/article/pii/S0098135402000339. pages 17

MINOTAUR [318] A. Mahajan, S. Leyffer, J. Linderoth, J. Luedtke, and T. Munson. MINOTAUR: a toolkit for solving mixed-integer nonlinear optimization. wiki-page, 2011. http://wiki.mcs.anl.gov/minotaur. pages 148

ferKirches:12 [319] A. Mahajan, S. Leyffer, and C. Kirches. Solving mixed-integer nonlinear programs by QP-diving. Preprint ANL/MCS-2071-0312, Argonne National Laboratory, Mathematics and Computer Science Division, 2012. pages 136, 137, 191

ManO:69 [320] O. Mangasarian. *Nonlinear Programming*. McGraw-Hill Book Company, New York, 1969. pages 76

MaonanHu:91 [321] L. Maonan and H. Wenjun. The study of choosing opiimal plan of air quantities regulation of mine ventilation network. In *Proceedings of the 5th US Mine Ventilation Symposium*, pages 427–421, 1991. pages 125

MarN:78 [322] N. Maratos. *Exact penalty function algorithms for finite dimensional and control optimization problems*. Ph.D. thesis, University of London, 1978. pages 84

Mariaetal:09 [323] J. Maria, T. T. Truong, J. Yao, T.-W. Lee, R. G. Nuzzo, S. Leyffer, S. K. Gray, and J. A. Rogers. Optimization of 3D plasmonic crystal structures for refractive index sensing. *Journal of Physical Chemistry C*, 113(24):10493–10499, 2009. pages 18, 122

itzmanne:1957 [324] H. M. Markowitz and A. S. Manne. On the solution of discrete programming problems. *Econometrica*, 25(1):84–110, 1957. pages 165, 169

Martin [325] A. Martin, M. Möller, and S. Moritz. Mixed integer models for the stationary case of gas network optimization. *Mathematical Programming*, 105:563–582, 2006. pages 128, 193

asihabadi2011 [326] S. Masihabadi, S. Sanjeevi, and K. Kianfar. n-step conic mixed integer rounding inequalities. Optimization Online, November 2011. http://www.optimization-online.org/DB_HTML/2011/11/3251.html. pages 160

mccormick:76 [327] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976. pages 172, 174, 176, 177

messine2 [328] F. Messine. Deterministic global optimization using interval constraint propagation techniques. *RAIRO-RO*, 38(4):277–294, 2004. pages 178

meyer:pwl76 [329] R. Meyer. Mixed integer minimization models for piecewise-linear functions of a single variable. *Discrete Mathematics*, 16(2):163–71, 1976. pages 168

629701 [330] A. Miller, E. Muljadi, and D. Zinger. A variable speed wind turbine power control. *Energy Conversion, IEEE Transactions on*, 12(2):181 –186, jun 1997. ISSN 0885-8969. doi:10.1109/60.629701. pages 17

Milleretal:10 [331] R. Miller, Z. Xie, S. Leyffer, M. Davis, and S. Gray. Surrogate-based modeling of the optical response of metallic nanostructures. *Journal of Physical Chemistry C*, 114(48):20741–20748, 2010. DOI:10.1021/jp1067632. pages 18, 122

er2012glomiqo [332] R. Misener and C. Floudas. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, pages 1–48, 2012. pages 175, 217

rfloudas:mp12 [333] R. Misener and C. Floudas. Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. *Mathematical Programming*, pages 1–28, 2012. ISSN 0025-5610. URL http://dx.doi.org/10.1007/s10107-012-0555-6. 10.1007/s10107-012-0555-6. pages 184

shape [334] B. Mohammadi and O. Pironneau. Shape optimization in fluid mechanics. *Annual Review of Fluid Mechanics*, 36:255–279, 2004. pages 17

4840074 [335] J. Momoh. Smart grid design for efficient and flexible power networks operation and control. In *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pages 1 –8, march 2009. doi:10.1109/PSCE.2009.4840074. pages 17

Momoh [336] J. Momoh, R. Koessler, M. Bond, B. Stott, D. Sun, A. Papalexopoulos, and P. Ristanovic. Challenges to optimal power flow. *IEEE Transaction on Power Systems*, 12:444–455, 1997. pages 128

MorTor:91 [337] J. J. Moré and G. Toraldo. On the solution of quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113, 1991. pages 90

