Complete search in continuous global optimization and constraint satisfaction

Arnold Neumaier

Institut für Mathematik, Universität Wien,

Nordbergstraße 15, A-1090 Wien, Austria

E-mail: `Arnold.Neumaier@univie.ac.at`

`www.mat.univie.ac.at/~neum/`

This survey covers the state of the art of techniques for solving general-purpose constrained global optimization problems and continuous constraint satisfaction problems, with emphasis on complete techniques that provably find all solutions (if there are finitely many). The core of the material is presented in sufficient detail that the survey may serve as a text for teaching constrained global optimization.

After giving motivations for and important examples of applications of global optimization, a precise problem definition is given, and a general form of the traditional first-order necessary conditions for a solution. Then more than a dozen software packages for complete global search are described.

A quick review of incomplete methods for bound-constrained problems and recipes for their use in the constrained case follows; an explicit example is discussed, introducing the main techniques used within branch and bound techniques. Sections on interval arithmetic, constrained propagation and local optimization are followed by a discussion of how to avoid the cluster problem. Then a discussion of important problem transformations follows, in particular of linear, convex, and semilinear (= mixed integer linear) relaxations that are important for handling larger problems.

Next, reliability issues – centring on rounding error handling and testing methodologies – are discussed, and the COCONUT framework for the integration of the different techniques is introduced. A list of challenges facing the field in the near future concludes the survey.

1. Introduction

Consider everything. Keep what is good. Avoid evil whenever you recognize it.
St. Paul, ca. 50 A.D. (*The Bible*, 1 Thess. 5:21–22)

1.1. Early history

As the above quote shows, continuous global optimization or constraint satisfaction and the associated global search methods have been part of the art of successful living since antiquity. In the mathematical literature published before 1975, there are occasional references related to the topic, few and important enough to mention them individually. (Please inform me about other significant work on continuous global optimization published before 1975 not mentioned here!) Several independent strands of work (probably done in ignorance of each other) are discernible.

(1) Markowitz and Manne (1957) and Dantzig *et al.* (Dantzig, Johnson and White 1958, Dantzig 1960) used piecewise linear approximations for the approximate global minimization of separable nonconvex programs, formulating them as mixed integer linear programs. Land and Doig (1960) and Little, Murty, Sweeney and Karel (1963) introduced the branch and bound technique for discrete optimization, applicable to mixed integer linear programs. Motzkin and Strauss (1965) showed that solving the (discrete) maximum clique problem is equivalent to finding the global minimum (or maximum) of a special nonconvex quadratic program. Falk and Soland (1969) gave the first piecewise linear relaxations of nonconvex problems, thus making them available for obtaining bounds in a branch and bound scheme. Beale and Tomlin (1970) and Tomlin (1970) introduced special ordered sets, defining the way piecewise linear functions are still handled today in mixed integer linear programming solvers. McCormick (1972) introduced the now frequently used linear relaxations for products and quotients, which made the solution of general factorable global optimization problems accessible to the branch and bound technique.

(2) Moore (1962) showed in Part 4 of his PhD thesis, which introduced interval arithmetic to numerical analysis (following an unpublished technical report by Moore and Yang (1959)), that by repeated subdivision and simple interval evaluation, the range – hence in particular the global minimum – of a rational function over a box can be determined in principle to arbitrary accuracy. Skelboe (1974) improved this basic but excessively slow method by embedding it into (what would today be called) a branch and bound scheme for continuous variables, giving (what is now called) the Moore–Skelboe algorithm. Moore's thesis (Moore 1962) also showed that interval methods can be used to prove the nonexistence of solutions of nonlinear systems in a box (which nowadays is used to discard boxes in a branch and bound scheme) and to reduce the region where a solution can possibly lie (which is now used to avoid excessive splitting). Kahan (1968) discovered that interval techniques can also be used to prove the existence of solutions of nonlinear systems in a box (and hence to verify feasibility). Krawczyk (1969) simplified Moore's methods for systems of equations; the Krawczyk operator based on his paper is used in several state-of-the-art global solvers. (See also the historical remarks in Hansen (2001) and Madsen and Skelboe (2002).) Piyavskii (1972) introduced complete global optimization methods based on Lipschitz constants, which are similar in flavour to interval methods.

(3) Tsuda and Kiono (1964) introduced the Monte Carlo method for finding solutions of systems of equations; and the thesis by Mockus (1966) applied it to global optimization. Becker and Lago (1970) were the first to use clustering methods, and Törn (1972, 1974) suggested combining these with local optimization, a combination which currently defines the most efficient class of stochastic global optimization algorithms (Janka 2000). Holland (1973) introduced genetic algorithms, still a popular (although usually slow) stochastic heuristics for global optimization.

As a recognizable mathematical discipline with diverse solution methods for precisely formulated problems involving continuous variables, the field essentially dates back to 1975 when the first book containing exclusively global optimization papers appeared: the volume *Towards Global Optimization*, edited by Dixon and Szegő (1975). In the almost 30 years since the publication of this landmark volume, tremendous progress has been made, and many signs indicate that the field is now ripe for manifold applications in science and engineering.

1.2. Scope

Global optimization is the task of finding the absolutely best set of admissible conditions to achieve an objective under given constraints, assuming that both are formulated in mathematical terms. It is much more difficult than convex programming or finding local minimizers of nonlinear programs, since the gap between the necessary KKT (Karush–Kuhn–Tucker) conditions for optimality and known sufficient conditions for global optimality is tremendous.

Many famous hard optimization problems, such as the travelling salesman problem or the protein folding problem, are global optimization problems. The truth of the famous unresolved conjecture $P \neq NP$ (Garey and Johnson 1979) would imply (Murty and Kabadi 1987, Pardalos and Schnitger 1988) that there are no general algorithms that solve a given global optimization problem in time that is polynomial in the problem description length. However, some large-scale global optimization problems have been solved by current methods, and a number of software packages are available that reliably solve most global optimization problems in small (and sometimes larger) dimensions. The author maintains a web site on Global (and Local) Optimization (1995) that contains many links to online information about the subject.

The different algorithms can be classified according to the degree of rigour with which they approach the goal.

- An *incomplete* method uses clever intuitive heuristics for searching but has no safeguards if the search gets stuck in a local minimum.
- An *asymptotically complete* method reaches a global minimum with certainty, or at least with probability one, if allowed to run indefinitely long, but has no means of knowing when a global minimizer has been found.

- A *complete* method reaches a global minimum with certainty, assuming exact computations and indefinitely long run-time, and knows after a finite time that an approximate global minimizer has been found (to within prescribed tolerances).
- A *rigorous* method reaches a global minimum with certainty and within given tolerances even in the presence of rounding errors, except in near-degenerate cases, where the tolerances may be exceeded.

(Often, the label *deterministic* is used to characterize the last two categories of algorithms; however, this label is slightly confusing since many incomplete and asymptotically complete methods are deterministic, too.)

1.3. Complete search

Complete methods (and *a fortiori* rigorous ones) are (in exact arithmetic) guaranteed to find the global minimizer (within some tolerances) with a predictable amount of work. Here predictable only means relative to known problem characteristics such as Lipschitz constants or other global information (needed for the convergence proof, but usually not for the algorithm itself). The bound on the amount of work is usually very pessimistic – exponential in the problem characteristics. It is only a weak guarantee that does not ensure that the algorithm is efficient in any sense, but it guarantees the absence of systematic deficiencies that prevent finding (ultimately) a global minimizer.

The simplest complete method for bound-constrained problems is *grid search*, where all points on finer and finer grids are tested, and the best point on each grid is used as a starting point for a local optimization. Since the number of points on a grid grows exponentially with the dimension, grid search is efficient only in one and two dimensions. More efficient complete methods generally combine branching techniques with one or several techniques from local optimization, convex analysis, interval analysis and constraint programming.

Generally, complete methods (including approximation methods that reduce the problem to one treated by complete methods) are more reliable than incomplete methods since, to the extent they work (which depends on the difficulty of the problem), they have built-in guarantees.

Complete methods with finite termination require more or less detailed access to global information about the problem. In most complete codes, this is obtained using interval arithmetic (which provides global control of nonlinearities) in an automatic differentiation-like manner (*cf.* Section 16), traversing a computational graph either explicitly, or implicitly by operator overloading. If only black box function (and sometimes gradient) evaluation routines are available, complete methods will find the global minimizer with certainty after a finite time, but will know when this is the case only after an exponentially expensive dense search (*cf.* Theorem 9.1 below). Thus, for complete black box algorithms, stopping must be based on heuristic recipes.

Good heuristics and probabilistic choices (similar to but usually simpler than those for incomplete methods) also play a role in complete methods, mainly to provide good feasible points cheaply that benefit the complete search.

1.4. About the contents

In this survey, the reader will be introduced to theory and techniques that form the backbone of the packages implementing complete or even rigorous algorithms. The core of the material is presented in sufficient detail that the survey may serve as a text for teaching constrained global optimization.

We deliberately exclude methods specific to special problem classes (such as distance geometry or protein folding (Neumaier 1997)), and methods specific to combinatorial optimization (Nemhauser and Wolsey 1988, 1989, Wolsey 1998). Moreover, the discussion of incomplete methods is limited to a short overview, and to techniques that remain useful for complete methods.

No attempt has been made to be objective in selection and evaluation of the material; even for the topics I discuss, there is often much more in the references quoted. Instead I have tried to give personal value judgements whenever I found it appropriate. At the present state of the art, where so many methods compete and reliable comparative information is only just beginning to become available, this seems justified. Thus I discuss the methods that I find most interesting, most useful, and most promising. I hope that my selection bias will be justified by the future. Also, while I try to give accurate references, I do not always refer to the first paper discussing a concept or method but rather quote convenient books or articles summarizing the relevant information, where available.

As one can see from the list of current codes for complete global optimization given in Section 6, none of these codes makes use of all available state-of-the-art techniques. Indeed, in the past, many research groups on global optimization worked with little knowledge of or consideration for related areas. It is hoped that this survey will help to change this lack of communication across the borders of the various traditions in global optimization.

Reviews from other perspectives, with less emphasis on the complete search aspect, are given in Gray *et al.* (1997), Pintér (1996*b*), Törn (2000), and Törn, Ali and Viitanen (1999). For recent books and other basic references, see Section 3.

2. Why global optimization?

Superficially, global optimization is just a stronger version of local optimization, whose great usefulness in practice is undisputed. Instead of searching for a locally unimprovable feasible point we want the globally best point in the feasible region. In many practical applications, finding the globally best point is desirable but not essential, since any sufficiently good feasible point is useful and usually an improvement over what is available without optimization. For such problems, there is little harm in doing an incomplete search; and indeed, this is all that can be achieved for many large-scale problems or for problems where function values (and perhaps derivatives) are available only through a black box routine that does not provide global information.

However, there are many problem classes where it is indispensable to do a complete search, in particular the following.

- **Hard feasibility problems** (*e.g.*, robot arm design, *cf.* Lee and Mavroidis (2002) and Lee, Mavroidis and Merlet (2002)), where local methods do not return useful information since they generally get stuck in local minimizers of the merit function, not providing feasible points (though continuation methods are applicable for polynomial systems in low dimensions).
- **Computer-assisted proofs** (*e.g.*, the proof of the Kepler conjecture by Hales (1998)), where inequalities must be established with mathematical guarantees.
- **Safety verification problems**, where treating nonglobal extrema as worst cases may severely underestimate the true risk (emphasized in the context of robust control by Balakrishnan and Boyd (1992)).
- Many problems in **chemistry** (*cf.* below), where often only the global minimizer (of the free energy) corresponds to the situation matching reality.
- **Semi-infinite programming**, where the optimal configurations usually involve global minimizers of auxiliary problems.

These problems, as well as the fact that algorithms doing a complete search are significantly more reliable and give rise to more interesting mathematics, justify our focus on complete solution techniques.

To show the relevance of global optimization for both pure and applied mathematics, we sketch here a number of typical applications. Of course, this is only the tip of an iceberg ....

(i) Graph structure. Many problems in graph theory are global optimization problems. For example, the *maximum clique problem* asks for the maximal number of mutually adjacent vertices in a given graph. By a well-known theorem of Motzkin and Strauss (1965), an equivalent formulation is the indefinite quadratic program

$$\begin{array}{ll} \max & x^TAx \\ \text{s.t.} & e^Tx = 1, \quad x \geq 0, \end{array}$$

where A is the adjacency matrix of the graph and e is the vector of all ones. Since the maximum clique problem is NP-hard, the same holds for all classes of global optimization problems that contain indefinite quadratic programming.

(ii) Packing problems. The problem is to place a number of k-dimensional ($k \leq 4$) objects of known shape within a number of larger regions of k-space of known shape in such a way that there is no overlap and a measure of waste is minimized. The simplest packing problem is the *knapsack problem* where a maximal number of objects of given weights is to be placed into a container with given maximum weight capacity. Many packing problems arise in industry; but there are also a number of famous packing problems in geometry, of which the 300 year-old *Kepler problem* of finding the densest packing of equal spheres in Euclidean 3-space was only solved recently by Hales (1998), reducing the problem to several thousand linear programs and some interval calculations to ensure rigorous handling of rounding errors. (The proof is still disputed because of the difficulty to check it for correctness; *cf.* Lagarias (2002). A proof based on rigorous global optimization algorithms would probably be more transparent.)

(iii) Scheduling problems. The problem is to match tasks (or people) and slots (time intervals, machines, rooms, airplanes, *etc.*) such that every task is handled in exactly one slot and additional constraints are satisfied. If there are several feasible matchings, one that minimizes some cost or dissatisfaction measure is wanted. Simple scheduling problems such as the *linear assignment problem* can be formulated as linear programs and are solved very efficiently, but already the related *quadratic assignment problem* is one of the hardest global optimization problems, where already most instances with about 30 variables are at the present limit of tractability: *cf.* Anstreicher (2003).

(iv) Nonlinear least squares problems. In many applications, we need to fit data to functional expressions. This leads to optimization problems with an objective function of a form such as

$$f(\theta) = \sum_l \|y_l - F(x_l, \theta)\|^2,$$

where x_l, y_l are given data vectors and θ is a parameter vector. Under certain assumptions, the most likely value of θ is the global minimizer; it generally must have a small objective function value at noise level if the model is to be deemed adequate. If the F_l are nonlinear in θ, a nonconvex optimization problem results that frequently has spurious local minima far above the noise level. A particularly obnoxious case is obtained for data fitting problems in *training neural networks*.

(v) Protein folding. The protein folding problem (Neumaier 1997) consists in finding the equilibrium configuration of the N atoms in a protein molecule with given amino acid sequence, assuming the forces between the atoms are known. These forces are given by the gradient of the $3N$-dimensional potential energy function $V(x_1, \ldots, x_N)$, where x_i denotes the coordinate vector of the ith atom, and the equilibrium configuration is given by the global minimizer of V. Because short-range repulsive forces act like packing constraints, there are numerous local minima.

(vi) Chemical equilibrium problems. (Floudas 1997, McDonald and Floudas 1995.) The task here is to find the number and composition of the phases of a mixture of chemical substances allowed to relax to equilibrium. Local optimization of the associated Gibbs free energy is notorious for giving wrong (nonglobal) solutions. The need to solve such problems was one of the main driving forces for the development of constrained global optimization packages in the chemical engineering community, which is still among the leaders in the field.

(vii) Robotics. For applications in robotics, see Neumaier (2003*b*).

(viii) Other applications. Many more applications can be found in the books by Pintér (1996*a*), Floudas and Pardalos (1990), Floudas *et al.* (1999) and Jaulin *et al.* (2001).

3. Basic ideas

In the following, we discuss complete methods for finding the global minimizer(s) of an objective function subject to constraints. Such problems are typically much more difficult than local optimization problems, since it is often hard to decide whether a local minimizer found is global, and since nonlocal space-covering techniques are needed to avoid being trapped in a region with only nonglobal local minimizers.

Basic to almost all complete global optimization algorithms is the *branching principle* (Section 9). This technique consists in splitting (branching) the original problem recursively into subproblems which are sooner or later easy to solve. In pure branching methods, the more prospective branches are split more frequently, while in *branch and bound methods* we compute bounds on the objective function for each subproblem, in the hope of being able to eliminate many subproblems at an early stage.

The very useful technique of *constraint propagation*, discussed in Section 14, allows us to reduce the feasible region in many cases by exploiting properties of *separable constraints* of the form

$$\sum_{k\in K} q_k(x_k) \in \mathbf{b}$$

with simple, often linear or quadratic functions q_k of a single variable only. This technique may save a lot of branching steps and thus speeds up the branch and bound procedure. This is a reason why special care should be taken in presenting (or transforming) the problem in a form which has as much separability as possible, and we introduce the notion of a *semiseparable program* adapted to this feature. Section 18 addresses ways to transform general problems into semiseparable form by introducing appropriate extra variables. Semiseparable programs are also amenable to approximation by a *mixed integer linear program* (MILP), the only class of global optimization problems that has long been successfully solvable, even for large problems. We shall not discuss techniques for solving MILPs (see, *e.g.*, Bixby *et al.* (2000), Nemhauser and Wolsey (1988, 1989), Wolsey (1998)), but we show how to approximate (and indeed rigorously relax) general global optimization problems by MILPs in Sections 17 and 18.

In order to be able to eliminate subproblems quickly, it is important that one can easily locate good feasible points. This is usually done by local optimization (often in a somewhat rudimentary form); see Section 13. However, especially for problems with many local extrema, it is important to use heuristics which (hopefully) prevent a local method being trapped in a high-lying local minimum. A suitable *tunnelling technique* is discussed in Section 13.

Another basic principle, discussed in Section 16, is that of *outer approximation* of the feasible domain and *underestimation* of the objective function, in order to obtain *relaxed problems* which are convex and hence solvable by local methods. Indeed, this is the traditional way to obtain the bounds on the subproblem. In particular, we consider the use of *cutting planes* and more general cutting surfaces. *Nonconvex relaxations* are also of interest if they can be solved efficiently.

A useful tool for the automatic construction of tight bound constraints, outer approximations and underestimating functions in nonlinear problems is *interval arithmetic*. Though little known in the optimization community, interval arithmetic is an elegant way of calculating with bound constraints, intervals, and simple higher-dimensional geometric shapes like boxes and parallelepipeds. Its most prominent feature is that it allows strict estimates of the approximation error in linear and quadratic approximations of nonlinear functions over a box, thereby providing non-local information even in large boxes. In Section 11, we shall give a very short introduction to this subject (just sufficient for writing programs); a more leisurely introduction embedded into a standard numerical analysis course can be found in Neumaier (2001*b*), and a much more extensive treatment in Neumaier (1990). Interval arithmetic can also be used to certify rigorously the validity of calculations with finite precision arithmetic, and some such applications to optimization are briefly treated in Section 20. The state of the art in 1996 of certified global optimization with interval methods may be found in Kearfott (1996*b*).

3.1. Basic references

A basic reference on most aspects of global optimization is the *Handbook of Global Optimization* by Horst and Pardalos (1995). It contains chapters written by the experts in the respective subfields, on global optimality conditions, complexity issues, concave minimization, dc methods, indefinite quadratic programming, complementarity problems, minimax problems, multiplicative programming, Lipschitz optimization, fractional programming, network problems, continuation methods, interval methods, and stochastic methods (including simulated annealing).

Some more recent books present the state of the art in deterministic global optimization from different perspectives: The interval point of view is in Kearfott's book *Rigorous Global Search* (Kearfott 1996*b*). The constraint propagation perspective is in the book *Numerica* by Van Hentenryck, Michel and Deville (1997*b*); see also the tutorial by Lustig and Puget (2001). The convex analysis perspective is in the books *Deterministic Global Optimization* by Floudas (1999) and *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming* by Tawarmalani and Sahinidis (2002*b*). An attempt to give a synthetic view of the field (but mostly restricted to discrete optimization) is in the book *Logic-Based Methods for Optimization* by Hooker (2000); see also his survey (Hooker 2002).

A comprehensive background on local optimization (needed as part of most good global optimization algorithms) can be found in the book *Numerical Optimization* by Nocedal and Wright (1999). For interior point methods, this should be complemented by Wright (1997).

Other books on global optimization methods include those of Forgó (1988), Hansen (1992*a*), Horst, Pardalos and Thoai (1995), Horst and Tuy (1990), Mockus (1989), Pardalos and Rosen (1987), Pintér (1996*a*), Ratschek and Rokne (1984, 1988), Törn and Žilinskas (1989), Van Laarhoven and Aarts (1987), Zhigljavsky (1991), and proceedings of conferences on global optimization include Bliek, Jermann and Neumaier (2003), Bomze, Csendes, Horst and Pardalos (1996), Dixon and Szegő (1975), Floudas and Pardalos (1992, 1996, 2000) and Grossmann (1996). The *Journal of Global Optimization* is devoted exclusively to papers on global optimization and its applications.

4. Problem formulation

In the present context, a global optimization problem is specified in the form

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in \mathbf{x}, \quad F(x) \in \mathbf{F}, \quad x_I \text{ integral.} \end{array} \tag{4.1}$$

Here

$$\mathbf{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R}^n \mid \underline{x} \leq x \leq \overline{x}\}$$

is a bounded or unbounded *box* in $\mathbb{R}^n$ (with $\underline{x} \in (\mathbb{R} \cup \{-\infty\})^n, \overline{x} \in (\mathbb{R} \cup \{\infty\})^n, \underline{x} \leq \overline{x}$), and x_I denotes the subvector $(x_{i_1}, \dots, x_{i_l})^T$ of x when $I = (i_1, \dots, i_l)$ is a list of indices. Inequalities between vectors are interpreted componentwise. Here $f : \mathbf{x} \to \mathbb{R}$ is a continuous objective function, $F : \mathbf{x} \to \mathbb{R}^m$ is a vector of m continuous constraint functions $F_1(x), \dots, F_m(x)$, and $\mathbf{F}$ is a box in $\mathbb{R}^m$ defining the constraints on $F(x)$. The *feasible domain* is defined by

$$C = \{x \in \mathbf{x} \mid x_I \text{ integral}, \ F(x) \in \mathbf{F}\}. \tag{4.2}$$

Points in C are called *feasible*, and a *solution* of (4.1) is a feasible point $\hat{x} \in C$ such that

$$f(\hat{x}) = \min_{x \in C} f(x). \tag{4.3}$$

A *local minimizer* only satisfies $f(\hat{x}) \leq f(x)$ for all $x \in C$ in some neighbourhood of $\hat{x}$, and the solutions are precisely the *global minimizers*, *i.e.*, the local minimizers with smallest objective function value. A *local* (*global*) *solver* is an algorithm or programming package designed for finding a local (global) minimizer. (We avoid the ambiguous term *optimizer* which may denote either a minimizer or a solver.)

The difficulties in global optimization stem mainly from the fact that there are generally many local minimizers but only one of them is the global minimizer (or just a few), and that the feasible region may be disconnected. (Consider, *e.g.*, the set of positions in the Alps above a certain altitude.) Already a linear objective function has one minimizer in each connected component of the feasible domain, and local descent methods usually fail if they start in the wrong component.

Even the *constraint satisfaction problem*, *i.e.*, the problem of deciding whether the feasible set is nonempty (and finding a feasible point) is frequently highly nontrivial, and may be essentially as difficult as the optimization problem itself (*cf.* Section 13). The usual device of minimizing a suitable measure of infeasibility does not work when the constraints are sufficiently nonlinear, since this measure has itself local minima in which descent methods often get stuck.

Usually, it is possible to reformulate a global optimization problem such that f and F are *smooth*, *i.e.*, twice continuously differentiable. Note that (4.1) is sufficiently flexible to take care of:

- free variables x_i: take $\underline{x}_i = -\infty$, $\overline{x}_i = \infty$;
- nonnegative variables x_i: take $\underline{x}_i = 0$, $\overline{x}_i = \infty$;
- binary variables x_i: take $\underline{x}_i = 0$, $\overline{x}_i = 1$, $i \in I$;
- equality constraints $F_i(x) = 0$: take $\underline{F}_i = \overline{F}_i = 0$;
- inequality constraints $F_i(x) \le 0$: take $\underline{F}_i = -\infty$, $\overline{F}_i = 0$.

If I is not empty then, if f and F are linear then (4.1) is called a *mixed integer linear program* (MILP); and if f and F are convex, and $\underline{F}_i = -\infty$ for all nonlinear F_i, (4.1) is called a *mixed integer nonlinear program* (MINLP). Strictly speaking, this term should apply for all problems (4.1); however, the current techniques for MINLP use the convexity in an essential way, so that it is appropriate to reserve the term for the convex case. Nonconvex mixed integer global optimization problems have received little attention, but see, *e.g.*, Floudas (1995), Grossmann (2002), Sahinidis (1996), and Tawarmalani and Sahinidis (2002*b*).

The only class of global optimization problems that can be reliably solved for many large problem instances (say, $\approx 10^5$ variables and $|I| \approx 10^3$) is the class of MILPs. This is due to the fact that after fixing the integer variables one is left with a linear program, which can be solved efficiently. Instead of trying all integer combinations separately, branching techniques (branch and bound, branch and cut), combined with preprocessing the resulting linear programs, drastically cut down the number of cases to be looked at. MINLP shares with MILP the feature that fixing all integer variables leads to a tractable problem, in this case a convex nonlinear program, for which every local minimizer is a solution; however, the dimensions are here more limited since nonlinear programming codes are significantly slower than their linear counterparts.

Most constrained global optimization is nowadays best viewed as an adaptation of mixed integer programming technology to nonlinear problems. Historically, however, many of the techniques were devised independently by groups working in integer programming, combinatorial optimization, unconstrained optimization, interval analysis, and constraint logic programming.

Other important classes of global optimization problems are as follows:

- simply constrained: if $\dim F = 0$,
- continuous: if $I = \emptyset$,
- bound-constrained: if simply constrained and continuous,
- separable: if $f(x) = \sum_{k=1}^{n} f_k(x_k)$ and $F(x) = \sum_{k=1}^{n} F_k(x_k)$,

- factorable: if f and F are obtained by applying a finite sequence of arithmetic operations and unary elementary functions to constants and the x_k,
- reverse convex: if f, F are concave, and $\underline{F}_i = -\infty$ for all nonlinear F_i,
- DC: if f, F are differences of convex functions.

5. First-order optimality conditions

Nonlinear programming provides the following necessary (*Karush–John*) *optimality conditions* for local minimizers. We assume that f, F are continuously differentiable, and denote by $f'(x)$ and $F'(x)$ the derivatives at x. Note that $f'(x)$ is a row vector and $F'(x)$ a matrix, the *Jacobian*.

Theorem 5.1. (Karush (1939), John (1948)) For every local minimizer $\hat{x}$ of (4.1) (which defines the notation) there are a number $\kappa \geq 0$ and a vector y, not both zero, such that the row vector

$$g^T = \kappa f'(\hat{x}) + y^T F'(\hat{x}) \tag{5.1}$$

satisfies

$$g_i \begin{cases} \geq 0 & \text{if } \underline{x}_i = \hat{x}_i < \overline{x}_i, \quad i \notin I, \\ \leq 0 & \text{if } \underline{x}_i < \hat{x}_i = \overline{x}_i, \quad i \notin I, \\ = 0 & \text{if } \underline{x}_i < \hat{x}_i < \overline{x}_i, \quad i \notin I, \end{cases} \tag{5.2}$$

$$y_i \begin{cases} \geq 0 & \text{if } \underline{F}_i < F_i(\hat{x}) = \overline{F}_i, \\ \leq 0 & \text{if } \underline{F}_i = F_i(\hat{x}) < \overline{F}_i, \\ = 0 & \text{if } \underline{F}_i < F_i(\hat{x}) < \overline{F}_i. \end{cases} \tag{5.3}$$

Note that there is no restriction on g_i if $i \in I$ or $\underline{x}_i = \overline{x}_i$, and no restriction on y_i if $\underline{F}_i = F_i(\hat{x}) = \overline{F}_i$. Buried implicitly in results of Mangasarian (1969), and spelled out explicitly in Neumaier and Schichl (2003), is the observation that one may in fact assume that either κ or the subvector y_J of y is nonzero, where J is the set of indices i such that either $F_i(x)$ is nonconvex and $F_i(\hat{x}) = \underline{F}_i$ or $F_i(x)$ is nonconcave and $F_i(\hat{x}) = \overline{F}_i$. (In particular, J does not contain any index i such that $F_i(x)$ is linear.) In view of the homogeneity of the statement of the theorem, one can therefore scale the multipliers such that

$$\kappa + y_J^T D y_J = 1, \tag{5.4}$$

where D is an arbitrary diagonal matrix with positive entries. This condition is relevant for the application of exclusion box techniques (*cf.* Section 15).

We say that $\hat{x}$ satisfies a *constraint qualification* (CQ) if (5.1)–(5.3) hold for some $\kappa > 0$. In this case, one can scale g, κ, y to enforce $\kappa = 1$ and obtains the more frequently used *Kuhn–Tucker conditions* (Kuhn and Tucker 1951, Kuhn 1991). A sufficient condition for the constraint qualification is that the rows of $F'(\hat{x})$ are linearly independent; various weaker conditions guaranteeing CQ are known.

If $\kappa = 1$ then y is called an optimal *Lagrange multiplier* corresponding to $\hat{x}$ (it need not be unique). In this case, g is the gradient of the associated *Lagrangian* (Lagrange 1797)

$$L(x, y) = f(x) + y^T F(x)$$

at $x = \hat{x}$.

Note that minimizers with huge Lagrange multipliers are best considered as points nearly violating the constraint qualification, so that (5.1) holds with $y = O(1)$ and tiny κ.

If there are only nonnegativity constraints and equality constraints,

$$C = \{x \geq 0 \mid F(x) = b\},$$

corresponding to $\underline{x}_i = 0$, $\overline{x}_i = \infty$, $\underline{F}_i = \overline{F}_i = b_i$ then the conditions (5.3) are vacuous, and (5.2) reduces to the traditional *complementarity condition*

$$\min(g_i, x_i) = 0 \quad \text{for all } \ i.$$

Example 5.2. We consider the problem

$$\begin{array}{ll} \min & f(x) = -x_1 - 2x_2 \\ \text{s.t.} & F(x) = (x_1 - 1)^2 + (x_2 - 1)^2 = 1, \quad x_1, x_2 \in [-1, 1]. \end{array} \tag{5.5}$$

The feasible region is a quarter circle, and the contour lines of the objective function are linear, decreasing in the direction indicated in Figure 5.1. This implies that there is a unique maximizer at P, a local minimizer at Q and a global minimizer at R. The solution is therefore $\hat{x} = (0, 1)$. Since there are only two variables, we analysed the problem graphically, but we could as well have proceeded symbolically as follows.

