Global Optimization: Software, Test Problems, and Applications

Chapter · January 2002

DOI: 10.1007/978-1-4757-5362-2_15

1 author:

Janos Pinter
Lehigh University
179 PUBLICATIONS **2,643** CITATIONS

Chapter 15

GLOBAL OPTIMIZATION: SOFTWARE, TEST PROBLEMS, AND APPLICATIONS

János D. Pintér
Pintér Consulting Services Inc. and Dalhousie University
129 Glenforest Drive
Halifax, NS
Canada B3M 1J2
jdpinter@hfx.eastlink.ca

Abstract This chapter provides a concise review of the most prominent global optimization (GO) strategies currently available. This is followed by a discussion of GO software, test problems and several important types of applications, with additional pointers. The exposition is concentrated around topics related to continuous GO, although in certain aspects it is also pertinent to analogous issues in combinatorial optimization.

Keywords: global optimization, solution strategies, software, test problems, illustrative applications.

1. Introduction

In engineering, economic and scientific studies, quantitative decisions are frequently modeled applying optimization concepts and tools. The decision-maker or modeler typically wants to find the 'absolutely best' decision which corresponds to the minimum (or maximum) of a suitable objective function, while it satisfies a given collection of feasibility constraints. The objective function expresses overall (modeled) system performance, such as profit, utility, loss, risk, or error. The constraints originate from physical, technical, economic – or possibly some other – considerations.

A large number of such models – mainly in the context of man-made systems – naturally belong to the realm of 'traditional' local scope continuous optimization: notably, to linear and convex programming. However, there exist also numerous cases in which the necessary structural (linearity or general convexity) requirements are not satisfied, or are not simply verifiable. The phenomena and processes modeled can be either highly nonlinear, or they may be given only implicitly (computationally). To simply illustrate the point of nonlinearity, one may think of the most frequently used general (basic) function forms appearing in applied mathematical, physical, chemical, biological, environmental, engineering and economic studies. Polynomials, power functions, the exponential/logarithmic pair and trigonometric functions – as well as far more complicated ones – play a primary role in descriptive system models. The low-order local polynomial (most often, linear or quadratic) approximations come after, to yield model forms that are then manageable using the more traditional armory of mathematics, including optimization. Illuminating and far-reaching discussions related to the general subject of nonlinearity in nature – in the most diverse contexts, and with literally hundreds of examples – are presented by Eigen and Winkler, 1975, Mandelbrot, 1983, Murray, 1989, Casti, 1990, Schroeder, 1991, Stewart, 1995, Wolfram, 1996, and Jacob, 2001.

In case of a possibly quite complex nonlinear system description, the associated decision model may – and it frequently will – have multiple locally optimal solutions. In most realistic cases, the number of such local solutions is not known *a priori*, and the quality of local and global solutions may differ substantially. Therefore these decision models can be very difficult, and – as a rule – standard optimization strategies are not directly applicable to solve them. Hence, one needs to rely on proper global optimization (GO) concepts and techniques.

Global optimization is arguably a most relevant and interesting area, since in principle, it covers 'all' traditional mathematical programming and (discretized) optimal control, while reaching into new territory. This implies that GO needs new, extended modeling paradigms and solution strategies, in addition to – i.e., in prudent combination with – classical optimization concepts and techniques.

The purpose of this chapter is to provide a concise review of available GO methodology and software, followed by a discussion of test problems and illustrative applications. The exposition will mainly concentrate on topics related to continuous GO, although in numerous aspects it is also pertinent to analogous issues in combinatorial optimization.

2. Solution strategies

2.1 The global optimization challenge

For the sake of completeness and subsequent use, we shall formulate the continuous global optimization problem (CGOP) as

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & x \in D \end{aligned} \tag{15.1}$$

where $D = \{x : l \leq x \leq u;\ g_j(x) \leq 0 \quad j = 1, \ldots, J\}$. In (15.1) we apply the following notation:

- $x \in \mathbb{R}^n$: real n-vector of decision variables,
- $f : \mathbb{R}^n \to \mathbb{R}$: continuous objective function,
- $D \subset \mathbb{R}^n$: non-empty set of feasible decisions (a proper subset of $\mathbb{R}^n$);

D is defined by

- l and u: explicit, finite (component-wise) lower and upper bounds on x, and
- $g : \mathbb{R}^n \to \mathbb{R}^m$: finite collection ($J$-vector) of continuous constraint functions.

Note that explicit bounds on the constraint function values can also be imposed, but such more specialized models are also amenable to the form (15.1).

The above listed basic analytical conditions – by the classical theorem of Weierstrass – guarantee that the optimal solution set of the CGOP is non-empty. At the same time, without further specific structural assumptions, (15.1) can be a very difficult problem. For instance, D could be disconnected, and even some of its components could be non-convex; furthermore, f could be *per se* highly multiextremal. Hence, there may exist an unknown number of global and local solutions to (15.1). For this reason, there is no 'straightforward' general algebraic characterization of global optimality. By contrast, in 'traditional' nonlinear programming, most exact algorithms aim to solve the Karush-Kuhn-Tucker system of (necessary) conditions: the corresponding system of equations and inequalities in the present framework becomes another GO problem, often at least as complex as the original model.

The CGOP class includes a number of well-structured, specific cases (such as e.g., concave minimization under convex constraints), as well as far more general (e.g., differential convex or Lipschitz-continuous) problems. Therefore one can expect that the corresponding 'most suitable' solution approaches will also vary to a considerable extent. On one hand, a very general optimization strategy should work for broad model classes, although its efficiency might be lower for the more special problem instances. On the other hand, highly tailored algorithms may not work for problem-classes outside of their declared scope.

In theory, we want to find all global solutions to model (15.1), tacitly assuming that the solution set is at most countable. The more practical aim of GO is to find suitable approximations of the set of global optima (denoted by X^*), and the corresponding optimum value ($z^* = f(x^*)$ for $x^* \in X^*$). This needs to be attained on the basis of a finite number of search steps, that is, objective and constraint function evaluations at algorithmically selected search points. (Note that in many GO methods there is no essential need to evaluate gradient, Hessian or higher order local information.) With this practical objective in mind, we shall briefly review the most frequently applied GO strategies. More extensive discussions can be found in the first volume of the *Handbook of Global Optimization* (Horst and Pardalos, 1995), in the present volume, as well as in the references cited below.

There are several logical ways to classify GO strategies. For our purposes – and in line with the objectives of the present volume – a natural dividing line lies between exact and heuristic methods. Deterministic algorithms belonging to the first group, at least in theory, have a rigorous guarantee for finding at least one, or all global solutions. However, the associated computational burden easily can become excessive, for larger dimensional models, and/or for more complicated model functions. Most continuous GO and combinatorial models of relevance are known to be NP-hard, therefore even the seemingly 'limitless' increase of computational power will not resolve their genuine intractability.

For this reason, in higher dimensions and without special model structure, there is perhaps more practical 'hope' in exact stochastic algorithms, and in intelligent heuristic methods which have a stochastic global search component. Heuristic methods – as a rule – do not have strict convergence guarantees; however, in many cases, they have a proven track record, and may offer practical tools to handle models that are (currently or forever) out of the reach of theoretically correct, rigorous methodology.

We would like to emphasize at this point that the issue of computational intractability crucially depends on the exact question posed. For example, in order to guarantee a given deterministic accuracy in approximating some x^* of X^* – for a sufficiently broad class of models belonging to (15.1) – typically would require an exponentially increasing number of search steps, as the model dimensionality grows. At the same time, under further mild and general analytical assumptions regarding (15.1), suitable stochastic methods will converge to an element of X^* with probability one (i.e., almost surely). Just as important, applying a reasonable numerical effort stochastic methods have a probabilistic guarantee – that is, a very good chance – to find a search point(s) that will be a 'reasonably good quality' solution. The exact terms of these statements will not be cited here (they can be found in the references cited later on), since our purpose is only to illustrate the importance of the question to answer.

For a balanced discussion, it is also worth pointing out that whenever rigorous approaches in fact can be applied, there has to be a convincing argument for choosing heuristics instead. Such arguments could be based on the lack of implementation (as yet) for a suitable exact algorithm or – more typically – the lack of computational resources and/or practical time limits when providing approximate solutions of acceptable accuracy, etc. However, it is essential to recognize also the inherent limitations of heuristic approaches, and draw performance conclusions only on the basis of extensive and representative computational tests.

For the sake of completeness, a brief annotated classification of the most frequently applied GO strategies is provided below, arranged alphabetically. For additional details – including discussions regarding the pros and cons of these methods – please consult corresponding chapters of one of the two volumes of the *Handbook of Global Optimization*, or the GO textbooks and Web sites cited below.

2.2 Exact methods

Adaptive stochastic search methods. These procedures are based – at least partially – on random sampling in D. Adaptive search strategy adjustments, sample clustering, deterministic solution refinement options, statistical stopping rules, etc. can be added as enhancements to the basic scheme of pure (that is, uniform in D) random sampling. Such methods are applicable to both discrete and continuous GOPs under very general conditions. See e.g., Zhigljavsky, 1991, Boender and Romeijn, 1995, or Pintér, 1996a.

Bayesian search algorithms. These are based upon some *a priori* postulated stochastic model of the function class of which the given function f (or given GO problem) is an instance of. The subsequent adaptive estimation of the problem-instance characteristics is based upon this model and the actual search results. Typically, 'myopic' (one-step optimal, hence easier to compute) approximate decisions govern the search procedure. Bayesian methods are applicable to general CGOPs: consult, e.g., Mockus et al., 1996.

Branch and bound algorithms. Adaptive partition, sampling, and bounding procedures (within subsets of the feasible set D) can be applied to continuous GO models, analogously to the well-known pure integer programming, or mixed integer linear programming methodology. This general approach subsumes many specific cases, and allows for significant generalizations. Branch-and-bound methods are applicable to diverse broadly structured GOPs, such as concave minimization, DC programming, and Lipschitz optimization problems. Consult, e.g., Ratschek and Rokne, 1988, Neumaier, 1990, Hansen, 1992, Horst and Tuy, 1996, Kearfott, 1996, Pintér, 1996a, Floudas, 1999, Strongin and Sergeyev, 2000.

Enumerative strategies. These methods are based upon a complete (streamlined) enumeration of all possible solutions. Applicable to combinatorial optimization problems, and to certain structured CGOP models (such as concave programming). Consult, e.g., Horst and Tuy, 1996.

Homotopy and trajectory methods. These strategies have the 'ambitious' objective of visiting (explicitly enumerating) all stationary points of the objective function f, within the set D. This search effort then leads to the list of all – global as well as local – optima. The methodology is applicable to smooth GO problems, but the computational demands can be very substantial. Consult, for instance, Diener, 1995 and Forster, 1995.

Integral methods. These methods are aimed at the determination of the essential supremum of the objective function f over D, by approximating the level sets of f. Consult, e.g., Zheng and Zhuang, 1995, or Hichert et al., 1997.

'Naive' (passive) approaches. These include, for instance, simultaneous grid search and pure random search: that is, there is no interrelation between the sample points selected (they could be as well sampled at the same time, without consideration to individual outcomes). Note that although such methods are obviously convergent under mild analytical assumptions, they are truly 'hopeless' in solving higher (often already in 3, 4, . . .) dimensional problems. For more details, see for instance Zhigljavsky, 1991 or Pintér, 1996a, with further references therein.

Relaxation (outer approximation) strategies. In this general approach, the GOP is replaced by a sequence of relaxed sub-problems that are easier to solve. Successive refinement of sub-problems to approximate the initial problem is applied: cutting planes and more general cuts, diverse minorant function constructions, and other customizations are possible. Relaxation algorithms are applicable to diverse structured GO – such as concave minimization, or DC programming – models. See, e.g., Horst and Tuy, 1996, or Benson, 1995.

2.3 Heuristic methods

Approximate convex underestimation. This strategy attempts to estimate the (possible large-scale, overall) convexity characteristics of the objective function based on directed sampling in D. Sometimes it works admirably well, but in other cases (when the postulated quadratic model is not suitable) it will not produce good approximate solutions. Convex underestimation strategies are applicable to smooth GO problems. See Dill et al., 1997.

Continuation methods. These approaches first transform the objective function into some more smooth, 'simpler' function with fewer local minimizers, and then use a local minimization procedure to attempt tracing all minimizers back to the original function. Continuation methods are applicable to smooth GOPs; see e.g. Moré and Wu, 1997.

Genetic algorithms, evolution strategies. Evolutionary optimization approaches heuristically 'mimic' biological evolution models. Various deterministic and stochastic algorithms can be constructed, based on diverse evolutionary 'game rules'. These strategies are applicable to both discrete and continuous GO problems under mild structural requirements. Consult, e.g., Michalewicz, 1996, Osman and Kelly, 1996, Glover and Laguna, 1997, or Voss et al., 1999.

'Globalized' extensions of local search methods. These practical strategies are based on a preliminary global search (passive grid or random search) phase, followed by local scope search. Applicable to smooth GO problems: differentiability is usually postulated for the sake of the embedded local search component. Consult, for instance, Zhigljavsky, 1991 or Pintér, 1996a.