MoritzPhD [338] S. Moritz. *A Mixed Integer Approach for the Transient Caseof Gas Network Optimization*. PhD thesis, Technische Universität Darmstadt, 2007. pages 17

MuellerPhD [339] J. Müller. *Surrogate Model Algorithms for Computationally Expensive Black-Box Global Optimization Problems*. PhD thesis, Tampere University of Technology, Tampere, Finland, 2012. pages 122

Mueller2012 [340] J. Müller, C. A. Shoemaker, and R. Piché. SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers & Operations Research*, 2012. doi:10.1016/j.cor.2012.08.022. To appear. pages 122

i2012rounding [341] G. Nannicini and P. Belotti. Rounding-based heuristics for nonconvex MINLPs. *Mathematical Programming Computation*, 4:1–31, 2012. pages 188

cini2008local [342] G. Nannicini, P. Belotti, and L. Liberti. A local branching heuristic for MINLPs. arXiv:0812.2188v1 [math.CO], 2008. http://arxiv.org/abs/0812.2188. pages 192

ser.wolsey:88 [343] G. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York, 1988. pages 121, 180

nemwol:88 [344] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, New York, 1988. pages 169

sigismondi:94 [345] G. L. Nemhauser, M. W. P. Savelsbergh, and G. C. Sigismondi. MINTO, a Mixed INTeger Optimizer. *Operations Research Letters*, 15:47–58, 1994. pages 215

NemiTodd:08 [346] A. Nemirovski and M. Todd. Interior-point methods for optimization. *Acta Numerica*, 17:181–234, 2008. pages 81

NestNemi:94 [347] Y. Nesterov and A. Nemirovskii. *Interior Point Polynomial Algorithms in Convex Programming*. Number 13 in Studies in Applied Mathematics. SIAM, 1994. pages 80

NocWri:99 [348] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, New York, 1999. pages 84, 98, 121

Nocedal-00 [349] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2000. pages 194

n.vigerske:03 [350] I. Nowak, H. Alperin, and S. Vigerske. LaGO — an object oriented library for solving MINLPs. In C. Bliek, C. Jermann, and A. Neumaier, editors, *Proceedings of the 1st Global Optimization and Constraint Satisfaction Workshop (COCOS 2002)*, number 2861 in Lecture Notes in Computer Science, pages 32–42, Berlin/Heidelberg, 2003. Springer. pages 175, 217

Oldenburg2003 [351] J. Oldenburg, W. Marquardt, D. Heinz, and D. Leineweber. Mixed logic dynamic optimization applied to batch distillation process design. *AIChE Journal*, 49(11):2900–2917, 2003. pages 129

OPTPDE [352] OPTPDE. OPTPDE — a collection of problems in PDE-constrained optimization. URL http://www.optpde.net. http://www.optpde.net. pages 202, 205

OutKocZow:98 [353] J. Outrata, M. Kocvara, and J. Zowe. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints*. Kluwer Academic Publishers, Dordrecht, 1998. pages 102

Ozdogan_04 [354] U. Ozdogan. Optimization of well placement under time-dependent uncertainty. Master's thesis, Stanford University, 2004. pages 193, 199

padberg:89 [355] M. Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Mathematical Programming, Series B*, 45(1):139 – 72, 1989. pages 183

padberg:orl00 [356] M. Padberg. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters*, 27(1):1–5, 2000. pages 169

PangLeyf:04 [357] J. Pang and S. Leyffer. On the global minimization of the value-at-risk. *Optimization Methods and Software*, 19(5):611–631, 2004. pages 101

936493 [358] A. Piccolo, L. Ippolito, V. zo Galdi, and A. Vaccaro. Optimisation of energy flow management in hybrid electric vehicles via genetic algorithms. In *Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on*, volume 1, pages 434 –439 vol.1, 2001. doi:10.1109/AIM.2001.936493. pages 17

osborn [359] R. Piwko, D. Osborn, R. Gramlich, G. Jordan, D. Hawkins, and K. Porter. Wind energy delivery issues - transmission planning and competitive electricity market operation. *IEEE Power and Energy Magazine*, 3:47–56, 2005. pages 9

PowMJD:78a [360] M. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. Watson, editor, *Numerical Analysis, 1977*, pages 144–157. Springer–Verlag, Berlin, 1978. pages 78