Assuming for simplicity the validity of the CQ, we find for the gradient of the Lagrangian

$$g = \begin{pmatrix} -1 \\ -2 \end{pmatrix} + y \begin{pmatrix} 2x_1 - 2 \\ 2x_2 - 2 \end{pmatrix}.$$

The Kuhn–Tucker conditions require that $g_i \geq 0$ if $\hat{x}_i = -1$, $g_i \leq 0$ if $\hat{x}_i = 1$, and $g_i = 0$ otherwise. This leaves three cases for each component, and a total of $3 \cdot 3 = 9$ cases. If we assume $|\hat{x}_1|, |\hat{x}_2| < 1$ we must have $g = 0$, hence $\hat{x}_1 = 1 + 1/2y$, $\hat{x}_2 = 1 + 1/y$. Since $\hat{x}$ must be feasible, $y < 0$, and since $F(\hat{x}) = 1$, $y = -\frac{1}{2}\sqrt{5}$, $\hat{x} = (1 - 1/\sqrt{5}, 1 - 2/\sqrt{5})^T$, which is the local maximizer P. If we assume $\hat{x}_1 = -1$ or $\hat{x}_2 = -1$, or $\hat{x}_1 = \hat{x}_2 = 1$,

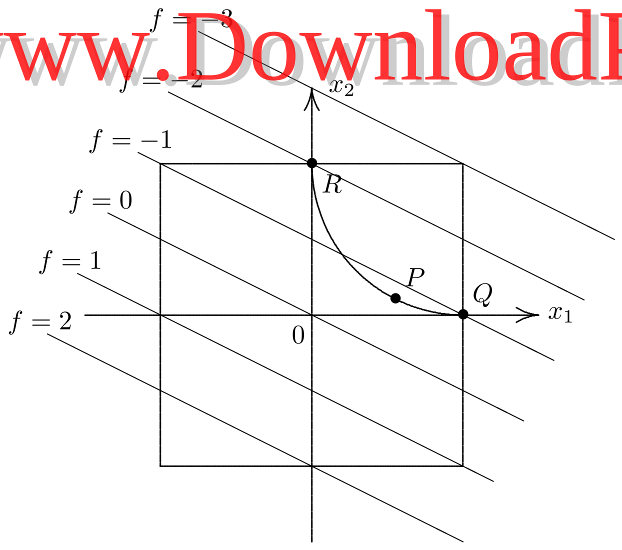

Figure 5.1. Feasible region and contour lines for Example 5.2.

we find a contradiction with $F(\hat{x}) = 1$. (These are 6 cases!) If we assume $|\hat{x}_2| < 1 = \hat{x}_1$ we find Q, and for $|\hat{x}_1| < 1 = \hat{x}_2$ we find R. Thus we have three feasible points satisfying the Kuhn–Tucker conditions, and a comparison of their function values shows that R is the global minimizer.

In general, we have three cases for each two-sided inequality and two for each one-sided inequality; since the number of independent choices must be multiplied, the total number of cases grows exponentially with the number of inequalities in the problem formulation. Hence this symbolic approach is limited to problems with few inequality constraints. Even then it only works if the resulting nonlinear equations are symbolically solvable and have few solutions only. Thus, in general, we need to resort to numerical methods.

We draw several conclusions from the example. First, there is a combinatorial aspect to the continuous global optimization problem, so that it resembles a mixed integer problem. Second, several cases can often be excluded by a single argument, which is the basis for the branch and bound approach to global optimization. Third, the Karush–John or Kuhn–Tucker conditions do not distinguish between maxima and minima (and other 'stationary' points); all these would have to be enumerated in a naive approach. Since there may be an exponential number of Kuhn–Tucker points, additional techniques are needed to reduce the search space. Lagrange multiplier techniques involving second-order conditions will address this last point; *cf.* Theorem 15.1.

6. Software for complete global optimization

Here we list some of the better complete global optimization codes available on the Web, with short comments on scope and method. Several of the codes (LGO, BARON, SBB, DICOPT) can be called from the GAMS modelling system (GAMS World 2003), allowing for very convenient input. Input from the AMPL modelling system (Fourer, Gay and Kernighan 1993) will be possible through an AMPL to GAMS translator available within the COCONUT environment; *cf.* Section 22. Only two of the codes (GlobSol and Numerica) are rigorous solvers.

6.1. Some branching codes using function values only

The codes listed use black box function evaluation routines, and have heuristic stopping rules, so that the actual implementation yields an incomplete search only.

(i) DIRECT: Divide Rectangles (in Fortran)
Gablonsky and Kelley (2001)
`ftp://ftp.math.ncsu.edu/FTP/kelley/iffco/DIRECTv204.tar.gz`

Direct.m, a MATLAB implementation of DIRECT
`www4.ncsu.edu/~definkel/research/`

Implementations of a simple and efficient global optimization method by Jones, Perttunen and Stuckman (1993) for bound-constrained problems. DIRECT is based on branching and a Pareto principle for box selection.

(ii) MCS: Multilevel Coordinate Search
Huyer and Neumaier (1999)
`www.mat.univie.ac.at/~neum/software/mcs/`

A MATLAB program for bound-constrained global optimization using function values only. MCS is based on branching and sequential quadratic programming.

(iii) LGO: Lipschitz Global Optimization (commercial)
Pintér (1996*a*, 1999)
`is.dal.ca/~jdpinter/l_s_d.htm`

An integrated development environment for global optimization problems with Lipschitz-continuous objective and constraints. LGO is based on branching and stochastic estimation of Lipschitz constants; constraints other than simple bounds are handled by L_1 penalty terms, but interior convex constraints by projection penalties. (LGO also has options for incomplete search methods; in many cases, these give better results than the branching option.)

The codes listed use global information (generally from required symbolic problem input). They have finite termination with guarantee that the global minimizer is found within certain tolerances; in difficult cases storage or time limits may be exceeded, however, leading to appropriate error messages. All codes use at least basic constraint propagation, but differ considerably in the other techniques implemented.

Not listed are the many MILP codes available (see the Global Optimization web page mentioned in the Introduction).

(i) BARON: Branch-And-Reduce Optimization Navigator
(commercial)
Sahinidis *et al.* (Ryoo and Sahinidis 1996, Sahinidis 1996, 2000, 2003, Tawarmalani and Sahinidis 2002*b*, 2004)
`archimedes.scs.uiuc.edu/baron/baron.html`

A general-purpose solver for optimization problems with nonlinear constraints and/or integer variables. Fast specialized solvers for many linearly constrained problems. BARON is based on branching and box reduction using convex and polyhedral relaxation and Lagrange multiplier techniques.

(ii) GlobSol: Global Solver (in Fortran 90)
Kearfott (1996*b*, 2003)
`www.mscs.mu.edu/~globsol/`

Branch and bound code for global optimization with general factorable constraints, with rigorously guaranteed results (even round-off is accounted for correctly). GlobSol is based on branching and box reduction using interval analysis to verify that a global minimizer cannot be lost.

(iii) LINGO (commercial)
Gau and Schrage (2004)
`www.lindo.com/cgi/frameset.cgi?leftlingo.html;lingof.html`

Branch and bound code for global optimization with general factorable constraints, including nondifferentiable expressions. LINGO is based on linear relaxations and mixed integer reformulations. C and Excel interfaces are available.

(iv) Frontline Interval Global Solver (commercial)
Nenov and Fylstra (2003)
`www.solver.com/technology5.htm`

This solver is based on interval methods and linear relaxations. Visual Basic and Excel interfaces are available.

(v) ALIAS (in C)
Merlet (ALIAS-C++ 2003, Merlet 2001)

`www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS-C++/ALIAS-C++.html`

Branch and bound environment for solving constraint satisfaction problems (and rudimentary global optimization). A toolkit of interval analysis and constraint programming techniques with a Maple interface for symbolic preprocessing.

(vi) Numerica
Van Hentenryck *et al.* (1997*b*)
Branch and bound code for constrained optimization (with mathematically rigorous results). This code (no longer available) was based on branching and box reduction using interval analysis and deeper constraint propagation techniques. The box reduction and interval analysis algorithms of Numerica are now available in *ILOG Solver* (commercial) at
`www.ilog.com/products/solver/`

(vii) αBB
Adjiman, Floudas and others (Adjiman *et al.* 1996, 1998*a*, 1998*b*, Androulakis *et al.* 1995)
`titan.princeton.edu/soft.html#abb`

Branch and bound code for nonlinear programs. The site currently has the description only, no code. αBB is based on branch and bound by convex underestimation, using interval analysis to write nonlinearities in DC (difference of convex functions) form.

(viii) LaGO
Nowak, Alperin and Vigerske (2003)
`www-iam.mathematik.hu-berlin.de/~eopt/#Software`

Branch and bound code for mixed integer nonconvex nonlinear programming, using block-separable structure and convex underestimation. The site currently has the description only, no code.

(ix) GloptiPoly (in MATLAB)
Henrion and Lasserre (2003*a*, 2003*b*, 2003*c*)
`www.laas.fr/~henrion/software/gloptipoly/`

Global optimization of polynomial nonlinear programs, using semidefinite relaxations. Currently limited to problems with fewer than 20 variables.

(x) SOSTOOLS (in MATLAB)
Prajna, Papachristodoulou and Parrilo (2002)
`control.ee.ethz.ch/~parrilo/sostools/index.html`

MATLAB toolbox for solving sums of squares (SOS) optimization programs. Allows the solution of polynomial global optimization problems.

(xi) cGOP
Visweswaran and Floudas (1996*a*, 1996*b*)
`titan.princeton.edu/soft.html`

Branch and bound code for linearly constrained global optimization problems with an objective containing linear, bilinear, and convex terms, using convex relaxations.

(xii) MINLPBB (commercial)
Fletcher and Leyffer (1994, 1998)
`www-unix.mcs.anl.gov/~leyffer/solvers.html`

Branch and bound code for mixed integer nonlinear programming; finding the global optimum is guaranteed only if all constraints become convex when all integer variables are fixed. Problems with AMPL input can be solved online via NEOS at
`www-neos.mcs.anl.gov/neos/solvers/MINCO:MINLP-AMPL/`

MINLP uses standard mixed integer programming techniques and filter methods for the local subproblems.

(xiii) DICOPT (commercial)
Duran and Grossmann (1986), Kocis and Grossmann (1989)
`www.gams.com/solvers/dicopt/main.htm`

Solver for mixed integer monlinear programming (MINLP) problems. Finding the global optimum is guaranteed only if all constraints become convex when all integer variables are fixed.

(xiv) SBB (commercial)
Bussiek and Drud (2001)
`www.gams.com/solvers/solvers.htm#SBB`

Branch and bound code for mixed integer nonlinear programming; finding the global optimum is guaranteed only if all constraints become convex when all integer variables are fixed. Problems with GAMS input can be solved online via NEOS at
`www-neos.mcs.anl.gov/neos/solvers/MINCO:SBB-GAMS/`

SBB uses standard mixed integer programming techniques and sequential quadratic programming methods for the local subproblems.

MINLPBB, DICOPT and SBB are borderline cases in our list since they do not use truly global techniques for the continuous variables, and will not be discussed further.

7. Incomplete methods for simple constraints

As mentioned in the Introduction, numerical methods for global optimization can be classified into four categories according to the available guarantees. We shall be mainly concerned with complete methods; however, since incomplete and asymptotically complete methods are frequently successful and, for many difficult problems, are the only feasible choice, we give an overview of the main possibilities. In these categories are several deterministic and most stochastic methods. For some of the latter, it is possible to prove convergence with probability arbitrarily close to 1 (if running arbitrarily long), but this does not yet guarantee convergence. (Moreover, the assumptions underlying the convergence proofs are frequently not verifiable for particular examples.)

The simplest incomplete method is *multiple random start*, consisting of picking random starting points and performing local optimizations from these points, in the hope that one of them is in the basin of attraction of the global minimizer. Most stochastic techniques can be regarded as devices for speeding up this basic method, by picking the points more carefully and by doing only rudimentary local optimization, or optimizing only selectively.

Most of the research on incomplete search has been concentrated on global optimization methods for simply constrained problems only. Many different incomplete methods are known for simply constrained problems, and we sort them into four categories: *local descent techniques*, including among others multiple random start, clustering (Boender, Rinnooy Kan, Timmer and Stougie 1982), tunnelling (Levy and Montalvo 1985), and smoothing methods (Guddat, Guerra Vasquez and Jongen 1990, Kostrowicki and Scheraga 1996, Moré and Wu 1997, Stillinger 1985); *response surface techniques*, including Bayesian stochastic techniques Mockus (1989, 1994) and related techniques (Björkman and Holmström 2000, Jones 2001, Jones, Schonlau and Welch 1998); *nonmonotonic search techniques*, including among others tabu search (Glover 1989/90, Glover and Laguna 1997), simulated annealing (Kirkpatrick, Geddat, Jr. and Vecchi 1983, Ingber 1993, Van Laarhoven and Aarts 1987), and deterministic variants such as threshold accepting (Dueck and Scheuer 1990); *ensemble methods*, including genetic algorithms (Holland 1973, Forrest 1993, Michalewicz 1996) and variants such as ant colony minimization (Dorigo, Maniezzo and Colorni 1991).

No attempt is made to be representative or complete on referencing or describing the large literature on incomplete techniques; we only mention the 1975 book by Dixon and Szegő (1975), which marks the start of a tradition of comparing different global optimization methods, an excellent exposition of stochastic global optimization methods for bound-constrained problems on the Web by Törn (2000), and another web survey of (mainly incomplete) methods by Gray *et al.* (1997). For incomplete search in combinatorial optimization (where the underlying ideas are also called *metaheuristics*), see, *e.g.*, Aarts and Lenstra (1997), Yagiura and Ibaraki (2001).

Instead of describing technical details of the various methods (these vary from author to author and even from paper to paper), we give an informal view of the ideas, strengths and weaknesses of one method from every category, each based on analogies to natural processes where more or less global optima are reached. While these techniques are motivated by nature it is important to remember that processes in nature need not be the most efficient ones; at best they can be assumed to be efficient *given the conditions under which they have to operate* (namely an uncertain and changing environment that is potentially hazardous to those operating in it). Indeed, much of our present technology has vastly surpassed natural efficiency by unnatural means, and it would be surprising if it were different in global optimization. Even assuming that nature solves truly *global* optimization problems (a disputable assumption), simple lower estimates for the number of elementary steps – roughly corresponding to function evaluations – available to natural processes to converge are (in chemistry and in biology) in the range of 10^{15} or even more. This many function evaluations are unacceptable for present day computers, and will be so in the near future.

With a limited number of function evaluations, the quality of incomplete methods depends a lot on details of the implementation; comparisons on relative efficiency are virtually missing. Indeed, the techniques must generally be tuned to special classes of applications in order to be fast and competitive, which makes general-purpose comparisons difficult and inconclusive.

Smoothing (= **homotopy** = **continuation**) **methods** are based on the intuition that, in nature, macroscopic features are usually an average effect of microscopic details; averaging smooths out the details in such a way as to reveal the global picture. A huge valley seen from far away has a well-defined and simple shape; only by looking more closely, the many local minima are visible, more and more at smaller and smaller scales. The hope is that by smoothing a rugged objective function surface, most or all local minima disappear, and the remaining major features of the surface only show a single minimizer. By adding more and more details, the approximations made by the smoothing are undone, and finally one ends up at the global minimizer of the original surface.

In mathematical terms, one has to define a homotopy by introducing an additional parameter t into the problem in such a way that $t = 0$ gives the original problem, while $t = 1$ gives either a related convex problem or a related problem with a unique and known global minimizer. (There are various ways of doing this; homotopies whose parameter has a natural interpretation in the context of the original problem usually perform better.) Then a sequence of local problems is solved for $t = t_1, t_2 \ldots, t_N$, where the t_i

form a decreasing sequence starting at 1 and ending at 0. Each time, the solution of the previous problem is taken as the starting point for the current problem. The quality of the final local minimizer depends on the homotopy, and is frequently the global or at least a good local minimizer.

There is no theoretical work on conditions that would ensure convergence to the global minimum. In particular, it is quite possible for such a method to miss the global minimum. However, for properly chosen homotopies, smoothing methods at least give good local minima with a small number of function evaluations. (For more theory on homotopy methods, see, *e.g.*, Guddat *et al.* (1990, Section 6.3).)

Response surface techniques are designed specifically for the global optimization of functions that are very expensive to evaluate. They construct in each iteration an interpolation or approximation *surrogate function* of known analytic form. The surrogate function is then subjected to global optimization, *e.g.*, by some form of multiple random start (started at a selection of the current points). The resulting optimizers (or some points where the feasible region has only been sparsely explored) are taken as new evaluation points. Since this is sequential global optimization, each step is much more expensive than the others, but the reduction of the number of function values needed gives (for sufficiently expensive function evaluations) a net gain in speed.

In principle, these methods may have convergence guarantees if the point selection strategy is well chosen; but this is irrelevant in view of the fact that for expensive functions, only few (perhaps up to 1000) function evaluations are admissible.

Simulated annealing takes its intuition from the fact that the heating (annealing) and slow cooling of a metal brings it into a more uniformly crystalline state that is believed to be the state where the free energy of bulk matter takes its global minimum. (Incidentally, even for the simplest potential energy functions, it is still an unsolved problem whether this is indeed true with mathematical rigour. Apart from that, even very pure crystals still have defects; *i.e.*, the global minimum is not quite achieved in nature.) The role of temperature is to allow the configurations to reach higher energy states with a probability given by Boltzmann's exponential law, so that they can overcome energy barriers that would otherwise force them into local minima. This is quite unlike line search methods and trust region methods on which good local optimization programs are based.

In its original form, the simulated annealing method is provably convergent in a probabilistic sense but exceedingly slow; various *ad hoc* enhancements make it much faster. In particular, except for simple problems, success depends very much on the implementation used.

Genetic algorithms make use of analogies to biological evolution by allowing mutations and crossing over between candidates for good local optima in the hope of deriving even better ones. At each stage, a whole population of configurations is stored. Mutations have a similar effect to random steps in simulated annealing, and the equivalent of lowering of the temperature is a rule for more stringent selection of surviving or mating individuals.

The ability to leave regions of attraction to local minimizers is, however, drastically enhanced by crossing over. This is an advantage if, with high probability, the crossing rules produce offspring of similar or even better fitness (objective function value); if not, it is a severe disadvantage. Therefore the efficiency of a genetic algorithm (compared with simulated annealing-type methods) depends in a crucial way on the proper selection of crossing rules. The effect of interchanging coordinates is beneficial mainly when these coordinates have a nearly independent influence on the fitness, whereas if their influence is highly correlated (such as for functions with deep and narrow valleys not parallel to the coordinate axes), genetic algorithms have much more difficulties. Thus, unlike simulated annealing, successful tuning of genetic algorithms requires a considerable amount of insight into the nature of the problem at hand.

Both simulated annealing methods and genetic algorithms are, in their simpler forms, easy to understand and easy to implement, features that invite potential users of optimization methods to experiment with their own versions. The methods often work, if only slowly, and may be useful tools for applications where function values are not very expensive and the primary interest is to find (near-)solutions *now*, even when the reliability is uncertain and only subglobal optima are reached.

To make simulated annealing methods and genetic algorithms efficient, clever enhancements exploiting expert knowledge about the problem class at hand are essential. Theoretical work on explaining the effectiveness of useful enhancements is completely lacking. Also, I have not seen careful comparisons of the various options available and their comparative evaluation on standard collections of test problems.

In general, incomplete methods tend to fail systematically to find the global optimum on the more difficult problems in higher dimensions, but they frequently give relatively good points with a reasonable amount of effort. Beyond a certain number of function evaluations (that depends on the problem), progress slows down drastically if the global optimum has not yet been located already. This is unlikely to change in the future, although new heuristics and variations of old ones are discovered almost every year.

For general-purpose global optimization, the most promising incomplete methods appear to be clustering methods (see the recent comparison by Janka (2000)), being fairly robust and fast. In particular, the multilevel clustering algorithm by Boender *et al.* (1982), as implemented by Csendes (1988), can be recommended. Among incomplete algorithms adapted to problem structure, I would favour smoothing methods (if a natural homotopy is available) and tabu-search-like strategies (since these have a kind of memory).

8. Bound-constrained approximation

For general constraints, incomplete techniques are much less developed. Only the smoothing techniques extend without difficulties to general constraints. To use the other incomplete techniques, it is customary to rewrite problems with general constraints in an equivalent or approximately equivalent form with either simple constraints only, for which the methods of the previous section apply, or into a mixed integer linear problem (MILP), since highly efficient software is available for solving the latter (Bixby *et al.* 2000). Both transformations are of great practical importance and widely used. Solving the transformed (equivalent or approximate) problem yields an approximate solution for the original problem, and local optimization from this approximate solution gives the global minimizer of the original problem if the approximation was good enough, and usually a good local minimizer otherwise.

In this section we treat the approximation of general constrained problems by bound-constrained problems using penalty and barrier functions. The approximation of nonlinear problems by mixed integer linear programs is treated in Section 18.

8.1. Penalty and barrier formulations

Traditionally (see Fiacco and McCormick (1990)), constraints that cannot be handled explicitly are accounted for in the objective function, using simple l_1 or l_2 penalty terms for constraint violations, or logarithmic barrier terms penalizing the approach to the boundary. In both cases, the reformulation changes the solution, so that this is an instance of an approximation method, and the result should be used as a starting point for a subsequent local optimization of the original problem. There are also so-called exact penalty functions whose optimization gives the exact solution (see, for example, Nocedal and Wright (1999)); however, this only holds if the penalty parameter is large enough, and what is large enough cannot be assessed without having global information.

The use of more general transformations gives rise to more precisely quantifiable approximation results. In particular, if it is known in advance that all constraints apart from the simple constraints are soft constraints only (so that some violation is tolerated), one may pick a transformation that incorporates prescribed tolerances into the reformulated simply constrained problem, using the following variation of a similar, but less flexible result of Dallwig, Neumaier and Schichl (1997), given in Huyer and Neumaier (2003).

Theorem 8.1. (Soft optimality theorem) Given $\Delta, \underline{\sigma}_i, \overline{\sigma}_i > 0, f_0 \in \mathbb{R}$, let

$$q(x) = \frac{f(x) - f_0}{\Delta + |f(x) - f_0|},$$

$$\delta_i(x) = \begin{cases} (F_i(x) - \underline{F}_i)/\underline{\sigma}_i, & \text{if } \ F_i(x) \leq \underline{F}_i, \\ (F_i(x) - \overline{F}_i)/\overline{\sigma}_i, & \text{if } \ F_i(x) \geq \overline{F}_i, \\ 0, & \text{otherwise,} \end{cases}$$

$$r(x) = \frac{2 \sum \delta_i^2(x)}{1 + \sum \delta_i^2(x)}.$$

Then the merit function

$$f_{\mathrm{merit}}(x) = q(x) + r(x)$$

has its range bounded by $]-1, 3[$, and the global minimizer $\hat{x}$ of f_{merit} in $\mathbf{x}$ either satisfies

$$F_i(\hat{x}) \in [\underline{F}_i - \underline{\sigma}_i, \overline{F}_i + \overline{\sigma}_i] \quad \text{for all} \ \ i, \tag{8.1}$$
$$f(\hat{x}) \leq \min\{f(x) \mid F(x) \in \mathbf{F}, x \in \mathbf{x}\}, \tag{8.2}$$

or one of the following two conditions holds:

$$\{x \in \mathbf{x} \mid F(x) \in \mathbf{F}\} = \emptyset, \tag{8.3}$$
$$f_0 < \min\{f(x) \mid F(x) \in \mathbf{F}, x \in \mathbf{x}\}. \tag{8.4}$$

Here (8.1) says that a soft version of the constraints is satisfied. The numbers $\underline{\sigma}_i$ and $\overline{\sigma}_i$ measure the degree to which the lower and upper bounds in the constraint $F_i(x) \in \mathbf{F}_i$ may be softened; suitable values are in many practical applications available from the meaning of the constraints.

Inequality (8.2) says that f_{merit} has a lower global minimum value (attained at a point satisfying the soft constraints) than the global minimum value of the original problem (on the hard version of the constraints). Thus little is lost from a practical point of view.

The degenerate cases (8.3)–(8.4) account for the possibility of an empty feasible set (8.3), and for a choice of f_0 that was too small. If a feasible point is already known we may choose f_0 as the function value of the best feasible point known (at the time of posing the problem), thus eliminating the possibility (8.3). If none is known, f_0 should be chosen as a fairly large value to avoid (8.4); it can be reset (and the optimization restarted) when a feasible point becomes available during the search.

In spite of the absolute value in the definition of $q(x)$, f_{merit} is continuously differentiable if f and F have this property. A suitable value for Δ is the median of the values $|f(x) - f_0|$ for an initial set of trial points x (in the context of global optimization often determined by a space-filling design (McKay, Beckman and Conover 1979, Owen 1992, Owen 1994, Sacks, Welch, Mitchell and Wynn 1989, Tang 1993)).

8.2. Projection penalties

A little-known result by Pintér (1996*a*) may be used to get in certain cases (in particular, for linear and convex quadratic constraints) an exact reformulation as a non-smooth but Lipschitz-continuous simply constrained problem. The idea is to project infeasible points to the feasible domain.

To accommodate linear constraints (or convex quadratic ones), Pintér assumes that x_0 is a known interior point. For arbitrary $\gamma > 0$ we now define the modified objective function

$$\overline{f}(x) := f(\overline{x}) + \gamma\|\overline{x} - x\|^2, \tag{8.5}$$

where

$$\overline{x} = \lambda x_0 + (1 - \lambda)x \tag{8.6}$$

and $\lambda = \lambda_x \geq 0$ is minimal such that $\overline{x}$ satisfies the linear constraints. This is well-defined, since $\lambda = 1$ always works by the choice of x_0. Each constraint contributes a lower bound $\in [0, 1]$ for λ, and the largest of these bounds is the desired value. In particular, a linear constraint $a^Tx \leq \alpha$ contributes a nonzero lower bound

$$\lambda \geq (a^Tx - \alpha)/(a^Tx - a^Tx_0)$$

if both numerator and denominator of the right-hand side are positive. A convex quadratic constraint similarly yields a quadratic inequality that can easily be solved for λ. (Convexity can be weakened to star-shapedness with respect to x_0.)

The modified objective function (8.5) is Lipschitz-continuous, but non-smooth at all points where the ray (8.6) hits a lower-dimensional face of the feasible domain. Note that to evaluate (8.5), function values are needed only at points satisfying the linear (or convex quadratic) constraints.

An interior point can be found by solving a linear program or convex second-order cone program. If no interior point exists since the feasible set is in a lower-dimensional subspace, each feasible point has the form $x = x_0 + Cz$ with $z \in \mathbf{z}$, where x_0 is in the relative interior of the feasible domain, and $\mathbf{z}$ a box with $0 \in \operatorname{int} \mathbf{z}$. Both x_0 and C can be found by techniques from convex analysis for finding a maximal independent set of points in the affine subspace spanned by the feasible set. Reposing the optimization problem in terms of z reduces the dimension and yields a problem in which 0 is an interior point.

9. Pure branching methods

We begin our analysis of complete methods for global optimization by looking at the options for methods that can access no global information about a problem. The information is made available via black box routines that provide *local information* only, *i.e.*, function values and possibly gradients or Hessians at single points. A necessary and sufficient condition for complete methods based on local information only is given by the following important *density theorem* due to Törn and Žilinskas (1989). It formalizes the simple observation that after finitely many local evaluations there are still many 'holes', *i.e.*, balls not containing an already evaluated point, and there are many functions (Neumaier 2003*a*) that have the known function values, gradients and Hessians at the evaluation points but an arbitrarily low function value at the centre of such a ball.

Theorem 9.1. Any method based on local information only that converges for every continuous f to a global minimizer of f in a feasible domain C must produce a sequence of points $x^1, x^2, \dots$ that is dense in C.

Conversely, for any such method,

$$\liminf_{l\to\infty} f(x^l) = \min\{f(x) \mid x \in C\}.$$

A global optimization method based on local information only is called *convergent* if it satisfies the hypothesis of this density theorem. (Actual implementations of a convergent global optimization method usually are not truly convergent since they must have built-in termination criteria that are necessarily heuristic.)

Convergence is a *minimal* requirement and does not make an algorithm good! For example, exhaustive grid search is convergent but far too slow in dimensions > 2. (Compare with local optimization with line searches along the steepest descent direction, which is globally convergent but frequently very slow.) In a sense, the density theorem says that any convergent method must be ultimately exhaustive, though it may delay the detailed exploration of unpromising regions. Since, in practice, only a limited number of points can be explored, the behaviour of a pure branching method is governed by its ability to find a good ordering of the points to be evaluated for which premature termination has no severe effect.

Three good asymptotically complete general-purpose global optimization algorithms based on local information only are currently available: DIRECT (Jones *et al.* 1993), MCS (Huyer and Neumaier 1999) and LGO (Pintér

1996*a*). All work for bound-constrained problems only and need the approximation techniques of Section 8 for more general problems. (Some of these are built into LGO, but must be coded by the user for DIRECT and MCS.) All three algorithms enforce convergence by employing a branching scheme. They differ in how and when to split, and what is done within each box.

A *branching scheme* generates a sequence of rooted trees of boxes whose leaves cover the feasible set. At least one point in each box is evaluated. The first tree just has the original box as root and only leaf. Each other tree is obtained from the previous one by splitting one or several leaves. If the diameters of all boxes at all leaves converge to zero, convergence of the algorithm is straightforward.

The convergence to zero of the diameters is ensured by appropriate *splitting rules* that define when and how a box is split. For example, convergence is guaranteed when in each of a sequence of rounds,

- we always split the oldest box along the oldest side, and possibly split finitely many other boxes, or
- we always split the longest box along the longest side, and possibly split finitely many other boxes (where length = sum of length of sides),

provided that each split of the oldest (or longest) box produces boxes whose volume is at most a fixed fraction < 1 of the unsplit box. The possibility of 'and finitely many other boxes' (but not many if the code is to be robust!) can be used with considerable flexibility without destroying the convergence property.