Sequential improvement of local optima. These approaches – including tunneling, deflation, and filled function methods – typically operate on adaptively constructed auxiliary functions, to assist the search for gradually better optima (while avoiding the ones found so far). These strategies are applicable to smooth GO problems. Consult, for instance, Levy and Gomez, 1985.

Simulated annealing. These techniques are based upon the physical analogy of cooling crystal structures that spontaneously arrive at a stable configuration, characterized by – globally or locally – minimal potential energy. Simulated annealing is applicable to both discrete and continuous GOPs under mild structural requirements. See, for instance, Osman and Kelly, 1996, or Glover and Laguna, 1997.

Tabu search (TS). The essential idea of this popular meta-heuristic is to 'forbid' search moves to points already visited in the (usually discrete) search space, at least within the next few steps. Tabu search methodology has been primarily used (so far) to solve combinatorial optimization problems, but it can also be extended to handle continuous GOPs. Consult, e.g., Osman and Kelly, 1996, Glover and Laguna, 1997, or Voss et al., 1999.

One can observe that certain overlaps may exist among the algorithm categories listed above. Moreover, search strategy combinations are often both desirable and possible: this, in turn leads to non-trivial issues in algorithm design.

3. Software

3.1 Introductory notes

In spite of significant theoretical advances, GO software development and standardized usage still seem to lag behind. This can be explained by several factors among which paramount is the inherent (theoretical and numerical) difficulty of GO models.

From a purely mathematical point of view, even the 'simpler' GO model instances – for example, concave minimization, or indefinite quadratic programming – belong to the hardest class of mathematical programming problems. The computational difficulty of any general class of such models can be expected to increase exponentially, as a function of the problem dimensionality n. (This point is extensively discussed in OR and computer science textbooks; hence, we will not repeat the exposition here.) The practical implication is that GO problems stated in

$\mathbb{R}^n$, with n being just 5, 10, 20, 50, or 100 – as a general rule – may have very rapidly increasing computational demands. Consequently, even if raw computer power seems to grow at a very fast pace, the 'curse of dimensionality' is here to stay.

This fact directly leads to a basic consequence with respect to all possible implementations of GO methodology. Since all rigorous (deterministic) algorithms, as a rule, imply an exponentially increasing computational demand, they – at least from a practical point of view – should ideally be completed by a correct and efficient local optimization phase (started from the global solution estimate generated in that search phase). Global convergence, however, can be guaranteed only by the global scope algorithmic component: hence, the latter should be applied – at least, in theory – in a complete, exhaustive search mode. This controversy between theoretical rigor and numerical efficiency is unavoidable, and poses the essential difficulty in developing robust and efficient GO software. One either has to sacrifice theoretically guaranteed, deterministic convergence properties, or some (possibly very much) numerical efficiency – at least when solving practical problems of non-trivial size and complexity, within a realistic time frame. All software solutions offered to address this paradox in merit need to aim at a careful balance, to avoid both 'practically hopeless, although entirely correct' and 'it always works like magic, no need for theory' approaches.

These notes are more than purely 'philosophical' reflections. In formulating and solving a practical decision model, one of the essential aspects to consider is the choice of method and software to use. Ideally, we want to select and apply the method/software that would be most 'suitable' for solving the problem at hand. The term 'suitability' aggregates different criteria for different people. For example, one user may emphasize the reliability and speed of the algorithm to find a 'sufficiently good' solution under most circumstances, even though the solution found is not guaranteed to be close to optimal. Another user may give more attention to rigorous guarantees and to the accuracy of the solution obtained, while still paying some (secondary) attention to efficiency issues. In commercial environments, the software provider's name and market share/visibility, reliable documentation and user support may dominate all other (perhaps more professional) aspects. Depending on user criteria, the choice is often far from easy.

Based also on extensive earlier discussions in the optimization literature, Khompatraporn et al., 2001 summarize the following key features considered in evaluating the overall quality of GO software.

- *Generality.* The algorithm is insensitive to secondary details of the problem structure. In other words, the algorithm theoretically converges, without any further restrictions on the structure of the problem instance (assuming, of course, that the instance and the entire problem-class belong to the scope of the algorithm).

- *Reliability.* The algorithm is reliable and solves the given problem within a reasonable degree of accuracy, which can be specified by the user.
- *Efficiency.* The algorithm keeps the amount of calculations as small as possible so that the actual computation time is realistic; it requires relatively few iterations; and it is insensitive to the initial starting point and other specifications (when applicable).
- *Ease of Use.* The application of the algorithm is relatively easy to understand even by less experienced users. The algorithm has as few parameters that require problem-dependent 'tuning' as possible.

It is easy to see that tradeoffs between these attributes are usually inevitable. Notably, when robustness and ease of use increase, efficiency typically decreases and *vice versa.* The stated general attributes are also interrelated. For instance, algorithms that are very sensitive to the selection of the tuning parameters are often problem dependent, and hence not sufficiently general. An algorithm tailored to a specific class of problems with common structure may be more efficient for that particular problem class than an algorithm that could be applied to a more general class of problems, but a lot less efficient – or even unusable – when applied to problems outside that class. For further related discussions of these points, consult Khompatraporn et al., 2001, Neumaier, 2001 or Pintér, 2001a.

3.2 Software availability and design

A few years ago, a global optimization software survey was prepared for the newsletter of the Mathematical Programming Society (Pintér, 1996b). This survey is also made available on the World Wide Web, see e.g. the site maintained by Mittelmann and Spellucci, 2001. Based on the responses solicited from software authors as well as on information collected from the Web, over fifty software products were listed in that review. It is most likely that by now the number of different software projects aimed at solving various GOP instances could be in the order of a few hundreds. For this reason – and also because of the rapidly changing scenery (new product and implementation versions, transitions from free distributions to commercial versions, disappearing Web and download sites, and so on) – there is no sound rationale to provide here a detailed and exact list of specific GO software products available today.

As of 2001, there is no '*Global Optimization Software Guide*' as such which could be similar to the annotated collection presented by Moré and Wright, 1993, probably since the field is still changing so fast. However, there exist several informative Web sites devoted partly or entirely to the subject. For prominent examples, three well maintained Web sites are mentioned here:

- *Nonlinear Programming Frequently Asked Questions* (Fourer, 2001).
- *Decision Tree for Optimization Software* (Mittelmann and Spellucci, 2001).
- The *Global Optimization* site of Neumaier, 2001.

These sites present a solid general framework, including insightful commentary on model classification, algorithms and available software, in addition to collecting facts and providing extensive pointers towards further information. (The latter alone is done by numerous other useful sites.) Specifically, the sites mentioned above provide concisely annotated listings of global optimization software (mainly publicly available software, but also selected commercial ones).

As it can be expected, the quality of GO software varies largely, in terms of theoretical rigor and precise convergence conditions, as well as in their implementation quality (execution efficiency, reliability, parameterizations, user-friendliness, documentation, etc.). Furthermore – again, as it can be rightfully expected – most freely available software comes without technical support and usability guarantees. However, the software developer or user who is willing to invest time into figuring out the details and spend time on evaluating the information found, can make good use of – and certainly can learn from – the information provided.

The (possibly somewhat subjective) overall impression of the author is that, with a few notable exceptions, the currently available GO software products are still dominantly of 'academic research level', as opposed to 'industrial strength level'. Of course, both types of software are useful and necessary: differences between such software products, however, need to be recognized.

A list of highly desirable – and in case of professional application software products, *de facto* indispensable – implementation features include the following.

- Well-specified target hardware platforms and software environments.

- Quality user guidance: software manual and/or on-line help, including a clearly outlined model development and solution procedure, sensible modeling and troubleshooting tips, sample user files, and non-trivial(!) numerical examples.
- Functional, robust and friendly user interface.
- Flawless and informative runtime communication, including clear system messages for all foreseeable program execution scenarios.
- 'Fool-proof' solver selection and execution procedures, including solver option usage tips and error messages, and proper exception handling (as needed).
- Automatic generation of result file(s) that include all essential runtime messages, in addition to final results.
- Model visualization (at least as an option): this can be considered particularly desirable in nonlinear systems modeling, to assist in the model development procedure.
- Flexible connectivity to other (external) application programs.
- Reliable, high-quality user support.
- Continued product maintenance and development (since hardware platforms, operating systems, as well as development environments are also in perpetual change).

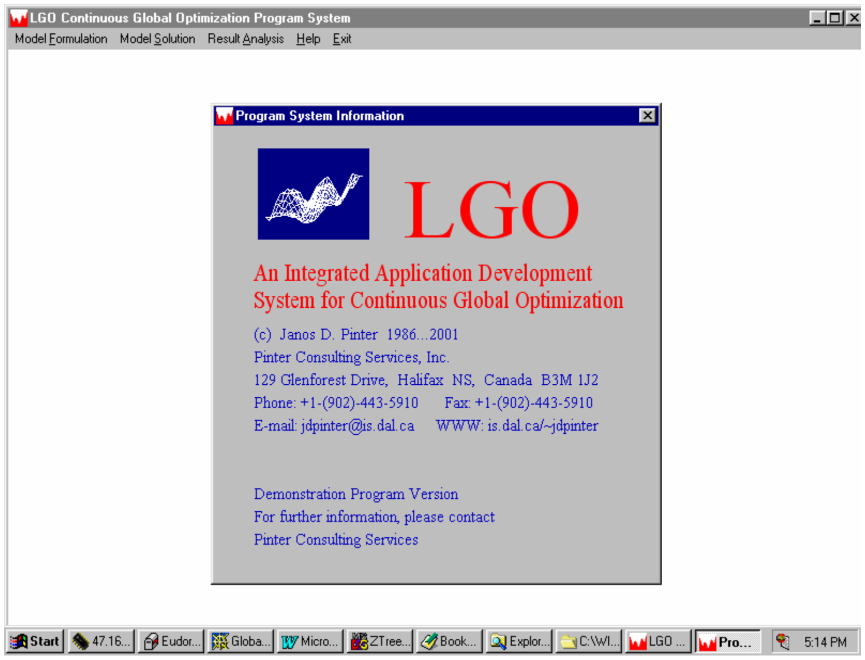
As the listed requirements indicate, the task is not quite simple, and goes well beyond of developing some 'adequately functional' machinery for sheer number crunching. When one adds to this tentative 'wish-list' the higher level (primary) criteria of theoretical correctness, generality, reliability and efficiency, then it is easy to see significant room for further work.

In recent years, the developers of professional modeling and optimization environments devoted a considerable effort to providing (relatively) 'bullet-proof' and user-friendly software products. Examples of such work are documented in relation to AMPL (Fourer et al., 1993, GAMS (Brooke et al., 1988), the LINDO Solver Suite (LINDO Systems, 1996), MPL (Maximal Software, 1998), or the advanced level Excel PSP Solvers (Frontline Systems, 2001), just to name a few. A similar level of development work and resulting software products will have to set the standards – at least for professional GO software – if the field wants to be fully recognized also by practitioners.

For illustration (and also because of its use in some numerical tests discussed later) let us mention here the LGO integrated modeling and solver system, as one of the professionally developed and maintained GO software products. The core of LGO is a suite of global and local scope solvers that can be activated in automatic and interactive operational modes. LGO can also be used as a standard (convex) solver applying only its local search components from a given starting point.

The software is available in several implementations. These (as of 2001) include a stand-alone MS Windows version equipped with a menu-driven user interface; a command-line version; and a 'silent operational mode' version that can be built into other modeling environments and user applications. LGO for Windows environments is documented in Pintér, 2000b; consult also the peer review by Benson and Sun, 2000. The tutorial (Pintér, 2001a) includes a demonstration of this program system, as well as several executable program examples. The documentation by Frontline Systems, 2001 is related to the Excel Solver platform implementation of LGO.

Figure 15.1 displays the LGO user interface (under MS Windows). The main menu items – Model Formulation, Solution, Result Analysis,



LGO

An Integrated Application Development
System for Continuous Global Optimization

(c) Janos D. Pinter 1986...2001
Pinter Consulting Services, Inc.
129 Glenforest Drive, Halifax NS, Canada B3M 1J2
Phone: +1-(902)-443-5910 Fax: +1-(902)-443-5910
E-mail: jdpinter@is.dal.ca WWW: is.dal.ca/~jdpinter

Demonstration Program Version
For further information, please contact
Pinter Consulting Services

Figure 15.1. LGO model development and solver system (MS Windows version): user interface.

together with built-in visualization, external program call and help features – effectively assist the development of GO models, without the need of leaving the LGO development environment.

4. Software testing

The thorough evaluation of optimization algorithms and software demands devotion, time and (hardware) resources, in addition to professional objectivity. This general remark is particularly valid with respect to global optimization software since – as already emphasized – GO literally encompasses all mathematical programming models. It is easy not only to 'fabricate' very challenging test problems, but also to find realistic GO problems that pose a formidable task for any algorithm of today (and of tomorrow). Since there is clearly no one universal algorithm that performs best in all categories of optimization problems, numerical experiments which compare new algorithms and software with existing ones complement the theoretical analysis of optimization methods. We will discuss the issue of selecting test problems, followed by a presentation of suggested evaluation criteria.