Po07 [361] R. Powell. Defending against terrorist attacks with limited resources. *American Political Science Review*, 101(3):527–541, 2007. pages 129

Prata2008 [362] A. Prata, J. Oldenburg, A. Kroll, and W. Marquardt. Integrated scheduling and dynamic optimization of grade transitions for a continuous polymerization reactor. *Computers & Chemical Engineering*, 32:463–476, 2008. pages 128

Pruitt2012 [363] K. A. Pruitt, S. Leyffer, A. M. Newman, and R. Braun. Optimal design and dispatch of distributed generation systems. Preprint ANL/MCS-2004-0112, Argonne National Laboratory, Mathematics and Computer Science Division, January 2012. pages 128

4495554 [364] W. Qiao, W. Zhou, J. Aller, and R. Harley. Wind speed estimation based sensorless output maximization control for a wind turbine driving a DFIG. *Power Electronics, IEEE Transactions on*, 23(3):1156–1169, may 2008. ISSN 0885-8993. doi:10.1109/TPEL.2008.921185. pages 17

lizza.etal:12 [365] A. Qualizza, P. Belotti, and F. Margot. Linear programming relaxations of quadratically constrained quadratic programs. In *Mixed Integer Nonlinear Programming*, volume 154 of *IMA Volume Series in Mathematics and its Applications*, pages 407–426. Springer, 2012. pages 183, 216

.grossmann:92 [366] I. Quesada and I. E. Grossmann. An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems. *Computers & Chemical Engineering*, 16:937–947, 1992. pages 146, 194

st.gemeert:98 [367] A. J. Quist, R. van Gemeert, J. E. Hoogenboom, T. Ílles, C. Roos, and T. Terlaky. Application of nonlinear optimization to reactor core fuel reloading. *Annals of Nuclear Energy*, 26:423–448, 1998. pages 18, 128

Quist:98 [368] A. J. Quist, E. de Klerk, C. Roos, T. Terlaky, R. van Geemert, J. Hoogenboom, and T. Illés. Finding optimal nuclear reactor core reload patterns using nonlinear optimization and search heuristics. *Engineering Optimization*, 32(2):143–176, 1999. pages 18

Rabitzetal:00 [369] H. Rabitz, R. de Vivie-Riedle, M. Motzkus, and K. Kompa. Whither the future of controlling quantum phenomena? *Science*, 288(5467):824–828, 2000. pages 17

RaghBieg:05 [370] A. Raghunathan and L. T. Biegler. An interior point method for mathematical programs with complementarity constraints (MPCCs). *SIAM Journal on Optimization*, 15(3):720–750, 2005. pages 107

Rashid12 [371] K. Rashid, S. Ambani, and E. Cetinkaya. An adaptive multiquadric radial basis function method for expensive black-box mixed-integer nonlinear constrained optimization. *Engineering Optimization*, to appear:1–22, 2012. doi:10.1080/0305215X.2012.665450. pages 122

Reinke_PRE_11 [372] C. M. Reinke, T. M. D. la Mata Luque, M. F. Su, M. B. Sinclair, and I. El-Kady. Group-theory approach to tailored electromagnetic properties of metamaterials: An inverse-problem solution. *Physical Review E*, 83(6):066603–1–18, 2011. pages 193

RobSM:74 [373] S. M. Robinson. Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms. *Mathematical Programming*, 7(1):1–16, 1974. pages 91

Romeroetal:02 [374] R. Romero, A. Monticelli, A. Garcia, and S. Haffner. Test systems and mathematical models for transmission network expansion planning. *IEEE Proceedings — Generation, Transmission and Distrbution.*, 149(1):27–36, 2002. pages 18, 128

tesandwich:92 [375] G. Rote. The convergence rate of the sandwich algorithm for approximating convex functions. *Computing*, 48:337–61, 1992. pages 167

tein2004cross [376] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer, New York, 2004. pages 187

.sahinidis:95 [377] H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19:552–566, 1995. pages 175, 180

.sahinidis:96 [378] H. S. Ryoo and N. V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8:107–139, 1996. pages 175

Sager2005 [379] S. Sager. *Numerical methods for mixed-integer optimal control problems*. Der andere Verlag, Tönning, Lübeck, Marburg, 2005. ISBN 3-89959-416-9. pages 126, 128, 196, 217