Apart from the convergence requirement, the key to efficiency is a proper balance of global and local search. This is achieved in DIRECT by splitting in each round all boxes for which the pair (v, f) (where v is the volume and f the midpoint function value) is not dominated by another such pair. Here (v, f) is dominated by (v', f') if both $v' < v$ and $f' > f$. In particular, the box of largest volume and the box with the best function value are never dominated and hence always split. MCS instead uses domination of pairs (l, f), where l is a suitably assigned level, and in addition employs local optimization steps (using line searches and sequential bound-constrained quadratic programs) from appropriate candidate points. LGO uses lower bounds

$$L \geq \max_{k,l} \|f(x_k) - f(x_l)\| / \|x_k - x_l\|$$

on Lipschitz constants L obtained from the previous function evaluations to decide on the promising boxes to split first. (Upper bounds on L, and hence bounds on function values, cannot be obtained from local information only.)

The combination of a suitable branching strategy with the heuristic methods discussed earlier would make the latter complete, and appears to be a fruitful research direction.

To improve on the density theorem we must find ways to throw away irrelevant parts of the feasible domain that are guaranteed not to contain a global minimizer. To be able to do this reliably, some kind of global information is necessary. This is utilized by box reduction techniques, discussed in Section 10 using a simple example, and afterwards in more depth.

10. Box reduction: an example

Box reduction techniques are based on a more or less sophisticated interplay of several components: logical constraint propagation, interval analysis, convex relaxations and duality arguments involving Lagrange multipliers. Before giving a more formal treatment, we illustrate simple arguments of each of these components by reconsidering Example 5.2.

Suppose that a local solver has already produced the local minimizer $\hat{x} = \binom{0}{1}$ for the problem (5.5) discussed in Example 5.2, perhaps as the best local minimizer found by minimizing from a few random starting points. We use box reduction to check whether there is possibly a better feasible point. In fact, we know already that this is not the case, but we obtained this knowledge in a way that works only for very simple problems. Thus we want to do it again, using only techniques of wide applicability.

The idea of box reduction is to use various arguments that allow us to shrink the box without losing any feasible point that is at least as good as the best point found already. Since $\hat{x}$ is feasible with objective function value -2, any such point satisfies

$$f(x) = -x_1 - 2x_2 \le -2, \tag{10.1}$$

$$F(x) = (x_1-1)^2 + (x_2-1)^2 = 1, \tag{10.2}$$

$$x_1 \in [-1,1], \quad x_2 \in [-1,1]. \tag{10.3}$$

Constraint propagation (see Section 14) is a very cheap and easily formalizable process that gives important initial range reductions in many otherwise difficult problems. It consists in deducing better bounds for a variable by using the other bounds and one of the constraints. In particular, (10.1) implies $x_2 \ge 1 - x_1/2 \ge 0.5$ since $x_1 \le 1$, and $x_1 \ge 2 - 2x_2 \ge 0$ since $x_2 \le 1$, reducing the bounds to

$$x_1 \in [0,1], \quad x_2 \in [0.5,1].$$

Similarly, (10.2) implies $(x_1-1)^2 = 1 - (x_2-1)^2 \ge 1 - 0.25 = 0.75$, hence $x_1 \le 1 - \sqrt{0.75} < 0.14$, giving the improved bound

$$x_1 \in [0, 0.14].$$

This bound could be used to improve again x_2 using (10.1); and by alternating use of (10.1) and (10.2) one would obtain a sequence of boxes shrinking towards $\hat{x}$. This is a special feature of this simple example. In most cases, this simple substitution process gives no or only very little improvements after the first few good reductions. (Look at a problem with the constraints $x_1 + x_2 = 0$, $x_1 - x_2 = 0$, $x_1, x_2 \in [-1, 1]$ to see why.)

Interval analysis (see Section 11) can be applied in a number of different ways. Here we use it to produce linear relaxations of the nonlinear constraint. The Jacobian of $F(x)$ at $x \in \mathbf{x} = ([0, 0.14], [0.5, 1])^T$ is

$$F'(x) = (2x_1 - 2, 2x_2 - 2) \in ([-2, -1.72], [-1, 0]) = F'(\mathbf{x}).$$

The mean value theorem implies that, for any $\tilde{x} \in \mathbf{x}$,

$$F(x) \in F(\tilde{x}) + F'(\mathbf{x})(x - \tilde{x}) \quad \text{if} \quad x \in \mathbf{x}.$$

Using $\tilde{x} = \hat{x}$ we find

$$1 \in 1 + [-2, -1.72]x_1 + [-1, 0](x_2 - 1) = [1 - 2x_1, 2 - 1.72x_1 - x_2];$$

the interval evaluation needs no case distinction since x_1 and $x_2 - 1$ happen to have constant sign. The lower bound gives no new information, but the upper bound leads to the new constraint

$$1.72x_1 + x_2 \le 1.$$

By its derivation, this constraint is weaker than (10.2).

But since it is linear, the constraint is quite useful for *relaxation techniques* (see Section 16). It allows us to create a convex relaxation of the problem. Indeed, we may look at the relaxed linear program

$$\begin{array}{ll} \min & -x_1 - 2x_2 \\ \text{s.t.} & 1.72x_1 + x_2 \le 1, \quad 0 \le x_1 \le 0.14, \quad 0.5 \le x_2 \le 1. \end{array} \tag{10.4}$$

By construction, every feasible point better than the best point is feasible for (10.4), hence the minimum of (10.4) will be a *lower bound* on the best-possible objective function value of the original problem. Solving (10.4) gives the solution $\hat{x} = \binom{0}{1}$ with function value -2. Since this lower bound equals the best function value found so far for the original problem, the original problem has global minimum -2. This is a happy accident due to special circumstances: our problem already had a linear objective function, and the global minimizer was at a corner of the feasible set. (But as we shall see, we can adapt the technique to work much more generally if the box is narrow enough.)

Multiplier techniques can be used in a variety of ways. Here we use them to check whether there is a second, undiscovered global minimizer. We use the Lagrange multiplier $\hat{\lambda} = 2$ associated with the linear constraint of (10.4) at the solution. The associated linear combination $-x_1 - 2x_2 + 2(1.72x_1 + x_2 - 1)$ is bounded by the best-known function value -2 of the original problem, giving $2.44x_1 - 2 \le -2$, hence $x_1 \le 0$. Thus we must have $x_1 = 0$, and constraint propagation using (10.1) implies $x_2 = 1$. Thus the box has been reduced to $\hat{x}$, showing that it is the only global minimizer.

What generalizes? The problem discussed was deliberately kept simple so that the complete solution process could be demonstrated explicitly. In general, constraint propagation only gives limited reduction. Similarly, relaxed linear or convex programs usually only give a lower bound on the smallest-possible objective function value, but the linear combination derived from the Lagrange multipliers frequently contains useful information that can be exploited by constraint propagation to get a further significant box reduction.

If the reduction process stalls or becomes slow, the box is split into two or more smaller boxes. On the smaller boxes, the same techniques may prove effective, and we alternate box reduction and box splitting until all box sizes are below some termination threshold. Usually, only very few boxes remain if sufficiently good reduction techniques are used (pathological exceptions include min $x - x$ s.t. $x \in [0, 1]$). If no box remains, the problem is guaranteed to have no feasible point.

The total number of boxes processed is a measure of the difficulty of a problem for the particular algorithm used. Simple problems (like the example discussed above) only need a single box; in the worst case, an exponential number of boxes may be needed. In the latter case, time and storage limitations may force a premature termination; in this case the best point found is not verified to be a global minimizer.

11. Interval arithmetic

Interval analysis, the study of theory and algorithms for computing with intervals, is a large subject; see Moore (1979) (introductory), Neumaier (2001*b*) (embedded in a numerical analysis context) and Neumaier (1990) (advanced). Its importance for global optimization stems from several, interrelated facts.

- Interval analysis gives easily computable (though sometimes only very crude) bounds on the range expressions.
- Interval analysis allows one to control nonlinearities in a simple way (via centred forms).
- Interval analysis extends classical analysis in its ability to provide *semi-local* existence and optimality conditions, valid within a *pre-specified* local region around some point, while classical analysis generally only asserts the existence of such neighbourhoods without providing a simple way to find them.

We give here a short introduction to the basics and mention the main techniques useful for global optimization. General references on interval methods in global optimization include Adjiman *et al.* (1998*a*), Benhamou, McAllister and Van Hentenryck (1994), Benhamou and Older (1997), Berner (1996), Dallwig *et al.* (1997), Epperly and Pistikopoulos (1997), Epperly and Swaney (1996), Jansson (1994), Jansson and Knüppel (1992, 1995), Hansen (1980, 1992*a*), Kearfott (1996*b*), Kearfott, Novoa and Chenyi Hu (1991), Neumaier (1996), Ratschek and Rokne (1984, 1988), and Van Hentenryck *et al.* (1997*b*).

If $\mathbf{a}$ and $\mathbf{b}$ are two intervals we define for $\circ \in \{+,-,*,/,\hat{}\ \}$ the binary operation

$$\mathbf{a} \circ \mathbf{b} := \Box\{\tilde{a} \circ \tilde{b} \mid \tilde{a} \in \mathbf{a}, \tilde{b} \in \mathbf{b}\}, \tag{11.1}$$

provided the right-hand side is defined. Here

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

denotes the *interval hull* of a set of real numbers, *i.e.*, the tightest interval containing S. A monotonicity argument gives for addition and subtraction

$$\mathbf{a} + \mathbf{b} = [\underline{a} + \underline{b}, \overline{a} + \overline{b}], \tag{11.2}$$

$$\mathbf{a} - \mathbf{b} = [\underline{a} - \overline{b}, \overline{a} - \underline{b}], \tag{11.3}$$

and for multiplication and division

$$\mathbf{a} * \mathbf{b} = \Box\{\underline{a}\underline{b}, \underline{a}\overline{b}, \overline{a}\underline{b}, \overline{a}\overline{b}\}, \tag{11.4}$$

$$\mathbf{a}/\mathbf{b} = \Box\{\underline{a}/\underline{b}, \underline{a}/\overline{b}, \overline{a}/\underline{b}, \overline{a}/\overline{b}\} \quad \text{if} \quad 0 \notin \mathbf{b}; \tag{11.5}$$

in most cases only two of these products or quotients need to be computed. We also define elementary functions $\varphi \in \{\mathrm{sqr}, \mathrm{sqrt}, \exp, \log, \sin, \cos, \mathrm{abs}, \ldots\}$ of an interval $\mathbf{a}$ (and similarly $-\mathbf{a}, \mathbf{a}_+$, *etc.*) by

$$\varphi(\mathbf{a}) := \Box\{\varphi(\tilde{a}) \mid \tilde{a} \in \mathbf{a}\} \tag{11.6}$$

whenever the right-hand side is defined. Again $\varphi(\mathbf{a})$ can be computed from the value of φ at the end-points of $\mathbf{a}$ and the interior extremal values, depending on the monotonicity properties of φ. Note, however, that $|\mathbf{a}|$ is defined as $\sup \mathrm{abs}(\mathbf{a})$, since this expression figures prominently in estimates involving interval techniques.

For interval vectors (=boxes) $\mathbf{x}$, analogous definitions apply. We also need the *interior*

$$\mathrm{int}\ \mathbf{x} = \{\tilde{x} \in \mathbb{R}^n \mid \underline{x} < \tilde{x} < \overline{x}\}$$

of a box $\mathbf{x} \subseteq \mathbb{R}^n$.

For details and a systematic study of interval operations see Neumaier (1990); we only remark here that some rules familiar from real arithmetic fail, and in particular the interval evaluation of different expressions equivalent in real arithmetic may give different results. For example (with $-\mathbf{a} := 0 - \mathbf{a} = [-\overline{a}, -\underline{a}]$),

$$\mathbf{a} + (-\mathbf{a}) = \mathbf{a} - \mathbf{a} \neq 0 \quad \text{except when} \quad \underline{a} = \overline{a}.$$

Therefore, we also use the converse *inner operations*

$$\mathbf{a} \oplus \mathbf{b} := [\underline{a} + \overline{b}, \overline{a} + \underline{b}], \tag{11.7}$$

$$\mathbf{a} \ominus \mathbf{b} := [\underline{a} - \underline{b}, \overline{a} - \overline{b}]. \tag{11.8}$$

Here, expressions of the form $\pm\infty\mp\infty$ in (11.7) or (11.8) must be interpreted as $-\infty$ for the lower bounds and as $+\infty$ for the upper bounds. Note that the result of an inner operation is not necessarily an interval since it may happen that the lower bound is larger than the upper bound; giving an empty 'interval'.

All these operations are very simple to program. Note that many implementations of interval arithmetic are rather slow since they take care to guarantee correct (and often optimal) outward rounding, needed when interval arithmetic is used for mathematically rigorous certification (see Section 20). For global optimization without certification, *unsafe* interval arithmetic, which uses the standard rounding for floating point operations, and hence is significantly faster but may lose containment of points that lie too close to the boundary, usually suffices if certain safety measures are taken. But it is significantly harder to ensure robust behaviour with unsafe interval arithmetic since occasionally the solution is lost, too.

Important. When using unsafe interval arithmetic, proper safeguards must be taken at places (such as inner operations and intersections) where intervals might become (spuriously) empty owing to accumulation of round-off errors. In place of an empty result, a thin interval formed from the arithmetic mean of the two intersecting bounds should be returned in a robust implementation.

As already mentioned, an interval evaluation $f(\mathbf{x})$ of some expression f often overestimates the desired *range*

$$\operatorname{Range}(f, \mathbf{x}) = \{f(x) \mid x \in \mathbf{x}\}$$

of a function. However, under very mild conditions (Neumaier 1990, Section 1.4), the evaluation over small boxes satisfies

$$f(\mathbf{x}) \subseteq \operatorname{Range}(f, \mathbf{x}) + O(\varepsilon) \quad \text{if} \quad \overline{x} - \underline{x} = O(\varepsilon);$$

we refer to this as the *linear approximation property* of simple interval evaluation.

Better enclosures, especially for small ε, can be obtained by *centred forms*; the simplest of these (but not the most efficient: see Chapter 2 of Neumaier

(1990) for better methods based on slopes) is the *mean value form*. Owing to the mean value theorem, we have

$$f(x) \in f(z) + f'(\mathbf{x})(x - z) \quad \text{if} \quad x, z \in \mathbf{x}. \tag{11.9}$$

In particular, $\text{Range}(f, \mathbf{x})$ is contained in $f(z) + f'(\mathbf{x})(\mathbf{x} - z)$, and it can be shown that, under mild conditions,

$$f(z) + f'(\mathbf{x})(\mathbf{x} - z) \subseteq \text{Range}(f, \mathbf{x}) + O(\varepsilon^2) \quad \text{if} \quad \overline{x} - \underline{x} = O(\varepsilon);$$

we say that the mean value form (as other centred forms) has the *quadratic approximation property*. Recently, centred forms based on higher-order Taylor expansions have found considerable attention; these are able to give significantly sharper bounds in cases where simple interval evaluation suffers from severe dependence. See the survey Neumaier (2002) and the numerical comparisons in Makino and Berz (2003); *cf.* also Carrizosa, Hansen and Messine (2004).

Apart from interval evaluation and centred forms, we need *interval Newton methods* for verifying solutions of nonlinear systems of equations. The prototype (but again not the most efficient method; see Neumaier (1990, Chapter 5) for better methods based on slopes and Gauss–Seidel iteration) is the Krawczyk (1969) method. To check for solutions of $F(x) = 0$ with $x \in \mathbf{x}$, Krawczyk multiplies the vector version of (11.9) by a matrix C and subtracts it from x to find

$$x \in K(\mathbf{x}, z) := z - CF(z) + (I - CF'(\mathbf{x}))(\mathbf{x} - z).$$

For $z \in \mathbf{x}$, the resulting *Krawczyk operator* $K(\mathbf{x}, z)$ (*cf.* Krawczyk (1969), Kahan (1968)) has the following properties, typical for interval Newton operators:

(i) any zero $x \in \mathbf{x}$ of F lies in $\mathbf{x} \cap K(\mathbf{x}, z)$,

(ii) if $\mathbf{x} \cap K(\mathbf{x}, z) = \emptyset$ then $\mathbf{x}$ contains no zero of F,

(iii) if $K(\mathbf{x}, z) \subseteq \text{int}\, \mathbf{x}$ then $\mathbf{x}$ contains a unique zero of F.

Properties (i) and (ii) follow directly from the above derivation, while (iii) is a simple consequence of Banach's fixed point theorem.

The most important part is (iii), since, applied to the Karush–John conditions, it allows the elimination of large regions around a local minimizer; *cf.* Section 15. However, (i) and (ii) are also useful as ways of reducing a box or eliminating it, if it contains no zero. This is implemented in GlobSol (Kearfott 1996*b*) and Numerica (Van Hentenryck *et al.* 1997*b*).

Another useful interval Newton operator with analogous properties is

$$x \in N(\mathbf{x}, z) := z - (CF[\mathbf{x}, z])^H (CF(z)),$$

where C is an approximate inverse of the interval slope $F[\mathbf{x}, z]$ and $\mathbf{A}^H\mathbf{b}$ is an enclosure for the set of solutions of $Ax = b$, $A \in \mathbf{A}$, $b \in \mathbf{b}$ computed, *e.g.*, by the Hansen–Bliek method (Bliek 1992, Hansen 1992*b*, Neumaier 1999).

11.1. Convexity check

Interval analysis can be used to check the convexity of a function $f : \mathbf{x} \to \mathbb{R}$ in some box $\mathbf{x}$. Let $\mathbf{G}$ be a matrix of intervals (usually simply called an *interval matrix*), calculated as an enclosure of $f''(x)$ for $x \in \mathbf{x}$, then, with $r = \max\{\overline{x}_k - \underline{x}_k \mid k = 1, \dots, n\}$, the linear approximation property implies that $|\overline{G} - \underline{G}| = O(r)$. Such a statement implies that $|\mathbf{G} - \tilde{G}| = O(r)$ for all individual matrices $\tilde{G} \in \mathbf{G}$, with absolute values taken component-wise. In particular, if $\hat{G}$ is positive definite then, provided the underlying box is not too wide, all matrices in $\mathbf{G}$ are definite, too; and if this is the case, f is convex in $\mathbf{x}$. The following constructive criterion for simultaneously checking the definiteness of all members of an interval matrix was given in Neumaier (1996).

Theorem 11.1. (Sufficient conditions for convexity) Let $f : \mathbf{x} \to \mathbb{R}$ be twice continuously differentiable on the compact box $\mathbf{x}$, and suppose that $\mathbf{G}$ is a symmetric interval matrix such that

$$f''(x) \in \mathbf{G} \quad \text{for all} \quad x \in \mathbf{x}. \tag{11.10}$$

(i) If some symmetric matrix $G_0 \in \mathbf{G}$ is positive definite and all symmetric matrices in $\mathbf{G}$ are nonsingular then they are all positive definite, and f is uniformly convex in $\mathbf{x}$.

(ii) In particular, this holds if the midpoint matrix

$$\check{G} = (\sup \mathbf{G} + \inf \mathbf{G})/2$$

is positive definite with inverse C, and the preconditioned radius matrix

$$\Delta = |C| \operatorname{rad} \mathbf{G},$$

where $|C|$ is the componentwise absolute value of C and

$$\operatorname{rad} \mathbf{G} = (\sup \mathbf{G} - \inf \mathbf{G})/2,$$

satisfies the condition

$$\|\Delta\| < 1 \tag{11.11}$$

(in an arbitrary norm).

Proof. **(i)** Since the eigenvalues are continuous functions of the matrix entries and the product of the eigenvalues (the determinant) cannot vanish, no eigenvalue changes sign. Hence the eigenvalues of all matrices in $\mathbf{G}$ are positive, since this is the case for the positive definite member. Thus all symmetric matrices in $\mathbf{G}$ are positive definite. By well-known results, uniform convexity of f now follows from (11.10).

(ii) $G_0 = \check{G}$ belongs to $\mathbf{G}$, and condition (11.11) implies strong regularity of the interval matrix $\mathbf{G}$ (Neumaier 1990, Section 4.1) and hence nonsingularity of all matrices in $\mathbf{G}$. Thus (i) applies. $\square$

In many cases, the Hessian of the augmented Lagrangian can be shown to have the form

$$f''(x) = \sum u_i A_i, \quad u_i \in \mathbf{u}_i,$$

with constructively available real matrices A_i and intervals $\mathbf{u}_i = [\check{u}_i - r_i, \check{u}_i + r_i]$. In this case, the above result can be strengthened (with virtually the same proof) by replacing $\check{G}$ and Δ with

$$\check{G} = \sum \check{u}_i A_i$$

and

$$\Delta' = \sum r_i |CA_i|,$$

respectively. Indeed, it is not difficult to see that for $\mathbf{G} = \sum \mathbf{u}_i A_i$, we always have $0 \le \Delta' \le \Delta$, so that the refined test is easier to satisfy.

Other sufficient conditions for convexity based on scaled Gerschgorin theorems and semidefinite programming, form the basis of the αBB method (Adjiman *et al.* 1996, Androulakis *et al.* 1995) and are given in Adjiman *et al.* (1998*a*) and Adjiman *et al.* (1998*b*).

12. The branch and bound principle

The branch and bound principle is a general label (invented in Land and Doig (1960) and Little *et al.* (1963)) to denote methods to split a problem recursively into subproblems which are sooner or later eliminated by showing that the subproblem cannot lead to a point better than (or as least as good as) the best point found so far. The latter is often checked by computing lower bounds on the objective function, and the splitting produces new branches in the tree of all subproblems tried, according to so-called *branching rules*; hence the name 'branch and bound'. But in practice, the subproblems are best treated in a more flexible fashion, allowing us also to eliminate subproblems only partially.

General references for branch and bound in global optimization include Beale (1979, 1988), Berner (1996), Epperly and Pistikopoulos (1997), Epperly and Swaney (1996), Floudas (1995), Grossmann (1990), Horst *et al.* (1995), Jansson and Knüppel (1992), Kearfott (1996*b*), Pintér (1996*a*) and Van Hentenryck *et al.* (1997*b*). A thorough discussion of branch and bound in discrete optimization, with many algorithmic choices that are of potential interest in general global optimization, is given in Parker and Rardin (1988).

For a global optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in \mathbf{x}^{\text{init}}, \quad F(x) \in \mathbf{F}, \quad x_I \text{ integral}, \end{array} \tag{12.1}$$

a natural way to define subproblems is to choose boxes $\mathbf{x} \subseteq \mathbf{x}^{\text{init}}$ of the initial box $\mathbf{x}^{\text{init}}$, and to consider the subproblems

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in \mathbf{x}, \quad F(x) \in \mathbf{F}, \quad x_I \text{ integral}, \end{array} \tag{12.2}$$

that is, each subproblem is characterized by (and stored as) the box over which the problem is solved. The branching process then consists in splitting a box $\mathbf{x}$ into two or several smaller boxes whose union is $\mathbf{x}$. The most typical branching rule is to select a *bisection coordinate* j and to split the jth component of the box at a *bisection point* ξ. Thus, the current box $\mathbf{x}$ is replaced by two sub-boxes $\mathbf{x}^{\text{low}}, \mathbf{x}^{\text{upp}}$ with

$$\begin{array}{l} \mathbf{x}_k^{\text{low}} = \mathbf{x}_k^{\text{upp}} = \mathbf{x}_k \quad \text{if} \quad k \neq j, \\ \mathbf{x}_j^{\text{low}} = [\underline{x}_j, \xi], \quad \mathbf{x}_j^{\text{upp}} = [\xi, \overline{x}_j]. \end{array} \tag{12.3}$$

This branching rule is termed *bisection*. The bisection point ξ is often taken as the *midpoint* $\xi = (\overline{x}_j + \underline{x}_j)/2$ of the interval $\mathbf{x}_j$; but this fails when there are infinite bounds and is inefficient when the interval ranges over several orders of magnitude. In this case, a more useful bisection point is a *safeguarded geometric mean*, defined by

$$\xi = \operatorname{sign} \underline{x}_j \sqrt{\underline{x}_j \overline{x}_j} \quad \text{if} \quad 0 < \underline{x}_j \overline{x}_j < \infty,$$

and otherwise

$$\begin{array}{ll} \xi = 0 & \text{if} \quad \underline{x}_j < 0 < \overline{x}_j, \\ \xi = \min(\mu, q\overline{x}_j) & \text{if} \quad \underline{x}_j = 0, \\ \xi = \max(-\mu, q\underline{x}_j) & \text{if} \quad \overline{x}_j = 0, \\ \xi = q^{-1}\underline{x}_j & \text{if} \quad \underline{x}_j > 0, \\ \xi = q^{-1}\overline{x}_j & \text{if} \quad \overline{x}_j < 0, \end{array}$$

where $q \in \ ]0,1[$ is a fixed constant (such as $q = 0.01$) and variables whose initial interval contains 0 are assumed to be most likely of magnitude μ.

The branching coordinate is more difficult to choose, but the speed of a branch and bound algorithm may be heavily affected by this choice. For a good algorithm, the choice should be scaling-invariant, but the details depend on how the algorithm treats the individual subproblems.

Sometimes, a *trisection* branching rule is used which splits some component of a box into three intervals. Also, *multisection* branching rules may be employed; only one natural choice is described here. Suppose we know that a sub-box $\mathbf{x}^0$ of $\mathbf{x}$ cannot contain a solution of (12.1). (In practice, $\mathbf{x}^0$ would be the intersection of an exclusion box with $\mathbf{x}$; see Section 11.) Then we can cover $\mathbf{x} \setminus \mathbf{x}^0$ by (at most) $2n$ sub-boxes, namely, for $j = 1, \ldots, n$,

$$\begin{aligned} &\mathbf{x}_k^{2j-1} = \mathbf{x}_k^{2j} = \mathbf{x}_k^0 \quad \text{if} \quad k < j, \\ &\mathbf{x}_j^{2j-1} = [\underline{x}_j, \underline{x}_j^0], \quad \mathbf{x}_j^{2j} = [\overline{x}_j^0, \overline{x}_j], \\ &\mathbf{x}_k^{2j-1} = \mathbf{x}_k^{2j} = \mathbf{x}_k \quad \text{if} \quad k > j. \end{aligned} \tag{12.4}$$

However, this may yield long and thin slices and is then rather inefficient.

For a comparison of some branching rules for bound-constrained problems see Csendes and Ratz (1997), Ratz (1996), Ratz and Csendes (1995).

The *bounding rule* in its classical variant requires the solution of a *convex relaxation*, *i.e.*, a convex (and often linear) optimization problem whose feasible set contains the feasible set of the subproblem (*outer approximation*) and whose objective function is at no feasible point larger than the original objective function (*underestimation*). If the convex problem is infeasible, the subproblem is infeasible, too, and can be discarded. If the convex problem is feasible, its solution provides a lower bound on $f(x)$, and when this lower bound is larger than the value of f^{best} for some feasible point x^{best} known (stored in a list of *best feasible points found so far*) we conclude that the subproblem no longer contributes to the solution of the global optimization problem and hence can be discarded.

Clearly, this procedure is equivalent to adding the constraint $f(x) \le f^{\text{best}}$ to the definition of the subproblem and checking infeasibility of the resulting reduced subproblem. This suggests a more general approach to defining subproblems by adding other *cuts*, *i.e.*, derived inequalities that have to be satisfied at a global minimizer. If these inequalities are linear, the cuts define hyperplanes and are referred to as *cutting planes*; *cf.* Section 16. Branch and bound methods using cuts are frequently labelled *branch and cut*.

Another important approach to handling subproblems uses constraint propagation and related techniques that define *reduction* (also called *tightening*, *narrowing*, *filtering* or *pruning*) *rules* which serve to reduce (as much as easily possible) the box defining a subproblem without changing its feasible set. If reduction results in an empty box, the subproblem is eliminated; if not, the subproblem may still have been reduced so much that many branching steps are saved. Fast and simple reduction rules use constraint propagation, discussed in Section 14; more expensive rules are discussed in Section 15. The balancing of work done in reduction versus work saved through less branching is a delicate matter, which at present more or less depends on *ad hoc* recipes.

Note that reduction techniques may be applied not only to the original constraints but to all constraints that must be satisfied at the global minimizer. This includes cutting planes (see Section 16) and the equations and inequalities derived from the Karush–John optimality conditions (see Section 5). In particular, software based on interval techniques (GlobSol (Kearfott 1996*b*), Numerica (Van Hentenryck *et al.* 1997*b*)) make essential use of the latter.

13. The role of local optimization

Local optimization routines are an important part of most global solvers. They are used for two different purposes.

(i) To find feasible points if the feasible domain has a complicated definition, and to find better local minimizers when (after successful tunnelling) a feasible point better than the previously best local minimizer has been found.

(ii) To solve auxiliary optimization problems such as relaxations of the original problem (for generating improved bounds) or bound-constrained approximations (for tunnelling).

13.1. Relaxation

The auxiliary local optimization problems that need to be solved are simpler in structure since they 'relax' the problem in some way. A *relaxation* is a modification of the original problem whose solution is tractable and gives some information about the possible location of the global minimizer. In the past, mainly linear and convex relaxation have been used, since for these, local optimization provides global solutions, which usually implies useful global information about the original problem. We shall discuss various ways of obtaining and using linear and convex relaxations in Section 16. Nonconvex relaxations may be useful, too, if they are reliably solvable to global optimality. We therefore discuss semilinear relaxations – which can be solved by MILP techniques – in Section 18.

13.2. Tunnelling

One may consider solving a global optimization problem as a *sequential nonlinear programming method* (SNLP), where local optimization (NLP) steps that improve a feasible point to local optimality alternate with tunnelling steps that produce better (nearly) feasible points by some *tunnelling procedure*. For complete methods based on branching, the 'tunnelling' is done by finding nearly feasible points during inspection of the subproblems.