4.1 Test problems

In principle, any (well-formulated) practically motivated nonlinear optimization model may serve as a valid test for some suitable GO algorithm. However, many such models are too specific and complicated, making their reproduction cumbersome; and/or their details may be confidential. These circumstances may prevent their use in general purpose, comparative tests. Hence, the use of widely available, reproducible and transferable test problems is also important.

There is a considerable variety of standardized test problem suites available in some coded form (often in one of the traditional or more modern programming languages, or in specific modeling environments). These test problems frequently originate from real-world applications, although especially in the global optimization community a number of purely academic test problems are also in use.

Perhaps this is a suitable point to comment upon some types of academic test functions that should *not* be used – at least exclusively – to prove the applicability of some GO algorithm or software. Figure 15.2 serves to illustrate the point. (Note that these functions are actually often used in reported tests, but our purpose is to suggest something perhaps better, so that references to the original sources or to their usage are deliberately omitted.) The pictures show the test functions 'upside down' (i.e. the objective $-f$ is maximized), to see better their behavior around the global optimum. (This comment applies also to the subsequent Figures 15.3 to 15.5.)

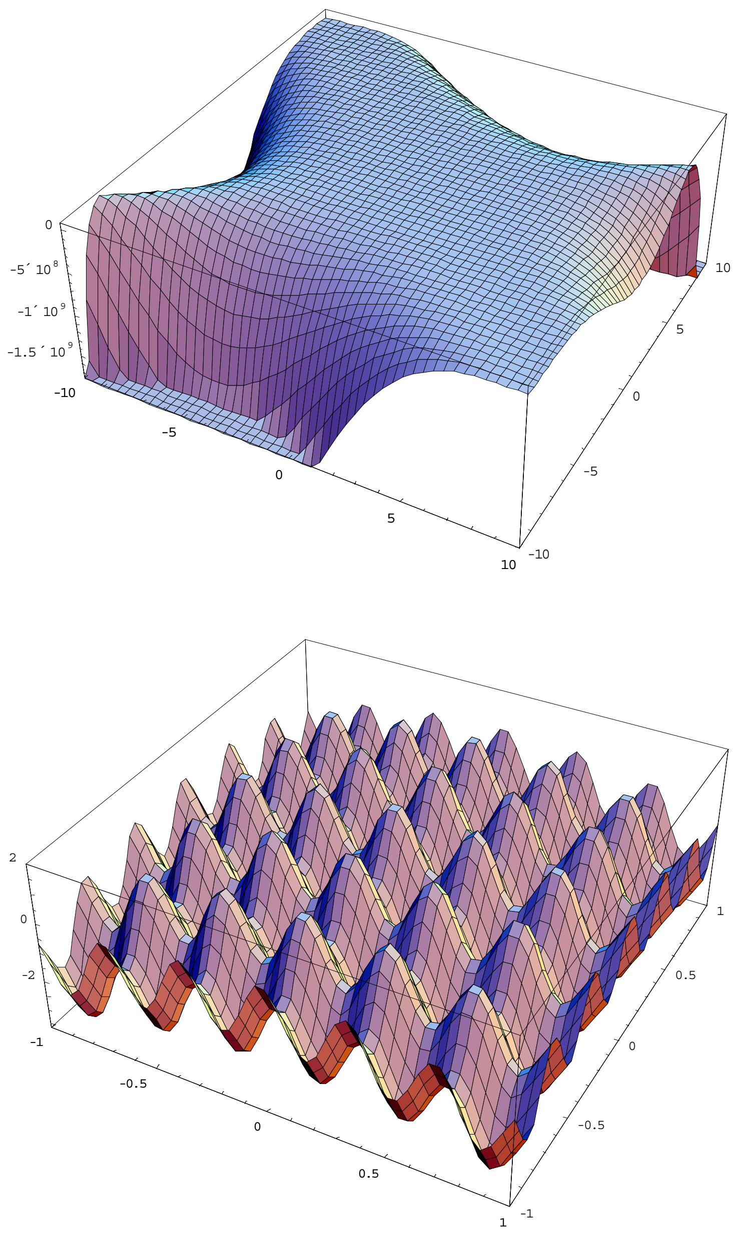


Figure 15.2. Two standard global optimization test function types.

The first graph in Figure 15.2 displays a function which is 'nearly convex' around the global optimum, with a very sizeable 'plateau' (region of attraction) around it. A magnification of this picture near to the optimum would indicate some more nonlinearity, but the key point is that for most nonlinear programming solvers – whether purely convex or global – such a function should not pose any difficulty.

The second graph in Figure 15.2 has a large number of pseudo-solutions, in addition to a singular global optimum. However, the search region is exactly symmetric around the optimal point, and this would give an almost 'ridiculous' advantage to any method that happens to sample also at the center of the search region. (This could occur, for instance, in the initialization phase of a branch-and-bound method, or in using a simple convex estimation procedure.) In the real world, if we have a model that possesses such remarkable symmetry, then any practical modeler would immediately look for ways to correct it (e.g., to reduce it to a smaller equivalent model). Furthermore, the pronounced symmetry of the test problem is not realistic at all. By all means, there exist models (for instance, energy models in computational chemistry) that possess symmetries and have a very large number of near-optimal solutions, but in these cases lexicographic ordering helps to reduce the solution space to rather different configurations, and their energy landscapes are far from simplistic. A perfectly symmetrical solution set typically spells some sort of modeling error, and poses also serious computational issues. (How would one know the number of solutions in advance? How many of these should be found? Are they truly equivalent?) A test problem which has overall 'looks' similar to the second graph in Figure 15.2 seems to be utterly 'idealized'.

To propose something a bit more challenging than the two function types discussed above, a relatively simple test function is displayed below, see Figure 15.3. (This function has been used for illustration in the GO software review Pintér, 1996b.)

The picture shows (again) a merely two-dimensional box-constrained GO problem: the function is the sum of four trigonometric terms with very simple linear arguments in two variables. However, the resulting function does not seem to have any obvious or easy structure, and hence should pose a more realistic challenge to GO solvers.

An even (much) better strategy is to generate randomized test functions that will have a randomly selected unique global solution (or a set of such solutions) and perhaps some other randomness in the model function definition(s). This way – assuming sufficiently complex function forms – the chance of 'tuning' some GO method to the test functions becomes much less, than for given deterministic test problems. The sit-

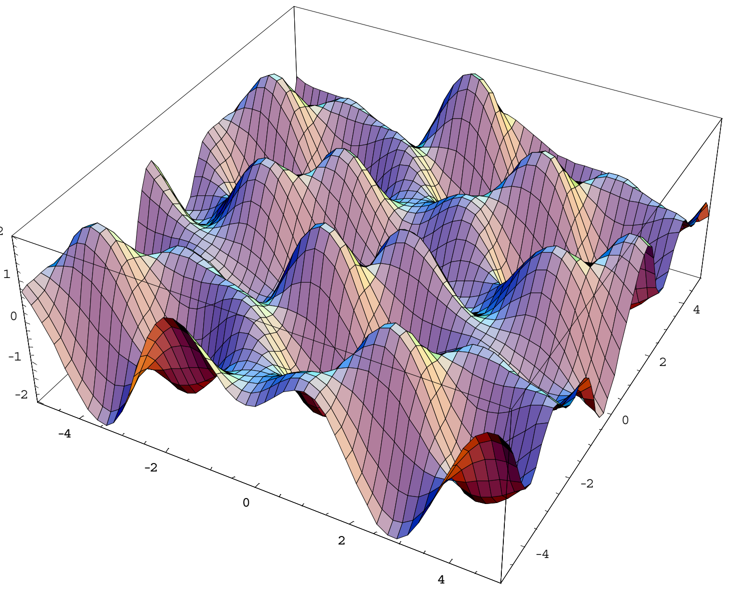


Figure 15.3. An illustrative test function (Pintér, 1996b).

uation also better corresponds to real life, in which problems to solve are received in some 'random' fashion.

To illustrate this point by examples, one can consider e.g., the following general test function type:

$$f(x) = s \sum_{i=1}^{n} (x_i - x_i^*)^2 + \sum_{k=1}^{k_{\max}} a_k \sin^2[f_k P_k(x - x^*)] \tag{15.2}$$

In (15.2), x^* is the unique global solution (chosen randomly for each function instance); $s > 0$ and $a_k > 0$ are scalars which can be changed, to make the test problem more or less difficult; $P_k(\cdot)$ are polynomials that vanish at the zero vector; and $f_k > 0$ are integer (frequency) multipliers, again to make the test more or less difficult.

For illustration, two different randomly generated instances of such test functions in the one- and two-dimensional cases are shown in Figures 15.4 and 15.5 respectively. All these models have a unique global solution, in addition to a large (exponentially increasing) number of local optima.

The last two figures in this section show model visualization examples generated by LGO (Windows version). Figure 15.6 displays a subspace projection of the merit (exact penalty) function in a standard LGO test problem. (This non-trivial model has 10 variables, and 6 nonlinear constraints.) The picture also shows the scatterplot of improving search points, and the projected location of the optimal solution in the variable pair subspace that can be interactively selected by the user.

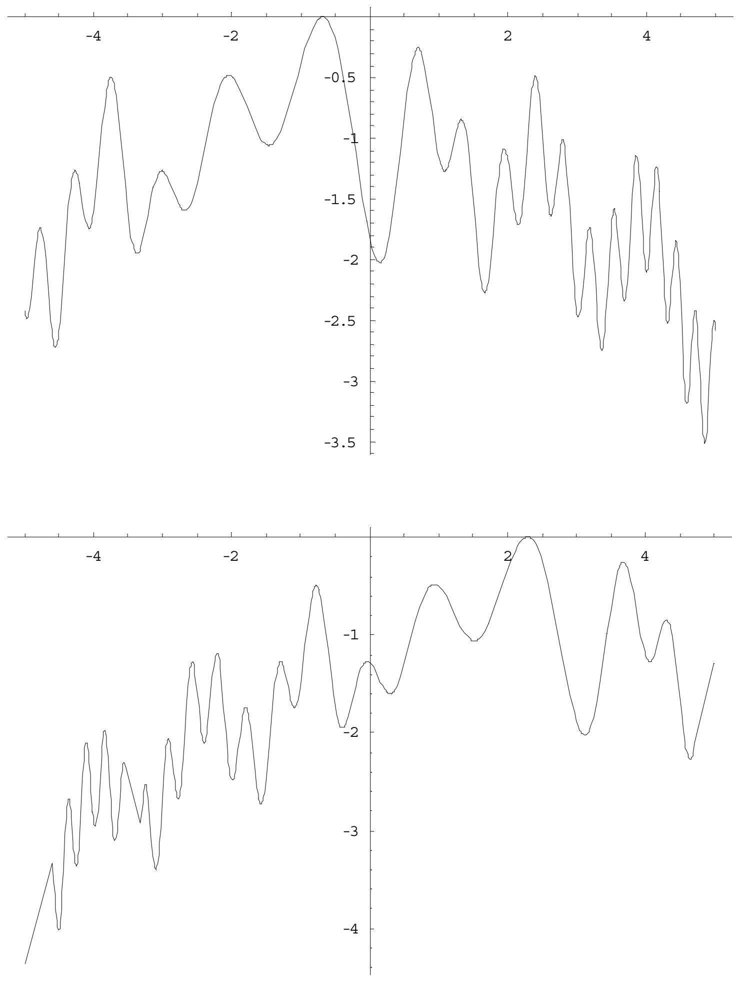
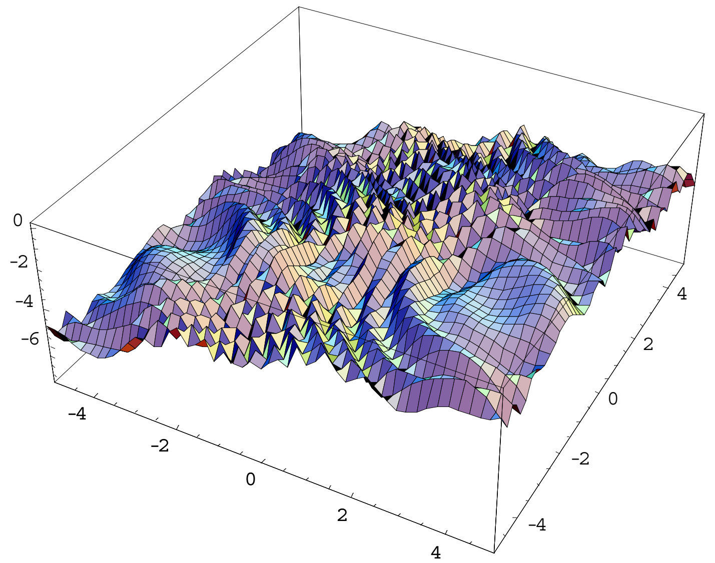


Figure 15.4. Two randomly generated one-dimensional global optimization test functions (see formula (15.2), $n = 1$).

An LGO feasibility plot is shown in Figure 15.7: again, the constraint and the variable subspace shown can be interactively selected by the LGO user. The picture clearly demonstrates that the optimal solution lies on the boundary of an active inequality constraint, as expected. (Note that in real world models often more complicated feasible set projections are seen than in this simple example.)



