J Glob Optim (2007) 38:79–101
DOI 10.1007/s10898-006-9084-2

ORIGINAL ARTICLE

Nonlinear optimization with GAMS /LGO

János D. Pintér

Received: 9 August 2006 / Accepted: 11 August 2006 / Published online: 7 October 2006
© Springer Science+Business Media B.V 2006

Abstract The Lipschitz Global Optimizer (LGO) software integrates global and local scope search methods, to handle a very general class of nonlinear optimization models. Here we discuss the LGO implementation linked to the General Algebraic Modeling System (GAMS). First we review the key features and basic usage of the GAMS /LGO solver option, then present reproducible numerical results to illustrate its performance.

Keywords Nonlinear (global and local) optimization · LGO solver suite · GAMS modeling system · GAMS /LGO solver option · Numerical performance analysis · Illustrative applications

1 Introduction

Nonlinearity is a key characteristic of a vast range of objects, formations, and processes in nature and in society. Consequently, nonlinear descriptive models are relevant in many areas of the sciences and engineering. For related discussions (targeted toward various professional audiences) consult for instance Aris (1999), Bracken and McCormick (1968), Dörner (1996), Gershenfeld (1999), Hansen and Jørgensen (1991), Murray (1983), Lopez (2005), Steeb (2005), and Stojanovic (2003). Managing nonlinear systems leads to nonlinear optimization—a subject that has been of great practical interest, at least since the beginnings of mathematical programming. For technical discussions and further examples, see the topical chapters in Bartholomew-Biggs (2005), Chong and Zak (2001), Diwekar (2003), Edgar et al. (2001), Pardalos and Resende (2002), and Hillier and Lieberman (2005).

Algorithmic advances and progress in computer technology have enabled the development of sophisticated nonlinear optimization software implementations.

J. D. Pintér (✉)
Pintér Consulting Services Inc., 129 Glenforest Drive, Halifax, NS, B3M 1J2, Canada
e-mail: jdpinter@hfx.eastlink.ca

Among the currently available software products, one can mention LANCELOT (Conn et al. 1992) which implements an augmented Lagrangian based solution approach. Sequential quadratic programing methods are implemented in EASY-FIT (Schittkowski 2002), filterSQP (Fletcher and Leyffer 1998, 2002), GALAHAD (Gould et al. 2002), and SNOPT (Gill et al. 2003). Another prominent class of methods is based on the reduced gradient (RG) approach and its generalization (GRG). The RG methodology is implemented, e.g., in MINOS (Murtagh and Saunders 1995) which includes also other solvers. GRG strategies are implemented in LSGRG (Lasdon and Smith 1992; Edgar et al. 2001), and in CONOPT (Drud 1996). Interior point methodology is used in LOQO (Vanderbei 1999) and in KNITRO (Waltz and Nocedal 2003). Finally—but without claiming completeness—one can mention gradient-free quadratic model-based solver implementations such as UOBYQA (Powell 2002), and heuristic direct search methods reviewed by Wright (1996). A recent issue of *SIAG/OPT Views and News* (Leyffer and Nocedal 2003) provides concise, informative reviews regarding the state-of-art in nonlinear optimization, although the contributed articles mainly discuss optimization software with a local search scope. A detailed review of local and global optimization algorithms is provided by Bliek et al. (2001).

Without a suitable starting point, even the best local search methods could encounter difficulties in solving general nonlinear models. Specifically, such methods will typically find only a local solution (when better solutions may exist), or they may return a locally infeasible result in (globally) feasible models. Clearly, if one does not have sufficient insight to guarantee an essentially convex model structure, or does not have access to a good starting point that will lead to the best possible solution, then the application of a global scope search strategy becomes desirable. We wish to point out that this line of argument does not "dismiss" high-quality local optimization software that has been in use for decades with considerable success. A global scope search, however, can bring tangible benefits to both model development (by enabling more general and thereby perhaps more realistic formulations) and solution (by making possible global search when it is not guaranteed that local search will suffice).

The field of global optimization (GO) has been gaining increasing attention in the past few decades, and in recent years it has reached a certain level of maturity. The number of textbooks focused on GO is well over one hundred worldwide. For illustration, the *Handbook of Global Optimization* volumes edited by Horst and Pardalos (1995) and by Pardalos and Romeijn (2002) are mentioned. These two volumes cover the most frequently used GO model types and solution strategies, including information also on software and various application areas.