The success of the tunnelling step depends on the details of looking for such points. One strategy (Dallwig *et al.* 1997) proceeds by solving on selected sub-boxes nonlinear least squares problems that minimize the sum of squares of the constraint violations, and (if a best feasible point with function value f^{best} is already available) the violation of $f(x) \le f^{\text{best}} - \Delta$, where $\Delta \ge 0$ is some measure of minimal gain in function value. Alternatively, one may use the soft optimality theorem (Theorem 8.1) in place of least squares. (See also Guddat *et al.* (1990) for tunnelling by continuation.) Thus, in a sense, the global optimization of (4.1) consists in solving a sequence of harder and harder feasibility problems

$$\begin{array}{ll} \text{find} & x \\ \text{s.t.} & x \in \mathbf{x}, \quad F(x) \in \mathbf{F}, \quad x_I \text{ integral}, \\ & f(x) \le f^{\text{best}} - \Delta. \end{array} \tag{13.1}$$

Typical global optimization methods spend perhaps 5% of their time on finding a global minimizer, and the remaining 95% on the verification that there is no significantly better feasible point, *i.e.*, showing that the feasibility problem (13.1) has no solution. Also, hard problems need a significant amount of time to find the first feasible point. Thus the initial and final (dominant) stages of a global optimization solution process are essentially identical to that for a feasibility problem.

In particular, general feasibility problems, also called *constraint satisfaction problems*, can be as hard as general global optimization problems, and the techniques needed for solving constraint satisfaction problems are essentially the same as those for solving global optimization problems.

13.3. General considerations

Considerations of superlinear convergence of local optimization algorithms imply that one generally uses *sequential quadratic programming* (SQP) techniques, which solve a sequence of related quadratic programs whose solution converges (under certain conditions, *cf.* below) to a local minimizer of the original problem; if the starting point is feasible (which, initially, need not be the case), the function value of the local minimizer is at or below that of the starting point.

To give the reader a rough idea of times and difficulties, here are some (completely unreliable but catchy) rules of thumb. If the time needed to solve a linear program of a certain size is LP then solving a problem of comparable size and sparsity structure may take perhaps the time

$$QP = 5 * LP$$

for a convex quadratic program,

$$QP' = 10 * LP$$

for a local minimizer of a nonconvex quadratic program,

$$SQP = 30 * QP$$

for a convex nonlinear program,

$$SQP' \geq 200 * QP$$

for a local minimizer of a nonconvex nonlinear program,

$$GLP_f \geq 100 * SQP$$

for *finding* a global minimizer of a nonconvex nonlinear program, and

$$GLP_v \geq 1000 * SQP$$

for *verifying* that it is a global minimizer.

We now comment on the properties of local optimization software that are important for their use in global optimization. Usually, it is more important that the local solver is fast than that it is very robust (*i.e.*, guaranteed to succeed), since lack of robustness in some of the local optimizations is made up for by the structure of the global solution process. To help control the amount of work done in the local part, it should be possible to force a premature return with a less than optimal point when some limit (of time or number of function values) is exceeded. Nevertheless, the local solver should be good to ensure that solving a problem with a unique minimizer (which is automatically global) by the global solver does not take much longer than a good local solver would need.

Modern nonlinear programming codes are usually 'globally convergent' in some sense. The global convergence proofs (to a *local* minimizer only!) usually make more or less stringent assumptions that imply the absence of difficulties in finding feasible points. Formally, we may say that a local optimization algorithm is *globally convergent* if there is a continuous function $d_{\text{feas}} : \mathbb{R}^n \to \mathbb{R}$ (defining a 'distance to feasibility') such that

$$d_{\text{feas}}(x) \geq 0, \quad \text{with equality if and only if } \ x \text{ is feasible}$$

and the algorithm produces for arbitrary continuous problems and arbitrary starting points a sequence of $x^l \in \mathbb{R}^n$ satisfying one of the following conditions:

(i) x^l converges to the set of points satisfying the Karush–John conditions (and, possibly, second-order necessary conditions);

(ii) $d_{\text{feas}}(x^l) \to 0$ and $f(x^l) \to -\infty$;

(iii) x_l converges to the set of points where the objective or some constraint function is not continuously differentiable;

(iv) $d_{\text{feas}}(x^l) \to 0$, $\|x^l\| \to \infty$;

(v) $d_{\text{feas}}(x^l)$ converges to a nonzero local minimum of d_{feas}.

Conditions (i) and (ii) characterize the achievement of the optimization goal, while conditions (iii)–(v) characterize various modes of unavoidable failure. Failures of type (iii) or (iv) are usually attributed to bad modelling or bad choice of the optimization methods. Some methods such as bundle methods can cope with lack of differentiability, hence do not lead to case (iii).

A failure of type (v) is unavoidable if there is no feasible point. However, failures of type (v) may happen for problems with nonconvex constraints even though feasible points exist. One could say that from a local point of view, an optimization problem is *easy* (for an algorithm) if (v) cannot occur whenever a feasible point exists. A local algorithm may be considered good if among its easy problems are all problems with convex constraints only, and all problems satisfying certain strong versions (Burke 1991) of the Mangasarian–Fromovitz constraint qualification (Mangasarian and Fromovitz 1967). Ideally, a good local algorithm would provide in these cases a certificate of infeasibility whenever it detects case (v).

14. Constraint propagation

In many cases, general constraints can be used to reduce the size of a box in the branching scheme. The general technique is called *constraint propagation* and was pioneered in constraint logic (Cleary 1987, Older and Vellino 1993) and interval analysis (Neumaier 1988), but has also forerunners in presolve techniques in mathematical programming (Mangasarian and McLinden 1985, Lodwick 1989, Anderson and Anderson 1995). See Babichev *et al.* (1993), Benhamou *et al.* (1994), Benhamou and Older (1997), Chen and van Emden (1995), Hager (1993), Hyvönen and De Pascale (1996), Kearfott (1991), Van Hentenryck (1989), Van Hentenryck, Michel and Benhamou (1997*a*), Van Hentenryck *et al.* (1997*b*) for further developments, and the COCONUT report (Bliek *et al.* 2001) for an extensive recent survey.

We follow here the set-up by Dallwig *et al.* (1997), which handles linear constraints (and more generally block-separable constraints) without the need to decompose the constraints into primitive pieces defined by single operations. (In the following, if J is a list of indices, x_J denotes the subvector of x formed by the components with index in J.)

Proposition 14.1. Let the q_k be real-valued functions defined on $\mathbf{x}_{J_k}$.

(i) If (for suitable $\overline{q}_k, \overline{s}$)

$$\overline{q}_k \geq \sup\{q_k(x_{J_k}) \mid x_{J_k} \in \mathbf{x}_{J_k}\}, \quad \overline{s} \geq \sum_k \overline{q}_k, \tag{14.1}$$

then, for arbitrary $\underline{a}$,

$$x \in \mathbf{x}, \quad \underline{a} \leq \sum_k q_k(x_{J_k}) \quad \Rightarrow \quad q_k(x_{J_k}) \geq \underline{a} - \overline{s} + \overline{q}_k \quad \text{for all} \quad k. \tag{14.2}$$

(ii) If

$$\underline{q}_k \le \inf\{q_k(x_{J_k}) \mid x_{J_k} \in \mathbf{x}_{J_k}\}, \quad \underline{s} \le \sum_k \underline{q}_k, \tag{14.3}$$

then, for arbitrary $\underline{a}$,

$$x \in \mathbf{x}, \quad \sum_k q_k(x_{J_k}) \le \overline{a} \quad \Rightarrow \quad q_k(x_{J_k}) \le \overline{a} - \underline{s} + \underline{q}_k \quad \text{for all} \quad k. \tag{14.4}$$

Proof. The assumptions of part (i) imply

$$q_k(x_{J_k}) \ge \underline{a} - \sum_{l \ne k} q_l(x_{J_l}) \ge \underline{a} - \sum_{l \ne k} \overline{q}_l \ge \underline{a} + \overline{q}_k - \overline{s},$$

hence the conclusion in (14.2) holds. Part (ii) is proved in the same way. □

The proposition is applied as follows to reduce the size of boxes by tightening bound constraints. Suppose that $x \in \mathbf{x}$. For any constraint of the form

$$\underline{a} \le \sum_k q_k(x_{J_k}) \tag{14.5}$$

we form the quantities (14.1). (This is straightforward if the q_k depend on a single variable x_k only, $J_k = \{k\}$; in the most important cases, q_k is linear or quadratic in x_k, and the supremum is very easy to calculate; in more complicated cases, upper (resp. lower) bounds can be calculated with interval arithmetic.) Then one checks the condition

$$\underline{a} \le \overline{s}. \tag{14.6}$$

If it is violated then (14.5) is clearly inconsistent with $x \in \mathbf{x}$ (and in the branch and bound application, the corresponding subproblem can be discarded). If (14.6) holds, one can exploit the conclusion in (14.2), provided that one can compute the set of $x_{J_k} \in \mathbf{x}_{J_k}$ (or a superset) such that

$$q_k(x_{J_k}) \ge \overline{q}_k + \underline{a} - \overline{s}. \tag{14.7}$$

If $\underline{a}$ is sufficiently close to $\overline{s}$ then x_{J_k} will be forced to be close to the global maximum of q_k over the interval $\mathbf{x}_{J_k}$, thus reducing the component $\mathbf{x}_{J_k}$ and hence the box $\mathbf{x}$. This procedure can be applied for each k in turn to get an optimally reduced box. One can similarly proceed for block-separable constraints of the form $\sum q_k(x_{J_k}) \le \overline{a}$. (The reader might wish to reconsider the example in Section 10 in the light of the above result.)

In the separable case ($J_k = \{k\}$), computing the set of x_k with (14.7) is easy, especially for linear or quadratic q_k. If q_k is nonmonotonic, it may happen that the resulting set is disconnected; then one has to make a choice between taking its convex hull – which is an interval – or of considering splitting the box into sub-boxes corresponding to the connected components.

In the case of two-sided constraints $\sum q_k(x_{J_k}) \in \mathbf{a}$, which includes the equality constraint $\sum q_k(x_{J_k}) = q_0$ for $\mathbf{a} = q_0$, one can combine (14.2) and (14.4) using interval arithmetic as follows. (See (11.7) for the inner addition $\oplus$.)

Proposition 14.2. Suppose that

$$\mathbf{q}_k \supseteq \square\{q_k(x_{J_k}) \mid x_{J_k} \in \mathbf{x}_{J_k}\}, \quad \mathbf{r} \supseteq \mathbf{a} - \sum_k \mathbf{q}_k. \tag{14.8}$$

(i) If $0 \notin \mathbf{r}$ then the conditions

$$x \in \mathbf{x}, \quad \sum_k q_k(x_{J_k}) \in \mathbf{a} \tag{14.9}$$

are inconsistent.

(ii) Any x satisfying (14.9) also satisfies

$$q_k(x_{J_k}) \in \mathbf{r} \oplus \mathbf{q}_k \quad \text{for all} \quad k. \tag{14.10}$$

Proof. (14.9) implies $0 \in \mathbf{a} - \sum q_k(x_{J_k}) \subseteq \mathbf{a} - \sum \mathbf{q}_k$, hence $0 \in \mathbf{r}$. Now suppose that $0 \in \mathbf{r}$. In the notation of the previous proposition we have

$$q_k(x_{J_k}) \in [\underline{a} - \overline{s} + \overline{q}_k, \overline{a} - \underline{s} + \underline{q}_k] = [\underline{a} - \overline{s}, \overline{a} - \underline{s}] \oplus \mathbf{q}_k,$$

and since $\mathbf{r} = \mathbf{a} - [\underline{s}, \overline{s}] = [\underline{a} - \underline{s}, \overline{a} - \overline{s}]$, this implies (14.10). $\square$

Again, condition (14.10) can be used to reduce $\mathbf{x}_{J_k}$ whenever

$$\mathbf{q}_k \not\subseteq \mathbf{r} \oplus \mathbf{q}_k. \tag{14.11}$$

We give details for the most important case of quadratic (and including linear) functions, dropping indices for a moment.

Proposition 14.3. Let $\mathbf{c}$ be an interval, $a, b \in \mathbb{R}$, and put

$$\mathbf{d} := (b^2 + 4a\mathbf{c})_+, \quad \mathbf{w} := \sqrt{\mathbf{d}} \text{ (if } \mathbf{d} \neq \emptyset).$$

Then

$$\{x \in \mathbb{R} \mid ax^2 + bx \in \mathbf{c}\} = \begin{cases} \emptyset & \text{if } \mathbf{d} = \emptyset, \\ \emptyset & \text{if } a = b = 0 \notin \mathbf{c}, \\ \mathbb{R} & \text{if } a = b = 0 \in \mathbf{c}, \\ \frac{\mathbf{c}}{b} & \text{if } a = 0, \\ \frac{-b-\mathbf{w}}{2a} \cup \frac{-b+\mathbf{w}}{2a} & \text{otherwise.} \end{cases}$$

Proof. $ax^2 + bx = \tilde{c} \in \mathbf{c}$ is equivalent to $x = \tilde{c}/b$ when $a = 0$, and to $x = (-b \pm \sqrt{b^2 + 4a\tilde{c}})/2a$ otherwise; in the latter case, the expression under the square root must be nonnegative and hence lies in $\mathbf{d}$. Since the varying $\tilde{c}$ occurs in these formulas only once, the range over $\tilde{c} \in \mathbf{c}$ is given by $\mathbf{c}/b$ if $a = 0$ and by $(-b \pm \sqrt{\mathbf{d}})/2a$ otherwise (use monotonicity!). $\square$

Note that the differences in Proposition 14.1 and the numerators in Proposition 14.3 may suffer from severe cancellation of leading digits, which requires attention in an actual implementation.

In the application to reducing boxes, one must of course intersect these formulae with the original interval. If the empty set results, the subproblem corresponding to the box $\mathbf{x}$ can be eliminated. (But remember to be cautious when using unsafe interval arithmetic!) If a disjoint union of two intervals results we either split the box into two boxes corresponding to the two intervals or leave $\mathbf{x}_k$ unchanged; the first alternative is advisable only when the gap in the interval is quite large.

All reduction techniques may be used together with the technique of *shaving*, which may be seen as an adaptation of the probing technique in mixed integer programming. The idea is to try to remove a fraction of the range $[\underline{x}_i, \overline{x}_i]$ of some variable x_i by restricting the range to a small subrange $[\underline{x}_i, \xi]$ or $[\xi, \overline{x}_i]$ at one of the two end-points of that variable, and testing whether reducing the small slab obtained in this way results in an empty intersection. If this is the case, the range of x_i can be restricted to the complementary interval $[\xi, \overline{x}_i]$ and $[\underline{x}_i, \xi]$, respectively. While more expensive, it reduces the overestimation in the processing of constraints which contain a variable several times. In practice, one would perhaps try to shave away 10% of the length of an interval.

14.1. Consistency concepts

In constraint logic programming (see the book by Van Hentenryck *et al.* (1997*b*) and the references at the beginning of this section), there are many consistency concepts that describe the strength of various reduction techniques. Essentially, a box is *consistent* with respect to a set of reduction procedures if their application does not reduce the box. A simple recursive argument invoking the finiteness of machine-representable boxes shows that every box can be reduced to a consistent box with finitely many applications of the reduction procedures in an arbitrary order. (Depending on the rules used, the resulting reduced box – called a *fixed point* of the reduction procedures – may or may not depend on the order of applying the rules.)

From a practical point of view, it is not advisable to apply the available rules until the fixed point is reached. The reason is that frequently the first few reductions are substantial, and later ones only reduce the box by tiny fractions; the convergence speed may be arbitrarily slow. For example, for the pair of constraints $x_1 + x_2 = 0, x_1 - qx_2 = 0$ with $q \in]0, 1[$, the unique fixed point (with respect to the simple reduction described above) reduces the volume in each step by a factor of q. For q close to one this is very inefficient compared to, say, a linear programming relaxation (which gives the result immediately).

Thus one has to be selective in practice, using suitable strategic rules for when to use which reduction strategy. The choice is usually done by various *ad hoc* recipes that balance the likely gain and the amount of work needed. Moreover, fine-grained interaction between different computations to avoid some unnecessary computation, such as that described in Granvilliers (2001) may be decisive in getting optimal performance.

14.2. Semiseparable constraints

With some more work, the above techniques can also be utilized for semi-separable constraints. We need the following result (essentially a form of the Cauchy–Schwarz inequality).

Lemma 14.4. If A is a rectangular matrix such that A^TA is nonsingular then

$$|u_k| \le \sqrt{((A^TA)^{-1})_{kk}}\|Au\|_2 \quad \text{for all } u.$$

Proof. Let $A = QR$ be an orthogonal factorization of A with $Q^TQ = I$ and R square nonsingular. Then $A^TA = R^TR$ and $\|Au\|_2 = \|Ru\|_2$. Since

$$|u_k| = |(R^{-T}e^{(k)})^TRu| \le \|R^{-T}e^{(k)}\|_2\|Ru\|_2,$$
$$\|R^{-T}e^{(k)}\|_2^2 = (e^{(k)})^TR^{-1}R^{-T}e^{(k)} = ((R^TR)^{-1})_{kk},$$

the assertion follows. □

Now suppose that we have a semiseparable inequality of the form

$$\sum_k q_k(x_k) + (x-x^0)^TH(x-x^0) \le \overline{a}, \tag{14.12}$$

with possibly nonsymmetric H. Using a modified Cholesky factorization (Gill and Murray 1974, Schnabel and Eskow 1990)

$$H + H^T = R^TR - D$$

with a (nonnegative) diagonal matrix D, we can rewrite (14.12) as

$$0 \le \frac{1}{2}\|R(x-x^0)\|_2^2 \le \overline{a} - \sum_k q_k(x_k) + \frac{1}{2}(x-x^0)^TD(x-x^0). \tag{14.13}$$

The right-hand side of (14.13) is a separable quadratic form, hence can be written as $\overline{a} - \sum \tilde{q}_k(x_k)$ with $\tilde{q}_k(x_k) = q_k(x_k) - \frac{1}{2}D_{kk}(x_k - x_k^0)^2$. Therefore, Proposition 14.1(ii) applies. Moreover, one gets the extra inequality

$$\|R(x-x^0)\|_2^2 \le 2(\overline{a} - \underline{s}),$$

which together with the lemma gives the further inequalities

$$|x_k - x_k^0| \le \sqrt{2(\overline{a} - \underline{s})((R^T R)^{-1})_{kk}}, \qquad (14.14)$$

which may help to reduce $\mathbf{x}_k$.

14.3. Block-separable constraints

For only block-separable constraints ($|J_k| > 1$), the q_k are multivariate, and we need to resort to suboptimal interval techniques.

How to exploit the enclosures from Proposition 14.1 and 14.2 to reduce the box depends on the special form of the q_k. In many cases, we can in turn solve the conditions directly for each variable involved, substituting an enclosing interval for all other variables.

If this is not possible directly we can use the mean value form (or another centred form) to rewrite a constraint $F_i(x) \in \mathbf{F}_i$ as

$$F_i(\xi) + F'(\mathbf{x})(x - \xi) \cap \mathbf{F}_i \ne \emptyset;$$

this is now a separable expression with interval coefficients that can be processed as above to reduce the box. This way of proceeding, dating back to Neumaier (1988), is called *conditioning* in Van Hentenryck *et al.* (1997*b*), and is used in the Numerica package. Similarly, by using a Taylor expansion to second order with an interval Hessian, we get a semiseparable expression with interval coefficients that can in principle be processed as above. (However, the interval coefficients cause here additional complications.) In the context of Taylor models of arbitrary order, a variation of this (with thin coefficients and an interval remainder term) has been used in the linear dominated bounder of Makino (1998); *cf.* the discussion in Makino and Berz (2003) and Neumaier (2002).

15. The cluster problem and second-order information

When programming a simple branch and bound algorithm for global optimization, one quickly notices that it is fairly easy to eliminate boxes far away from the global minimizer, while, especially in higher dimensions, there remains a large cluster of tiny boxes in a neighbourhood of the global minimizer that is difficult to eliminate. The occurrence of this situation is called the *cluster problem*. Often, algorithms try to avoid the cluster problem by providing only a Δ-optimal solution; *i.e.*, the program stops when it has shown that there is no feasible point with an objective function value of $f^{\text{best}} - \Delta$, where f^{best} is the function value of the best feasible point found so far. However, when Δ is small (as we want) then the cluster problem is still present, although to a less pronounced degree.

Kearfott and Du (1994) studied the cluster problem for unconstrained global optimization, and discovered that the source of the problem was the limited accuracy with which the function values were bounded. In particular, they showed that the cluster problem disappears if, for x in a box of diameter $O(\varepsilon)$, one can bound the overestimation of $f(x^{\text{best}}) - f(x)$ by $O(\varepsilon^3)$. Here we give a simplified version of their result.

Let $\widehat{x} \in \mathbb{R}^n$ be a global minimizer of $f(x)$, and let $\widehat{G}$ be the Hessian at $\widehat{x}$. Near the global minimizer, we have

$$f(x) = f(\widehat{x}) + \frac{1}{2}(x - \widehat{x})^T \widehat{G}(x - \widehat{x}) + O(\|x - \widehat{x}\|^3)$$

since the gradient vanishes at $\widehat{x}$. Suppose we can bound the objective function value over a box of diameter ε with an accuracy of $\Delta = K\varepsilon^{s+1}$, $s \le 2$. Then no box of diameter ε containing a point x with $\frac{1}{2}(x - \widehat{x})^T \widehat{G}(x - \widehat{x}) + O(\|x - \widehat{x}\|^3) \le \Delta$ can be eliminated. For sufficiently small Δ, this describes a nearly ellipsoidal region with volume proportional to $\sqrt{(2\Delta)^n / \det \widehat{G}}$, and any covering by boxes of diameter ε contains at least $\text{const}\sqrt{(2\Delta)^n / (\varepsilon^n \det \widehat{G})}$ boxes. The number of uneliminated boxes is therefore proportional to at least

$$\begin{array}{ll} (\text{const}/\varepsilon)^{n/2} / \sqrt{\det \widehat{G}} & \text{if} \quad s = 0, \\ \sqrt{\text{const}^n / \det \widehat{G}} & \text{if} \quad s = 1, \\ \sqrt{(\text{const} \cdot \varepsilon)^n / \det \widehat{G}} & \text{if} \quad s = 2. \end{array}$$

We see that for $s = 0$, the number grows immensely as ε gets small. For $s = 1$, the number of boxes needed – while (for small ε) essentially independent of ε – may still grow exponentially with the dimension, and it is especially large for problems where the Hessian at the solution is ill-conditioned. However, the number is guaranteed to be small (for small ε) when $s = 2$.

For pure constraint satisfaction problems, a similar cluster effect is present (Schichl and Neumaier 2004), but with order reduced by one; thus the quadratic approximation property available from methods exploiting first-order information only (such as centred forms) already avoids the cluster effect, except in degenerate cases. However, especially near poorly conditioned solutions, the size of boxes that can be eliminated is significantly larger if second-order information is used (Schichl and Neumaier 2004). In the case of nonisolated solution sets, some clustering seems unavoidable, but Lyapunov–Schmidt reduction techniques (Neumaier 2001*a*) might prove useful. The problem of covering nonisolated solution sets efficiently with a small number of boxes is discussed in considerable algorithmic detail by

Xuan-Ha Vu, Sam-Haroud and Silaghi (2002, 2003) for the case of pure constraint satisfaction problems; see also Chapter 7 of the COCONUT report (Bliek *et al.* 2001).

For constrained global optimization, similar arguments as for the unconstrained case apply in a reduced manifold with the result that, in the formulas, n must be replaced by $n - a$, where a is the maximal number of constraints active at the solution, with linearly independent constraint gradients.

Clearly, to bound the overestimation over a box of diameter $O(\varepsilon)$ by $O(\varepsilon^3)$ requires that we know the Hessian up to $O(\varepsilon)$, and that we are able to bound the deviation from a quadratic model. (Actually, the above argument shows that $o(\varepsilon^2)$ is sufficient, but this still requires the knowledge of the Hessian up to $o(1)$.) Thus it is necessary to have access to second-order information. Unfortunately, in higher dimensions, no cheap method is known that bounds function values over an arbitrary narrow box of diameter $O(\varepsilon)$ close to a minimizer by $O(\varepsilon^3)$. In a single dimension, cheap methods are known; see Cornelius and Lohner (1984) and Neumaier (1990, Section 2.4). In dimension > 1, peeling methods together with Taylor expansions work with an effort that grows like $O(n^3 \cdot 3^n)$; see the discussion in Neumaier (2002, Section 5).

Fortunately, however, it turns out that by using interval Hessian matrices (which, for 3-times differentiable functions have the required $O(\varepsilon)$ accuracy: see Neumaier (1990, Section 1.4)), there are several ways to avoid the cluster problem, at least when the global minimizer is nondegenerate, *i.e.*, satisfies the second-order sufficient conditions for a local minimizer.

15.1. Using Hessian information

Explicit global Hessian information can be used, as in GlobSol (Kearfott 1996*b*) and Numerica (Van Hentenryck *et al.* 1997*b*), by interval Newton methods (see Section 11) applied to the Karush–John conditions discussed in Section 5. These may verify the existence of a unique solution of the Karush–John conditions (Theorem 5.1 and equation (5.4)) in some box around the best point found, and hence allow us to shrink that box to a single point.

Alternatively, we may use global Hessian information to verify the second-order sufficient conditions for a global minimizer given in Neumaier (1996). They apply to smooth nonlinear programs of the form

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in \mathbf{x}, \quad F(x) = 0. \end{array} \tag{15.1}$$

Thus it is necessary to introduce slack variables to rewrite general inequality constraints as equality constraints. The sufficient condition is as follows.

Theorem 15.1. Let $\widehat{x}$ be a Kuhn–Tucker point for the nonlinear program (15.1), with associated multiplier z, and let

$$y := f'(\widehat{x})^T - F'(\widehat{x})^T z, \tag{15.2}$$

$$D = \mathrm{Diag}\left(\sqrt{\frac{2|y_1|}{\overline{x}_1 - \underline{x}_1}}, \dots, \sqrt{\frac{2|y_n|}{\overline{x}_n - \underline{x}_n}}\right). \tag{15.3}$$

If, for some continuously differentiable function $\varphi : \mathbb{R}^m \to \mathbb{R}$ with

$$\varphi(0) = 0, \quad \varphi'(0) = z^T, \tag{15.4}$$

the *generalized augmented Lagrangian*

$$\widehat{L}(x) := f(x) - \varphi(F(x)) + \frac{1}{2}\|D(x - \widehat{x})\|_2^2 \tag{15.5}$$

is convex in $[u, v]$, then $\widehat{x}$ is a global solution of (15.1). Moreover, if $\widehat{L}(x)$ is strictly convex in $[u, v]$, this solution is unique.

A choice for φ that works in some neighbourhood of a strong global minimizer (*i.e.*, one in which sufficient second-order conditions for local optimality hold) is given in Neumaier (1996), together with further implementation hints. The convexity can be checked by means of interval arithmetic; see Section 11. If these conditions hold in some box, we can shrink this box to a single point.

We can use any of these techniques to construct boxes $\mathbf{y}$ that are guaranteed to contain no global minimizer unless already detected, resulting in exclusion constraints. An *exclusion constraint* is a constraint of the form

$$x \notin \mathbf{y}.$$

It can be used to reduce an arbitrary box $\mathbf{x}$ by intersecting it with $\mathbf{y}$ and taking the interval hull, which may result in a smaller box. If there were no reduction but the intersection were strictly contained in $\mathbf{x}$, then we might also want to resort to multisection: *cf.* (12.4). Interesting exclusion boxes are those that are constructed around local minimizers, since this helps fighting the cluster problem.

It is possible (though probably not most efficient) to base global optimization algorithms on exclusion methods alone; see the work of Georg *et al.* (Allgower, Erdmann and Georg 2002, Georg 2001, Georg 2002), who also give associated complexity results.

15.2. Backboxing

Whenever we have a tentative approximate global minimizer $\tilde{x}$, we try to find simultaneously a large box $\mathbf{x}$ and a tiny box $\mathbf{z}$ such that any global minimizer $\widehat{x} \in \mathbf{x}$ satisfies $\widehat{x} \in \mathbf{z}$. This allows us to use $\mathbf{x}$ as an exclusion region while $\mathbf{z}$ is stored in an output list as a box containing a putative minimizer; this is termed *backboxing*. (After terminating the branching process, these boxes need to be checked again for possible elimination.)

Since we expect that $\tilde{x}$ has a function value optimal within $O(\varepsilon)$, but knowing that this only enforces that $\tilde{x}$ has an accuracy of $O(\sqrt{\varepsilon})$ (possibly less in the case of singular Hessians), we start with a box

$$\mathbf{x} = [\tilde{x} - \sqrt{\varepsilon}u, \tilde{x} + \sqrt{\varepsilon}u]$$

for some vector u reflecting the scaling of the variables, and apply the available reduction techniques until no significant improvement results. Call the resulting box $\mathbf{z}$. If second-order techniques are used to do the box reduction, then $\mathbf{z}$ is usually a tiny box or empty.

If $\mathbf{z}$ is empty, $\tilde{x}$ was not a good approximation but we know that $\mathbf{x}$ contains no solution. If $\mathbf{z}$ is nonempty, it is likely that $\mathbf{z}$ contains a solution. Indeed this is always the case if *only* interval Newton-like reduction techniques are used and $\mathbf{z} \subseteq \operatorname{int} \mathbf{x}$. (This requires some qualifying conditions that ensure that we can verify sufficient existence conditions such as those in Neumaier (1990, Chapter 5.3–4).) Thus we may store $\mathbf{z}$ in a list of output boxes together with a flag whether existence (and possibly uniqueness) was verified.

If $\mathbf{z}$ is still a box of significant size, we must have been close to a degeneracy; splitting would probably not improve this and lead to an exponential number of boxes; thus it is preferable to put this box also in the list of output boxes to indicate that a low resolution candidate for a solution has been found. (This way of handling degeneracies is due to Kearfott (1996*a*).)