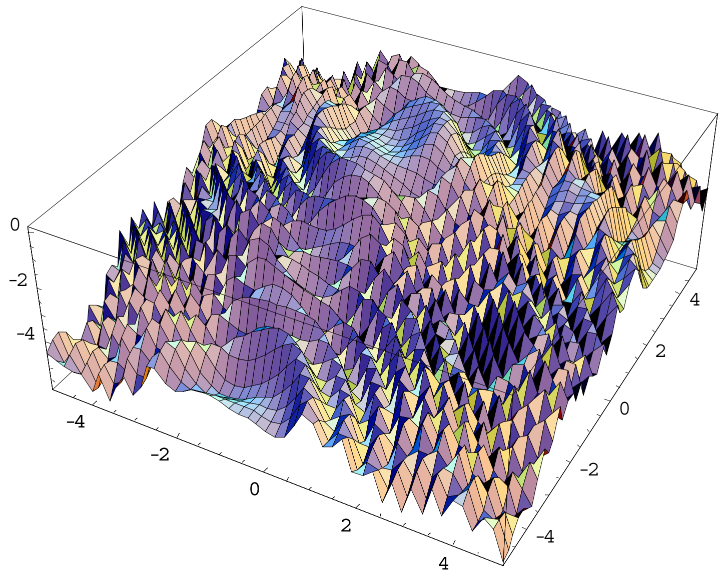
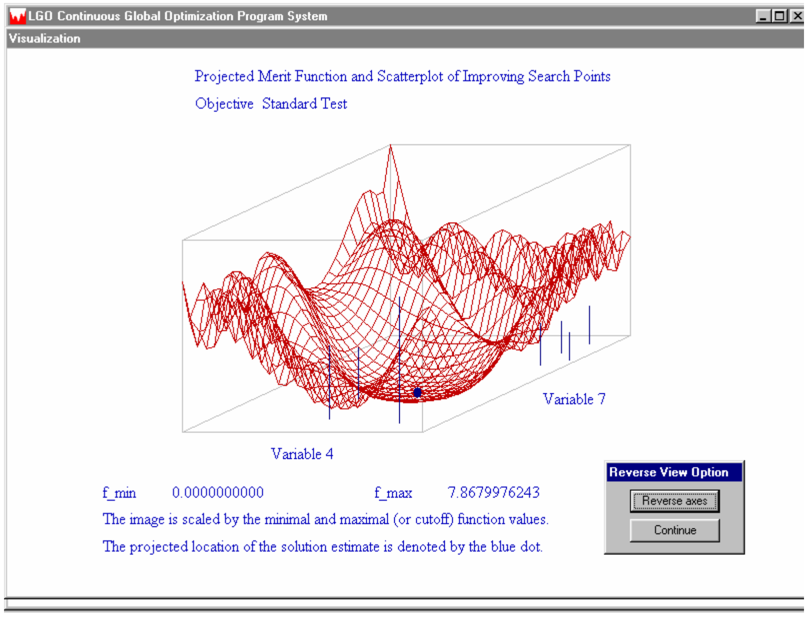


Figure 15.5. Two randomly generated two-dimensional global optimization test functions (see formula (15.2), $n = 2$).



Projected Merit Function and Scatterplot of Improving Search Points

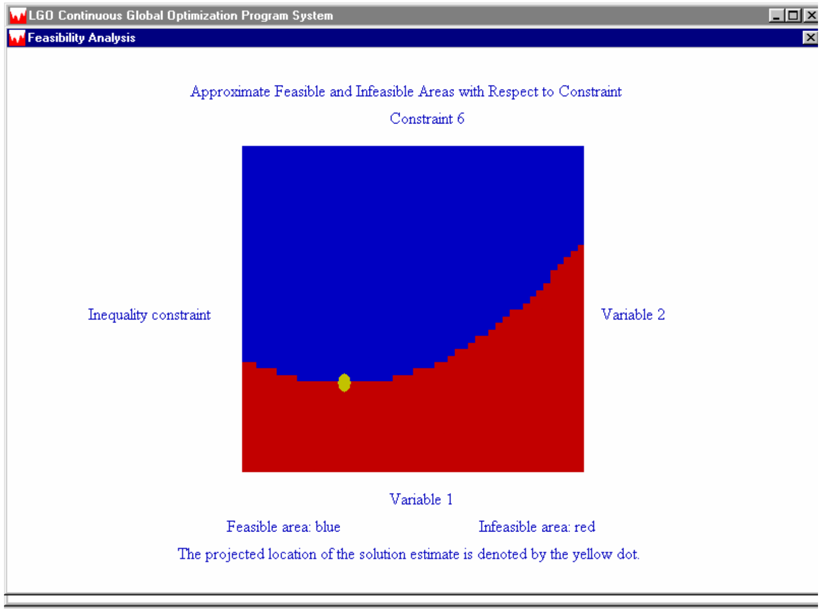
Objective Standard Test

f_min 0.0000000000 f_max 7.8679976243

The image is scaled by the minimal and maximal (or cutoff) function values.

The projected location of the solution estimate is denoted by the blue dot.

Figure 15.6. LGO model development and solver system: merit function visualization.



Approximate Feasible and Infeasible Areas with Respect to Constraint

Constraint 6

Feasible area: blue Infeasible area: red

The projected location of the solution estimate is denoted by the yellow dot.

Figure 15.7. LGO model development and solver system: feasibility visualization.

While obviously there is no recipe to produce a ‘perfect’ GO test problem suite, these pictures illustrate the substantial optimization challenge. The capability of solving models of high complexity (even in lower dimensions, with a good average behavior) seems to indicate quality algorithm and software development. It is also clear that the use of such – or similarly difficult – tests will illustrate the genuine need for GO methodology. (Evidently, traditional nonlinear solvers are not capable to handle the models shown in Figures 15.3 to 15.6, unless the search is started out from the immediate vicinity of the global solution.) Facing real challenges will motivate quality research into the right direction.

4.2 Test problem sources

At least in the past three decades, a considerable effort has been devoted to collecting and presenting test problems for nonlinear and global optimization. For collections of both convex and non-convex nonlinear programming test problems, consult, e.g., Moré et al., 1981, or Hock and Schittkowski, 1981. (Several other examples will be mentioned later on.)

The two volumes edited by Dixon and Szegö, 1975, 1978 discuss several early attempts to solve GO models: the contributed chapters in these volumes also include a few classical GO tests. Floudas and Pardalos, 1990, Jansson and Knüppel, 1992, Floudas et al., 1999 provide further and much more extensive suites of GO test problems.

On the Web, the pages of Fourer, 2001, Mittelmann and Spellucci, 2001, Neumaier, 2001, and the sites of Argonne National Laboratories, 1993 and Sandia National Laboratories, 1997 present useful benchmarking information. Neumaier also provides detailed discussions of several test problem types and suggestions on reporting numerical test results.

Note that the test problem collections mentioned above include practically motivated models, as well as a number of *ad hoc* test problems suggested by researchers in optimization.

A concise and very broad classification of GO test problems is the following:

- unconstrained nonlinear optimization (often using also starting point information)

- bound constrained nonlinear and global optimization

- nonlinear and global optimization under general constraints

- problems with an ‘exploitable’ special structure.

Regarding this primary classification, note that in global optimization bounds are required, to guarantee the existence of the solution(s). Furthermore – in very general terms – the difficulty of GO tests increases (often significantly), as we move from bound constrained to general constrained models.

The special structure referred to may or may not make the model easier, but it is probably worth to consider in the solution procedure. For instance, in solving systems of nonlinear equations it is valuable information that the global optimum value is zero (assuming that the system has at least one solution, and that in solving the problem we globally minimize the norm of the total error).

Note also that – in accordance with our earlier related remarks – combinatorial optimization (CO) tests could also be considered in a CGOP framework, since all CO models can be equivalently formulated as CGOPs. This would add a very significant model-group to the above classification. However, CO models will not be discussed here. Further refinements of this typology can also be useful, of course: for example, one could distinguish models that have only linear constraints; models with reverse convex, explicitly known differential convex or Lipschitz structure, and so on.

The model classification leads to a challenging issue. To establish objective guidelines and benchmarks, a GO solution approach and its software implementation should be subject to extensive testing, on *all* model types encompassed by its solver(s). Neumaier, 2001 correctly states that any reasonable GO algorithm should be able to cope with standard – convex nonlinear, or even linear – problems, without a huge extra overhead, when compared to suitable local search methods. This is a perfectly understandable criterion that, however, is not easy to meet. For example, one can not expect a 'general purpose' continuous GO solver to be competitive with a large scale linear programming (LP) solver, unless it also includes some model processing mechanism that recognizes the special structure of the LP, and then uses the corresponding specialized solver option. A natural way of handling simultaneously several specific classes of GO models encompassed by a broader class – and to tailor the solution strategy accordingly – is the pre-classification of models submitted to the solver system in question. However, this may either give the user an extra – and not necessarily welcome – burden when submitting models to the solver, or it requires a software development effort to support automated model classification and pre-processing.

Note also that – as argued above – to achieve rigorous global convergence and acceptable numerical efficiency are partially contradicting criteria. In solving GOPs – just as in most combinatorial models –

the global optimum is often 'found' (that is, well approximated) in an early phase of the search, and a large part of the computational effort is spent on verifying the (approximate) global optimality of the solution found. The consideration of proper stopping rules is again far from trivial, since these are mostly based on postulated properties of the model class of which the concrete problem is an instance. Rigorous GO methods typically spend a very significant effort on guaranteeing (deterministic) bounds, while heuristic GO strategies frequently do not address this issue at all. According to (my own) implementation and application experience, suitable statistical decision rules lead to both satisfactory precision and acceptable efficiency, in most tests and practical problems. The proper implementation of such concepts is, however, quite technical – and there is always a small chance of premature algorithm termination, or (more often) of 'wasting' significant amounts of search effort. Global optimization problems are and will remain inherently difficult.

4.3 Evaluation and comparison of test results

4.3.1 Related earlier work. The objective experimental assessment of various nonlinear optimization algorithms has been of significant practical interest for at least several decades. We will briefly review some benchmarking studies: for more details, consult the original sources referred below, the already mentioned Web sites, as well as Khompatraporn et al., 2001.

Colville, 1968 made a pioneering attempt at comparing the performance of nonlinear optimization algorithms. Eight test problems were submitted to 30 codes. The test models were all based on practically motivated problems received from large companies with access to (mainframe) computers: IBM, Shell, Electricité de France, and so on. The participants were required to submit the result of their 'best effort' on each problem, and the corresponding program execution time. This work was a useful step towards establishing objective evaluation criteria. The study had some serious flaws, however, since the exact conditions of the tests were rather loosely specified – computational nonlinear optimization was a new field indeed.

Eason and Fenton, 1974 performed a comparative study of 20 optimization codes based on solving 13 test problems (the latter included some of Colville's tests, and a few rather simple new ones). The study primarily focused on penalty-type methods and all of the computations were performed on the same computer. The study did not include other types of methods available at the time, and some of the models were too easy (!) to solve. Around the same time, Dembo, 1974 provided the first collection of geometric programming test problems and their solutions: these models were based on chemical and design engineering problems.

A comparative analysis of nonlinear programming methods was done by Sandgren and Ragsdell, 1980: the methods included are described in their paper, and are discussed also by Reklaitis et al., 1983. This study was more thorough and systematic, than the previous ones: it was based on solving 31 models collected from the earlier three studies, and the results were also analyzed in more precise detail. (Relative performance measures, in terms of normalized solution speed were also used.) The findings were aggregated in the form of utility curves.

A major computational study of nonlinear programming algorithms and software was completed by Schittkowski, 1980. The study included 20 different codes: these were applied to a set of 180 randomly generated problems with predetermined characteristics and using multiple starting points. The codes were evaluated based upon their efficiency, reliability, convergence, ability to solve degenerate or ill-posed problems, ability to solve indefinite problems, sensitivity to starting points and problem variations, and ease of use. Then a(n obviously somewhat subjective) weighting scheme was applied to arrive at a final, aggregated ranking.

In addition to the studies listed above, in recent years a number of challenging tests and real-world problems were proposed and investigated in articles appearing in the *Journal of Global Optimization* and elsewhere. This journal is also a good source of some real-world problem types. Examples of challenging problems with a practical flavor presented in *JoGO* are the test function classes proposed by Schoen, 1993, Mathar and Žilinskas, 1994, the circuit design model discussed by Ratschek and Rokne, 1993, or the minimal (log-potential) energy problem posed by Pardalos, 1995.

In summary, computational studies for global optimization should be based on these and similar – or even more challenging – tests. Specifically, they should consider multiple measures of performance, a comprehensive selection of test problems, and a well-defined procedure to conduct the evaluations.

4.3.2 Evaluation criteria. A report on computational experiments should cover the following aspects:

- model or problem-class statement (model dimensionality, number of constraints, feasible region type, model parameters and generation rules (if applicable)

- number of global and local optima (if known); optimum value or best solution known
- practical background or other reason for choice (when available)
- related earlier work (if available, for comparative assessment purposes)
- solution approach used in test: algorithm(s) and their implementation, concise outline of code design, algorithm parameterization, and input data structure
- hardware platform(s), operating system(s), and software environment (compiler(s) and possible additional software) used in test
- testing methodology: exact description of all performance measures used (standardized runtime based on model function evaluation time units, number of function/gradient/Hessian evaluations, required precision and other stopping criteria, etc. as applicable)
- report of quantitatively well-defined successes and failures, with added explanation (when available)
- analysis of parameterization effects
- statistical characteristics for randomized problem-classes and/or for randomized solution methods
- summary of results, in tabular and graphical forms (as suitable)
- additional comments and recommendations.