Sager2007 [380] S. Sager, M. Diehl, G. Singh, A. Küpper, and S. Engell. Determining SMB superstructures by mixed-integer control. In *Proceedings of OR2006*, pages 37–44, Karlsruhe, 2007. Springer. pages 129

sahinidis:96 [381] N. V. Sahinidis. BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8:201–205, 1996. pages 175, 216

SA03 [382] T. Sandler and D. G. Arce M. Terrorism and game theory. *Simulation Gaming*, 34:319–337, 2003. pages 129

SS06 [383] T. Sandler and K. Siqueira. Global terrorism: Deterrence versus preemption. *Canadian Journal of Economics*, 39(4):1370–1387, 2006. pages 129

nn.sargent:79 [384] I. E. G. R. W. H. Sargent. Optimal design of multipurpose batch plants. *Industrial & Engineering Chemistry Process Design and Development*, 18:343–348, 1979. pages 122

avelsbergh:94 [385] M. W. P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6:445–454, 1994. pages 147, 178, 180

nami.lee:mp10 [386] A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. *Mathematical Programming*, 124:383–411, 2010. pages 184

nami.lee:mp11 [387] A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Mathematical Programming*, 130:359–413, 2011. pages 183

SchSch:00 [388] H. Scheel and S. Scholtes. Mathematical program with complementarity constraints: Stationarity, optimality and sensitivity. *Mathematics of Operations Research*, 25:1–22, 2000. pages 102, 103, 104, 105

chl2004global [389] H. Schichl. Global optimization in the COCONUT project. *Numerical Software with Result Verification*, pages 277–293, 2004. pages 216

SchS:01 [390] S. Scholtes. Convergence properties of regularization schemes for mathematical programs with complementarity constraints. *SIAM J. Optimization*, 11(4):918–936, 2001. pages 102, 107

sch:86 [391] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, New York, 1986. pages 133

er1999process [392] C. A. Schweiger. *Process Synthesis, Design, and Control: Optimization with Dynamic Models and Discrete Decisions*. PhD thesis, Princeton University, Princeton, NJ, 1999. pages 215

Shaik2008 [393] O. Shaik, S. Sager, O. Slaby, and D. Lebiedz. Phase tracking and restoration of circadian rhythms by model-based optimal control. *IET Systems Biology*, 2:16–23, 2008. pages 128

Sharma:13 [394] S. Sharma. Mixed-integer nonlinear programming heuristics applied to a shale gas production optimization problem. Master's thesis, Norwegian University of Science and Technology, 2013. http://www.diva-portal.org/smash/get/diva2:646797/FULLTEXT01.pdf. pages 193

ikhGhafoor:07 [395] W. Sheikh and A. Ghafoor. An optimal bandwidth allocation and data droppage scheme for differentiated services in a wireless network. *Wireless Communications and Mobile Computing*, 10(6): 733–747, 2010. pages 128

rali.adams:98 [396] H. Sherali and W. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer, Dordrecht, 1998. pages 182

herali1992new [397] H. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2(4):379–410, 1992. pages 182

eralismith:97 [398] H. Sherali and E. Smith. A global optimization approach to a water distribution network design problem. *Journal of Global Optimization*, 11:107–132, 1997. pages 122

sherali:orl01 [399] H. D. Sherali. On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. *Operations Research Letters*, 28:155 – 160, 2001. pages 168

celli.sherali [400] H. D. Sherali and B. M. P. Fraticelli. Enhancing RLT relaxations via a new class of semidefinite cuts. *Journal of Global Optimization*, 22:233–261, 2002. pages 183

eraliwater:01 [401] H. D. Sherali, S. Subramanian, and G. V. Loganathan. Effective relaxations and partitioning schemes for solving water distribution network design problems to global optimality. *Journal of Global Optimization*, 19:1–26, 2001. pages 122

Simon-08 [402] R. Simon. *Multigrid Solver for Saddle Point Problems in PDE-Constrained Optimization*. PhD thesis, Johannes Kepler Universitat Linz, 2008. pages 194