No matter what case we have been in, we always know that $\mathbf{x}$ cannot contain a solution not yet in the output list. Therefore, we may add the exclusion constraint $x \notin \mathbf{x}$ to the problem description. However, one can often make $\mathbf{x}$ even bigger. So we try recursively

$$\mathbf{x}^0 = \mathbf{x},\ \mathbf{z}^0 = \mathbf{z}, \quad \text{but} \ \ \mathbf{z}^0 = \operatorname{mid}(\mathbf{x}) \ \text{ if } \ \mathbf{z} = \emptyset,$$
$$\mathbf{x}^l = 2\mathbf{x}^{l-1} \ominus \mathbf{z}^{l-1}, \quad \mathbf{z}^l = \operatorname{reduce}(\mathbf{x}^l);$$

using the available ways of reducing $\mathbf{x}^l$, stopping when $\mathbf{z}^l \subseteq \mathbf{x}^{l-1}$ or $\mathbf{z}^l = \mathbf{x}^l$. (For the inner subtraction $\ominus$, see (11.8).) Then we have the generally stronger new exclusion constraint $x \notin \mathbf{x}^l$. (This way of generating exclusion constraints, using interval Newton methods, is due to Van Iwaarden (1986), who calls the technique *backboxing*, and is part of GlobSol (Kearfott 1996*b*).) Recent methods by Schichl and Neumaier (2004) for constructing large exclusion boxes can be combined with this iterative approach.

15.3. Finite termination

Closely related to the cluster problem is the question of finite termination, *i.e.*, whether branch and bound algorithms find (assuming exact arithmetic) a global optimizer with a finite amount of branching only. This is not easy to achieve, and in practice, most algorithms are content with working towards ε-optimality, *i.e.*, finding a (nearly) feasible point within ε of the true but unknown optimal function value.

Theoretical finite termination guarantees are available only for problems where the optimum is attained at extreme points (Al-Khayyal and Sherali 2000, Shectman and Sahinidis 1998). However, in practice, algorithms based on the explicit use of second-order interval information (either via interval Newton operators or via second-order sufficient conditions) have finite termination behaviour on problems with a nondegenerate global minimizer, and it is likely that this can be proved theoretically.

In the case of degeneracies, behaviour of branch and bound methods can become arbitrarily poor. However, the situation may improve in cases where the degeneracy can be removed by identifying and eliminating redundant constraints causing the degeneracy. To do this rigorously requires care; see Huyer and Neumaier (2004) for first results in this direction.

16. Linear and convex relaxations

One of the highly developed sides of global optimization is the use of *linear and convex relaxations* to find a lower bound for the value of the objective function, which makes it possible to discard boxes where this lower bound is larger than the function value f^{best} of the best feasible point found so far. The details are well covered in several books (Floudas 1995, 1999, Tawarmalani and Sahinidis 2002*b*), so we are brief here and only describe the basic issues and recent extensions that are not widely known.

16.1. Reformulation-linearization

McCormick (1976) introduced the notion of a factorable function (composed of finitely many unary or binary operations), and constructed non-smooth relations for such functions. Kearfott (1991), and perhaps others before him, noticed that by introducing intermediate variables, every factorable optimization problem can be rewritten in a form in which all constraints are unary, $z = \varphi(x)$, or binary, $z = x \circ y$. Independently, Ryoo and Sahinidis (1996) proposed using in place of these constraints implied linear constraints (so-called *linear relaxations*) to generate a set of linear inequalities defining a polyhedral outer approximation. Since the objective can be represented by a single variable and another constraint, this allows one to find a linear programming relaxation for arbitrary factorable optimization problems.

Linear relaxations for unary operations are easily found by a simple graphical analysis of the various elementary functions. In particular, for a convex function, the secant between the end-points of the graph is an overestimate, and any tangent is an underestimate; frequently, taking the two tangents at the end points is already quite useful. For concave functions, the reverse situation holds, and in the general case one may also need to consider bitangents, and tangent secants. Since powers can be written in terms of exp, log and the product, the only binary operations that need to be analysed are products and quotients.

Assuming that bounds $x \in \mathbf{x}$, $y \in \mathbf{y}$ for the factors are available, McCormick proposed for the product $z = xy$ the relaxations

$$\underline{y}x + \underline{x}y - \underline{x}\underline{y} \le z \le \underline{y}x + \overline{x}y - \overline{x}\underline{y},$$
$$\overline{y}x + \overline{x}y - \overline{x}\overline{y} \le z \le \overline{y}x + \underline{x}y - \underline{x}\overline{y},$$

which follow immediately from $(x - \underline{x})(y - \underline{y}) \ge 0$ and three similar inequalities. Al-Khayyal and Falk (1983) showed later that these inequalities are indeed best-possible in the sense that any other generally valid linear inequality is a consequence of these and the bound constraints. (One says they form the *convex and concave envelope*.)

For the quotient $x = z/y$, exactly the same formulas are valid with $\mathbf{x} = \mathbf{z}/\mathbf{y}$, but remarkably, one does not get the envelope in this way. For example, the following inequality, due to Zamora and Grossmann (1999), is not implied.

Proposition 16.1. Let $\mathbf{x}, \mathbf{y}, \mathbf{z}$ be nonnegative intervals. If $x \in \mathbf{x}$, $y \in \mathbf{y}$, and $z = xy \in \mathbf{z}$ then

$$xy \ge \left(\frac{z + \sqrt{\underline{z}\overline{z}}}{\sqrt{\underline{z}} + \sqrt{\overline{z}}} \right)^2. \tag{16.1}$$

Inequality (16.1) describes a convex set in the nonnegative orthant of $\mathbb{R}^3$, although the inequality itself is not convex. However, it is the prototype of a convex conic constraint (see below) and can be exploited by solvers for second-order cone programs. Therefore, adding this constraint gives a relaxation that may be tighter than the McCormick relaxation. The general formulas for the convex envelope of a quotient, derived explicitly in Tawarmalani and Sahinidis (2002*b*), are quite complicated.

Crama (1993) showed that the following bounds define the optimal convex relaxation of a product of factors bounded in $[0, 1]$.

Proposition 16.2. If $x_i \in [0, 1]$ $(i = 1 : n)$ and $z = x_1 \cdots x_n$ then

$$1 - n + \sum_{k=1}^{n} x_k \le z, \quad 0 \le z \le x_i \le 1 \ (i = 1 : n). \tag{16.2}$$

More generally, arbitrary multilinear functions have an optimal convex relaxation (the envelope) defined by finitely many linear inequalities. For this result, and for other methods for getting linear relaxations of nonconvex programs based on the *reformulation-linearization* technique: see Rikun (1997), Sherali (1997), Sherali and Tuncbilec (1995, 1997*a*, 1997*b*), Al-Khayyal, Larsen and van Voorhis (1995), and Audet, Hansen, Jaumard and Savard (2000). Also related is the *lift-and-project* technique in mixed integer linear programming; see, *e.g.*, Balas, Ceria and Cornuejols (1993).

Note that this approach using the factorable or a nearly factorable form generally results in problems with many more variables than in the original problem formulation. Nevertheless, since the resulting linear or convex programs are extremely sparse, the technique can be very useful, especially for larger problems. In particular, this is the main workhorse of the global optimization packages BARON (Tawarmalani and Sahinidis 2002*b*) and LINGO (Gau and Schrage 2004). Because linear programming solvers are currently much more reliable and faster than general convex solvers, the convex envelopes used in BARON are in fact approximated by a number of linear constraints computed adaptively with a variant of the sandwich algorithm of Rote *et al.* (Burkard, Hamacher and Rote 1992, Rote 1992).

Independently of the way a linear relaxation is produced (see below for alternatives that work without additional variables), the information in the linear relaxation can be exploited not only to get lower bounds on the objective or to eliminate a subproblem, but also to reduce the box. Cheap marginals-based range reduction techniques using Lagrangian multipliers are described in Tawarmalani and Sahinidis (2002*b*) and are implemented in BARON. Recent results of Lebbah *et al.* (2004) and Lebbah, Rueher and Michel (2002) show that the more expensive approach of minimizing and maximizing each variable with respect to a linear relaxation (which BARON 5.0 did only at the root node of the branch tree) may give a significant speed-up on difficult constraint satisfaction problems, and are now part of the default strategy in BARON 6.0.

16.2. Semidefinite relaxations

Starting quite recently, a large number of papers appeared that propose the use of *semidefinite relaxations*, or *convex conic relaxations*, to solve polynomial constraint satisfaction and global optimization problems: see, *e.g.*, Chesi and Garulli (2001), Jibetean and De Klerk (2003), Fujie and Kojima (1997), Kim and Kojima (2001, 2003), Kim, Kojima and Waki (2003), Kojima, Kim and Waki (2003), Kojima and Tuncel (2000), Lasserre (2001), Meziat (2003), Gatermann and Parrilo (2004), Parrilo (2003), Parrilo and Lall (2003), Parrilo and Sturmfels (2003). These techniques are implemented in two software packages, GloptiPoly (Henrion and Lasserre 2003*a*, 2003*b*, 2003*c*) and SOSTOOLS (Prajna *et al.* 2002). Being developed completely independently from the mainstream in global optimization, these packages do not incorporate any of the other global techniques, and hence are currently restricted to problems with few variables (say, below 20). But since they are able to solve many of these problems to global optimality without doing any branching, their combination with the other techniques, in particular with branch and bound, appears to be highly promising.

The background of these methods is that constraints of the form

$$\sum_k x_k A_k \quad \text{is positive semidefinite,} \tag{16.3}$$

where the A_k are symmetric (or complex Hermitian) matrices, so-called *semidefinite constraints*, define convex sets, and that constraints of the form

$$\|Ax - b\|_2 \le a^T x + \alpha$$

or

$$\|Ax - b\|_2^2 \le x_i x_j, \quad x_i,\, x_j \in \mathbb{R}_+,$$

so-called *second-order cone constraints*, describe convex conic sections. Problems with a linear or convex quadratic objective and an arbitrary number of such constraints (in addition to linear constraints) can be efficiently solved using interior point methods. Therefore, convex conic and semidefinite relaxations of nonlinear constraints can be efficiently exploited. Books and surveys emphasizing the nonlinear case include Alizadeh (2003), de Klerk (2002), Lobo, Vandenberghe, Boyd and Lebret (1998), Todd (2001), Vandenberghe and Boyd (1996) and Wolkowicz, Saigal and Vandenberghe (2000); for software, see the semidefinite programming homepage (Helmberg 2003) and the package SeDuMi (see SeDuMi (2001) and Sturm (1999)), on which both GloptiPoly and SOSTOOLS are based.

The basic idea behind semidefinite relaxations is the observation that, given any set of basis functions $\varphi_i(x)$ and any nonnegative weight function $w(x)$, the matrix M with components

$$M_{ik} = w(x)\varphi_i(x)\varphi_k(x)$$

is always symmetric and positive semidefinite. If the φ_i and w are polynomials then the entries of M are also polynomials, and by introducing auxiliary variables z_j for the elements of a basis of polynomials of sufficiently high degree, one can write both the entries of M and any polynomial objective or constraint as a linear combination of the z_j. The condition that M is positive semidefinite therefore gives rise to a semidefinite constraint. Possible choices for $w(x)$ can easily be made up from the constraints. Moreover, given an equality constraint, any multiple by a polynomial is another equality constraint, and given two inequality constraints $u(x) \ge 0$ and $v(x) \ge 0$, their product is again such a constraint. Thus lots of additional polynomial constraints can be generated and used. Results from algebraic geometry can then be invoked to show that infeasibility and ε-optimality can always be achieved by using sufficiently high degrees, without the need for any problem splitting. Apparently, in many cases, relatively low degrees suffice, which is fortunate since the number of intermediate variables would otherwise become excessively large. Moreover, problem symmetry can be exploited by using basis sets with corresponding symmetry properties (Gatermann and Parrilo 2004).

The conic and semidefinite relaxations produced in this way also result in problems with many more variables than in the original problem formulation, but since semidefinite relaxations are often much stronger than linear relaxations, the effort required to solve these large problems may be well spent if a subproblem is solved without the need to split it into many smaller pieces. Since problems with semidefinite constraints involving larger matrices are more expensive to solve than those with convex conic constraints, the latter are in principle preferable, but conclusive results on the best way of using or combining the various possible relaxations are not yet available.

For semidefinite relaxations of certain fractional functions see Tawarmalani and Sahinidis (2001, 2002*a*, 2002*b*).

16.3. Relaxations without extra variables

In place of introducing additional variables for nonlinear intermediate expressions, it is also possible to relax the original constraints directly. Apart from McCormick's non-smooth convex relations (McCormick 1976), which are difficult to use, this can be done in two different ways.

The first possibility is to write the constraints as a difference of convex functions (DC representation). The package αBB (see Adjiman *et al.* (1996, 1998*a*, 1998*b*), Adjiman and Floudas (1996), Androulakis *et al.* (1995)) uses DC-techniques, by separating in each inequality constraint $h(x) \le 0$ a recognizable linear, convex or concave parts from a 'general' remainder. Linear and convex parts are kept, concave parts are overestimated by secant-type constructions, and general terms are made convex by adding a nonpositive separable quadratic function. This ensures that a convex underestimating inequality results. More specifically, if $f(x)$ is twice continuously differentiable at all x in a neighbourhood of a box $\mathbf{x}$ and D is a diagonal matrix with nonnegative entries, then

$$f_{\mathrm{rel}}(x) := f(x) + \frac{1}{2}(x - \underline{x})^T D (x - \overline{x})$$

is an underestimator of $f(x)$ on the box, and the amount of underestimation is bounded by

$$|f_{\mathrm{rel}}(x) - f(x)| \le \frac{1}{8} \operatorname{rad} \mathbf{x}^T D \operatorname{rad} \mathbf{x}, \tag{16.4}$$

attained at the midpoint. (At the vertices there is no overestimation.) If the Hessian $G(x)$ lies in the interval matrix $\mathbf{G}$ for all $x \in \mathbf{x}$ (such a $\mathbf{G}$ can be found by interval evaluation of the Hessian, *e.g.*, using automatic differentiation) and all symmetric matrices in $\mathbf{G} + D$ are positive semidefinite then f_{rel} is convex. The latter condition can be checked as in Theorem 11.1; the difficulty is to choose D in such a way that this condition holds and the underestimation bound in (16.4) is kept small but the work in getting D remains reasonable (Adjiman *et al.* 1998*a*, 1998*b*). Recent, more advanced convexity-enforcing corrections are discussed in Akrotirianakis and Floudas (2004).

More general DC-techniques are treated extensively from a mostly theoretical point of view in the book by Horst and Tuy (1990); see also the overview in Tuy (1995). Apart from what is used in αBB (and described above), these techniques have not materialized in available codes; however, see, *e.g.*, Le Thi Hoai An and Pham Dinh Tao (1998) for some recent numerical results.

General techniques for recognizing convexity automatically are discussed in forthcoming work of Fourer (2003) and Maheshwari, Neumaier and Schichl (2003). Other questions related to the semidefiniteness of an interval matrix are discussed in Jaulin and Henrion (2004).

The second possibility is to use centred forms. The Frontline Interval Global Solver constructs linear enclosures based on a centred form (in fact a first-order Taylor form),

$$f(x) \in \mathbf{f} + c^T(x - z), \tag{16.5}$$

using forward propagation in an automatic differentiation-like manner, described in Kolev and Nenov (2001). Since the coefficients of the linear term in (16.5) are real numbers, this directly gives two parallel linear functions which underestimate and overestimate $f(x)$:

$$\underline{f} + c^T(x - z) \le f(x) \le \overline{f} + c^T(x - z).$$

The COCONUT environment constructs a centred form using slopes and automatic differentiation-like backward propagation, according to formulas given in Schichl and Neumaier (2003), from which linear enclosures are constructed. Indeed, given a centred form

$$f(x) = \tilde{f} + \tilde{s}^T(x - z), \quad \tilde{f} \in \mathbf{f},\ \tilde{s} \in \mathbf{s},$$

we have the linear underestimator

$$f(x) \ge \gamma + c^T(x - \underline{x}),$$

where

$$\gamma = \underline{f} + \overline{s}^T(\underline{x} - z), \quad c_i = \frac{\underline{s}_i(\overline{x}_i - z_i) - \overline{s}_i(\underline{x}_i - z_i)}{\overline{x}_i - \underline{x}_i}.$$

A similar formula provides a linear overestimator. Geometrically, the formulas amount to enclosing the double cone defined by the centred form by a pair of hyperplanes; since linear functions are separable, the formulas derived from an analysis of the univariate case can be applied componentwise.

17. Semilinear constraints and MILP

Let us call a constraint *semilinear* if, for arguments x in a *bounded* box $\mathbf{x}$, it is equivalent to a finite list of linear constraints and integer constraints; usually the latter involve additional auxiliary variables. The objective function $f(x)$ is called *semilinear* if the inequality $f(x) \le x_0$, where x_0 is an additional variable, is semilinear. A *semilinear program* is an optimization problem with a semilinear objective function and a bounded feasible domain defined by semilinear constraints only. Since we can rewrite an arbitrary global optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in C \end{array}$$

in the form

$$\begin{array}{ll} \min & x_0 \\ \text{s.t.} & x \in C,\ f(x) \le x_0, \end{array}$$

it is clear from the definition that any semilinear program can be rewritten as a mixed integer linear program by the introduction of additional variables.

The remarkable fact that every factorable optimization problem can be arbitrarily closely approximated by semilinear programs (see Section 18) implies that we can use MILP software to obtain arbitrarily good approximate solutions of factorable optimization problems. To make this observation computationally useful we need to handle two tasks.

(i) Find interesting classes of semilinear constraints and constructive procedures for translating such constraints into linear and integer constraints.

(ii) Show how to approximate factorable constraints by semilinear constraints: see Section 18.

In this section we look at task (i). This is in principle well known, but usually considered to be part of the modelling process. For good overviews of the modelling related issues see, *e.g.*, Floudas (1995, Section 7.4), Williams (1999) and (in German) Kallrath (2002). Here we simply give the underlying mathematical substance.

All linear constraints and integer constraints are trivially semilinear. A *binary constraint*

$$z \in \{0, 1\}$$

is semilinear, since it can be written in the equivalent form

$$z \in [0, 1], \quad z \in \mathbb{Z}.$$

We call a list x_K of variables constrained by

$$\sum_{k\in K} x_k = 1, \quad x_k \in \{0, 1\}\ (k \in K), \tag{17.1}$$

where K is some index set, a *binary special ordered set* (BSOS) (*cf.* Dantzig *et al.* (1958), Beale and Tomlin (1970)). Clearly, the constraint

$$x_K \quad \text{is a BSOS} \tag{17.2}$$

is also semilinear. Because (17.1) can hold only if all but one of the x_k ($k \in K$) vanish, (17.2) is equivalent to requiring that

$$x_K = e^{(k)} \quad \text{for some} \quad k, \tag{17.3}$$

where $e^{(k)}$ is the unit vector with a one in position k and zeros elsewhere. (A binary special ordered set of size two is just a pair of complementary binary variables, and one of its variables is redundant.) Note that a BSOS is a special ordered set of type 1, and can be handled efficiently by most MILP codes. Since special ordered sets, defined more generally as sets of variables such that at most one – type 1 – or two (which then must be adjacent) – type 2 – are nonzero, are ubiquitous in MILP formulations, any MILP solver has special facilities to make efficient use of special ordered sets.

Many techniques for translating semilinear constraints are consequences of the following basic result.

Theorem 17.1. Let $F_k : C_0 \to \mathbb{R}^{m_k} (k = 1, \dots, d)$ be scalar- or vector-valued functions such that

$$F_k(x) \ge \underline{F}_k \quad \text{for all} \quad x \in C_0 \tag{17.4}$$

with $\underline{F}_k \in \mathbb{R}^{m_k}$. Then there is a point $x \in C_0$ such that

$$F_1(x) \ge 0 \quad \vee \quad \cdots \quad \vee \quad F_d(x) \ge 0 \tag{17.5}$$

if and only if there are $z \in \mathbb{R}^d$ and $x \in C_0$ such that

$$z \quad \text{is a BSOS}, \tag{17.6}$$

$$F_k(x) \ge \underline{F}_k(1 - z_k) \quad \text{for all} \quad k = 1, \dots, d. \tag{17.7}$$

(The symbol $\vee$ denotes the logical operation **or**. The operation **and** is implicitly given by the comma, and we follow the convention that **and** is binding more strongly than **or**.)

Proof. If (17.5) holds then $F_k(x) \geq 0$ for some k, and $z = e^{(k)}$ satisfies (17.6) and (17.7). Conversely, if (17.6) holds then $z = e^{(k)}$ for some k, and (17.7) implies $F_k(x) \geq 0$; the other constraints in (17.7) are automatically satisfied because of (17.4). □

Note that (17.4) can always be satisfied if C_0 is bounded and the F_k are continuous.

A constraint of the form (17.5) is called a *disjunctive constraint*. The theorem implies that *linear* disjunctive constraints, where all $F_k(x)$ are affine functions of x, are semilinear if the F_k have *known*, finite lower bounds on the feasible domain (*bound qualification*), since then (17.7) consists of linear constraints. In the following, *we shall always silently assume the bound qualification*. (In practice, this is usually enforced where necessary by *ad hoc* 'big M' domain restrictions. In rigorous solvers, this is of course forbidden.)

More generally, linear disjunctive constraints of the form

$$A_1 x \in \mathbf{b}_1 \quad \vee \quad \cdots \quad \vee \quad A_d x \in \mathbf{b}_d \tag{17.8}$$

are semilinear, since we can rewrite each $A_k x \in \mathbf{b}_k$ in the form

$$\begin{pmatrix} A_k x - \underline{b}_k \\ \overline{b}_k - A_k x \end{pmatrix} \geq 0.$$

Note that we can rewrite (17.8) in the equivalent form

$$A_k x \in \mathbf{b}_k \quad \text{for some} \quad k \in \{1 : d\}. \tag{17.9}$$

Since many practically relevant constraints can be cast in the form (17.8), this makes the theorem a very useful tool for recognizing semilinear constraints and translating them into a MILP formulation. (There is also an extended literature on disjunctive programming not based on transformations to MILP; for pointers see Balas (1979), Jeroslow (1977), Sherali and Shetty (1980).)

For example, *semicontinuous* (*semi-integer*) *variables* are variables x_k constrained by

$$x_k = 0 \quad \vee \quad x_k \in \mathbf{a} \tag{17.10}$$

and

$$x_k = 0 \quad \vee \quad x_k \in \mathbf{a}, \quad x_k \in \mathbb{Z}, \tag{17.11}$$

respectively, which are semilinear constraints.

A *numerical special ordered set* (NSOS) is a vector $\lambda \in \mathbb{R}^d$ such that

$$\lambda \geq 0, \quad \sum_{k=1}^{d} \lambda_k = 1,$$

at most two λ_k are nonzero, and nonzero λ_k must have adjacent indices.

Since the latter condition can be formulated as

$$\lambda_k + \lambda_{k+1} = 1 \quad \text{for some} \quad k,$$

it is disjunctive; hence the constraint

$$x_K \text{ is an NSOS} \tag{17.12}$$

is semilinear. Note that an NSOS is a special ordered set of type 2, and can be handled efficiently by most MILP codes.

An *exclusion constraint* of the form

$$x \notin \operatorname{int} \mathbf{x}, \tag{17.13}$$

where $\mathbf{x}$ is a box, is semilinear since it is a disjunction of the constraints

$$x_k \le \underline{x}_k \quad \vee \quad x_k \ge \overline{x}_k.$$

Propositional constraints are semilinear: if x_k denotes a binary variable which has the value 1 if and only if a corresponding logical proposition P_k holds, then

$$\begin{aligned}
P_1 \vee \cdots \vee P_K \;&\text{ iff }\; x_1 + \cdots + x_K \ge 1,\\
P_1 \wedge \cdots \wedge P_K \;&\text{ iff }\; x_k = 1 \;\text{ for }\; k = 1:K,\\
P_1 \Leftrightarrow P_2 \;&\text{ iff }\; x_1 = x_2,\\
P_1 \Rightarrow P_2 \;&\text{ iff }\; x_1 \le x_2,\\
P_1 \vee \cdots \vee P_K \Rightarrow P_{K+1} \vee \cdots \vee P_L \;&\text{ iff }\; x_k \le x_{K+1} + \cdots + x_L \;\text{ for }\; k = 1:K.
\end{aligned}$$

Conditional linear constraints of the form

$$Ax \in \mathbf{a} \quad \text{if} \quad Bx < b \tag{17.14}$$

are semilinear since (17.14) is equivalent to

$$Ax \in \mathbf{a} \quad \vee \quad (Bx)_1 \ge b_1 \quad \vee \quad \cdots \quad \vee \quad (Bx)_d \ge b_d,$$

where d is the number of rows of B and b. (Conditional linear constraints with $=$ or $\le$ in place of $<$ in (17.14) are apparently not semilinear in general since their disjunctive form contains strict inequalities, which – according to our definition – are not regarded as linear constraints. However, conditional linear constraints where the condition involves only integer variables and rational coefficients are semilinear since the condition can be replaced by an equivalent strict inequality condition.)

Certain *minimum and maximum constraints* are also semilinear. A constraint of the form

$$a^T x \le \min_{i=1:d} (Ax - b)_i \tag{17.15}$$

is equivalent to the linear constraints

$$a^T x \le (Ax - b)_i \quad \text{for} \quad i = 1:d.$$

The reverse constraint

$$a^Tx \geq \min_{i=1:d}(Ax-b)_i \tag{17.16}$$

is equivalent to the linear disjunctive constraint

$$a^Tx \geq (Ax-b)_1 \quad \vee \quad \cdots \quad \vee \quad a^Tx \geq (Ax-b)_d.$$

Similarly, a constraint of the form

$$a^Tx \geq \max_{i=1:d}(Ax-b)_i \tag{17.17}$$

is equivalent to the linear constraints

$$a^Tx \geq (Ax-b)_i \quad \text{for} \quad i = 1:d,$$

and the reverse constraint

$$a^Tx \leq \max_{i=1:d}(Ax-b)_i \tag{17.18}$$

is equivalent to the linear disjunctive constraint

$$a^Tx \leq (Ax-b)_1 \quad \vee \quad \cdots \quad \vee \quad a^Tx \leq (Ax-b)_d.$$

The constraints

$$a^Tx = \min_{i=1:d}(Ax-b)_i, \tag{17.19}$$

$$a^Tx = \max_{i=1:d}(Ax-b)_i \tag{17.20}$$

are also semilinear, since they are equivalent to (17.15), (17.16) and (17.17), (17.18), respectively. In particular, *linear complementarity constraints* (Billups and Murty 2000), defined by

$$\min(a^Tx-\alpha, b^Tx-\beta) = 0 \tag{17.21}$$

are semilinear. Their MILP reformulation needs a single binary variable only since the associated BSOS has size two.

Linear complementarity constraints arise in bilevel programming (see, *e.g.*, Ben-Ayed (1993), Luo, Pang and Ralph (1996), Outrata, Kočvara and Zowe (1998), Shimizu, Ishizuka and Bard (1997), Vicente and Calamai (1994)), in which the inner optimization problem is a linear program. See, *e.g.*, Grossmann and Floudas (1987) for solving bilevel programs as mixed integer problems.

Constraints of the form

$$a^Tx - \alpha \leq |b^Tx - \beta|, \tag{17.22}$$

$$a^Tx - \alpha = |b^Tx - \beta|, \tag{17.23}$$

$$a^Tx - \alpha \geq |b^Tx - \beta| \tag{17.24}$$

are semilinear since we can write the absolute value in the form $|b^Tx - \beta| = \max(\beta - b^Tx, b^Tx - \beta)$; again a single binary variable suffices for the MILP formulation. In particular, a constraint

$$\alpha \le |x| \le \beta \tag{17.25}$$

can be modelled as

$$(\alpha + \beta)z - \beta \le x \le (\alpha + \beta)z - \alpha, \quad z \in \{0, 1\}.$$

Certain other *piecewise linear constraints* are also semilinear. Of particular interest are those of the form

$$a^Tx \le \varphi(x_i), \tag{17.26}$$

where φ is a continuous, piecewise linear function of a *single* variable with a finite number of derivative discontinuities. Let $\xi_0 < \xi_1 < \cdots < \xi_d$ be a list of *nodes* such that $x_i \in [\xi_0, \xi_d]$ and φ is linear in each interval $[\xi_{k-1}, \xi_k]$. Then

$$\varphi(\xi) = \varphi_k + \varphi_k'(\xi - \xi_k) \quad \text{for} \quad \xi \in [\xi_{k-1}, \xi_k], \tag{17.27}$$

where

$$\varphi_k = \varphi(\xi_k), \quad \varphi_k' = \frac{\varphi(\xi_k) - \varphi(\xi_{k-1})}{\xi_k - \xi_{k-1}}.$$

Therefore, (17.26) can be rewritten as a disjunction of the d constraints

$$x_i \in [\xi_{k-1}, \xi_k], \quad a^Tx \le \varphi_k + \varphi_k'(x_i - \xi_k)$$

for $k = 1, \ldots, d$. Since these are linear constraints, (17.26) is equivalent to a linear disjunctive constraint. The constraints

$$a^Tx \ge \varphi(x_i), \tag{17.28}$$

$$a^Tx = \varphi(x_i), \tag{17.29}$$

are semilinear by the same argument, with $\ge$ or $=$ in place of $\le$.

Piecewise linear constraints may also be modelled by NSOS (*cf.* (17.12)); see Beale (1988, Section 10.3) and Beale and Forrest (1976), Beale and Tomlin (1970), Burkard *et al.* (1992), Dantzig *et al.* (1958) and Tomlin (1970). Indeed, if $\varphi(x)$ is piecewise linear with nodes $\xi_{1:d}$ and corresponding function values $\varphi_k = \varphi(\xi_k)$ then we may write an arbitrary argument x as

$$x = \sum \xi_k \lambda_k, \quad \lambda \text{ is an NSOS}, \tag{17.30}$$

and find

$$\varphi(x) = \sum \varphi_k \lambda_k.$$

Therefore, if we add the semilinear constraints (17.30), we may replace each occurrence of $\varphi(x)$ by $\sum \varphi_k \lambda_k$. This even works for unbounded variables and for general separable constraints $\sum \varphi_l(x_l) \in \mathbf{a}$ with piecewise linear φ_l. Many modern MILP programs have special features that allow them to handle piecewise linear constraints using special ordered sets of type 2.

Many combinatorial constraints are semilinear. For example, *all-different constraints* of the form

$$\text{the components of } x_K \text{ are distinct integers} \tag{17.31}$$

are semilinear, since we can rewrite them as

$$x_k \in \mathbb{Z} \quad \text{for} \quad k \in K; \qquad |x_j - x_k| \geq 1 \quad \text{for } j, k \in K,\ j \neq k.$$

A *cardinality constraint*

$$\text{the number of nonzero } x_k\ (k \in K) \text{ is in } \mathbf{s}$$

is semilinear if we know that x_K is integral and nonnegative. Indeed, an equivalent condition is the existence of binary numbers z_k such that

$$\begin{aligned} z_k = 1 \quad &\text{if} \quad x_k > 0, \\ z_k = 0 \quad &\text{if} \quad x_k < 1, \\ \sum_{k \in K} z_k &\in \mathbf{s}, \end{aligned}$$

and these are semilinear constraints.