The key theoretical developments in GO have been followed by increasingly efficient solution algorithms and their software implementations. While most GO software products reviewed by Pintér (1996b) have been perhaps "academic" rather than "professional", a decade later a number of companies offer professionally developed and maintained GO software. To illustrate this point, it suffices to visit the web sites of Frontline Systems, the GAMS Development Corporation, LINDO Systems, Maplesoft, Maximal Software, Paragon Decision Technology, TOMLAB, or Wolfram Research, to check out platform-specific GO software information. One should also mention here at least a few informative, non-commercial web sites that discuss GO models, algorithms, and technology. For instance, the web site of Neumaier (2005a) is devoted to global optimization in its entirety; Fourer (2005) and Mittelmann and

Spellucci (2005) also provide valuable discussions of nonlinear programing methods and software, with numerous further links and pointers.

In this article, we present the GAMS /LGO software implementation for handling nonlinear optimization models. First we formulate and briefly analyze the general GO model, then review the key features of the LGO solver suite, and discuss its GAMS-specific implementation. We also present fully reproducible numerical results, to illustrate the performance of GAMS /LGO in comparison with several state-of-the-art global and local solvers available with GAMS.

2 Global optimization: model statement and specifications

Consider the continuous global optimization (CGO) model stated as

$$\min f(x) \quad \text{subject to } x \in D := \{x : l \leq x \leq u \; g_j(x) \leq 0 \; j = 1, \ldots, m\}. \tag{1}$$

In (1) we apply the following notation and assumptions:

- $x \in \boldsymbol{R}^n$ — n-dimensional real-valued vector of decision variables,
- $f: \boldsymbol{R}^n \to \boldsymbol{R}$ — continuous (scalar-valued) objective function,
- $D \subset \boldsymbol{R}^n$ — non-empty set of feasible solutions, a proper subset of $\boldsymbol{R}^n$: this feasible set is defined by
- $l \in \boldsymbol{R}^n, u \in \boldsymbol{R}^n$ — component-wise finite lower and upper bounds on x, and
- $g: \boldsymbol{R}^n \to \boldsymbol{R}^m$ — a finite collection (m-vector) of continuous constraint functions.

Let us note that the constraints $g_j \; j = 1, \ldots, m$ in (1) could be followed in (1) by arbitrary ($\leq$, $=$, $\geq$) relation signs, and that explicit bounds on the constraint function values could also be imposed. Such—formally more general—models are directly deducible to the model statement (1). Without going into details that are not relevant here, let us also point out that models with bounded integer variables can be brought to the form (1). This, of course, also implies the formal coverage of mixed integer models by the CGO model.

The compactness of D and the continuity of f (by the Bolzano–Weierstrass theorem) guarantee that the global solution set X^* of the CGO model is non-empty. In many cases of practical relevance, X^* consists of a unique point x^*. However, it is easy to construct GO model-instances in which X^* is finite (with cardinality greater than one), countable, non-countable, or it can be even a subset of D with a positive volume. For the sake of meaningful algorithmic convergence statements, we typically assume that X^* is at most countable. This is rarely a restriction in well-posed, practically motivated problems.

Without further structural assumptions, model-instances of (1) could lead to very difficult numerical problems. For example, the feasible set D could be disconnected, and some of its components could be non-convex; furthermore, the objective function f could well be multi-extremal over D. In such cases, the CGO model (1) can have an unknown number of global (as well as local) solutions. Let us recall that there is no generally applicable, constructive algebraic characterization of global optimality. In traditional nonlinear programming, numerical methods frequently aim at solving the Lagrange or Karush–Kuhn–Tucker (KKT) system of necessary optimality conditions, to find local solutions. The corresponding feasibility system of equations and inequalities to find points from X^* becomes another GO problem, often at least as complex as the original model (1). Neumaier (2004) presents an interesting discussion

of this point, indicating that the number of KKT points to check for optimality could grow exponentially as the model size (number of variables n and/or constraints m) increases. A very simple illustration of this general observation is to minimize a concave function over the vertex set of an n-dimensional box region: here all 2^n vertices are local optima.