Sinhaetal:02 [403] R. Sinha, A. Yener, and R. D. Yates. Noncoherent multiuser communications: Multistage detection and selective filtering. *EURASIP Journal on Applied Signal Processing*, 12:1415–1426, 2002. pages 128

fvars19981829 [404] H. Skrifvars, S. Leyffer, and T. Westerlund. Comparison of certain minlp algorithms when applied to a model structure determination and parameter estimation problem. *Computers & Chemical Engineering*, 22(12):1829 – 1835, 1998. ISSN 0098-1354. doi:10.1016/S0098-1354(98)00238-5. URL http://www.sciencedirect.com/science/article/pii/S0098135498002385. pages 123

pantelides:97 [405] E. M. B. Smith and C. C. Pantelides. Global optimization of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791–S796, 1997. pages 172, 175

kolowski:1992 [406] J. Sokolowski and J. P. Zolasio. *Introduction to Shape Optimization: Shape Sensitivity Analysis*, volume 16. Springer-Verlag, 1992. pages 17

ipouretal:02 [407] M. Soleimanipour, W. Zhuang, and G. H. Freeman. Optimal resource management in wireless multimedia wideband CDMA systems. *IEEE Transactions on Mobile Computing*, 1(2):143–160, 2002. pages 128

Soler2011 [408] M. Soler, A. Olivares, E. Staffetti, and P. Bonami. En-route optimal flight planning constrained to pass through waypoints using MINLP. In *Proceedings of 9th USA/Europe Air Traffic Management Research and Development Seminar*, Berlin, June 14–17 2011. pages 128

Song2000293 [409] Y. Song, B. Dhinakaran, and X. Bao. Variable speed control of wind turbines using nonlinear and adaptive algorithms. *Journal of Wind Engineering and Industrial Aerodynamics*, 85(3): 293 – 308, 2000. ISSN 0167-6105. doi:10.1016/S0167-6105(99)00131-2. URL http://www.sciencedirect.com/science/article/pii/S0167610599001312. pages 17

StackH:52 [410] H. V. Stackelberg. *The Theory of Market Economy*. Oxford University Press, 1952. pages 101

inbach2007pde [411] M. C. Steinbach. On PDE solution in transient optimization of gas networks. *Journal of computational and applied mathematics*, 203(2):345–361, 2007. pages 193

westerlund:06 [412] C. Still and T. Westerlund. Solving convex MINLP optimization problems using a sequential cutting plane algorithm. *Comput. Optim. Appl.*, 34(1):63–83, 2006. pages 147, 194

s.mehrotra:99 [413] R. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86:515–532, 1999. pages 137, 138, 156, 157, 158, 163, 194

s.mehrotra:02 [414] R. Stubbs and S. Mehrotra. Generating convex polynomial inequalities for mixed 0-1 programs. *Journal of Global Optimization*, 24:311–332, 2002. pages 158

DOEexafusion [415] W. Tang and D. Keyes. Scientific grand challenges: Fusion energy sciences and the role of computing at the extreme scale, March 2009. URL http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/Fusion_report.pdf. pages 18

Fusion [416] W. Tang, D. Keyes, N. Sauthoff, N. Gorelenkov, J. R. Cary, A. H. Kritz, S. Zinkle, J. N. Brooks, R. Betti, W. Mori, A. Bhattacharjee, W. Daughton, and A. Shoshani. Scientific grand challenges: Fusion energy science and the role of computing at the extreme scale. Report from the workshop held March 18-20, 2009, U.S. Department of Energy, Office of Fusion Energy Sciences and the Office of Advanced Scientific Computing Research, 2009. pages 17

.sahinidis:02 [417] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Boston MA, 2002. pages 165, 172, 175, 177, 180, 194

.sahinidis:04 [418] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004. pages 175

65216.1065219 [419] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, June 2005. ISSN 0025-5610. doi:10.1007/s10107-005-0581-8. URL http://dx.doi.org/10.1007/s10107-005-0581-8. pages 150, 151

mohitjp:10 [420] M. Tawarmalani, J.-P. Richard, and K. Chung. Strong valid inequalities for orthogonal disjunctions and bilinear covering sets. *Mathematical Programming*, 124:481–512, 2010. 10.1007/s10107-010-0374-6. pages 184, 185