Cardinality rules (Yan and Hooker 1999), *i.e.*, constraints of the form

$$\geq j \text{ components of } x_J \text{ equal } 1 \quad \Rightarrow \quad \geq k \text{ components of } x_K \text{ equal } 1$$

for binary $x_{J \cup K}$, can clearly be written in terms of cardinality constraints and hence are semilinear, too.

18. Semilinear relaxations

The preceding results are of importance for general global optimization since every factorable global optimization problem can be approximated arbitrarily well by semilinear programs. Furthermore, these approximations can be made in a way to provide rigorous relaxations, so that solving the resulting semilinear programs after a MILP reformulation can be used to obtain lower bounds in a branch and bound scheme.

The ideas go back to Markowitz and Manne (1957) and Dantzig (1960) for approximate separable nonconvex programming using piecewise linear constraints. (A Lagrangian method by Falk and Soland (1969) gives piecewise linear relaxations, but in general, these do not yield arbitrarily good approximations.) With a trick due to Pardalos and Rosen (1987) (for the special case of indefinite quadratic programs, but not in the context of approximations) that allows one to transform multivariate quadratic expressions into separable form, everything extends easily to the semiseparable case; see (18.3) below. For indefinite quadratic programs, this is discussed in detail in Horst *et al.* (1995).

With a suitable reformulation, arbitrary factorable optimization problems (and many nonfactorable ones) can be rewritten in such a way that the objective function is linear and all constraints are either semilinear, or of the form (17.26), (17.28), (17.29) with continuous functions of a single variable. To see this, we introduce an auxiliary variable for every intermediate result; then the objective function is just a variable, hence linear, and the constraints are simple constraints or equations involving a single operation only,

$$x_k = \varphi(x_i), \tag{18.1}$$

$$x_k = x_i \circ x_j \quad (\circ \in \{+,-,*,/,\hat{\ }\}). \tag{18.2}$$

The problem formulation in terms of constraints of the form (18.1) and (18.2) together with a simple objective $\min \pm x_i$ and simple bounds (and possibly integrality constraints) is called the *ternary form* of a global optimization problem.

To find a semilinear relaxation, we note that the equations (18.1) have the form (17.29) and hence can be handled as in the previous section. The equations (18.2) are linear if $\circ \in \{+,-\}$. For $\circ = /$, we get equivalent constraints $x_i = x_k x_j$, and for $\circ = \hat{\ }$ (the power), we can rewrite the constraint $x_k = x_i^{x_j}$ as

$$y_k = x_j y_i, \quad y_k = \log x_k, \quad y_i = \log x_i.$$

(Powers with constant exponents are treated as a case of (18.1).) It remains to consider products. But $x_k = x_i x_j$ is equivalent to

$$\alpha x_i + \beta x_j = u, \quad \alpha x_i - \beta x_j = v,$$
$$w = v^2, \quad w + 4\alpha\beta x_k = u^2,$$

for arbitrary α, $\beta \neq 0$. The first two are linear constraints in x_i, x_j, u, v, and the others are of the form (17.29). This proves that the reformulation can always be done.

However, it is clear that in most cases many fewer intermediate variables need to be introduced since affine expressions $a^T x + \alpha$ can be left intact, as can all expressions depending only on a single variable. Moreover, as we shall see in a moment, quadratic and bilinear expressions can be handled more efficiently.

Therefore, it is advisable to do in a first step only those substitutions needed to transform the problem such that the new objective function

$f(x) =: F_0(x)$ and the components $F_i(x)$ $(i = 1 : m)$ of the new constraint function vector $F(x)$ are *semiseparable*, *i.e.*, of the form

$$F_i(x) = \sum_{(j,k)\in K_i} \varphi_j(x_k) + x^T H_i x + c_i^T x + \gamma_i \quad (i = 0 : m) \tag{18.3}$$

with nonlinear univariate functions φ_j and (in general extremely sparse) matrices H_i. Note that linear terms may be absorbed into the sum, and quadratic and bilinear terms into $x^T H_i x$.

In a second step, the quadratic terms are rewritten as a weighted sum of squares,

$$x^T H_i x = \frac{1}{2} \sum_{j\in J_i} d_j (r_j^T x)^2. \tag{18.4}$$

This is always possible, usually in many ways; for example, by a spectral factorization

$$H_i + H_i^T = QDQ^T, \quad D \text{ diagonal},$$

which gives

$$2x^T H_i x = (Q^T x)^T D (Q^T x) = \sum D_{kk} (Q^T x)_k^2.$$

(For numerical stability we need to take care of scaling issues, to ensure that no unavoidable cancellation of significant digits takes place.) Using (18.4) and substituting new variables for the $r_j^T x$, we see that we can achieve in this second step the *separable form*

$$F_i(x) = \sum_{(j,k)\in K_i} \varphi_j(x_k) + c_i^T x + \gamma_i \quad (i = 0 : m) \tag{18.5}$$

with increased K_i. Constraints of the form $F_i(x) \le \overline{F}_i$ are now replaced by

$$\sum_{(j,k)\in K_i} y_j + c_i^T x + \gamma_i \le \overline{F}_i,$$

$$y_j \ge \varphi_j(x_k) \quad \text{for} \quad (j,k) \in K_i,$$

and similarly for the objective function. Constraints of the form $F_i(x) \ge \underline{F}_i$ are replaced by

$$\sum_{(j,k)\in K_i} y_j + c_i^T x + \gamma_i \ge \underline{F}_i,$$

$$y_j \le \varphi_j(x_k) \quad \text{for} \quad (j,k) \in K_i.$$

Finally two-sided constraints $F_i(x) \in \mathbf{F}_i$ with finite $\mathbf{F}_i$ are replaced by

$$\sum_{(j,k)\in K_i} y_j + c_i^T x + \gamma_i \in \mathbf{F}_i,$$

$$y_j = \varphi_j(x_k) \quad \text{for} \quad (j,k) \in K_i.$$

Thus, in this third step, the required form has been achieved, and generally much more parsimoniously. (A few more variables could be saved by leaving in each nonlinear $F_i(x)$ one of the nonlinear terms unsubstituted.)

So far, no approximation has occurred; the reformulated problem is equivalent to the original one. In a final *approximation step*, constraints of the form (17.26), (17.28), (17.29) are replaced by piecewise linear constraints. If we only need an approximation (as is traditionally done (Beale and Forrest 1976)), then we use a piecewise linear interpolant to φ.

However, with only a little more work, *outer approximations* can be constructed if we have two piecewise linear approximations $\underline{\varphi}$, $\overline{\varphi}$ with the same nodes $\xi_0 < \cdots < \xi_d$, satisfying

$$\underline{\varphi}(\xi) \le \varphi(\xi) \le \overline{\varphi}(\xi) \quad \text{for} \quad \xi \in [\xi_0, \xi_d]. \tag{18.6}$$

To get $\xi_0 = \underline{x}_i$ and $\xi_d = \overline{x}_i$ we need good bounds $\mathbf{x}_i$ on x_i, which can usually be calculated by constraint propagation (see Section 14). The bounding functions $\underline{\varphi}$ and $\overline{\varphi}$ can be found by exploiting convexity properties of φ, which are well known for elementary functions and can be determined with interval analysis for factorable univariate functions. Given (18.6), the constraint (17.26) *implies* (and not only approximates) the semilinear constraints

$$x_i \in [\xi_{k-1}, \xi_k], \quad a^T x \le \overline{\varphi}_k + \overline{\varphi}_k'(x_i - \xi_k) \quad \text{for some} \quad k,$$

the constraint (17.28) implies the semilinear constraints

$$x_i \in [\xi_{k-1}, \xi_k], \quad \underline{\varphi}_k + \underline{\varphi}_k'(x_i - \xi_k) \le a^T x \quad \text{for some} \quad k,$$

and the constraint (17.29) implies the semilinear constraints

$$x_i \in [\xi_{k-1}, \xi_k], \quad \underline{\varphi}_k + \underline{\varphi}_k'(x_i - \xi_k) \le a^T x \le \overline{\varphi}_k + \overline{\varphi}_k'(x_i - \xi_k) \quad \text{for some} \quad k.$$

Moreover, by adaptively adding additional nodes we can make the gap between the bounds in (18.6) arbitrarily small, and the approximation by these semilinear constraints becomes arbitrarily good (at the cost of higher complexity, of course).

As one can see, the complexity of the resulting MILP formulation depends on the number of nonlinear operations (but in a problem-dependent fashion because of the quadratic bilinear terms), and grows linearly with the number of nodes used in the piecewise linear approximation. Hence it is an efficient technique only if the number of nonlinear operations is not too large, and the approximation not too close.

19. Other problem transformations

Linear or convex relaxations of a global optimization problem may be viewed as transformations of the problem or of its nonlinear (resp. nonconvex) constraints. There is a number of other useful transformations of constraints or groups of constraints.

19.1. General cuts

A *redundant constraint*, or simply a *cut*, is an inequality (or sometimes an equation) not in the original problem formulation that must hold for any global minimizer; if the inequality is linear, it is called a *cutting plane* (Gomory 1960). A lot is known about cutting planes in mixed integer linear programming (see, *e.g.*, Nemhauser and Wolsey (1988), Nemhauser and Wolsey (1989), Wolsey (1998)); we are here particularly interested in techniques for the smooth case.

We have already met several kinds of derived constraints that cut off part of the feasible region.

- The constraint $f(x) \leq f^{\text{best}}$ cuts off points worse than the best feasible point found so far (with function value f^{best}).
- Exclusion constraints, discussed in Section 15, cut off a region around local minimizers that do not contain any other, better minimizer.
- The linear, convex and semilinear relaxations of constraints, discussed in Sections 16 and 18, are of course special cases of cuts.

For bound-constrained indefinite quadratic programs, Vandenbussche (2003) generalized techniques for mixed integer linear programming to find cuts which lead to excellent results on this problem class.

Surely there is much more to be explored here.

19.2. Symbolic transformations

The quality of all techniques considered so far may depend strongly on the form in which a problem is posed. Symbolic techniques may be employed to change the given form into another, perhaps more advantageous form. Unfortunately, it is not clear which transformations are most valuable, and the best transformations must usually be found on a problem-specific *ad hoc* basis. A recent example of a very difficult constraint satisfaction problem in robotics that only yielded to such an approach is described by Lee *et al.* (2002). We simply list here a few of the techniques that may be of interest.

Frequently, problems involving *trigonometric variables* can be replaced by equivalent problems with only polynomial equations, using (as recommended in ALIAS-C++ (2003)) the additional constraint

$$s^2 + c^2 = 1$$

together with the substitution

$$s = \sin\varphi, \quad c = \cos\varphi, \quad s/c = \tan\varphi, \quad c/s = \cot\varphi,$$

and similar rules for trigonometric functions of half or double angles.

Techniques from *algebraic geometry* can be applied to 'solve' polynomial equations symbolically or to bring them into a special form that may be useful. In particular, *Gröbner basis* methods (see, *e.g.*, Buchberger and Winkler (1998), Faugere, Gianni, Lazard, and Mora (1993), Stetter (1997)) provide normal forms that have a triangular structure and thus allow a complete enumeration of solutions for small polynomial problems. The work grows exponentially with the number of variables. Elimination theory (see, *e.g.*, Cox, Little and O'Shea (1998), Emiris and Canny (1995), Emiris and Mourrain (1999), Jónsson and Vavasis (2001), Moller and Stetter (1995), Bondyfalat, Mourrain and Pan (2000), Mourrain and Pan (1997), Mourrain, Pan and Ruatta (2003)) provides different, often less expensive techniques for potential simplifications by the elimination of variables. The results are often expressed in terms of determinants, and their exploitation by global solution techniques is not well explored. While the matrices arising in elimination theory appear to be related to those in semidefinite relaxations of polynomial systems, the connection apparently received little attention (Datta 2001, Hanzon and Jibetean 2003).

Unfortunately, the equations resulting from completely automatic algebraic techniques are often numerically unstable. If this is the case, function evaluations need either rational arithmetic or higher precision. In such cases, interval evaluation including their refinements suffer from excessive cancellation and provide only very weak global information. It would be very desirable to have flexible tools that do only partial elimination but provide stable reduced equations of some sort.

Hanzon and Jibetean (2003) apply these techniques to find the global minimizer of multivariate polynomials, giving attention also to the case where the minimum is achieved at infinity.

Automatic differentiation (Berz, Bischof, Corliss and Griewank 1996, Coleman and Verma 1998, Griewank 1989, 1991, 1992, Griewank and Corliss 1991) is a now classical technique for obtaining high-quality derivatives analytically and cheaply by transforming a program for function evaluation into a program for the evaluation of derivatives. This technique can be applied directly to create the Karush–John optimality conditions (Theorem 5.1 and equation (5.4)) as additional constraints for constraint propagation or for verifying optimality, as is done in the COCONUT environment (*cf.* Section 22) and (with a weaker form of (5.4)) in Numerica (Van Hentenryck *et al.* 1997*b*) and GlobSol (Kearfott 1996*b*).

On the other hand, the automatic differentiation techniques can also be adapted to provide evaluations of interval derivatives, slopes (Bliek (1992, 1997), with improvements in Schichl and Neumaier (2003)), linear enclosures (Nenov and Fylstra 2003), and second-order slopes (Kolev 1997).

20. Rigorous verification and certificates

The reliability of claimed results is the most poorly documented aspect of current global optimization software. Indeed, as was shown by Neumaier and Shcherbina (2004), even famous state-of-the-art solvers like CPLEX8.0 (and many other commercial MILP codes) may lose an integral global solution of an innocent-looking mixed integer linear program. In our testing of global solvers within the COCONUT project we noticed many other cases where global solutions were lost or feasible problems were declared infeasible, probably because of ill-conditioned intermediate calculations that lead to rounding errors not covered by the built-in tolerances.

For the solution of precise mathematical problems (such as the Kepler problem (Hales 1998)), but also for safety-critical optimization problems, it is necessary to have a complete mathematical guarantee that the global minimizer has been found. This requires special attention since numerical computations are affected by rounding errors. Fortunately, interval arithmetic, if performed with directed (outward) rounding, is able to give mathematical guarantees even in the presence of rounding errors.

20.1. Rounding in the problem definition

Many problems contain floating-point constants in their formulation. Therefore, frequently, the translation of the problems into an internal format involves floating-point computations which introduce rounding errors. Unfortunately, none of the currently available modelling systems allows one to control these rounding errors or any rounding errors made in a presolve phase used to simplify the problem formulation. The rigorous solvers available (GlobSol and Numerica) have special input modes for constants, but cannot be fed with problems generated from AMPL or GAMS input (the format for most test problems in the collections available: see Section 21). It is hoped that future releases of modelling systems provide options that allow for the passing of either symbolic constants or interval-valued coefficients computed from the input, so that the exact problem, or at least nearly exact but rigorous relaxations of the exact problem, can be recovered.

20.2. Rounding in the solution process

Most current solvers simply implement algorithms valid in exact arithmetic, and do not consider rounding errors, except by allowing for certain nonrigorous *ad hoc* tolerances in testing feasibility.

On the other hand, certain solvers (in the above list of solvers, GlobSol and Numerica) do only rigorous computations – by enclosing all numbers in intervals accounting for the rounding errors. However, they do not make use of convexity arguments leading to linear or convex programs that can be solved by local techniques, and hence have a competitive disadvantage for numerous problems. The main reason seems to be that, until recently, making linear (or convex) programming rigorous was very expensive compared with the traditional approximate approach, and time-consuming to implement.

Neumaier and Shcherbina (2004) showed that it is possible to certify the results of linear optimization problems with finite bounds by simple pre- and post-processing, without having to modify the solvers. Jansson (2003, 2004*b*) extended this to the case of unbounded variables (where only a little more work is needed unless a large number of unbounded variables is present), and Jansson (2004*a*) extended the approach further to the case of convex programs.

The availability of these cheap, easy-to-use methods for certifying the results of linear and convex optimization programs is likely to change this in the near future. First results in this direction are presented by Lebbah *et al.* (2002, 2004), who report rigorous results for a combination of constraint propagation, interval Newton and linear programming methods that significantly outperform other rigorous solvers (and also the general-purpose solver BARON) on a number of difficult constraint satisfaction problems.

20.3. Certification of upper bounds

Apart from controlling rounding errors in the computation of bounds, care must also be taken in using objective function values as upper bounds on the objective function. This is permitted only if the argument is feasible. However, especially in the presence of equality constraints, the arguments are often not exactly feasible but satisfy the constraints only within certain tolerances. In these cases, a rigorous upper bound on the objective can be obtained only if the existence of a feasible point in a small box around the approximate point can be proved rigorously, and the objective function is then evaluated at this box. This requires the use of interval Newton techniques (*cf.* Section 11). However, since there are frequently fewer equality constraints than variables, the standard existence tests must be modified to take account of this, and also to handle inequalities correctly. For a description of the main techniques currently available to certify the existence of feasible points, see, *e.g.*, Kearfott (1996*a*, 1996*b*).

20.4. Certificates of infeasibility

If an optimization problem (or a subproblem in a box generated by branch and bound) has no feasible point, a *certificate of infeasibility* can often be given allowing an easy check that this is the case. For linear constraints, the following result applies (Neumaier and Shcherbina 2004), which only uses basic interval arithmetic.

Theorem 20.1. The set of points satisfying

$$x \in \mathbf{x}, \quad Ax \in \mathbf{b} \tag{20.1}$$

is empty if and only if there is a multiplier vector y such that

$$(y^T A)\mathbf{x} \cap y^T \mathbf{b} = \emptyset. \tag{20.2}$$

Proof. If x satisfies (20.1) then the left-hand side of (20.2) contains $y^T Ax$ and hence is nonempty. Thus (20.2) implies that (20.1) cannot be satisfied. The converse is a simple consequence of the Lemma of Farkas and the fact (Neumaier 1990, Section 3.1) that $a^T\mathbf{x} = \{a^T x \mid x \in \mathbf{x}\}$. $\square$

Thus a certificate of infeasibility consists in a multiplier vector y satisfying (20.2), and is, *e.g.*, a byproduct of phase 1 of a simplex algorithm.

If there are nonlinear constraints, there are simple certificates for infeasibility of

$$x \in \mathbf{x}, \quad F(x) \in \mathbf{F},$$

such as a multiplier vector y with

$$y^T F(\mathbf{x}) \cap y^T \mathbf{F} = \emptyset, \tag{20.3}$$

where $F(\mathbf{x})$ is an interval evaluation of F at the box $\mathbf{x}$, or

$$y^T F(\xi) + \big(y^T F'(\mathbf{x})\big)(\mathbf{x} - \xi) \cap y^T \mathbf{F} = \emptyset, \tag{20.4}$$

where $F'(\mathbf{x})$ is an interval evaluation of the Jacobian F' at the box $\mathbf{x}$. Similarly, if a feasible point with objective function value f^{best} is known, then a multiplier vector y with

$$\min_{x\in\mathbf{x}} (f(x) + y^T F(x)) > f^{\text{best}} + \sup y^T \mathbf{F} \tag{20.5}$$

is a certificate that the box $\mathbf{x}$ cannot contain a global minimizer. The left-hand side can be bounded from below by interval evaluation or a centred form, giving a verifiable sufficient condition. In the linear case, (20.5) reduces to half of Theorem 20.1.

It is not difficult to show that, for convex constraints, a certificate of infeasibility can be constructed in complete analogy to the linear case. But in the nonconvex case, there is no guarantee that such a certificate exists (or can be found easily if it exists). Moreover, local solvers may fail because they are not able to find a feasible point, even if one exists. Indeed, finding a feasible point is in the latter case already a global problem that cannot be handled by local methods.

Good certificates of infeasibility of the form (20.4) are, however, available for small boxes not too close to the feasible domain. This follows from the quadratic approximation property of centred forms. A suitable multiplier vector y can be obtained in this case from the linearized problem.

Thus, in combination with branching, we can certify the nonexistence of a solution in a covering of almost all of the initial box.

20.5. Certification of global minimizers

One may also be interested in providing a minimal number of mathematically rigorous certificates that constitute a proof that some point in a narrow computed box is in fact a global minimizer. These certificates are mathematically valid only if the corresponding conditions have been evaluated in exact arithmetic; and additional safeguards are needed to ensure their validity in finite precision arithmetic. Virtually nothing has been done so far with regard to this problem.

21. Test problems and testing

An important part of the development of global optimization software is the careful testing of proposed methods.

For useful test problem collections, see, *e.g.*, The COCONUT Benchmark (2002), Floudas *et al.* (1999), GLOBAL Library (2002), Huyer and Neumaier (1999), Janka (2000), Jansson and Knüppel (1995) and Walster, Hansen and Sengupta (1985). In particular, Huyer and Neumaier (1999) contains a test suite containing the traditional global optimization test set of low-dimensional problems by Dixon and Szegő (1975), together with test results for DIRECT, MCS, and many incomplete global optimization methods. Janka (2000) (see also Khompatraporn, Pintér and Zabinsky (2004), Mongeau, Karsenty, Rouz and Hiriart-Urruty (2000)) contains a comparison of stochastic global optimization routines on a large number of low-dimensional test problems from different sources, and Jansson and Knüppel (1995) and Walster *et al.* (1985) contain test results for some interval methods on a large number of low-dimensional test problems. Testing in higher dimensions has been much more limited, although this is about to change.

The documentation and availability of test problems has been considerably simplified by coding them in one of the widely used modelling languages. AMPL (Fourer *et al.* 1993) and GAMS (GAMS World 2003) are two flexible and convenient algebraic modelling languages enabling rapid prototyping and model development. They are of widespread use in the optimization community, as attested by the large number of existing interfaces with state-of-the-art optimization solvers.

The recent *Handbook of Test Problems in Local and Global Optimization* (Floudas *et al.* 1999) contains a large collection of test problems for local and global optimization problems, both academic and from real applications. (Unfortunately, the book contains a significant number of inaccuracies; see Shcherbina (2002).) The algebraic test problems of this collection are available in the GAMS modelling language, and the differential-algebraic problems are supplied in the MINOPT modelling language. All test problems can be downloaded from the *Handbook*'s web site. A recent web site by GAMS World (GLOBAL Library 2002) started collecting a library, GlobalLib, of real-life global optimization problems with industrial relevance, coded in GAMS, but currently most problems on this site are without computational results. 131 algebraic test problems from the *Handbook* are all included and constitute about a third of the 397 test problems currently available at GlobalLib.

Test problems for local optimization should also pass global optimization solvers; the traditional test set for low-dimensional unconstrained problems is that by Moré, Garbow and Hillstrom (1981), with optional bounds from Gay (1984). A number of these problems have in fact several local minimizers and are therefore global optimization problems. Bob Vanderbei maintains a large collection of AMPL files for constrained nonlinear optimization problems (Vanderbei 2000) from practical applications; also included are the major part of the CUTE collection and the more academic but useful low-dimensional problem collection of Hock and Schittkowski (1981).

The COCONUT Benchmark (2002) (*cf.* Shcherbina *et al.* (2003)) is a collection of nearly 1300 AMPL models, containing the CUTE part of the Vanderbei test collection, AMPL versions of the problems from GlobalLib (collection from summer 2002), and a large collection of pure constraint satisfaction problems from various places. All problems are annotated with best-known function values (or even solutions) and some statistical information such as the number of variables and constraints.

The COCONUT project has extensive test results on its benchmark for a number of solvers, made public on the COCONUT web site. Extensive benchmarking results for *local* optimization by Mittelmann (2002) are also available online. See also Dolan and Moré (2000, 2001).

Bussiek, Drud, Meeraus and Pruessner (2003) report on obtaining reliable and repeatable comparisons in global optimization. The ACM Transaction of Mathematical Software (TOMS 2002) recommends considering advice in Johnson (2002) for performing computational experiments. The Mathematical Programming Society has guidelines (Jackson, Boggs, Nash and Powell 1990/91) for reporting results of computational experiments based on Crowder, Dembo and Mulvey (1979), Greenberg (1990) and Ratliff and Pierskalla (1981). See also Barr *et al.* (1995).

22. The COCONUT environment

This survey is part of an attempt to integrate various existing complete approaches to global optimization into a uniform whole. This is the goal of the COCONUT project, sponsored by the European Union. The COCONUT consortium provides on its homepage (COCONUT 2001) a modular solver environment for nonlinear global optimization problems with an open-source kernel, which can be expanded by commercial and open-source solver components (inference engines). The following information is taken from Schichl (2004).

The application programming interface (API) of the COCONUT environment is designed to make the development of the various module types independent of each other and independent of the internal model representation. It is a collection of open-source C++ classes protected by the LGPL license model (GNU Lesser General Public License 1999), so that it can be used as part of commercial software. It uses the FILIB++ (Lerch *et al.* 2001) library for interval computations and the matrix template library MTL (Siek and Lumsdaine 1999) for the internal representation of various matrix classes. Support for dynamic linking relieves the user from recompilation when modules are added or removed. In addition, it is designed for distributed computing, and will probably be developed further (in the years after the end of the COCONUT project) to support parallel computing as well.

The solution algorithm is an advanced branch and bound scheme which proceeds by working on a set of search nodes, each representing a subproblem of the optimization problem to be solved. A complete optimization problem is always represented by a *single* directed acyclic graph (DAG). The vertices of the graph represent operators similar to computational trees. Constants and variables are sources, objective and constraints are sinks of the DAG. This DAG is optimally small in the sense that it contains every subexpression of objective function and constraints only once.

For expression DAGs, special forward and backward evaluators are provided to allow function evaluation and automatic differentiation-like tasks. Currently implemented are real function values, function ranges, gradients (real, interval), and slopes. In the near future, evaluators for Hessians (real, interval) and second-order slopes (see, *e.g.*, Schichl and Neumaier (2004)) will be provided as well.

A strategy engine is the main part of the algorithm. It makes decisions, directs the search, and invokes the various modules. The strategy engine consists of the logic core ('search') which is essentially the main solution loop, special decision makers for determining the next action at every point in the algorithm. It calls management modules, report modules, and inference engines in a sequence defined by programmable search strategies.

The strategy engine can be programmed using a simple strategy language based on the language Python (in which, for instance, most of the interface of the web search engine Google is written). Since it is interpreted, (semi-)interactive and automatic solution processes are possible, and even debugging and single-stepping of strategies is supported. The language is object-oriented, provides dynamically typed objects, and is garbage-collecting. These features make the system easily extendable. Furthermore, the strategy engine manages the search graph and the search database. The strategy engine uses a component framework to communicate with the inference engines. This makes it possible to launch inference engines dynamically (on need) to avoid memory overload. Since the strategy engine is itself a component, even multilevel strategies are possible.

Corresponding to every type of problem change, a class of inference engines is designed: model analysis (*e.g.*, find convex part), model reduction (*e.g.*, pruning, fathoming), model relaxation (*e.g.*, linear relaxation), model splitting (*e.g.*, bisection), model glueing (*e.g.*, undo excessive splitting), computing of local information (*e.g.*, probing, local optimization). Several state-of-the-art techniques are already provided, including interfaces to local nonlinear solvers and linear programming systems, rigorous point verifiers, exclusion box generators, constraint propagation, linear relaxations, a splitter, and a box-covering module.

Changes suggested by an inference engine and approved of by the strategy engine are performed by appropriate management modules. Report modules produce human-readable or machine-readable files for checkpointing or external use.

The open design of the solver architecture, and its extensibility to include both open source modules and commercial programs, was chosen in the hope that the system will be a unique platform for global optimization in the future, serving the major part of the community, bringing their members closer together.

Several researchers and companies from outside the COCONUT project have already agreed to complement our efforts in integrating the known techniques by contributing to the COCONUT environment.

23. Challenges for the near future

We end the survey by listing a number of challenges that researchers in global optimization, and those working on software systems and support, may wish to face to improve the state of the art.

(1) Ensuring reliability is perhaps the most pressing issue. While in theory essentially all techniques discussed here can be made fully rigorous, many of them with little computational overhead (see Section 20), only very few solvers do this. As a result, even otherwise excellent solvers (such as CPLEX 8.0 for linear mixed integer problems) occasionally lose the solution and give completely misleading results without warning, and global solvers based on these inherit the problems unless properly safeguarded. Safe bounds can guard against all errors due to finite precision arithmetic. Programming bugs are another possible source of loss of solutions, and can be discovered only through extensive testing on benchmarking suites with known solutions.

Also under the reliability heading are improvements relevant for computer-assisted proofs, especially the documentation of certificates that give a short and complete proof (that can be checked independently) that the solution is indeed correct.

(2) Better compiler (or even hardware) support for automatic differentiation, outward rounded interval arithmetic, and related techniques (Schichl and Neumaier 2003) based on computational graphs would significantly simplify its use in global optimization codes, and probably speed up the programs. (Code optimization would, however, need to provide an option that ensures that simplifications are only performed if they are mathematically safe even in finite precision arithmetic.) The SUN FORTE compiler (Walster 2000) already supports interval arithmetic. NAG (Cohen, Naumann and Riehme 2003, NAG 2003) is investigating the possible integration of automatic differentiation capabilities into its Fortran 95 compiler.

(3) Unbounded variables are perhaps the dominant reason for failure of current complete global optimization codes on problems with few variables. Unless the initial constraint propagation phase provides useful finite bounds, interval estimates are frequently meaningless since calculations with unbounded intervals rarely generate tight enclosures. Thus the bounding part of the search remains weak, and an excessive number of boxes is generated. Better techniques for handling problems with unbounded variables are therefore highly desirable. For unconstrained polynomial problems see Hanzon and Jibetean (2003).

(4) Unconstrained problems and bound-constrained problems in higher dimensions are harder for current solvers than highly constrained ones, since the lack of constraints gives little basis for attack with the known methods.

In particular, current complete solvers are quite slow on many nonzero residual least squares problems. Until better techniques become available, users should take advantage of available freedom in modelling by providing as many constraints as possible, *e.g.*, by adding for a least squares problem $\min \|F(x)\|_2^2$ the additional constraints $F(x) \in \mathbf{F}$ for some reasonable box $\mathbf{F}$. While this may change the solution, it might be fully adequate for the application.