Again, without going into a detailed discussion, one can observe that the criteria listed are often partially conflicting. Consider, for instance, the tradeoff between the solution accuracy required and a (possibly pre-set) maximal number of objective function evaluations. In such cases, concepts and techniques used in multi-objective optimization – specifically including the selection of non-dominated, Pareto optimal algorithms for a given set of test problems – can be brought to the subject. Such analysis can lead to valuable insights regarding the comparative strengths and weaknesses of optimization algorithms.

4.4 An example of software evaluation using randomized test problems

As an example, a set of randomized test functions will be used to present an illustrative computational report. Specifically, the random test function class proposed in Section 4.1 has the following general form:

$$f(x) = s\sum_{i=1}^{n}(x_i - x_i^*)^2 + \sum_{k=1}^{k_{\max}} a_k \sin^2[f_k P_k(x - x^*)]$$

A relatively simple instance of $f(x)$ is defined, for fixed n, as follows:

- $s = 0.025n$: scaling factor (by definition, depends on model dimension)
- $l = -5$, $u = 5$: lower and upper bounds, identical for each component of x
- x^*: randomly generated solution, chosen from a uniform distribution on $[l, u]$
- $a_k = f_k = 1$ $(k = 1, \ldots, k_{\max})$: amplitude scalars and frequency multipliers; $k_{\max} = 2$ is chosen throughout

and

$$\begin{aligned} P_1(x - x^*) &:= \sum_{i=1}^{n}(x_i - x_i^*) + \sum_{i=1}^{n}(x_i - x_i^*)^2 \\ P_2(x - x^*) &:= \sum_{i=1}^{n}(x_i - x_i^*) \end{aligned}$$

are polynomial 'noise terms' with the required properties.

In spite of the relative simplicity of this example, these functions already lead to non-trivial test problems. In fact, Figure 15.5 shows different two-dimensional instances of this test function type. The large number of local optima seen on these pictures clearly indicates the need for using proper global scope search approaches.

For illustration, the LGO solver system is used to solve instances of this box-constrained test problem, for $n = 1, \ldots, 10$. The details of these tests are summarized below.

4.4.1 Evaluation criteria.

- Model or problem-class statement (model dimensionality, number of constraints, feasible region type, model parameters): see above.
- Number of global optima: 1, the corresponding solution value is 0.
- Number of local optima: *a priori* unknown; increases exponentially as a function of n.

- Practical background or other reason for choice: choice is motivated by the importance of trigonometric functions in general modeling practice, and by the visible difficulty of (even) low-dimensional instances.
- Related earlier work: not directly available.
- Solution approach used in test: the LGO software is applied in its Microsoft Windows-style implementation.
- Algorithms used: automatic branch-and-bound based global search, followed by the built-in sequence of local search methods.
- Input data structure: model functions are defined in the so-called user function file; bounds and several key algorithm parameters are defined in the so-called input parameter file. Consult the LGO user documentation (Pintér, 2000b, Pintér, 2001a) for details.
- Algorithm parameterization and stopping criteria used:
  - acceptability threshold: 0.1 (upon attaining a search point in which the objective function value does not exceed this value, the global search phase is terminated, and local search is launched from this point)
  - tolerance in satisfying the Kuhn-Tucker conditions (i.e., the required precision of the KT equation for the objective/merit function, in finishing local search phase): 10^{-6}.
- Hardware platform: Pentium-Pro 200 MHz personal computer.
- Operating system: Windows NT Workstation, Version 4, Service Pack 6.
- Software environment (compiler) used: Lahey Fortran 90, Version 4.5 (Lahey Computer Systems, 1999).
- Timings: in these illustrative runs, only real runtimes are measured.
- Number of model function evaluations: given below, note that higher order information – direct gradient/Hessian evaluations – is not used by LGO. (Finite difference gradient approximations are used in the local search phase.)
- Report of successes and failures: all test models were solved up to at least 10^{-10} precision, in terms of the optimal function value.

- Analysis of parameterization effects: not done in these illustrative runs. (One testproblem instance is solved just once, for each n.)
- Statistical characteristics for randomized problem-classes and/or for randomized solution methods: not done in these illustrative runs.
- Summary of results, in tabular form: see below.

Table 15.1. Summary of results

Number of variables	*Number of function evaluations*	*Runtime (seconds)*
1	36	0.02
2	760	0.07
3	2,222	0.11
4	33,500	0.55
5	6,705	0.19
6	51,590	1.05
7	14,963	0.42
8	19,665	0.57
9	120,700	3.13
10	32,656	1.02

  - *Optimum value* (found by LGO): less than 10^{-10}, in all runs above (vs. the theoretical optimum value of zero).
  - *Optimal solution*: the intrinsic random number generator of the compiler was reset in each run (in fact, the same can be done also within LGO), to assure the reproducibility of results in repeated runs. For illustration, the randomly generated 'exact' solution vector $x^{(10)^*}$ and the estimated solution vector $x_{LGO}^{(10)^*}$ for $n = 10$ are shown below. The solution for all other $n < 10$ cases can be derived, by simply 'truncating' the vector $x^{(10)^*}$.

The randomly generated solution vector (up to 10^{-10} precision) equals:

$$x^{(10)^*} = \\ (2.3139631748, -2.4219650030, 0.3140348196, 3.7724709511, \\ -3.6655817926, 2.1954119205, -0.1030561328, -0.5903974175, \\ 4.7747606039, -0.0936746597).$$

The solution found by LGO for $n = 10$ equals:

$$x_{LGO}^{(10)^*} =$$

$$(2.3139630534, -2.4219648189, 0.3140349154, 3.7724704203,
-3.6655820156, 2.1954119704, -0.1030561668, -0.5903967541,
4.7747602149, -0.0936743558).$$

Note that the solver precision obtained could probably be somewhat increased, but it seems to be adequate for illustrative purposes.

With a view towards more detailed future experiments – and to assist other researchers in using this test problem class – the exact (randomly generated) solution vector for $n = 100$ is also included below:

$$x^{(100)^*} =$$

$$(2.3139631748, -2.4219650030, 0.3140348196, 3.7724709511,
-3.6655817926, 2.1954119205, -0.1030561328, -0.5903974175,
4.7747606039, -0.0936746597, 2.7728712559, 4.5591384172,
2.0799285173, -1.1147031188, 3.9711540937, -3.2205986977,
-0.0075784326, 4.6699059010, 2.4387961626, -0.2425742149,
4.4414567947, -4.1198746860, -4.6567079425, 0.0459510088,
4.4965875149, -0.9529367089, -4.6436333656, 3.9077073336,
4.3915367126, 1.9217687845, -0.8288607001, -3.5522249341,
-2.4924343824, 3.8403457403, -1.8877500296, 0.1746726036,
-3.1836619973, -2.1224305034, 2.4154454470, 2.8571480513,
-0.3390669823, -0.8383473754, -1.6837617755, -3.9999020845,
-0.6876027584, -3.0761498213, -3.2940532267, 4.7059488297,
-4.5894131809, -4.7277861834, -0.6674996018, 3.6198598146,
-4.0551695973, 1.8542295694, -4.5681066811, 4.5124012232,
-4.3568804115, 3.5457694530, 0.1703751087, -4.4784013927,
-2.7483130991, 0.2916604280, 3.7373530865, -0.0596478581,
3.4482359886, 4.1347974539, 2.0059490204, 4.4946092367,
-2.1391028166, -0.2948442101, -2.5199368596, -4.6839887649,
-0.5125197768, 3.2239007950, -2.0734250546, 3.4358721972,
4.8674798012, -2.1683278680, 1.1255329847, -2.0943489671,
-4.1337385774, -0.6612965465, -4.5193142444, 2.9797691107,
-1.6175588965, -3.8230604678, -4.6569699422, 0.3034192324,
-4.0681108832, 2.3098903894, -1.9086727500, -0.4662388563,
-2.0102167130, 1.8418258429, 2.4033385515, -4.7985471413,
1.2285727262, -1.4984077215, -3.6602240801, -0.8291804790).$$

4.4.2 Additional notes and recommendations. Although it would be far-fetched to draw very general conclusions on the basis of solving just a few test examples from the same artificially 'fabricated' model-class, one can make a few valid observations. These are summarized below.

- The 'blind' (that is, fully automatic) usage of LGO leads to successful numerical solution, for relatively small instances from a randomized, non-trivial test function-class.

- The solution times (using rather modest – as of 2001, nearly 4 years old technology based – hardware) are quite reasonable, but one can observe the general tendency of (as expected, rapidly and erratically) increasing runtimes. Empirical functional relations regarding this point could be derived in the usual manner. Namely, one can fit a parameterized curve to a set of runtime (or numerical effort) *vs.* model size data-pairs (see e.g. Pintér, 1996a, Chapter 4.1): however, this would require a more extensive set of runs, for given well-defined classes of problems.

- Run characteristics – most notably, the number of function evaluations needed to attain a prefixed threshold solution quality in the global search phase – may vary, perhaps significantly. Of course, the threshold parameter could be increased to 'cut off' the global search, but there is definitely a trade-off between (perhaps much) faster solution times and solver robustness in finding the global optimum. In certain applications, rigorous solution is the key aspect, while in many practical problems solution speed is a dominant factor to consider. (Note that in LGO the maximal global search effort can be effectively limited by two complementary input parameter settings.)

- LGO runs executed in the Windows environment (due to features of the graphical user interface, such as an online iteration counter, and so on) are measurably slower, than the same runs done using e.g. a 'silent' executable version of LGO. (At the same time, this environment can be rather convenient to use, and it also has some added features such as model visualization.)

- LGO can be used in several flexible ways, including interactive solver choices, and changes in the input parameter file. In solving difficult models, it is most advisable to try various solver options, and also to 'tweak' the input parameters, in order to obtain reasonable approximations of the global solution within an acceptable time frame. To illustrate this point, LGO was used to solve approximately the 100-variable instance of the test problem shown above. Applying interactive search mode, the following numerical result was obtained:

– Estimated optimum: 1.1201451967

– Total number of function evaluations: 23732

– Solution time (seconds): 20.25

Note that this solution time also includes user input (search mode selection) operations. Using the visualization and detailed result generation capabilities of LGO, it is easy to verify that this solution is well within the best 1% of all function values in the 100-dimensional search interval. Furthermore, one can also verify that the corresponding solution vector is within a few percent of the true solution, in all of its components. Indeed, simply repeating an LGO run within a smaller search 'box' around the approximate solution obtained leads to a precise estimate of the true optimum (similarly to the small-dimensional test results shown above).

The coded test functions, including all numerical data presented above, are available from the author upon request.

5. Some prominent application areas

Since GO problems are literally ubiquitous in scientific, engineering and economic modeling and decision-making, one could easily devote entire books to discussing such applications in detail. For the purposes of the present work, we shall only touch upon a few important general application-types (in this section) and then list a large number of further application areas (in Section 6).

The Reader will notice overlaps among the problems studied in different works cited later on: this often indicates important applications appearing in various specific contexts. Please also keep in mind the test case studies discussed above and the recommended Web sites for further examples.

5.1 'Black box' systems optimization

The realistic analysis of industrial design, chemical or pharmaceutical production, biological, environmental (as well as many other) processes often requires the development of sophisticated coupled systems of submodels. Such descriptive model-systems are then connected to suitable optimization (solver) modules. For examples of various complexity, consult for instance, Dorfman et al., 1974, Loucks et al., 1981, Haith, 1982, Somlyódy and van Straten, 1983, Beck, 1985, Beck, 1987, Pintér, 1996a, Papalambros and Wilde, 2000, Edgar et al., 2001. We shall illustrate this point below, by briefly discussing a modeling framework for river water quality management: for additional details, see Pintér, 1996a and references therein.

Assume that the ambient water quality in a river at time t is characterized by a certain vector $s(t)$. The components in $s(t)$ can include, for instance the following descriptors: suspended solids concentration, dissolved oxygen concentration, biological oxygen demand, chemical oxygen demand, concentrations of micro-pollutants and heavy metals, and so on. Naturally, the resulting water quality is influenced by a number of controllable and external factors. These include the often stochastically fluctuating (discharge or non-point source) pollution load, as well as the regional hydro-meteorological conditions (streamflow rate, water temperature, etc). Some of these factors can be directly observed, while some others may not be completely known. In a typical integrated model development process, sub-models are constructed to describe all physical, chemical, biological, and ecological processes of relevance. (As for an example, one can refer to the Streeter-Phelps differential equations that approximate the longitudinal evolution of biological oxygen demand and dissolved oxygen concentration profiles in a river; consult for instance, Orlob, 1983.)