JPE:JPE979 [421] C. M. Taylor and A. Hastings. Finding optimal control strategies for invasive species: a density-structured model for Spartina alterniflora. *Journal of Applied Ecology*, 41(6):1049–1057, 2004. ISSN 1365-2664. doi:10.1111/j.0021-8901.2004.00979.x. URL http://dx.doi.org/10.1111/j.0021-8901.2004.00979.x. pages 17

Terwen2004 [422] S. Terwen, M. Back, and V. Krebs. Predictive powertrain control for heavy duty trucks. In *Proceedings of IFAC Symposium in Advances in Automotive Control*, pages 451–457, Salerno, Italy, 2004. pages 129

Tomlin [423] J. Tomlin. A suggested extension of special ordered sets to non-separable non-convex programming problems. *Annals of Discrete Mathematics*, 11:359–370, 1981. pages 169, 171

oriello:pwl12 [424] A. Toriello and J. P. Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219:86–95, 2012. pages 167

roeltzsch1984 [425] F. Tröltzsch. The generalized bang-bang-principle and the numerical solution of a parabolic boundary-control problem with constraints on the control and the state. *Zeitschrift für Angewandte Mathematik und Mechanik*, 64(12):551–556, 1984. ISSN 0044-2267. doi:10.1002/zamm.19840641218. pages 205

eltzsch2010:1 [426] F. Tröltzsch. *Optimal Control of Partial Differential Equations*, volume 112 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, 2010. pages 202, 203

rkay1996logic [427] M. Türkay and I. E. Grossmann. Logic-based MINLP algorithms for the optimal synthesis of process networks. *Computers & Chemical Engineering*, 20(8):959–978, 1996. pages 149

UlbUlbVic:04 [428] M. Ulbrich, S. Ulbrich, and L. Vicente. A globally convergent primal-dual interior-point filter method for nonlinear programming. *Mathematical Programming*, 100(2):379–410, 2004. pages 94

387896 [429] T. Van Cutsem. An approach to corrective control of voltage instability using simulation and sensitivity. *Power Systems, IEEE Transactions on*, 10(2):616 –622, may 1995. ISSN 0885-8950. doi:10.1109/59.387896. pages 17

vanroy:83 [430] T. J. Van Roy. Cross decomposition for mixed integer programming. *Mathematical Programming*, 25:145–163, 1983. pages 141

.nemhauser:05 [431] D. Vandenbussche and G. L. Nemhauser. A polyhedral study of nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102:531–557, 2005. pages 184

emhauser:05-2 [432] D. Vandenbussche and G. L. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102:559–575, 2005. pages 184

a2011pwlinlog [433] J. P. Vielma and G. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1):49–72, 2011. pages 170

Vielma2008 [434] J. P. Vielma, S. Ahmed, and G. L. Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs. *INFORMS Journal on Computing*, 20(3):438–450, 2008. pages 160

elma2010pwlin [435] J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010. pages 166, 168, 169, 171

n1990combined [436] J. Viswanathan and I. E. Grossmann. A combined penalty function and outer-approximation method for MINLP optimization. *Computers & Chemical Engineering*, 14(7):769–782, 1990. pages 215

er.biegler:06 [437] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. pages 215

4739156 [438] M. Walker, E. Schuster, D. Mazon, and D. Moreau. Open and emerging control problems in tokamak plasma control. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3125 –3132, dec. 2008. doi:10.1109/CDC.2008.4739156. pages 17

.lundqvist:01 [439] T. Westerlund and K. Lundqvist. Alpha-ECP, Version 5.01: An interactive MINLP-solver based on the Extended Cutting Plane Method. Technical Report Report 01-178-A, Process Design Laboratory at Åbo University, 2001. pages 145

.lundqvist:05 [440] T. Westerlund and K. Lundqvist. Alpha-ECP, version 5.101: An interactive MINLP-solver based on the Extended Cutting Plane Method. Technical Report Report 01-178-A, Process Design Laboratory at Åbo University, 2005. pages 214

pettersson:95 [441] T. Westerlund and F. Pettersson. A cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19:s131–s136, 1995. pages 141, 145

nd2002solving [442] T. Westerlund and R. Pörn. Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. *Optimization and Engineering*, 3(3):253–280, 2002. pages 214

williams:99 [443] H. P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, 1999. pages 122, 123, 126, 134

wilson:98 [444] D. L. Wilson. *Polyhedral Methods for Piecewise-Linear Functions*. PhD thesis, University of Kentucky, 1998. pages 169, 171