(5) Integrating techniques from mixed integer and semidefinite programming into the current solver frameworks appears to be a promising direction. Work in this direction has begun at various places, and it is already apparent that we can expect major improvements in speed.

(6) Problems with symmetries have many solutions that differ only in trivial rearrangements, sign changes, *etc.* However, it is not easy to avoid finding the solutions repeatedly or having to exclude repeatedly regions equivalent under symmetries. There are significant applications in cluster optimization (The Cambridge Cluster Database 1997), packing problems, and the optimal design of experiments (Sloane 2003). A recent paper by Margot (2003) handles the integer linear programming case, and Gatermann and Parrilo (2004) address the case of polynomial systems.

(7) The representation of nonisolated solution sets is another challenge where papers are slowly forthcoming (*e.g.*, Xuan-Ha Vu *et al.* (2002, 2003) for the case of continuous constraint satisfaction problems) and which has important applications in the modelling of devices with uncertain parameters or flexible parts (Neumaier 2003*b*). A related problem is that of parametric global optimization, where the same parametrized problem needs to be solved in dependence on a few parameters. (The result is then a function of these parameters instead of a single solution.) Apart from some discussion in Guddat *et al.* (1990), very little seems to have been done in this area.

(8) Problems with severe dependence among the variables have poor interval extensions and hence create difficulties for complete solvers. This applies in particular to problems containing nested functions $f(x)$ such as those arising from volume-preserving discrete dynamics, where $f(x) = f_n(x)$ with $f_1(x) = \varphi(x)$, $f_n(x) = \varphi(f_{n-1}(x))$, which suffer from a severe wrapping effect (Nedialkov and Jackson 2001, Neumaier 1993), and problems involving determinants of matrices of size > 3. Taylor methods (Makino and Berz 2003, Neumaier 2002) and reformulation techniques might help overcome these problems.

(9) Differential constraints are not of the factorable form that is the basis of all current global solvers. But they arise in optimal control problems, and it is well known that many of these (especially in space mission design and in chemical engineering) have multiple minima and hence would need a global optimization approach. Recently, some approximate methods (Esposito and Floudas 2000, Meyer, Floudas and Neumaier 2002) and a complete method (Papamichail and Adjiman 2002) (for the inverse monotone case) have become available, though these work at present only in very low dimensions.

(10) Constraints involving integrals are also not factorable; so far, they have received no attention in a global optimization context. They arise naturally in many stochastic optimization problems defined in terms of continuous random variables, since expectations or probabilities involving these are given by integrals. Examples are probabilistic safety factors in engineering and value at risk in finance.

(11) Large-scale problems are obviously hard owing to their size and the worst-case exponential behaviour of branch and bound algorithms. However, as in many combinatorial optimization problems, there may be many large-scale problems that are tractable if their problem structure is exploited. Extending the current methods to take advantage of such structure would make them much more widely applicable. Recent work of Boddy and Johnson (2003), who solved to completion large quadratic constraint satisfaction problems arising in oil refinery, including one with 13 711 variables, 17 892 constraints (of which 2 696 were nonlinear) gives rise to optimism.

All these problems show that much remains to be done and that we can expect further progress in the future.

Acknowledgements

I want to thank the Mathematics Department of the University of Colorado at Denver, and in particular Weldon Lodwick, for the opportunity to give there (in April/May 2001) a course with the same title, and the Mathematics and Computer Science Division of the Argonne National Laboratory, in particular Sven Leyffer, for the invitation to present my vision of the current techniques and challenges in global optimization in a lecture within the Global Optimization Theory Institute (2003).

I also want to thank Hermann Schichl for many discussions and for comments on earlier versions of this survey, and Christian Jansson and Nick Sahinidis for additional comments that improved the paper.

This survey is part of work done in the context of the COCONUT project (COCONUT 2001) sponsored by the European Union, with the goal of integrating various existing complete approaches to global optimization into a uniform whole (*cf.* Section 22). Funding by the European Union under the IST Project Reference Number IST-2000-26063 within the FET Open Scheme is gratefully acknowledged. A preliminary version of the survey was contained in Chapter 4 of the unpublished COCONUT report (Bliek *et al.* 2001).

REFERENCES

E. H. L. Aarts and J. K. Lenstra, eds (1997), *Local Search in Combinatorial Optimization*, Wiley, Chichester.

C. S. Adjiman, I. P. Androulakis, C. D. Maranas and C. A. Floudas (1996), 'A global optimization method αBB for process design', *Comput. Chem. Engin.* **20**, 419–424.

C. S. Adjiman, S. Dallwig, C. A. Floudas and A. Neumaier (1998*a*) 'A global optimization method, αBB, for general twice-differentiable constrained NLPs, I: Theoretical advances', *Comput. Chem. Engin.* **22**, 1137–1158.

C. S. Adjiman, I. P. Androulakis and C. A. Floudas (1998*b*), 'A global optimization method, αBB, for general twice-differentiable constrained NLPs, II: Implementation and computational results', *Comput. Chem. Engin.* **22**, 1159–1179.

C. S. Adjiman and C. A. Floudas (1996), 'Rigorous convex underestimators for general twice-differentiable problems', *J. Global Optim.* **9**, 23–40.

I. G. Akrotirianakis and C. A. Floudas (2004), 'A new class of improved convex underestimators for twice continuously differentiable constrained NLPs', *J. Global Optim.*, in press.

ALIAS-C++ (2003), *A C++ Algorithms Library of Interval Analysis for Equation Systems, Version 2.2*. User manual: `manual-alias-C++2.2.ps` at `www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS-C++/`

F. Alizadeh (2003), Semidefinite and second order cone programming foundations, algorithms and applications. Slides: `tutorialProsper.pdf` at `www.ima.umn.edu/talks/workshops/3-11.2003/alizadeh/`

F. A. Al-Khayyal and J. E. Falk (1983), 'Jointly constrained biconvex programming', *Math. Oper. Res.* **8**, 273–286.

F. A. Al-Khayyal and H. D. Sherali (2000), 'On finitely terminating branch-and-bound algorithms for some global optimization problems', *SIAM J. Optim.* **10**, 1049–1057.

F. A. Al-Khayyal, C. Larsen and T. van Voorhis (1995), 'A relaxation method for nonconvex quadratically constrained quadratic programs', *J. Global Optim.* **6**, 215–230.

E. Allgower, M. Erdmann and K. Georg (2002), 'On the complexity of exclusion algorithms for optimization', *J. Complexity* **18**, 573–588.

Le Thi Hoai An and Pham Dinh Tao (1998), 'A branch-and-bound method via DC optimization algorithm and ellipsoidal technique for box constrained nonconvex quadratic programming problems', *J. Global Optim.* **13**, 171–206.

E. D. Anderson and K. D. Anderson (1995), 'Presolving in linear programming', *Math. Program.* **71**, 221–245.

I. P. Androulakis, C. D. Maranas and C. A. Floudas (1995), 'αBB: A global optimization method for general constrained nonconvex problems', *J. Global Optim.* **7**, 337–363.

K. Anstreicher (2003), 'Recent advances in the solution of quadratic assignment problems', *Math. Program. B* **97**, 27–42.

C. Audet, P. Hansen, B. Jaumard and G. Savard (2000), 'A branch and cut algorithm for nonconvex quadratically constrained quadratic programming', *Math. Program.* **87**, 131–152.

A. B. Babichev, O. B. Kadyreva, T. P. Kashevarova, A. S. Leshchenko, and A. L. Semenov (1993), 'UniCalc: A novel approach to solving systems of algebraic equations', *Interval Computations* **3**, 29–47.

V. Balakrishnan and S. Boyd (1992), Global optimization in control system analysis and design, in *Control and Dynamic Systems: Advances in Theory and Applications*, Vol. 53 (C. T. Leondes, ed.), Academic Press, New York, pp. 1–56.

E. Balas (1979), 'Disjunctive programming', *Ann. Discrete Math.* **5**, 3–51.

E. Balas, S. Ceria and G. Cornuejols (1993), 'A lift-and-project cutting plane algorithm for mixed 0–1 programs', *Math. Program. A* **58**, 295–323.

R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. R. Stewart (1995), 'Designing and reporting on computational experiments with heuristic methods', *J. Heuristics* **1**, 9–32.

E. M. L. Beale (1979), Branch and bound methods for mathematical programming systems, *Ann. Discrete Math.* **5** 201–219.

E. M. L. Beale (1988), *Introduction to Optimization*, Wiley, Chichester.

E. M. L. Beale and J. J. H. Forrest (1976), 'Global optimization using special ordered sets', *Math. Program.* **10**, 52–69.

E. M. L. Beale and J. A. Tomlin (1970), Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables, in *OR 69: Proc. Fifth Int. Conf. Oper. Res.* (J. Lawrence, ed.) Tavistock Publications, London, pp. 447–454.

R. W. Becker and G. V. Lago (1970), A global optimization algorithm, in *Proc. 8th Allerton Conf. Cicuits Systems Theory* (Monticello, Illinois, 1970), pp. 3–12.

O. Ben-Ayed (1993), 'Bilevel linear programming', *Comput. Oper. Res.* **20**, 485–501.

F. Benhamou and W. J. Older (1997), 'Applying interval arithmetic to real, integer, and boolean constraints', *J. Logic Program.* **32**, 1–24.

F. Benhamou, D. McAllister and P. Van Hentenryck (1994), CLP(intervals) revisited, in *Proc. International Symposium on Logic Programming*, MIT Press, Ithaca, NY, pp. 124–138.

S. Berner (1996), 'Parallel methods for verified global optimization: Practice and theory', *J. Global Optim.* **9**, 1–22.

M. Berz, C. Bischof, G. Corliss and A. Griewank, eds (1996), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia.

A. C. Billups and K. G. Murty (2000), 'Complementarity problems', *J. Comput. Appl. Math.* **124**, 303–318.

R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg and R. Wunderling (2000), MIP: Theory and practice: Closing the gap, in *System Modelling and Optimization: Methods, Theory and Applications* (M. J. D. Powell and S. Scholtes, eds), Kluwer, Dordrecht, pp. 19–49.

M. Björkman and K. Holmström (2000), 'Global optimization of costly nonconvex functions using radial basis functions', *Optim. Eng.* **1**, 373–397.

C. Bliek (1992), Computer methods for design automation, PhD thesis, Department of Ocean Engineering, MIT.

C. Bliek (1997), 'Fast evaluation of partial derivatives and interval slopes', *Reliable Comput.* **3**, 259–268.

C. Bliek, P. Spellucci, L. N. Vicente, A. Neumaier, L. Granvilliers, E. Monfroy, F. Benhamou, E. Huens, P. Van Hentenryck, D. Sam-Haroud and B. Faltings (2001), *Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art.* A progress report of the COCONUT project: `www.mat.univie.ac.at/~neum/glopt/coconut/StArt.html`

C. Bliek, C. Jermann and A. Neumaier, eds (2003), *Global Optimization and Constraint Satisfaction*, Vol. 2861 of *Lecture Notes in Computer Science*, Springer, Berlin.

M. Boddy and D. Johnson (2003), A new method for the global solution of large systems of continuous constraints, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 142–156.

C. G. E. Boender, A. H. G. Rinnooy Kan, G. T. Timmer and L. Stougie (1982), 'A stochastic method for global optimization', *Math. Program.* **22**, 125–140.

I. M. Bomze, T. Csendes, R. Horst and P. M. Pardalos, eds (1996), *Developments in Global Optimization*, Kluwer, Dordrecht.

D. Bondyfalat, B. Mourrain and V. Y. Pan (2000), 'Solution of a polynomial system of equations via the eigenvector computation', *Linear Algebra Appl.* **319**, 193–209.

B. Buchberger and F. Winkler (1998), *Groebner Bases: Theory and Applications*, Cambridge University Press.

R. E. Burkard, H. Hamacher and G. Rote (1992), 'Sandwich approximation of univariate convex functions with an application to separable convex programming', *Naval Res. Logist.* **38**, 911–924.

J. V. Burke (1991), 'An exact penalization viewpoint of constrained optimization', *SIAM J. Control Optim.* **29**, 968–998.

M. R. Bussiek and A. S. Drud (2001), SBB: A new solver for mixed integer nonlinear programming. Slides: `www.gams.com/presentations/or01/sbb.pdf`
SBB user manual (2002): `www.gams.com/solvers/sbb.pdf`

M. R. Bussiek, A. S. Drud, A. Meeraus and A. Pruessner (2003), Quality assurance and global optimization, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 223–238.

The Cambridge Cluster Database (1997), web document:
`brian.ch.cam.ac.uk/CCD.html`

E. Carrizosa, P. Hansen and F. Messine (2004), 'Improving interval analysis bounds by translations', *J. Global Optim.*, to appear.

H. M. Chen and M. H. van Emden (1995), Adding interval constraints to the Moore–Skelboe global optimization algorithm, in *Extended Abstracts of APIC'95, International Workshop on Applications of Interval Computations* (V. Kreinovich, ed.), *Reliable Comput.* (Supplement), pp. 54–57.

G. Chesi and A. Garulli (2001), On the characterization of the solution set of polynomial systems via LMI techniques, in *Proc. of the European Control Conference ECC 2001, Porto (Portugal), September 4–7, 2001*, pp. 2058–2063.

J. G. Cleary (1987), 'Logical arithmetic', *Future Computing Systems* **2**, 125–149.

COCONUT (2001), Continuous constraints: Updating the technology. Web site:
`www.mat.univie.ac.at/~neum/glopt/coconut.html`

The COCONUT Benchmark (2002), A benchmark for global optimization and constraint satisfaction. Web document:
`www.mat.univie.ac.at/∼neum/glopt/coconut/benchmark.html`

M. Cohen, U. Naumann and J. Riehme (2003), Differentiation-enabled Fortran 95 compiler technology. Manuscript:
`www-unix.mcs.anl.gov/∼naumann/nagfm.ps`

T. F. Coleman and A. Verma (1998), 'The efficient computation of sparse Jacobian matrices using automatic differentiation', *SIAM J. Sci. Comput.* **19**, 1210–1233.

H. Cornelius and R. Lohner (1984), 'Computing the range of values of real functions with accuracy higher than second order', *Computing* **33**, 331–347.

D. Cox, J. Little and D. O'Shea (1998), *Using Algebraic Geometry*, Springer, New York.

Y. Crama (1993), 'Concave extensions for nonlinear 0–1 maximization problems', *Math. Program.* **61**, 53–60.

H. P. Crowder, R. S. Dembo and J. M. Mulvey (1979), 'On reporting computational experiments with mathematical software', *ACM Trans. Math. Software* **5**, 193–203.

T. Csendes (1988), Nonlinear parameter estimation by global optimization: Efficiency and reliability. *Acta Cybernetica* **8**, 361–370. Fortran and C code is at
`ftp://ftp.jate.u-szeged.hu/pub/math/optimization/index.html`
A Fortran 95 version by A. Miller is at
`www.mat.univie.ac.at/∼neum/glopt/contrib/global.f90`

T. Csendes and D. Ratz (1997), 'Subdivision direction selection in interval methods for global optimization', *SIAM J. Numer. Anal.* **34**, 922–938.

S. Dallwig, A. Neumaier and H. Schichl (1997), GLOPT: A program for constrained global optimization, in *Developments in Global Optimization* (I. Bomze *et al.*, eds), Kluwer, Dordrecht, pp. 19–36.

G. B. Dantzig (1960), 'On the significance of solving linear programming problems with some integer variables', *Econometrica* **28**, 30–44.

G. B. Dantzig, S. Johnson and W. White (1958), 'A linear programming approach to the chemical equilibrium problem', *Management Science* **5**, 38–43.

R. S. Datta (2001), Using semidefinite programming to minimize polynomials. Manuscript: `www.math.berkeley.edu/∼datta/ee227apaper.pdf`

E. de Klerk (2002), *Aspects of Semidefinite Programming: Interior Point Algorithms and Selected Applications*, Kluwer, Dordrecht.

L. C. W. Dixon and G. P. Szegő (1975), *Towards Global Optimization*, Elsevier, New York.

E. D. Dolan and J. J. Moré (2000), Benchmarking optimization software with COPS. Technical report ANL/MCS-246, Argonne National Laboratory:
`www-unix.mcs.anl.gov/∼more/cops`

E. D. Dolan and J. J. Moré (2001), Benchmarking optimization software with performance profiles. Technical report ANL/MCS-P861-1200, Argonne National Laboratory: `www-unix.mcs.anl.gov/∼more/cops`

M. Dorigo, V. Maniezzo and A. Colorni (1991), 'The ant system: Optimization by a colony of cooperating agents', *IEEE Trans. Systems, Man, Cyber. Part B* **26**, 29–41.

G. Dueck and T. Scheuer (1990), 'Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing', *J. Comput. Physics* **90**, 161–175.

M. A. Duran and I. E. Grossmann (1986), 'An outer approximation algorithm for a class of mixed integer nonlinear programs', *Math. Program.* **36**, 307–339.

I. Z. Emiris and J. F. Canny (1995), 'Efficient incremental algorithms for the sparse resultant and the mixed volume', *J. Symbolic Comput.* **20**, 117–149.

I. Z. Emiris and B. Mourrain (1999), 'Matrices in elimination theory', *J. Symbolic Comput.* **28**, 3–44.

T. G. W. Epperly and E. N. Pistikopoulos (1997), 'A reduced space branch and bound algorithm for global optimization', *J. Global Optim.* **11**, 287–311.

T. G. W. Epperly and R. E. Swaney (1996), Branch and bound for global NLP, Chapters 1–2 in *Global Optimization in Engineering Design* (I. E. Grossmann, ed.), Kluwer, Dordrecht.

W. R. Esposito and C. A. Floudas (2000), 'Deterministic global optimization in nonlinear optimal control problems', *J. Global Optim.* **17**, 97–126.

J. E. Falk and R. M. Soland (1969), 'An algorithm for separable nonconvex programming', *Management Sci.* **15**, 550–569.

J. C. Faugere, P. Gianni, D. Lazard, and T. Mora (1993), 'Efficient computation of zero-dimensional Groebner bases by change of ordering', *J. Symbolic Comput.* **16**, 329–344.

A. Fiacco and G. P. McCormick (1990), *Sequential Unconstrained Minimization Techniques*, Vol. 4 of *Classics in Applied Mathematics*, SIAM, Philadelphia.

R. Fletcher and S. Leyffer (1994), 'Solving mixed integer nonlinear programs by outer approximation', *Math. Program.* **66**, 327–349.

R. Fletcher and S. Leyffer (1998), 'Numerical experience with lower bounds for MIQP branch-and-bound', *SIAM J. Optim.* **8**, 604–616.

C. A. Floudas (1995), *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*, Oxford University Press.

C. A. Floudas (1997), Deterministic global optimization in design, control, and computational chemistry, in *Large Scale Optimization with Applications, Part II: Optimal Design and Control* (L. T. Biegler *et al.*, eds), Springer, New York, pp. 129–184.
`ftp://titan.princeton.edu/papers/floudas/ima.pdf`

C. A. Floudas (1999), *Deterministic Global Optimization: Theory, Algorithms and Applications*, Kluwer, Dordrecht.

C. A. Floudas and P. M. Pardalos (1990), *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Vol. 455 of *Lecture Notes in Computer Science*, Springer, Berlin.

C. A. Floudas and P. M. Pardalos, eds (1992), *Recent Advances in Global Optimization*, Princeton University Press.

C. A. Floudas and P. M. Pardalos, eds (1996), *State of the Art in Global Optimization*, Kluwer, Dordrecht.

C. A. Floudas and P. M. Pardalos, eds (2000), *Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches*, Kluwer, Dordrecht.

C. A. Floudas, P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gümüs, S. T. Harding, J. L. Klepeis, C. A. Meyer and C. A. Schweiger (1999), *Handbook of Test Problems in Local and Global Optimization*, Kluwer, Dordrecht. `titan.princeton.edu/TestProblems/`

F. Forgó (1988), *Nonconvex Programming*, Akadémiai Kiadó, Budapest.

S. Forrest (1993), 'Genetic algorithms: Principles of natural selection applied to computation', *Science* **261**, 872–878.

R. Fourer (2003), Convexity recognition. In preparation.

R. Fourer, D. M. Gay and B. W. Kernighan (1993), *AMPL: A Modeling Language for Mathematical Programming, Duxbury Press*, Brooks/Cole Publishing Company. `www.ampl.com/cm/cs/what/ampl/`

T. Fujie and M. Kojima (1997), 'Semidefinite programming relaxation for nonconvex quadratic programs', *J. Global Optim.* **10**, 367–380.

J. M. Gablonsky and C. T. Kelley (2001), 'A locally-biased form of the DIRECT algorithm', *J. Global Optim.* **21**, 27–37.

GAMS World (2003), web document: `www.gamsworld.org`

M. R. Garey and D. S. Johnson (1979), *Computers and Intractability*, Freeman, San Francisco, CA.

K. Gatermann and P. Parrilo (2004), 'Symmetry groups, semidefinite programs, and sum of squares', math.AC/0211450, *J. Pure Appl. Algebra*, to appear. `www.zib.de/gatermann/publi.html`

C.-Y. Gau and L. Schrage (2004), Implementation and testing of a branch-and-bound based method for deterministic global optimization, in *Proceedings of the Conference Frontiers in Global Optimization, Santorini, Greece, June 2003* (C. A. Floudas and P. M. Pardalos, eds), Kluwer, Dordrecht.

D. M. Gay (1984), A trust-region approach to linearly constrained optimization, in *Numerical Analysis* (D. F. Griffiths, ed.), Vol. 1066 of *Lecture Notes in Mathematics*, Springer, Berlin, pp. 72–105.

K. Georg (2001), 'Improving the efficiency of exclusion algorithms', *Adv. Geom.* **1**, 193–210.

K. Georg (2002), 'A new exclusion test', *J. Comput. Appl. Math.* **152**, 147–160.

P. E. Gill and W. Murray (1974), 'Newton type methods for unconstrained and linearly constrained optimization', *Math. Program.* **7**, 311–350.

Global (and Local) Optimization (1995), web site: `www.mat.univie.ac.at/~neum/glopt.html`

GLOBAL Library (2002), web document: `www.gamsworld.org/global/globallib.htm`

Global Optimization Theory Institute (2003), Argonne National Laboratory, September 8–10: `www-unix.mcs.anl.gov/~leyffer/gotit/`

F. Glover (1989/90), 'Tabu Search'. Part 1: *ORSA J. Comput.* **1** (1989), 190–206. Part 2: *ORSA J. Comput.* **2** (1990), 4–32.

F. Glover and M. Laguna (1997), *Tabu Search*, Kluwer, Boston.

GNU Lesser General Public License (1999), web document: `www.gnu.org/copyleft/lesser.html`

R. E. Gomory (1960), *An Algorithm for the Mixed Integer Problem*, RM-2597, The Rand Corporation.

L. Granvilliers (2001), 'Progress in the solving of a circuit design problem', *J. Global Optim.* **20**, 155–168.

P. Gray, W. Hart, L. Painton, C. Phillips, M. Trahan and J. Wagner (1997), A survey of global optimization methods. Web document: `www.cs.sandia.gov/opt/survey/`

H. J. Greenberg (1990), 'Computational testing: Why, how, and how much', *ORSA J. Comput.* **2**, 94–97.

A. Griewank (1989), On automatic differentiation, in *Mathematical Programming* (M. Iri and K. Tanabe, eds), KTK Scientific Publishers, Tokyo, pp. 83–107.

A. Griewank (1991), Automatic evaluation of first and higher-derivative vectors, in *International Series of Numerical Mathematics*, Vol. 97, Birkhäuser, pp. 135–148.

A. Griewank (1992), 'Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation', *Optim. Methods Software* **1**, 35–54.

A. Griewank and G. F. Corliss (1991). *Automatic Differentiation of Algorithms*, SIAM, Philadelphia.

I. E. Grossmann (1990), 'Mixed-integer nonlinear programming techniques for the synthesis of engineering systems', *Research in Engineering Design* **1**, 205–228.

I. E. Grossmann, ed. (1996), *Global Optimization in Engineering Design*, Kluwer, Dordrecht.

I. E. Grossmann (2002), 'Review of nonlinear mixed-integer and disjunctive programming techniques', *Optim. Eng.* **3**, 227–252. `egon.cheme.cmu.edu/Group/Papers/MINLPOPTE.pdf`

I. E. Grossmann and C. A. Floudas (1987), 'Active constraint strategy for flexibility analysis in chemical processes', *Comput. Chem. Eng.* **11**, 675–693.

J. Guddat, F. Guerra Vasquez and H. T. Jongen (1990), *Parametric Optimization: Singularities, Path Following and Jumps*, Wiley, Chichester.

G. D. Hager (1993), 'Solving large systems of nonlinear constraints with application to data modeling', *Interval Computations* **3**, 169–200.

T. C. Hales (1998), An overview of the Kepler conjecture. Manuscript: math.MG/9811071–math.MG/9811078 `citeseer.nj.nec.com/hales98overview.html`

E. R. Hansen (2001), Publications related to early interval work of R. E. Moore. Web document: `interval.louisiana.edu/Moores_early_papers/bibliography.html`

E. R. Hansen (1980), 'Global optimization using interval analysis: The multidimensional case', *Numer. Math.* **34**, 247–270.

E. R. Hansen (1992*a*), *Global Optimization Using Interval Analysis*, Dekker, New York.

E. R. Hansen (1992*b*), 'Bounding the solution of interval linear equations', *SIAM J. Numer. Anal.* **29**, 1493–1503.

B. Hanzon and D. Jibetean (2003), 'Global minimization of a multivariate polynomial using matrix methods', *J. Global Optim.* **27**, 1–23. `homepages.cwi.nl/~jibetean/polopt.ps`

C. Helmberg (2003), Semidefinite programming. Web site: `www-user.tu-chemnitz.de/~helmberg/semidef.html`

D. Henrion and J. B. Lasserre (2003*a*), 'GloptiPoly: Global optimization over polynomials with MATLAB and SeDuMi', *ACM Trans. Math. Software* **29**, 165–194.

D. Henrion and J. B. Lasserre (2003*b*), Solving global optimization problems over polynomials with GloptiPoly 2.1, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 43–58.

D. Henrion and J. B. Lasserre (2003*c*), Detecting global optimality and extracting solutions in GloptiPoly. Manuscript: `www.laas.fr/~henrion/papers/extract.pdf`

W. Hock and K. Schittkowski (1981), *Test Examples for Nonlinear Programming Codes*, Vol. 187 of *Lecture Notes in Economics and Mathematical Systems*, Springer, Berlin. `ftp://plato.la.asu.edu/pub/donlp2/testenviron.tar.gz`

J. Holland (1973), 'Genetic algorithms and the optimal allocation of trials', *SIAM J. Comput.* **2**, 88–105.

J. N. Hooker (2000), *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, Wiley, New York.

J. N. Hooker (2002), 'Logic, optimization and constraint programming', *INFORMS J. Comput.* **14**, 295–321. `ba.gsia.cmu.edu/jnh/joc2.ps`

R. Horst and P. M. Pardalos, eds (1995), *Handbook of Global Optimization*, Kluwer, Dordrecht.

R. Horst and H. Tuy (1990), *Global Optimization: Deterministic Approaches*, 2nd edn, Springer, Berlin.

R. Horst, P. M. Pardalos and N. V. Thoai (1995), *Introduction to Global Optimization*, Kluwer, Dordrecht.

W. Huyer and A. Neumaier (1999), 'Global optimization by multilevel coordinate search', *J. Global Optim.* **14**, 331–355. See also further tests in `www.mat.univie.ac.at/~neum/glopt/contrib/compbound.pdf`

W. Huyer and A. Neumaier (2003), SNOBFIT: Stable Noisy Optimization by Branch and Fit. Manuscript: `www.mat.univie.ac.at/~neum/papers.html#snobfit`

W. Huyer and A. Neumaier (2004), 'Integral approximation of rays and verification of feasibility', *Reliable Comput.* **10**, 195–207. `www.mat.univie.ac.at/~neum/papers.html#rays`

E. Hyvönen and S. De Pascale (1996), Interval computations on the spreadsheet, in *Applications of Interval Computations* (R. B. Kearfott and V. Kreinovich, eds), Applied Optimization, Kluwer, Dordrecht, pp. 169–209.

L. Ingber (1993), 'Simulated annealing: Practice versus theory', *Math. Comput. Modelling* **18**, 29–57.

R. H. F. Jackson, P. T. Boggs, S. G. Nash and S. Powell (1990/91), 'Guidelines for reporting results of computational experiments: Report of the ad hoc committee', *Math. Program.* **49**, 413–426.

E. Janka (2000), A comparison of stochastic methods for global optimization. Web document: `www.mat.univie.ac.at/~vpk/math/gopt_eng.html`

C. Jansson (1994), On self-validating methods for optimization problems, in *Topics in Validated Computations* (J. Herzberger, ed.), Elsevier Science BV, pp. 381–438.

C. Jansson (2003), Rigorous error bounds for the optimal value of linear programming problems, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 59–70.

C. Jansson (2004*a*), 'A rigorous lower bound for the optimal value of convex optimization problems', *J. Global Optim.* **28**, 121–137.

C. Jansson (2004*b*), 'Rigorous lower and upper bounds in linear programming', *SIAM J. Optim.*, to appear.

C. Jansson and O. Knüppel (1992), A global minimization method: The multi-dimensional case. Technical report 92-1, TU Hamburg-Harburg, January.

C. Jansson and O. Knüppel (1995), 'Branch and bound algorithm for bound constrained optimization problems without derivatives', *J. Global Optim.* **7**, 297–333.

L. Jaulin and D. Henrion (2004), 'Linear matrix inequalities for interval constraint propagation', *Reliable Comput.*, to appear.
`www.laas.fr/~henrion/Papers/jaulin_lmi.pdf`

L. Jaulin, M. Kieffer, O. Didrit and E. Walter (2001), *Applied Interval Analysis*, Springer, London.

R. G. Jeroslow (1977), 'Cutting plane theory: Disjunctive methods', *Ann. Discrete Math.* **1**, 293–330.

D. Jibetean and E. De Klerk (2003), Global optimization of rational functions: A semidefinite programming approach. Manuscript:
`www.optimization-online.org/DB_HTML/2003/05/654.html`

F. John (1948), Extremum problems with inequalities as subsidiary conditions, in *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*, Interscience, New York, pp. 187–204. Reprinted as *Fritz John, Collected Papers*, Vol. 2 (J. Moser, ed.), Birkhäuser, Boston (1985), pp. 543–560.