In order to combine such system description with management (optimization) models and solvers, one has to be able to assess the quality of all decisions considered. Each given decision x can be related, *inter alia*, to the location and sizing of industrial and municipal wastewater treatment plants, the control of non-point source (agricultural) pollution, the design of a wastewater sewage collection network, the daily operation of these facilities, and so on. The analysis typically involves the numerically intensive evaluation of environmental quality (e.g., by solving a system of partial differential equations, for each decision option considered). The – possibly more realistic – stochastic extensions of such models will also require the execution of a (large) number of Monte Carlo simulation cycles.

Under such or similar circumstances, environmental management models can be very complex, consisting of a number of 'black box' sub-models. Consequently, the general conceptual modeling framework

$$\begin{aligned} \min \quad & TCEM(x) \\ \text{subject to} \quad & EQ_{\min} \leq EQ(x) \leq EQ_{\max} \end{aligned} \tag{15.3}$$

$$TF_{\min} \leq TF(x) \leq TF_{\max}$$

in which

- $TCEM(x)$: total (discounted, expected) costs of environmental management
- $EQ(x)$: resulting environmental quality (vector)
- $EQ_{\min}$ and $EQ_{\max}$: vector bounds on 'acceptable' environmental quality indicators
- $TF(x)$: resulting technical system characteristics (vector)
- $TF_{\min}$ and $TF_{\max}$: vector bounds on 'acceptable' technical characteristics

may – and often will – lead to multiextremal model instances requiring the application of suitable GO techniques.

Numerous other examples could be cited, similarly to the case outlined above. These may involve the solution of systems of (algebraic, ordinary or partial differential) equations, and/or the statistical analysis of the chemical, biological, environmental etc. system studied. For further examples – including industrial wastewater management, regional pollution management in rivers and lakes, risk assessment and control of accidental pollution – in the context of global optimization consult, for example, Pintér, 1996a, and references therein.

5.2 Calibration of descriptive system models

The incomplete or poor understanding of environmental – as well as many other complex – systems calls for descriptive model development as an essential tool of the related research. The following main phases of quantitative systems modeling can be distinguished:

- identification: formulation of principal modeling objectives, determination (selection) of suitable model structure
- calibration: (inverse) model fitting to available data and background information
- validation and application in analysis, forecasting, control, management

Consequently, the 'adequate' or 'best' parameterization of descriptive models is an important stage in the process of understanding complex systems. Practically motivated discussions of model calibration are presented by Beck, 1985, Pintér, 1996a, and Hendrix, 1998.

A fairly simple and commonly applied instance of the model calibration problem can be stated as follows. Given

- a descriptive system model (of a lake, river, groundwater or atmospheric system, etc.) that depends on certain unknown (physical, chemical) parameters; their vector is denoted by x
- the set of *a priori* feasible model parameterizations D
- the model output values $y_t^{(m)} = y_t^{(m)}(x)$ at time moments $t = 1, \ldots, T$
- a set of corresponding observations y_t at $t = 1, \ldots, T$
- a discrepancy measure denoted by f which expresses the overall distance between the vectors $\{y_t^{(m)}\}$ and $\{y_t\}$.

Then the model calibration problem can be formulated as

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in D \end{aligned} \tag{15.4}$$

where $f(x) := f\left\{\{y_t^{(m)}(x)\}, \{y_t\}\right\}$. Quite typically, D is a finite n-interval; furthermore, f is a continuous or somewhat more special (smooth, Lipschitz, etc.) function. Additional structural assumptions regarding f may be difficult to postulate however, due to the following reason. For each fixed parameter vector x, the model output sequence $\{y_t^{(m)}(x)\}$ may be produced by some implicit formulae, or by a computationally demanding numerical procedure (such as e.g., the solution of a system of partial differential equations). Consequently, although model (15.4) most typically belongs to the general class of continuous GO problems, a more specific classification may be elusive. Therefore one needs to apply a GO procedure that enables the solution of the calibration problem under the very general conditions outlined above.

To conclude the brief discussion of this example, note that in Pintér, 1996a several variants of the calibration problem statement are studied in detail. Namely, the model development and solver system LGO is applied to solve model calibration problems related to water quality analysis in rivers and lakes, river flow hydraulics, and aquifer modeling.

5.3 Dynamic modeling of complex systems

Dynamic (time-dependent) control problems arise in many scientific, engineering and economic investigations: their subsequent multi-stage discretization leads to standard mathematical programming models. A good exposition on control theory – including numerous examples of applications – is presented e.g. by Sethi and Thompson, 2000. To illustrate this very general problem-type, a simple fish stock management example – received from Erhan Erkut, University of Alberta, and discussed in Pintér, 2001a – will be presented below.

We shall consider the following problem. In a lake there is a (single-species) fish population that can be profitably managed, i.e. caught and then sold. Given a suitable growth process model, one wants to find the optimal stock management policy for a fixed time period.

To keep things simple in this illustrative example, we shall assume that the fish population grows from January to July, fishing takes place in July, and the remaining population simply carries over to the next year. It is to be noted immediately that – albeit non-trivial – the description below is much simplified indeed. Namely, it could be refined and extended in numerous respects, including for instance multiple (and quite possibly competing) species, more realistic individual population growth and extinction models, and so on. The inclusion of these elements in a more sophisticated model would even more motivate the use of a suitable GO approach.

Model input data:

- $t = 1, \ldots, T$: time periods
- i: interest rate
- a, b: parameters in population growth model
- $pop_{1,Jan}$: initial population
- $pop_{t,LB}$: lower bound to avoid stock depletion at any time
- $pop_{t,UB}$: upper bound of stock (carrying capacity of lake)
- $pop_{T,LB}$: prescribed final stock (in January of year $T+1$)
- $catch_{t,\max}$: prescribed maximal catch quota in any given time period.

We wish to determine the sequence of values $catch_t$ ($t = 1, \ldots, T$) so that it maximizes the net present value of the total catch:

$$\max \sum_{t=1}^{T} \{d_t \cdot catch_t\}, \quad \text{in which} \quad d_t = 1/(1+i)^t. \tag{15.5}$$

We shall postulate the explicit box constraints $0 \leq catch_t \leq catch_{t,\max}$, for each t.

The yearly natural population change is described as follows. The population grows at a rate of $a\%$ minus a factor that increases with the population (to model a congestion and saturation effect). We also include the explicit upper bound $pop_{t,UB}$ in the description:

$$\begin{aligned} &pop_{t,July} = f(pop_{t,Jan}, a, b) = \\ &\qquad \min[pop_{t,Jan} + pop_{t,Jan}(a - bpop_{t,Jan}), pop_{t,UB}] \quad t = 1, \ldots, T. \end{aligned}$$

The catch rate is possibly bounded by the required minimum carry forward stock:

$$catch_t \leq pop_{t,July} - pop_{t,LB}.$$

The population balance for the next year is simply expressed by

$$pop_{t+1,Jan} = pop_{t,July} - catch_t \quad t = 1, \ldots, T.$$

This way, the required stock at the end of time horizon (or, equivalently, at the beginning of time period $T + 1$) satisfies the constraint

$$pop_{T+1,Jan} \geq pop_{T,LB}.$$

In spite of the simplifications mentioned, this model is far from trivial and clearly illustrates the need for applying optimization tools. Local scope solvers will fail to produce good solutions, unless started from a 'sufficiently close guess' of a favorable stock management policy (as noted also by Erkut). The GO approach demonstrated in Pintér, 2001a leads to a close approximation of the global optimum (which – under suitable model parameterization – can be analytically determined for this simple model). As observed above, more realistic model extensions – which consider several competing fish species and more sophisticated population models – definitely require the use of a genuine GO approach.

5.4 Optimal object arrangement (configuration) problems

The feasible and 'well-balanced' distribution of points or other (two-, three- or higher dimensional) objects over a given area, surface or in an embedding volume is of significant interest in a variety of scientific investigations. These are related to applied mathematics (computational complexity, numerical approximation, packing problems), physics (multi-body potential energy models, celestial mechanics, electrostatics, crystallography), computational chemistry and biology (molecular structure modeling, viral morphology), as well as to engineering design issues.

For related expositions consult, for instance, Conway and Sloane, 1988, Greengard, 1988, Pain, 1994, Dean, 1995, Pardalos et al., 1996, Neumaier, 1997, Neumaier, 1999, Hartke, 1999, Wales and Scheraga, 1999, Floudas and Pardalos, 2000 and references therein.

The quality of configurations is typically expressed by a context-dependent criterion function: such objectives often have a very large – exponentially increasing – number of local optima. Therefore global scope search methodology is needed to analyze and solve such problems.

For illustration, we shall consider the problem of 'optimal' point distributions on the surface of the unit sphere in $\mathbb{R}^3$. In the context of spherical point arrangements, Saff and Kuijlaars, 1997 present a concise review of the theoretical background for several model versions. The exposition below draws mainly on Pintér, 2000a, where illustrative numerical results are also presented regarding the model versions introduced.

The general modeling framework is the following. Given the unit sphere B in $\mathbb{R}^3$, and a positive integer n, we want to find an n-tuple of unit vectors

$$x(n) = \{x_1, \ldots, x_n\} \quad x_i = (x_{i1}, x_{i2}, x_{i3}) \quad \|x_i\| = 1$$

(where $\|\cdot\|$ is the Euclidean norm). The components x_i are distributed on the surface S of B such that $x(n)$ optimizes the value of a given criterion function.

Let us denote by $d_{jk} = d(x_j, x_k) = \|x_j - x_k\|$ the distance between points x_j and x_k. We shall consider all such pair-wise distances d_{jk} $1 \le j < k \le n$, when defining the quality of point arrangements $x(n)$ by the subsequent formulae (15.6) to (15.9). Obviously, the optimization is related to all possible point configurations over S.

Four frequently considered model versions will be presented: these are distinguished by their respective objective functions given below.

Elliptic Fekete problem.

$$\max \prod_{j,k} d_{jk} \quad \text{subject to} \quad x(n) \in S(n) \tag{15.6}$$

(i.e. $x_i \in S$, $i = 1, \ldots, n$). For reasons of better numerical tractability, one can apply a logarithmic transformation. Hence, the objective in (15.6) is replaced by the equivalent logarithmic potential function

$$\max \sum_{j,k} \log d_{jk} \quad \text{subject to} \quad x(n) \in S(n). \tag{15.6$'$}$$

Fekete problem.

$$\min \sum_{j,k} d_{jk}^{-1} \quad \text{subject to} \quad x(n) \in S(n). \tag{15.7}$$

Note that this objective function corresponds to the classical Coulomb potential model.

Power sum problem.

$$\max \sum_{j,k} d_{jk}^{a} \quad \text{subject to} \quad x(n) \in S(n) \tag{15.8}$$

(where $a > 0$ is a given model parameter).

Tammes (hard-spheres) problem.

$$\max \min_{j,k} d_{jk} \quad \text{subject to} \quad x(n) \in S(n). \tag{15.9}$$

Notice the close relation of this last model-type to more general object packing problems.

One can also observe that the criterion functions in (15.6) to (15.9) pose various levels of numerical difficulty. Models (15.6′) and (15.7) need the exclusion of points that are 'too close' to each other, to avoid poles and implied numerical problems; (15.8) seems to have the analytically easiest objective function; and model (15.9) seems to be the hardest, especially since its objective is non-differentiable.

Let us note finally that numerous other (scientifically meaningful) objective function forms could also be considered. As a rule, such functions possess a high degree of symmetry, induced by equivalent permutations of point arrangements. The ambiguity of solutions can be excluded by imposing, e.g., the lexicographic arrangement of the points in a complete configuration $x(n)$.

In Pintér, 2000a a general global optimization framework is suggested to solve these problems. To illustrate the viability of this approach, the model development and solver system LGO is applied to the four different model versions stated above. Numerical results – including the visual representation of criterion functions in these models – are presented. The global optimization approach can be tailored to specific problem settings, and it is also applicable to a large variety of other (including also more general) model forms.

5.5 Systems of nonlinear equations and inequalities

The analysis of systems of equations is one of the most studied numerical problems, since the equilibrium of physical, chemical, biological, engineering or economic systems is typically characterized by a corresponding set of equations. Additional restrictions – in general, expressed by inequalities – may also be present.

The problem of solving a system of algebraic equations, in its full generality, can be naturally formulated as a global optimization problem. Namely, one can search for the minimum of a suitable aggregate error function (such as the sum of squared errors).

Mathematically, the problem of finding the solution of

$$\begin{array}{rll} & f_i(x) = 0 & i = 1, \dots, I \\ \text{subject to} & x \in D & \end{array} \tag{15.10}$$

(where $D := \{x : l \leq x \leq u;\ g(x) \leq 0\ j = 1, \dots, J\}$ is a subset of $\mathbb{R}^n$, and f_i are continuous functions on D) is equivalent to solving the GO model

$$\begin{array}{rl} \min & f(x) \\ \text{subject to} & x \in D \end{array}$$

where $f(x) := \|f_1(x), \dots, f_I(x)\|$ and $\|\cdot\|$ denotes an arbitrary norm function defined in $\mathbb{R}^I$. Without going into details, note that under- or over-determined systems are also encompassed by the present formulation. For a more comprehensive discussion and numerical examples, as well as for further references consult, for instance, Pintér, 1996a.