Woer:04 [445] C. Woerlen. Experience curves for energy technologies. 2:641–649, 2004. pages 15

WolfSmeers:00 [446] D. D. Wolf and Y. Smeers. The gas transmission problem solved by an extension of the simplex algorithm. *Management Science*, 46(11):1454–1465, 2000. pages 125

wolsey:98 [447] L. A. Wolsey. *Integer Programming*. John Wiley and Sons, New York, 1998. pages 121, 150

WriMH:92 [448] M. Wright. Interior methods for constrained optimization. *Acta Numerica*, 1:341–407, 1992. pages 81

4295002 [449] F. Wu, X.-P. Zhang, K. Godfrey, and P. Ju. Small signal stability analysis and optimal control of a wind turbine with doubly fed induction generator. *Generation, Transmission Distribution, IET*, 1(5): 751 –760, september 2007. ISSN 1751-8687. doi:10.1049/iet-gtd:20060395. pages 17

uzKarfakis:91 [450] X. Wu, E. Topuz, and M. Karfakis. Optimization of ventilation control device locations and sizes in underground mine ventilation systems. In *Proceedings of the 5th US Mine Ventilation Symposium*, pages 391–399, 1991. pages 125

yajimafuj:98 [451] Y. Yajima and T. Fujie. Polyhedral approach for nonconvex quadratic programming problems with box constraints. *Journal of Global Optimization*, 13(2):151 – 170, 1998. pages 183

Seideman_2008 [452] J. Yelk, M. Sukharev, and T. Seideman. Optimal design of nanoplasmonic materials using genetic algorithms as a multiparameter optimization tool. *The Journal of Chemical Physics*, 129(6):064706, 2008. URL http://arxiv.org/abs/0802.2899. pages 17

YouLeyf:10 [453] F. You and S. Leyffer. Oil spill response planning with MINLP. *SIAG/OPT Views-and-News*, 21 (2):1–8, 2010. URL http://www.mcs.anl.gov/uploads/cels/papers/P1846.pdf. pages 18

YouLeyf:10a [454] F. You and S. Leyffer. Oil spill response planning with MINLP. *SIAG/OPT Views-and-News*, 21(2): 1–8, 2010. pages 128, 193

YouLeyf:10b [455] F. You and S. Leyffer. Mixed-integer dynamic optimization for oil-spill response planning with integration of a dynamic oil weathering model. *AIChe Journal*, 2011. Published online: DOI: 10.1002/aic.12536. pages 128, 193

YouLeyf:11 [456] F. You and S. Leyffer. Mixed-integer dynamic optimization for oil-spill response planning with integration of a dynamic oil weathering model. *AIChe Journal*, 57(12):3555–3564, 2011. doi:10.1002/aic.12536. pages 18

014stochastic [457] V. M. Zavala. Stochastic optimal control model for natural gas networks. *Computers & Chemical Engineering*, 64:103–113, 2014. pages 193

Zhang-13 [458] P. Zhang, D. Romero, J. Beck, and C. Amon. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, chapter Solving Wind Farm Layout Optimization with Mixed Integer Programming and Constraint Programming. Springer Verlag, 2013. pages 193

zhu.kuno:96 [459] Y. Zhu and T. Kuno. A disjunctive cutting-plane-based branch-and-cut algorithm for 0-1 mixed-integer convex nonlinear programs. *Industrial and Engineering Chemistry Research*, 45:187–196, 2006. pages 159

Zh08 [460] J. Zhuang. *Modeling Secrecy and Deception in Homeland Security Resource Allocation*. Ph.D. thesis, University of Wisconsin-Madison, 2008. pages 129

ZB07a [461] J. Zhuang and V. M. Bier. Investment in security. *Industrial Engineer*, 39:53–54, 2007. pages 129

ZB07b [462] J. Zhuang and V. M. Bier. Balancing terrorism and natural disasters — defensive strategy with endogenous attacker effort. *Operations Research*, 55:976–991, 2007. pages 129

ZB07c [463] J. Zhuang and V. M. Bier. Modeling secrecy and deception in homeland security resource allocation, 2007. Submitted. pages 129

knitromanual [464] KNITRO. *KNITRO Documentation*. Ziena Optimization, dec. 2012. pages 215

Index