To illustrate the potential numerical difficulty of CGO models by an example, let us consider the problem of finding the solution(s) to the equations

$$\begin{aligned} \text{eqn1} &:= x - \sin(2x + 3y) - \cos(3x - 5y) = 0, \\ \text{eqn2} &:= y - \sin(x - 2y) + \cos(x + 3y) = 0. \end{aligned} \tag{2}$$

We will search for solutions in the postulated variable range $x \in [-2, 3], y \in [-2.5, 1.5]$.

Figure 1 shows the surface plot of the error function $(\text{eqn } 1)^2 + (\text{eqn } 2)^2$ which vanishes at the solution(s). Although this reformulation leads to a rather simple (merely two-variable, box-constrained) model-instance of (1), the resulting model has no apparent structure that could be easily exploited by a search procedure.

In line with the discussion above, problem (2) could have multiple global and local solutions. For example, one of the approximate numerical solutions is $x^* \approx 0.8388353863$, $y^* \approx 0.5371194096$; the residual absolute errors of the equations are $\text{eqn } 1 \approx 2.12628 \cdot 10^{-10}$, $\text{eqn } 2 \approx -5.21127 \cdot 10^{-11}$. This solution has been produced using the LGO solver implementation described by (Pintér and Kampas 2003): later on we will produce another solution using GAMS/LGO.

Theoretically, one would like to find *exactly* all global solutions $x^* \in X^*$, by applying a suitable search mechanism. However, even unconstrained local search methods in (general) nonlinear optimization require an infinite numerical procedure. Therefore a more realistic goal is to find suitable approximations of points in X^*, and of the corresponding optimum value f^*. In practice, this needs to be attained on the basis of a finite number of model function evaluations at algorithmically selected search

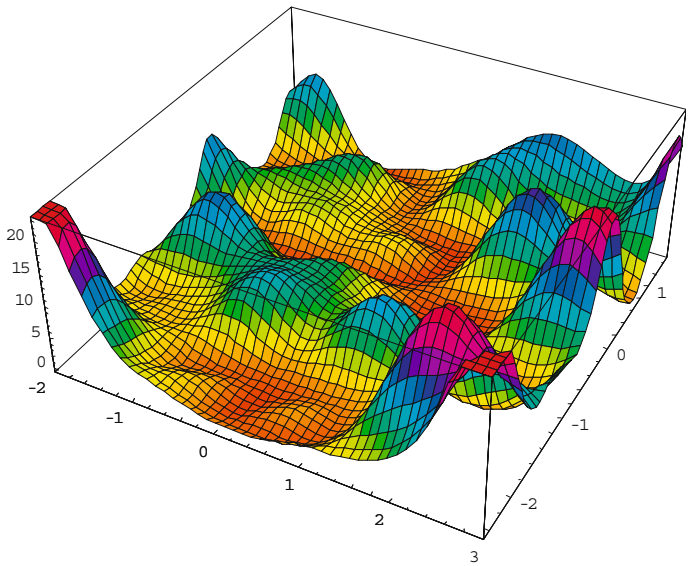

Fig. 1 An illustrative CGO model-instance

points. Formally, one could accept an approximate numerical solution x_a^* that satisfies the relation

$$\rho\left(x_a^*, X^*\right) := \min_{x^* \in X^*} \rho\left(x_a^*, x^*\right) \leq \gamma. \tag{3}$$

In (3) ρ is a given metric—typically defined by the Euclidean norm introduced in $\boldsymbol{R}^n$—and $\gamma > 0$ is a fixed tolerance parameter. (In words, x_a^* should be "sufficiently close" to at least one of the global solutions.) Similarly, one could accept an approximate solution $x_e^* \in D$ that satisfies the relation

$$f\left(x_e^*\right) \leq f^* + \varepsilon, \tag{4}$$

In (4) $\varepsilon > 0$ is another tolerance (accuracy) parameter. Here we formally assume that f^* is known, or that it can be properly estimated. In practice, one searches for feasible solutions that are within a specified tolerance from the "best possible" solution, or from its valid lower bound. Theoretically, we expect that the lower bounding procedure is consistent: i.e., that we can provide increasingly accurate bound estimates that converge to f^*. In numerical practice, this may also be a difficult problem, in "unstructured" GO models when such estimates are difficult (or even impossible) to produce.