5.6 Stochastic optimization problems

In deterministic management models, it is tacitly assumed that all essential problem characteristics are known with certainty. This assumption represents only an (often quite rudimentary) approximation of real-word decision-making situations, in which uncertainties and statistical variations of system behavior frequently play an important role.

Stochastic programming (SP) provides us with tools to describe, analyze and control stochastic systems, using a combination of concepts and techniques from 'standard' (deterministic) mathematical programming, applied probability theory and statistics. In SP models, 'rigid' deterministic constraints and objectives are replaced by carefully chosen stochastic counterparts (extensions) that make sense for all – or at least for most – possible random system states (scenarios). For instance, the general deterministic GO model

$$\begin{array}{rl} \min & f(x) \\ \text{subject to} & x \in D \end{array}$$

(cf. equation (15.1)) can be extended to incorporate uncertainty aspects as follows. Define $\xi = \xi(\omega) \in \mathbb{R}^m$ as the random variable vector outcome depending on the random instance ω which is defined as an elementary event in a suitable probability space $(\Omega, A, \mathbf{P})$. Further on, let us define $f(x, \xi)$ and $g(x, \xi)$ as the proper extensions of the functions f and g, when considering also all modeled system uncertainties and fluctuations (aggregated by ω). Then, introducing the operator $E[\cdot]$ for denoting the expected (mean) value of a random quantity, a proper stochastic extension of (15.1) may be written as

$$\begin{array}{rl} \min & E[f(x,\xi)] \\ \text{subject to} & x \in \underline{D} \end{array} \tag{15.11}$$

where $\underline{D} := \{x : l \leq x \leq u;\ E[g(x,\xi)] \leq 0 \quad j = 1, \ldots, J\}$. That is, we want to minimize the average value of $f(x, \xi)$, under the conditions that the decision vector x belongs to the interval $[l, u]$ (as before), and that the (random-valued) constraints $g(x, \xi) \leq 0$ are met on average.

There are a number of advanced textbooks devoted to stochastic programming: for instance, Prékopa, 1995 presents a far-reaching exposition of its models and solution approaches. Several chapters in Pintér, 1996a also discuss SP models, methodological issues and a range of applications. SP has been successfully applied, for instance, in managing water resources and electric power systems, manufacturing, inventory control, statistics, engineering design and operations, economics, insurance, finances, and environmental management.

In the framework of the present discussion, the main point to emphasize is that the evaluation of objective and constraint functions in many practical SP models is based on stochastic simulation. Applying a similar argument to that in Section 5.1, in general one can perceive the model functions $E[f(x, \xi)]$ and $E[g(x, \xi)]$, as 'black boxes'. Consequently, SP problems – with the notable exception of some important specific cases defined in terms of the functions f, g and the random variables ξ – may easily lead to non-convex models. This again shows the genuine relevance of GO concepts and strategies, also in the SP context.

6. Other applications and further pointers

In this section, we shall provide a list of numerous specific GO application areas, without going into details. The collected applications will be listed in their order of appearance (in the literature referred); and – within that classification – simple alphabetical ordering will be used. We also provide a list of further information sources.

The notable early work of Bracken and McCormick, 1968 discusses models from the following categories (all models discussed are nonlinear, and several of them are non-convex):

- alkylation process optimization
- bid evaluation
- chemical equilibrium
- deterministic equivalents to stochastic LP problems
- launch vehicle design and costing
- optimized stratified sampling
- parameter estimation in curve fitting
- structural optimization
- weapons assignment

The test problem collection by Floudas and Pardalos, 1990 includes application models from the following areas:

- chemical reactor network synthesis
- distillation column sequencing
- heat exchanger network synthesis
- indefinite quadratic programming
- mechanical design
- (general) nonlinear programming
- phase and chemical reaction equilibrium
- pooling/blending operations
- quadratically constrained problems
- reactor-separator-recycling systems
- VLSI design

The recent collection of test problems by Floudas et al., 1999 significantly expands upon the above material, adding more specific classes of nonlinear programming models, combinatorial optimization problems and dynamic models, as well as further practical examples. This work also includes models from the following application areas:

- batch plant design under uncertainty
- conformational problems in clusters of atoms and molecules
- dynamic optimization problems in parameter estimation
- homogeneous azeotropic separation system
- network synthesis
- optimal control problems
- parameter estimation and data reconcilliation
- pump network synthesis
- robust stability analysis
- trim loss minimization

The MINPACK-2 collection presented at the Argonne National Laboratories, 1993 Web site includes models related to the following types of problems:

- brain activity
- Chebychev quadrature
- chemical and phase equilibria
- coating thickness standardization
- combustion of propane
- control systems (analysis and design)
- database optimization
- design with composites
- elastic-plastic torsion
- enzyme reaction analysis
- exponential data fitting

- flow in a channel
- flow in a driven cavity
- Gaussian data fitting
- Ginzburg-Landau problem
- human heart dipole
- hydrodynamic modeling
- incompressible elastic rods
- isomerization of alpha-pinene
- Lennard-Jones clusters
- minimal surfaces
- pressure distribution
- Ramsey graphs
- solid fuel ignition
- steady-state combustion
- swirling flow between disks
- thermistor resistance analysis
- thin film design
- VLSI design

Pintér, 1996a and Pintér, 2001a provide a detailed discussion of several GO case studies and applications; the first of these works also includes a review of numerous further GO applications. The models discussed in these books (in detail) were all analyzed and solved by using various implementations of the LGO modeling and solver environment. The current list of LGO applications includes the following areas:

- bio-mechanical design
- 'black box' (closed, confidential, etc.) system design and operation
- combination of negotiated expert opinions (forecasts, votes, assessments, etc.)
- data classification, pattern recognition

- dynamic population and other environmental resource management problems
- engineering design
- financial modeling
- inverse model fitting to observation data (model calibration)
- laser equipment design
- logical (constraint system satisfiability) problems
- minimal energy configuration problems (free and surface-constrained forms)
- multi-facility location-allocation problems
- nonlinear approximation
- nonlinear regression, and other curve/surface fitting problems
- optimized tuning of equipment and instruments in medical research and other areas
- packing problems (in various settings)
- reactor maintenance policy analysis
- resource allocation (in cutting, loading, scheduling, sequencing, etc. problems)
- risk analysis and control in various environmental management contexts
- robotics equipment design
- robust product (base material mixture) design
- statistical modeling
- systems of nonlinear equations and inequalities
- therapy (dosage and schedule) optimization

Pintér, 2001b is an edited collection of detailed case studies related to the following GO applications:

- adaptive learning via neural nets
- agro-ecosystem management

- animal husbandry operations management
- assembly line design
- atomic and molecular conformation (several studies)
- bio-informatics
- cellular mobile network design
- chemical product design
- composite manufacturing
- controller design for induction motors
- laser equipment design
- mechanical engineering design
- phase unwrapping problem and a related telecommunication application
- radiotherapy (dose delivery) design
- solution of nonlinear equations, with applications in electrical engineering
- systems design in bio-technology and bio-engineering

The prominent Web site maintained by Neumaier, 2001 discusses, *inter alia*, the following important application areas:

- bases for finite elements
- boundary value problems
- chemical engineering problems (various models)
- chemical phase equilibria
- complete pivoting example
- distance geometry models
- extreme forms
- identification of dynamical systems with matrices depending linearly on parameters
- indefinite quadratic programming models

- minimax problems
- nonlinear circuits
- optimal control problems
- optimal design
- parameter identification with data of bounded error
- PDE defect bounds
- PDE solution by least squares
- pole assignment
- production planning
- propagation of discrete dynamical systems
- protein-folding problem
- pseudo-spectrum
- quadrature formulas
- Runge-Kutta formulas
- spherical designs (point configurations)
- stability of parameter matrices

For additional literature on real-world GO applications, the Reader may like to consult, e.g., the following works:

- Mockus et al., 1996 on network problems, combinatorial optimization (knapsack, travelling salesman, flow-shop problems), batch process scheduling
- Grossmann, 1996 on GO algorithms and their applications (primarily) in chemical engineering design
- Bomze et al., 1997, with contributions on decision support systems and techniques for solving GO problems, but also on molecular structures, queuing systems, image reconstruction, location analysis and process network synthesis
- Migdalas et al., 1997 on multilevel optimization algorithms and their applications

- Mistakidis and Stavroulakis, 1997 on engineering applications of the finite element method
- De Leone et al., 1998 on a variety of interesting applications of high performance NLP software
- Hendrix, 1998 on applications from the fields of environmental management, geometric design, robust product design, and parameter estimation
- Corliss and Kearfott, 1999 on industrial applications of rigorous global search
- Floudas and Pardalos, 2000 on optimization in computational chemistry and molecular biology
- Laguna and González-Velarde, 2000 on advanced computing tools for tackling various challenging optimization problems
- Papalambros and Wilde, 2000 on the principles and practice of optimal engineering design
- Atkinson et al., 2001 on optimum experimental design
- Edgar et al., 2001 on the optimization of chemical processes

Since the literature is growing so fast, the avid Reader is advised to visit regularly the Web sites of publishers who specialize in the areas of nonlinear and global optimization. Special reference is made here to the topical *Nonconvex Optimization and Its Applications* book series of Kluwer Academic Publishers which – at the time of this writing – includes nearly 60 volumes (several of these are briefly discussed in this chapter).

Finally, one can mention that numerous issues of professional OR/MS, natural science, engineering, economics and finance journals and books also publish work describing interesting GO applications: all these call for suitable methodology. Without doubt, the future of applicable global optimization is bright.

Acknowledgments

I would like to thank Panos Pardalos and Edwin Romeijn, the editors for their kind invitation to contribute to this Volume. Special thanks are due to Edwin for his comments and editorial assistance in the manuscript finalization stage.

My work on this chapter has benefited from information provided by colleagues, or collected from books, articles, and Web sites, as reflected also by the list of references. I wish to thank, at least collectively, a large number of colleagues who have been supporting my work by offering cooperation, feedback, comments, and criticism. Many of these people are explicitly acknowledged elsewhere, and their work is also referenced therein (see for instance, Pintér, 1996a; Pintér, 1996b; Pintér, 1999; Pintér, 2001a; Pintér, 2001b, as well as at the end of this chapter). Special thanks are due to Zelda Zabinsky and Charoenchai Khompatraporn for their contribution to recent joint work on which some of the observations related to software evaluation and tests are drawn.

Users of the professional LGO software have been significantly motivating my views on the practice of global optimization, including both research and software development.

This work has been partially supported by grants received from the National Research Council of Canada (NRC IRAP Project No. 362093) and from the Hungarian Scientific Research Fund (OTKA Grant No. T 034350). Wolfram Research Inc. (Champaign, IL) also supported my research in recent years by providing a visiting scholarship, valuable books and software products related to *Mathematica* – the powerful software used here (only) to generate all figures that were not generated by LGO.

References

Argonne National Laboratories (1993). *MINPACK-2 Test Problem Collection.* See also the accompanying notes titled 'Large-scale optimization: Model problems', by B.M. Averick and J.J. Moré. `http://www-c.mcs.anl.gov/home/more/tprobs/html`.

Atkinson, A., Bogacka, B., and Zhigljavsky, A.A., editors (2001). *Optimum Design 2000.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Beck, M.B. (1985). *Water Quality Management: A Review of the Development and Application of Mathematical Models.* Springer-Verlag, Berlin / Heidelberg / New York.

Beck, M.B., editor (1987). *Systems Analysis in Water Quality Management.* Pergamon Press, Oxford.

Benson, H.P. (1995). Concave minimization: theory, applications, and algorithms. In Horst, R. and Pardalos, P.M., editors, *Handbook of Global Optimization*, pages 43–148. Kluwer Academic Publishers, Dordrecht / Boston / London.

Benson, H.P. and Sun, E. (2000). LGO – Versatile tool for global optimization. *OR/MS Today*, 27(5):52–55.

Boender, C.G.E. and Romeijn, H.E. (1995). Stochastic methods. In Horst, R. and Pardalos, P.M., editors, *Handbook of Global Optimization*, pages 829–869. Kluwer Academic Publishers, Dordrecht / Boston / London.

Bomze, I.M., Csendes, T., Horst, R., and Pardalos, P.M., editors (1997). *Developments in Global Optimization*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Bracken, J. and McCormick, G.P. (1968). *Selected Applications of Nonlinear Programming*. John Wiley and Sons, New York.

Brooke, A., Kendrick, D., and Meeraus, A. (1988). *GAMS: A User's Guide*. The Scientific Press, Redwood City, California. Revised versions are available from the GAMS Corporation.

Casti, J.L. (1990). *Searching for Certainty*. Morrow & Co., New York.

Colville, A.R. (1968). A comparative study of nonlinear programming codes. Technical Report No. 320-2949, IBM Scientific Center, New York.

Conway, J.H. and Sloane, N.J.A. (1988). *Sphere Packings, Lattices and Groups*. Springer-Verlag, Berlin / Heidelberg / New York, 2nd edition.

Corliss, G.F. and Kearfott, R.B. (1999). Rigorous global search: Industrial applications. In Csendes, T., editor, *Developments in Reliable Computing*, pages 1–16. Kluwer Academic Publishers, Dordrecht / Boston / London.