D. S. Johnson (2002), A theoretician's guide to the experimental analysis of algorithms, in *Proc. 5th and 6th DIMACS Implementation Challenges* (M. Goldwasser, D. S. Johnson, and C. C. McGeoch, eds) AMS, Providence, RI, pp. 215–250. `www.research.att.com/~dsj/papers/experguide.pdf`

D. R. Jones (2001), 'A taxonomy of global optimization methods based on response surfaces', *J. Global Optim.* **21**, 345–383.

D. R. Jones, C. D. Perttunen and B. E. Stuckman (1993), 'Lipschitzian optimization without the Lipschitz constant', *J. Optim. Theory Appl.* **79**, 157–181.

D. R. Jones, M. Schonlau and W. J. Welch (1998), 'Efficient global optimization of expensive black-box functions', *J. Global Optim.* **13**, 455–492.

G. Jónsson and S. A. Vavasis (2001), Accurate solution of polynomial equations using Macaulay resultant matrices. Manuscript:
`www.cs.cornell.edu/home/vavasis/vavasis.html`

W. Kahan (1968), A more complete interval arithmetic: Lecture notes for an engineering summer course in numerical analysis, University of Michigan.

J. Kallrath (2002), *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis*, Vieweg.

W. Karush (1939), Minima of functions of several variables with inequalities as side constraints, MSc dissertation, Department of Mathematics, University of Chicago, IL.

R. B. Kearfott (1991), ‘Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems’, *Computing* **47**, 169–191.

R. B. Kearfott (1996*a*), A review of techniques in the verified solution of constrained global optimization problems, in *Applications of Interval Computations* (R. B. Kearfott and V. Kreinovich, eds), Kluwer, Dordrecht, pp. 23–60.

R. B. Kearfott (1996*b*), *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht.

R. B. Kearfott (2003), GlobSol: History, composition, and advice on use, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 17–31.

R. B. Kearfott and K. Du (1994), ‘The cluster problem in multivariate global optimization’, *J. Global Optim.* **5**, 253–265.

R. B. Kearfott, M. Novoa and Chenyi Hu (1991), ‘A review of preconditioners for the interval Gauss–Seidel method’, *Interval Computations* **1**, 59–85.

C. Khompatraporn, J. Pintér and Z. B. Zabinsky (2004), ‘Comparative assessment of algorithms and software for global optimization’, *J. Global Optim.*, to appear.

S. Kim and M. Kojima (2001), ‘Second order cone programming relaxation methods of nonconvex quadratic optimization problem’, *Optim. Methods Software* **15**, 201–224.

S. Kim and M. Kojima (2003), ‘Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations’, *Comput. Optim. Appl.* **26**, 143–154.

S. Kim, M. Kojima and H. Waki (2003), Generalized Lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. Manuscript: `math.ewha.ac.kr/~skim/Research/list.html`

S. Kirkpatrick, C. D. Geddat, Jr., and M. P. Vecchi (1983), ‘Optimization by simulated annealing’, *Science* **220**, 671–680.

G. R. Kocis and I. E. Grossmann (1989), ‘Computational experience with DICOPT solving MINLP problems in process systems engineering’, *Comput. Chem. Eng.* **13**, 307–315.

M. Kojima and L. Tuncel (2000), ‘Cones of matrices and successive convex relaxations of nonconvex sets’, *SIAM J. Optim.* **10**, 750–778.

M. Kojima, S. Kim and H. Waki (2003), ‘A general framework for convex relaxation of polynomial optimization problems over cones’, *J. Oper. Res. Soc. Japan* **46**, 125–144.

L. V. Kolev (1997), ‘Use of interval slopes for the irrational part of factorable functions’, *Reliable Comput.* **3**, 83–93.

L. V. Kolev and I. P. Nenov (2001), ‘Cheap and tight bounds on the solution set of perturbed systems of nonlinear equations’, *Reliable Comput.* **7**, 399–408.

J. Kostrowicki and H. A. Scheraga (1996), Some approaches to the multiple-minima problem in protein folding, in *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding* (P. M. Pardalos *et al.*, eds), AMS, Providence, RI, pp. 123–132.

R. Krawczyk (1969), ‘Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken’, *Computing* **4**, 187–201.

H. W. Kuhn (1991), Nonlinear programming: A historical note, in *History of Mathematical Programming* (J. K. Lenstra *et al.*, eds) North Holland, Amsterdam, pp. 82–96.

H. W. Kuhn and A. W. Tucker (1951), in *Nonlinear Programming*, *Proc. 2nd Berkeley Symp. Math. Stat. Prob.* (J. Neyman, ed.), University of California Press, Berkeley, CA, pp. 481–492.

J. C. Lagarias (2002), 'Bounds for local density of sphere packings and the Kepler conjecture', *Discrete Comput. Geom.* **27**, 165–193.

J. L. Lagrange (1797), *Théorie des Fonctions Analytiques*, Impr. de la République, Paris.

A. H. Land and A. G. Doig (1960), 'An automated method for solving discrete programming problems', *Econometrica* **28**, 497–520.

J. B. Lasserre (2001), 'Global optimization with polynomials and the problem of moments', *SIAM J. Optim.* **11**, 796–817.

Y. Lebbah, M. Rueher and C. Michel (2002), A global filtering algorithm for handling systems of quadratic equations and inequations, in *Principles and Practice of Constraint Programming, CP 2002* (P. van Hentenryck, ed.), Vol. 2470 of *Lecture Notes in Computer Science*, Springer, New York, pp. 109–123.

Y. Lebbah, C. Michel, M. Rueher, J.-P. Merlet and D. Daney (2004), Efficient and safe global constraints for handling numerical constraint systems, *SIAM J. Numer. Anal.*, to appear.

E. Lee and C. Mavroidis (2002), 'Solving the geometric design problem of spatial 3R robot manipulators using polynomial homotopy continuation', *J. Mech. Design, Trans. ASME* **124**, 652–661.

E. Lee, C. Mavroidis and J. P. Merlet (2002), Five precision points synthesis of spatial RRR manipulators using interval analysis, in *Proc. 2002 ASME Mechanisms and Robotics Conference, Montreal, September 29–October 2*, pp. 1–10. `robots.rutgers.edu/Publications.htm`

M. Lerch, G. Tischler, J. Wolff von Gudenberg, W. Hofschuster and W. Krämer (2001), The Interval Library filib 2.0: Design, features and sample programs. Preprint 2001/4, Universität Wuppertal. `www.math.uni-wuppertal.de/wrswt/software/filib.html`

A. V. Levy and A. Montalvo (1985), 'The tunneling algorithm for the global minimization of functions', *SIAM J. Sci. Statist. Comput.* **6**, 15–29.

J. D. Little, K. C. Murty, D. W. Sweeney and C. Karel (1963), 'An algorithm for the travelling salesman problem', *Oper. Res.* **11**, 972–989.

M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret (1998), 'Applications of second-order cone programming', *Linear Algebra Appl.* **284**, 193–228. `www.stanford.edu/~boyd/socp.html`

W. A. Lodwick (1989), 'Constraint propagation, relational arithmetic in AI systems and mathematical programs', *Ann. Oper. Res.* **21**, 143–148.

Z.-Q. Luo, J.-S. Pang and D. Ralph (1996), *Mathematical Programs with Equilibrium Constraints*, Cambridge University Press, Cambridge.

I. J. Lustig and J.-F. Puget (2001), Program does not equal program: Constraint programming and its relationship to mathematical programming, *Interfaces* **31**, 29–53.

K. Madsen and S. Skelboe (2002), The early days of interval global optimization. `interval.louisiana.edu/conferences/VC02/abstracts/MADS.pdf`

C. Maheshwari, A. Neumaier and H. Schichl (2003), Convexity and concavity detection. In preparation.

K. Makino (1998), Rigorous analysis of nonlinear motion in particle accelerators, PhD thesis, Department of Physics and Astronomy, Michigan State University: `bt.pa.msu.edu/makino/phd.html`

K. Makino and M. Berz (2003), 'Taylor models and other validated functional inclusion methods', *Int. J. Pure Appl. Math.* **4**, 379–456.

O. L. Mangasarian (1969), *Nonlinear Programming*, McGraw-Hill, New York. Reprinted as *Classics in Applied Mathematics*, SIAM, Philadelphia (1994).

O. L. Mangasarian and S. Fromovitz (1967), 'The Fritz John necessary optimality conditions in the presence of equality and inequality constraints', *J. Math. Anal. Appl.* **17**, 37–47.

O. L. Mangasarian and L. McLinden (1985), 'Simple bounds for solutions of monotone complementarity problems and convex programs', *Math. Program.* **32**, 32–40.

F. Margot (2003), 'Exploiting orbits in symmetric ILP', *Math. Program.* **98**, 3–21

H. M. Markowitz and A. S. Manne (1957), 'On the solution of discrete programming problems', *Econometrica* **25**, 84-110.

G. P. McCormick (1972), Converting general nonlinear programming problems to separable nonlinear programming problems. Technical report T-267, George Washington University, Washington, DC.

G. P. McCormick (1976), 'Computability of global solutions to factorable nonconvex programs, Part I: Convex underestimating problems', *Math. Program.* **10**, 147–175.

C. M. McDonald and C. A. Floudas (1995), 'Global optimization for the phase and chemical equilibrium problem: Application to the NRTL equation', *Comput. Chem. Eng.* **19**, 1111–1139.

M. McKay, R. Beckman and W. Conover (1979), 'A comparison of three methods for selecting values of input variables in the analysis of output from a computer code', *Technometrics* **21**, 239–245.

J.-P. Merlet (2001), 'A parser for the interval evaluation of analytical functions and its applications to engineering problems', *J. Symbolic Comput.* **31**, 475–486.

C. A. Meyer, C. A. Floudas and A. Neumaier (2002), 'Global optimization with nonfactorable constraints', *Ind. Eng. Chem. Res.* **41**, 6413–6424.

R. J. Meziat (2003), Analysis of non convex polynomial programs by the method of moments. Manuscript:
`www.optimization-online.org/ARCHIVE_DIGEST/2003-03.html`

Z . Michalewicz (1996), *Genetic Algorithm + Data Structures = Evolution Programs*, 3rd edn, Springer, New York.

H. Mittelmann (2002), Benchmarks. Web site:
`plato.la.asu.edu/topics/benchm.html`

J. Mockus (1966), Multiextremal problems in design, PhD Thesis, Nauka.

J. Mockus (1989), *Bayesian Approach to Global Optimization*, Kluwer, Dordrecht.

J. Mockus (1994), 'Application of Bayesian approach to numerical methods of global and stochastic optimization', *J. Global Optim.* **4**, 347–356.

H. M. Moller and H. J. Stetter (1995), 'Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems', *Numer. Math.* **70**, 311–329.

M. Mongeau, H. Karsenty, V. Rouzé and J.-B. Hiriart-Urruty (2000), 'Comparison of public-domain software for black box global optimization', *Optim. Methods Software* **13**, 203–226.

R. E. Moore (1962), Interval arithmetic and automatic error analysis in digital computing. PhD thesis, Appl. Math. Statist. Lab. Rep. 25, Stanford University, Stanford, CA:
`interval.louisiana.edu/Moores_early_papers/disert.pdf`

R. E. Moore (1979), *Methods and Applications of Interval Analysis*, SIAM, Philadelphia.

R. E. Moore and C. T. Yang (1959), Interval analysis, I. Space Division Report LMSD285875, Lockheed Missiles and Space Co.
`interval.louisiana.edu/Moores_early_papers/Moore_Yang.pdf`

J. Moré and Z. Wu (1997), 'Global continuation for distance geometry problems', *SIAM J. Optim.* **7**, 814–836.

J. J. Moré, B. S. Garbow and K. E. Hillstrom (1981), 'Testing unconstrained optimization software', *ACM Trans. Math. Software* **7**, 17–41.

T. S. Motzkin and E. G. Strauss (1965), 'Maxima for graphs and a new proof of a theorem of Turan', *Canad. J. Math.* **17**, 533–540.

B. Mourrain and V. Y. Pan (1997), Solving special polynomial systems by using structured matrices and algebraic residues, in *Proc. Workshop on Foundations of Computational Mathematics* (F. Cucker and M. Shub, eds), Springer, Berlin, pp. 287–304.

B. Mourrain, Y. V. Pan and O. Ruatta (2003), 'Accelerated solution of multivariate polynomial systems of equations', *SIAM J. Comput.* **32**, 435–454.

K. G. Murty and S. N. Kabadi (1987), 'Some NP-complete problems in quadratic and nonlinear programming', *Math. Program.* **39**, 117–129.

NAG (2003), Differentiation enabled Fortran compiler technology. Web document:
`www.nag.co.uk/nagware/research/ad_overview.asp`

N. S. Nedialkov and K. R. Jackson (2001), A new perspective on the wrapping effect in interval methods for initial value problems for ordinary differential equations, in *Perspectives on Enclosure Methods* (U. Kulisch *et al.*, eds), Springer, Berlin, pp. 219–264. `www.cs.toronto.edu/NA/reports.html#ned.scan00`

G. L. Nemhauser and L. A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley, New York.

G. L. Nemhauser and L. A. Wolsey (1989), Integer programming, Chapter VI in *Optimization* (G. L. Nemhauser *et al.*, eds), Vol. 1 of *Handbooks in Operations Research and Management Science*, North Holland, Amsterdam, pp. 447–527.

I. P. Nenov and D. H. Fylstra (2003), 'Interval methods for accelerated global search in the Microsoft Excel Solver', *Reliable Comput.* **9**, 143–159.

A. Neumaier (1988), The enclosure of solutions of parameter-dependent systems of equations, in *Reliability in Computing* (R. E. Moore, ed.), Academic Press, San Diego, pp. 269–286.
`www.mat.univie.ac.at/~neum/publist.html#encl`

A. Neumaier (1990), *Interval Methods for Systems of Equations*, Cambridge University Press.

A. Neumaier (1993), 'The wrapping effect, ellipsoid arithmetic, stability and confidence regions', *Computing Supplementum* **9**, 175–190.

A. Neumaier (1996), 'Second-order sufficient optimality conditions for local and global nonlinear programming', *J. Global Optim.* **9**, 141–151.

A. Neumaier (1997), 'Molecular modeling of proteins and mathematical prediction of protein structure', *SIAM Review* **39**, 407–460.

A. Neumaier (1999), 'A simple derivation of the Hansen–Bliek–Rohn–Ning–Kearfott enclosure for linear interval equations', *Reliable Comput.* **5**, 131–136. Erratum, *Reliable Comput.* **6** (2000), 227.

A. Neumaier (2001*a*), 'Generalized Lyapunov–Schmidt reduction for parametrized equations at near singular points', *Linear Algebra Appl.* **324**, 119–131.

A. Neumaier (2001*b*), *Introduction to Numerical Analysis*, Cambridge University Press, Cambridge.

A. Neumaier (2002), 'Taylor forms: Use and limits', *Reliable Comput.* **9**, 43–79.

A. Neumaier (2003*a*), 'Rational functions with prescribed global and local minimizers', *J. Global Optim.* **25**, 175–181.

A. Neumaier (2003*b*), Constraint satisfaction and global optimization in robotics. Manuscript: `www.mat.univie.ac.at/∼neum/papers.html#rob`

A. Neumaier and H. Schichl (2003), Sharpening the Karush–John optimality conditions. Manuscript: `www.mat.univie.ac.at/∼neum/papers.html#kj`

A. Neumaier and O. Shcherbina (2004), Safe bounds in linear and mixed-integer programming, *Math. Program. A* **99**, 283–296. `www.mat.univie.ac.at/∼neum/papers.html#mip`

J. Nocedal and S. J. Wright (1999), *Numerical Optimization*, *Springer Series in Operations Research*, Springer, Berlin.

I. Nowak, H. Alperin and S. Vigerske (2003), LaGO: An object oriented library for solving MINLPs, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 32–42. `www.math.hu-berlin.de/∼alpe/papers/LaGO/`

W. Older and A. Vellino (1993), Constraint arithmetic on real intervals, in *Constrained Logic Programming: Selected Research* (F. Benhameou and A. Colmerauer, eds), MIT Press.

J. Outrata, M. Kočvara and J. Zowe (1998), *Nonsmooth Approach to Optimization problems with Equilibrium Constraints*, Kluwer, Dordrecht.

A. B. Owen (1992), 'Orthogonal arrays for computer experiments, integration and visualization', *Statist. Sinica* **2**, 439–452.

A. B. Owen (1994), 'Lattice sampling revisited: Monte Carlo variance of means over randomized orthogonal arrays', *Ann. Statist.* **22**, 930–945.

I. Papamichail and C. S. Adjiman (2002), 'A rigorous global optimization algorithm for problems with ordinary differential equations', *J. Global Optim.* **24**, 1–33.

P. M Pardalos and J. B. Rosen (1987), *Constrained Global Optimization: Algorithms and Applications*, Vol. 268 of *Lecture Notes in Computer Science*, Springer, Berlin.

P. M. Pardalos and G. Schnitger (1988), 'Checking local optimality in constrained quadratic programming is NP-hard', *Oper. Res. Lett.* **7**, 33–35.

R. G. Parker and R. L. Rardin (1988), *Discrete Optimization*, Academic Press, San Diego, CA.

P. A. Parrilo (2003), ‘Semidefinite programming relaxations for semialgebraic problems’, *Math. Program. B* **96**, 293–320.

P. A. Parrilo and S. Lall (2003), ‘Semidefinite programming relaxations and algebraic optimization in control’, *Europ. J. Control* **9**, 307–321.

P. A. Parrilo and B. Sturmfels (2003), Minimizing polynomial functions, in *Algorithmic and quantitative real algebraic geometry*, Vol. 60 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, AMS, Providence, RI, pp. 83–99. `www.arxiv.org/abs/math.OC/0103170`

J. D. Pintér (1996*a*), *Global Optimization in Action*, Kluwer, Dordrecht.

J. D. Pintér (1996*b*), ‘Continuous global optimization software: A brief review’, *Optima* **52**, 1–8.

J. D. Pintér (1999), *LGO: A Model Development System for Continuous Global Optimization. User’s Guide*, Pintér Consulting Services, Inc., Halifax, NS.

S. A. Piyavskii (1972), An algorithm for finding the absolute extremum of a function, *USSR Comput. Math. and Math. Phys.* **12** 57–67.

S. Prajna, A. Papachristodoulou and A. Parrilo (2002), SOSTOOLS: Sum of Squares Optimization Toolbox for MATLAB – User’s guide. Manuscript: `www.optimization-online.org/DB_HTML/2002/05/483.html`

H. D. Ratliff and W. Pierskalla (1981), ‘Reporting computational experience in operations research’, *Oper. Res.* **29**, xi–xiv.

H. Ratschek and J. G. Rokne (1984), *New Computer Methods for the Range of Functions*, Ellis Horwood, Chichester.

H. Ratschek and J. G. Rokne (1988), *New Computer Methods for Global Optimization*, Wiley, New York.

D. Ratz (1996), On branching rules in second-order branch-and-bound methods for global optimization, in *Scientific Computation and Validation* (G. Alefeld *et al.*, eds), Akademie-Verlag, Berlin.

D. Ratz and T. Csendes (1995), ‘On the selection of subdivision directions in interval branch-and-bound methods for global optimization’, *J. Global Optim.* **7**, 183–207.

A. D. Rikun (1997), ‘A convex envelope formula for multilinear functions’, *J. Global Optim.* **10**, 425–437.

G. Rote (1992), ‘The convergence of the sandwich algorithm for approximating convex functions’, *Computing* **48**, 337–361.

H. S. Ryoo and N. V. Sahinidis (1996), ‘A branch-and-reduce approach to global optimization’, *J. Global Optim.* **8**, 107–139.

J. Sacks, W. J. Welch, T. J. Mitchell and H. P. Wynn (1989), ‘Design and analysis of computer experiments’, *Statist. Sci.* **4**, 409–435.

N. V. Sahinidis (1996), ‘BARON: A general purpose global optimization software package’, *J. Global Optim.* **8**, 201–205.

N. V. Sahinidis (2000), BARON: Branch And Reduce Optimization Navigator – User’s manual. Web document: `archimedes.scs.uiuc.edu/baron/baron.html`

N. V. Sahinidis (2003), Global optimization and constraint satisfaction: The branch-and-reduce approach, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 1–16.

H. Schichl (2004), Global optimization in the COCONUT project, in *Numerical Software with Result Verification*, Vol. 2991 of *Lecture Notes in Computer Science*, Springer, Heidelberg, pp. 243–249.
`www.mat.univie.ac.at/∼herman/papers.html`

H. Schichl and A. Neumaier (2003), Interval analysis on directed acyclic graphs for global optimization. Manuscript:
`www.mat.univie.ac.at/∼neum/papers.html#intdag`

H. Schichl and A. Neumaier (2004), 'Exclusion regions for systems of equations', *SIAM J. Numer. Anal.* **42**, 383–408.
`www.mat.univie.ac.at/∼neum/papers.html#excl`

R. B. Schnabel and E. Eskow (1990), 'A new modified Cholesky factorization', *SIAM J. Sci. Statist. Comput.* **11**, 1136–1158.

SeDuMi (2001), web site: `fewcal.kub.nl/sturm/software/sedumi.html`

O. Shcherbina (2002), Misprints and mistakes in Floudas *et al.* (1999). Web document:
`www.mat.univie.ac.at/∼neum/glopt/contrib/handbook_corr.html`

O. Shcherbina, A. Neumaier, D. Sam-Haroud, Xuan-Ha Vu and Tuan-Viet Nguyen (2003), Benchmarking global optimization and constraint satisfaction codes, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 211–222.
`www.mat.univie.ac.at/∼neum/papers.html#bench`

J. P. Shectman and N. V. Sahinidis (1998), 'A finite algorithm for global minimization of separable concave programs', *J. Global Optim.* **12**, 1–36.

H. D. Sherali (1997), 'Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets', *Acta Math. Vietnamica* **22**, 245–270.

H. D. Sherali and C. M. Shetty (1980), *Optimization with Disjunctive Constraints*, Springer, Berlin.

H. D. Sherali and C. H. Tuncbilec (1995), 'A reformulation-convexification approach for solving nonconvex quadratic programming problems', *J. Global Optim.* **7**, 1–31.

H. D. Sherali and C. H. Tuncbilec (1997*a*), 'New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems', *Oper. Res. Lett.* **21**, 1–9.

H. D. Sherali and C. H. Tuncbilec (1997*b*), 'Comparison of two reformulation-linearization technique based linear programming relaxations for polynomial programming problems', *J. Global Optim.* **10**, 381–390.

K. Shimizu, Y. Ishizuka and J. F. Bard (1997), *Nondifferentiable and Two-Level Programming*, Kluwer, Boston.

J. Siek and A. Lumsdaine (1999), 'The matrix template library: Generic components for high-performance scientific computing', *Computing in Science and Engineering* **18**, 70–78. `www.osl.iu.edu/research/mtl/`

S. Skelboe (1974), 'Computation of rational interval functions', *BIT* **14**, 87–95.
`www.diku.dk/∼stig/CompRatIntv.pdf`

N. J. A. Sloane (2003), web document with tables of packings and designs:
`www.research.att.com/∼njas/`

H. Stetter (1997), Stabilization of polynomial systems solving with Groebner bases, in *Proc. ACM Int. Symp. Symbolic Algebraic Computation*, pp. 117–124.

F. H. Stillinger (1985), ‘Role of potential-energy scaling in the low-temperature relaxation behavior of amorphous materials’, *Phys. Rev. B* **32**, 3134–3141.

J. F. Sturm (1999), ‘Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones’, *Optim. Methods Software* **11–12**, 625–653.
`fewcal.kub.nl/sturm/software/sedumi.html`

B. Tang (1993), ‘Orthogonal array-based Latin hypercubes’, *J. Amer. Statist. Assoc.* **88**, 1392–1397.

M. Tawarmalani and N. V. Sahinidis (2001), Semidefinite relaxations of fractional programs via novel convexification techniques, *J. Global Optim.* **20** 137–158.

M. Tawarmalani and N. V. Sahinidis (2002*a*), Convex extensions and envelopes of lower semi-continuous functions, *Math. Program.* **93** 247–263.

M. Tawarmalani and N. V. Sahinidis (2002*b*), *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer, Dordrecht.

M. Tawarmalani and N. V. Sahinidis (2004), ‘Global optimization of mixed-integer nonlinear programs: A theoretical and computational study’, *Math. Program.*, DOI 10.1007/s10107-003-0467-6.

M. Todd (2001), Semidefinite optimization, in *Acta Numerica*, Vol. 10, Cambridge University Press, pp. 515–560.

J. A. Tomlin (1970), Branch and bound methods for integer and non-convex programming, in *Integer and Nonlinear Programming* (J. Abadie, ed.), American Elsevier Publishing Company, New York, pp. 437–450.

TOMS (2002), Information for authors. Web document:
`www.acm.org/toms/Authors.html#TypesofPapers`

A. Törn (1972), Global optimization as a combination of local and global search, in *Proceedings of Computer Simulation Versus Analytical Solutions for Business and Economical Models, Gothenburg, August 1972* (W. Goldberg, ed.).

A. Törn (1974), Global optimization as a combination of global and local search, PhD Thesis, Abo Akademi University, HHAAA 13.

A. Törn (2000), Global optimization. Web document:
`www.abo.fi/∼atorn/Globopt.html`

A. Törn and A. Žilinskas (1989), *Global Optimization*, Vol. 350 of *Lecture Notes in Computer Science*, Springer, Berlin.

A. Törn, M. Ali and S. Viitanen (1999), ‘Stochastic global optimization: Problem classes and solution techniques’, *J. Global Optim.* **14**, 437–447.

T. Tsuda and T. Kiono (1964), Application of the Monte Carlo method to systems of nonlinear algebraic equations, *Numerische Matematik* **6** 59–67.

H. Tuy (1995), DC optimization: Theory, methods and algorithms, in *Handbook of Global Optimization* (R. Horst and P. M. Pardalos eds), Kluwer, Dordrecht, pp. 149–216.

M. Ulbrich and S. Ulbrich (1996), Automatic differentiation: A structure-exploiting forward mode with almost optimal complexity for Kantorovich trees, in *Applied Mathematics and Parallel Computing* (H. Fischer *et al.*, eds), Physica-Verlag, Heidelberg, pp. 327–357.

L. Vandenberghe and S. Boyd (1996), ‘Semidefinite programming’, *SIAM Review* **38**, 49–95. `www.stanford.edu/∼boyd/reports/semidef_prog.ps`

D. Vandenbussche (2003), Polyhedral approaches to solving nonconvex quadratic programs, PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology.

B. Vanderbei (2000), Nonlinear optimization models. Web site: `www.sor.princeton.edu/~rvdb/ampl/nlmodels/`

P. Van Hentenryck (1989), *Constraint Satisfaction in Logic Programming*, MIT Press, Cambridge, MA.

P. Van Hentenryck, L. Michel and F. Benhamou (1997*a*), 'Newton: Constraint programming over non-linear constraints', *Sci. Program.* **30**, 83–118.

P. Van Hentenryck, L. Michel and Y. Deville (1997*b*), *Numerica: A Modeling Language for Global Optimization*, MIT Press, Cambridge, MA.

R. Van Iwaarden (1986), An improved unconstrained global optimization algorithm, PhD thesis, University of Colorado at Denver, CO.

P. J. M. Van Laarhoven and E. H. L. Aarts (1987), *Simulated Annealing: Theory and Applications*, Kluwer, Dordrecht.

L. N. Vicente and P. H. Calamai (1994), 'Bilevel and multilevel programming: A bibliography review', *J. Global Optim.* **5**, 291–306.

V. Visweswaran and C. A. Floudas (1996*a*), New formulations and branching strategies for the GOP algorithm, in *Global Optimization in Chemical Engineering* (I. E. Grossmann, ed.), Kluwer, Dordrecht, pp. 75–100.

V. Visweswaran and C. A. Floudas (1996*b*), Computational results for an efficient implementation of the GOP algorithm and its variants, in *Global Optimization in Chemical Engineering* (I. E. Grossmann, ed.), Kluwer, Dordrecht, pp. 111–153.

Xuan-Ha Vu, D. Sam-Haroud and M.-C. Silaghi (2002), Approximation techniques for non-linear problems with continuum of solutions, in *Proc. 5th Int. Symp. Abstraction, Reformulation Approximation, SARA'2002* (B. Y. Choueiry and T. Walsh, eds), Vol. 2371 of *Lecture Notes in Artificial Intelligence*, Springer, pp. 224–241.

Xuan-Ha Vu, D. Sam-Haroud and M.-C. Silaghi (2003), Numerical constraint satisfaction problems with non-isolated solutions, in *Global Optimization and Constraint Satisfaction* (C. Bliek *et al.*, eds), Springer, Berlin, pp. 194–210.

W. Walster (2000), Interval arithmetic solves nonlinear problems while providing guaranteed results, FORTE TOOLS feature stories. Web manuscript: `wwws.sun.com/software/sundev/news/features/intervals.html`

G. Walster, E. Hansen and S. Sengupta (1985), Test results for a global optimization algorithm, in *Numerical Optimization 1984* (P. T. Boggs *et al.*, eds), SIAM, Philadelphia, pp. 272–287.

H. P. Williams (1999), *Model Solving in Mathematical Programming*, 4th edn, Wiley, Chichester.

H. Wolkowicz, R. Saigal and L. Vandenberghe, eds (2000), *Handbook on Semidefinite Programming*, Kluwer, Dordrecht.

L. A. Wolsey (1998), *Integer Programming*, Wiley.

S. J. Wright (1997), *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia.

M. Yagiura and T. Ibaraki (2001), 'On metaheuristic algorithms for combinatorial optimization problems', *Systems and Computers in Japan* **32**, 33–55.

H. Yan and J. N. Hooker (1999), 'Tight representations of logical constraints as cardinality rules', *Math. Program.* **85**, 363–377.

J. M. Zamora and I. E. Grossmann (1999), 'A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms', *J. Global Optim.* **14**, 217–249.

A. A. Zhigljavsky (1991), *Theory of Global Random Search*, Kluwer, Dordrecht.