To ensure the numerical solvability of model (1) in the sense of (4)—on the basis of a finite sample point sequence from D—we often require the Lipschitz-continuity of the model functions. Recall that a function h is Lipschitz-continuous in the set D, if the relation

$$|h(x_1) - h(x_2)| \leq L\|x_1 - x_2\| \tag{5}$$

is valid for all pairs x_1, x_2 from D. (In the right-hand side of (5), the Euclidean norm is used.) The value $L = L(D,h) \geq 0$ is a suitable Lipschitz constant of h over D. Let us emphasize that the smallest possible value of L is typically unknown (for a smooth function h, it is the global maximum of $\|\nabla h(x)\|$ over the set D). Therefore the availability of a proper overestimate (with respect to all model functions f and g_j $j = 1, \ldots, m$) is often postulated theoretically, or it is estimated in practice: for related discussions, consult Pintér (1996a), or Strongin and Sergeyev (2000). As it is well-known, if f is Lipschitz-continuous in $[l,u]$ and $L(D,f)$ is a valid Lipschitz constant, then even a single point x chosen from $[l,u]$ and the corresponding function value $f(x)$ supports the computation of a lower bounding function for f over the entire box $[l,u]$. Such basic (or more advanced) bounding information can be built into branch-and-bound search procedures that will then have rigorous global convergence properties (Horst and Tuy 1996, Pintér 1996a).

3 LGO solver suite: algorithm components and current implementations

The CGO model (1) with a postulated Lipschitz structure is still very general, and it encompasses most GO problem types that occur in practice. As a consequence, it includes also very difficult problem-instances that pose a tough challenge in any computational environment of today or tomorrow. For given CGO model-instances, the "most suitable" solution approach could vary to a considerable extent. A "universal scope" GO solver strategy—and the corresponding software—is expected to work for

broad model classes, although its efficiency could be lower for certain problem types when compared to more specialized solvers. On the other hand, highly specialized algorithms often will not work for GO models outside of their intended scope.

LGO—abbreviating a Lipschitz(-Continuous) Global Optimizer—has been designed to handle in principle the entire class of models defined by (1), without requiring any special structure beyond continuity or Lipschitz-continuity. For example, the objective function displayed in Fig. 1 is Lipschitz-continuous, but it could be difficult to place it into a more specific category in a constructive and algorithmically useful manner. This overall design principle and the corresponding choice of component algorithms makes LGO applicable even to "black box" models. The latter category specifically includes business confidential models provided as an object file or a dynamic link library. It also includes models that incorporate computational procedures such as special functions, parametric differential equations, integrals, stochastic simulation, and so on. At the same time, models with a given specific structure—e.g., an indefinite quadratic objective f over a convex set D—can also be solved by LGO, although more specialized software can be more efficient to handle such models.

LGO (as a proprietary solver system) has been gradually developed since 1990, with continuing development and maintenance. The theoretical results underpinning its global search algorithms are discussed in Pintér (1996a), with platform-specific implementations described in articles and user manuals (Pintér 1997, 2001a, 2002, 2003a, 2004, 2005a,b,c,d, Pintér and Kampas 2003; Pintér et al. 2004). Therefore only a concise review of its key features is included here.

The overall solution approach in LGO is based on the integration of theoretically convergent global and efficient local search strategies. Currently, the following search algorithms are offered as LGO components:

- adaptive partition and search (branch-and-bound) based global search (referred to as BB)
- adaptive global random search (single-start) (GARS)
- adaptive global random search (multi-start) (MS)
- constrained local search by the generalized reduced gradient method (LS).

Within a given LGO solver run, the user can choose any of the global solver modes BB, GARS, or MS. Upon completion, the selected global solver component is automatically followed by the local solver (search phase). The LS option can be used also in a stand-alone mode, started from a user-supplied initial solution or—in lack of such information—from an automatically generated (default) starting point.

The BB solver component implements a theoretically convergent algorithm, assuming Lipschitz-continuous model functions f and g. In models with a unique global solution x^*, the algorithmically generated search point sequence $\{x_k\}$ converges to x^*. (Theoretically, in models which have an at most countable set X^*, all elements of X^* are limit points of a corresponding sub-sequence of $\{x_k\}$.) The BB approach is based on the underlying assumption that one is able to provide valid overestimates of the Lipschitz constant, for each model function, throughout the iterations. In practice, such a condition is typically only approximated by algorithm implementations. The BB implementation in LGO combines its adaptive set partition steps with deterministic and randomized sampling strategies within the generated subsets. The latter strategy supports also the application of statistical bounding procedures. The BB solver

module is expected to generate a close approximation of at least one of the global solution points, before LGO switches over to local search.