De Leone, R., Murli, A., Pardalos, P.M., and Toraldo, G., editors (1998). *High Performance Software for Nonlinear Optimization*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Dean, P.M. (1995). *Molecular Similarity in Drug Design*. Chapman and Hall, London.

Dembo, R.S. (1974). A set of geometric programming test problems and their solutions. Working paper, Department of Management Science, University of Waterloo, Waterloo, Ontario, Canada.

Diener, I. (1995). Trajectory methods in global optimization. In Horst, R. and Pardalos, P.M., editors, *Handbook of Global Optimization*, pages 649–668. Kluwer Academic Publishers, Dordrecht / Boston / London.

Dill, K.A., Phillips, A.T., and Rosen, J.B. (1997). Molecular structure prediction by global optimization. In Bomze, I.M., Csendes, T., Horst, R., and Pardalos, P.M., editors, *Developments in Global Optimization*, pages 217–234. Kluwer Academic Publishers, Dordrecht / Boston / London.

Dixon, L.C.W. and Szegö, G.P., editors (1975, 1978). *Towards Global Optimisation, Volumes 1-2*. North-Holland, Amsterdam, The Netherlands.

Dorfman, R., Jacoby, H.D., and Thomas, H.A., editors (1974). *Models for Managing Regional Water Quality.* Harvard University Press, Cambridge, Massachusetts.

Eason, E.D. and Fenton, R.G. (1974). A comparison of numerical optimization methods for engineering design. *ASME J. Eng. Ind. Ser. B*, 96(1):196–200.

Edgar, T.F., Himmelblau, D.M., and Lasdon, L.S. (2001). *Optimization of Chemical Processes.* McGraw-Hill, Boston, Massachusetts, 2nd edition.

Eigen, M. and Winkler, R. (1975). *Das Spiel.* Piper & Co., München, Germany.

Floudas, C.A. (1999). *Deterministic Global Optimization: Theory, Algorithms and Applications.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Floudas, C.A. and Pardalos, P.M. (1990). *A Collection of Test Problems for Constrained Global Optimization Algorithms*, volume 455 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin / Heidelberg / New York.

Floudas, C.A. and Pardalos, P.M., editors (2000). *Optimization in Computational Chemistry and Molecular Biology – Local and Global Approaches.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Floudas, C.A., Pardalos, P.M., Adjiman, C., Esposito, W.R., Gumus, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A., and Schweiger, C.A. (1999). *Handbook of Test Problems in Local and Global Optimization.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Forster, W. (1995). Homotopy methods. In Horst, R. and Pardalos, P.M., editors, *Handbook of Global Optimization*, pages 669–750. Kluwer Academic Publishers, Dordrecht / Boston / London.

Fourer, R. (2001). Nonlinear programming frequently asked questions. Optimization Technology Center of Northwestern University and Argonne National Laboratory. `http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html`.

Fourer, R., Gay, D.M., and Kernighan, B.W. (1993). *AMPL – A Modeling Language for Mathematical Programming.* The Scientific Press, Redwood City, California. Reprinted by Boyd and Fraser, Danvers, Massachusetts, 1996.

Frontline Systems (2001). *Premium Solver Platform – Field-Installable Solver Engines. User Guide.* Frontline Systems, Inc., Incline Village, Nevada.

Glover, F. and Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Greengard, L. (1988). *The Rapid Evaluation of Potential Fields in Particle Systems*. The MIT Press, Cambridge, Massachusetts.

Grossmann, I.E., editor (1996). *Global Optimization in Engineering Design*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Haith, D.A. (1982). *Environmental Systems Optimization*. Wiley, New York.

Hansen, E.R. (1992). *Global Optimization Using Interval Analysis*. Marcel Dekker, New York.

Hartke, B. (1999). Global cluster geometry optimization by a phenotype algorithm with niches: Location of elusive minima, and low-order scaling with cluster size. *Journal of Computational Chemistry*, 20(16):1752–1759.

Hendrix, E.M.T. (1998). *Global Optimization at Work*. PhD thesis, Agricultural University Wageningen, Wageningen, The Netherlands.

Hichert, J., Hoffman, A., and Phú, H.X. (1997). Convergence speed of an integral method for computing the essential supremum. In Bomze, I.M., Csendes, T., Horst, R., and Pardalos, P.M., editors, *Developments in Global Optimization*, pages 153–170. Kluwer Academic Publishers, Dordrecht / Boston / London.

Hock, W. and Schittkowski, K. (1981). *Test Examples for Nonlinear Programming Codes*, volume 187 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin / Heidelberg / New York.

Horst, R. and Pardalos, P.M., editors (1995). *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Horst, R. and Tuy, H. (1996). *Global Optimization – Deterministic Approaches*. Springer, Berlin / Heidelberg / New York, 3rd edition.

Jacob, C. (2001). *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann Publishers, San Francisco, California.

Jansson, C. and Knüppel, O. (1992). A global minimization method: The multi-dimensional case. Bericht 92.1, FiuK, TUHH, Hamburg-Harburg, Germany.

Journal of Global Optimization. Kluwer Academic Publishers, Dordrecht / Boston / London. Published since 1991.

Kearfott, R.B. (1996). *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Khompatraporn, C., Zabinsky, Z.B., and Pintér, J.D. (2001). Comparative assessment of algorithms and software for global optimization. Submitted for publication.

Laguna, M. and González-Velarde, J.-L., editors (2000). *Computing Tools for Modeling, Optimization and Simulation*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Lahey Computer Systems (1999). *Fortran 90 User's Guide.* Lahey Computer Systems, Inc., Incline Village, Nevada.

Levy, A.V. and Gomez, S. (1985). The tunneling method applied for the global minimization of functions. In Boggs, P.T., editor, *Numerical Optimization*, pages 213–244. SIAM, Philadelphia, Pennsylvania.

LINDO Systems (1996). *Solver Suite.* LINDO Systems, Inc., Chicago, Illinois.

Loucks, D.P., Stedinger, J.R., and Haith, D.A. (1981). *Water Resources Systems Planning and Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey.

Mandelbrot, B.B. (1983). *The Fractal Geometry of Nature.* Freeman & Co., New York.

Mathar, R. and Žilinskas, A. (1994). A class of test functions for global optimization. *Journal of Global Optimization*, 5:195–199.

Maximal Software (1998). *MPL Modeling System.* Maximal Software, Inc., Arlington, Virginia.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, Berlin / Heidelberg / New York, 3rd edition.

Migdalas, A., Pardalos, P.M., and Värbrand, P., editors (1997). *Multilevel Optimization: Algorithms and Applications.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Mistakidis, E.S. and Stavroulakis, G.E. (1997). *Nonconvex Optimization. Algorithms, Heuristics and Engineering Applications of the F.E.M.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Mittelmann, H.D. and Spellucci, P. (2001). Decision tree for optimization software. `http://plato.la.asu.edu/guide.html`.

Mockus, J., Eddy, W., Mockus, A., Mockus, L., and Reklaitis, G. (1996). *Bayesian Heuristic Approach to Discrete and Global Optimization.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Moré, J.J., Garbow, B.S., and Hillström, K.E. (1981). Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7:17–41.

Moré, J.J. and Wright, S.J. (1993). *Optimization Software Guide*, volume 14 of *Frontiers in Applied Mathematics Series.* SIAM, Philadelphia, Pennsylvania.

Moré, J.J. and Wu, Z. (1997). Global continuation for distance geometry problems. *SIAM Journal on Optimization*, 7:814–836.

Murray, J.D. (1989). *Mathematical Biology.* Springer-Verlag, Berlin / Heidelberg / New York.

Neumaier, A. (1990). *Interval Methods for Systems of Equations.* Cambridge University Press, Cambridge.

Neumaier, A. (1997). Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review*, 39:407–460.

Neumaier, A. (1999). Molecular modeling of proteins. `http://solon.cma.univie.ac.at/~neum/protein.html`.

Neumaier, A. (2001). Global optimization. `http://www.mat.univie.ac.at/~neum/glopt.html`.

Nonconvex Optimization and its Applications. Kluwer Academic Publishers, Dordrecht / Boston / London. Bbook series, published since 1995. `http://www.wkap.nl/series.htm/NOIA`.

Orlob, G.T., editor (1983). *Mathematical Modeling of Water Quality: Streams, Lakes and Reservoirs*. Wiley, New York.

Osman, I.H. and Kelly, J.P., editors (1996). *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Pain, R.H., editor (1994). *Mechanisms of Protein Folding*. Oxford University Press, Oxford.

Papalambros, P.M. and Wilde, D.J. (2000). *Principles of Optimal Design – Modeling and Computation*. Cambridge University Press, Cambridge, 2nd edition.

Pardalos, P.M. (1995). An open global optimization problem on the unit sphere. *Journal of Global Optimization*, 6:213.

Pardalos, P.M., Shalloway, D., and Xue, G. (1996). *Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding*, volume 23 of *DIMACS Series*. American Mathematical Society, Providence, Rhode Island.

Pintér, J.D. (1996a). *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications)*. Kluwer Academic Publishers, Dordrecht / Boston / London.

Pintér, J.D. (1996b). Continuous global optimization software: A brief review. *Optima*, 52:1–8. Web version is available at: `http://plato.la.asu.edu/gom.html`.

Pintér, J.D. (1999). Continuous global optimization: An introduction to models, solution approaches, tests and applications. *Interactive Transactions in Operations Research and Management Science*, 2(2). `http://catt.bus.okstate.edu/itorms/pinter/`.

Pintér, J.D. (2000a). Globally optimized spherical point arrangements: Model variants and illustrative results. To appear in *Annals of Operations Research*.

Pintér, J.D. (2000b). *LGO – A Model Development System for Continuous Global Optimization. User's Guide*. Pintér Consulting Services, Inc., Halifax, Nova Scotia, Canada. For summary descriptions, see

`http://is.dal.ca/~jdpinter/lgoide.htm`;
`http://www.solver.com/xlslgoeng.htm`.

Pintér, J.D. (2001a). *Computational Global Optimization in Nonlinear Systems – An Interactive Tutorial.* Lionheart Publishing, Inc., Atlanta, Georgia. This work is also available electronically, at `http://www.lionhrtpub.com/books/globaloptimization.html`.

Pintér, J.D., editor (2001b). *Global Optimization: Selected Case Studies.* Kluwer Academic Publishers, Dordrecht / Boston / London. To appear.

Prékopa, A. (1995). *Stochastic Programming.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Ratschek, H. and Rokne, J. (1988). *New Computer Methods for Global Optimization.* Ellis Horwood, Chichester.

Ratschek, H. and Rokne, J. (1993). Experiments using interval analysis for solving a circuit design problem. *Journal of Global Optimization*, 3:501–518.

Reklaitis, G.V., Ravindran, A., and Ragsdell, K.M. (1983). *Engineering Optimization Methods and Applications.* John Wiley and Sons, New York.

Saff, E.B. and Kuijlaars, A.B.J. (1997). Distributing many points on a sphere. *The Mathematical Intelligencer*, 19(1):5–11.

Sandgren, E. and Ragsdell, K.M. (1980). The utility of nonlinear programming algorithms: A comparative study – parts 1 and 2. *ASME Journal of Mechanical Design*, 102(3):540–551.

Sandia National Laboratories (1997). *A Survey of Global Optimization Methods.* Prepared by Gray, P., Hart, W.E., Painton, L., Phillips, C., Trahan, M., and Wagner, J.
`http://www.cs.sandia.gov/opt/survey/main.html`.

Schittkowski, K. (1980). *Nonlinear Programming Codes: Information, Tests, Performance*, volume 183 of *Lecture Notes in Economics and Mathematical Systems.* Springer-Verlag, Berlin / Heidelberg / New York.

Schoen, F. (1993). A wide class of test functions for global optimization. *Journal of Global Optimization*, 3:133–138.

Schroeder, M. (1991). *Fractals, Chaos, Power Laws.* Freeman & Co., New York.

Sethi, S.P. and Thompson, G.L. (2000). *Optimal Control Theory. Applications to Management Science and Economics.* Kluwer Academic Publishers, Dordrecht / Boston / London, 2nd edition.

Somlyódy, L. and van Straten, G., editors (1983). *Modeling and Managing Shallow Lake Eutrophication.* Springer-Verlag, Berlin / Heidelberg / New York.

Stewart, I. (1995). *Nature's Numbers.* Basic Books. Harper and Collins, New York.

Strongin, R.G. and Sergeyev, Ya.D. (2000). *Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Voss, S., Martello, S., Osman, I.H., and Roucairol, C., editors (1999). *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publishers, Dordrecht / Boston / London.

Wales, D.J. and Scheraga, H.A. (1999). Global optimization of clusters, crystals, and biomolecules. *Science*, 285:1368–1372.

Wolfram, S. (1996). *The Mathematica Book.* Wolfram Media and Cambridge University Press, 3rd edition.

Zheng, Q. and Zhuang, D. (1995). Integral global minimization: algorithms, implementations and numerical tests. *Journal of Global Optimization*, 7:421–454.

Zhigljavsky, A.A. (1991). *Theory of Global Random Search.* Kluwer Academic Publishers, Dordrecht / Boston / London.