Regarding the GARS and MS solver modes, it is well-known that properly constructed stochastic search algorithms possess global convergence properties, under mild analytical conditions. Specifically, each convergent subsequence of the sequence $\{x_k^*\}$ of improving global solution estimates converges to a point of X^*, with probability 1. This statement applies to instances of model (1) defined by continuous functions f and g, even without the Lipschitz assumption. The GARS solver mode is based on a combination of random search methods and an attempt to focus the global search effort on the region which—on the basis of the actual sample results—is estimated to contain the global solution point (or, in general, one of these points). Similarly to BB, this method is used to generate an initial solution for subsequent local search.

The MS global search component MS is also based on the theoretical stochastic global convergence properties mentioned above. In MS the total allocated global sampling effort is distributed among several global searches. Each of these leads to a "promising" starting point used in subsequent LS. Typically, MS requires the most computational effort (due to its multiple local searches); however, in complicated models, it often finds the best numerical solution. Therefore the MS + LS combination is chosen as the recommended default solver mode, but it is straightforward to select another option (LS, BB + LS, GARS + LS).

All three global solvers are gradient-free, requiring only model function value information. Specifically, their operations are partially driven by iteratively calculated values of the exact penalty function

$$f(x) + \sum_{j \in E} |g_j(x)| + \sum_{j \in I} \max(g_j(x), 0). \tag{6}$$

In (6) the index sets E and I denote the subsets of equality and inequality constraints, respectively. In the LS phase finite difference based gradient approximations are used (tacitly assuming at least local smoothness, in order to guarantee theoretical local convergence). Again, this gradient-free approach supports also the optimization of "black box" systems.

Let us note here that "black box" model handling from GAMS may become useful, e.g. when interfacing GAMS with other (C/C++, Delphi, Excel, Java, MATLAB, Oracle, SQL, Visual Basic, etc.) environments: consult, e.g. Ferris (2005) and Kalwelagen (2005). A similar comment applies also to other LGO implementations linked to other modeling environments.

As already mentioned, each of the LGO global solver modes is automatically followed by the local search phase(s). The local solver embedded in LGO implements a dense nonlinear optimization algorithm, without postulating or exploiting any specific further model structure. This GRG solver implementation is based on classical nonlinear optimization techniques discussed, e.g. by Edgar et al. (2001). The application of the local search mode theoretically assumes that the CGO model (1) is defined by continuously differentiable functions, at least in the local region(s) of attraction where this solver mode is used.

As a result of using the outlined solution approach, LGO will return—barring numerical problems and "unsuitable" option settings—a global search based solution (BB + LS, GARS + LS), possibly several such solutions (MS + LS), or a LS based solution LS. The term "global search based solution" describes a solution that—according

to extensive numerical experience—often is very close to the global solution (or one of these solutions), or at least is a high-quality feasible solution. Let us emphasize here the gap between global convergence theory and software implementation and usage in practice, before dismissing such an outcome. It strongly depends on the practical circumstances and user demands, whether we wish to find a rigorously guaranteed "very precise" solution (that could require hours/days/weeks/months/years of program runtime), or we need/prefer to get a numerical solution in seconds/minutes. An honest look at the deterministically or stochastically guaranteed "gap" between the best solution found and the (unknown) best possible solution, in a time-limited run, often can be a humbling experience, when trying to solve models of realistic size and complexity.

Since 1990, LGO—equipped with a text input/output interface—has been implemented using several programing language platforms. These include professional Fortran compilers (Lahey Fortran 77/90, Lahey-Fujitsu Fortran 95, Digital/Compaq/Intel Visual Fortran 95, g77, g95, and some others), with direct connectivity to C/C++ models (using e.g. Borland C/C++, Microsoft Visual C/C++, gcc, lcc-win32, and others). The basic (compiler based) LGO implementation can be equipped with a Windows-style GUI, thereby providing an integrated development environment that can be used in conjunction with C and Fortran compilers. The compiler-based and GUI enhanced LGO implementations are discussed in (Pintér 1997, 2001a, 2002, 2005a).

In addition to the above core implementations, LGO is available as a callable library, to use in conjunction with several optimization modeling languages and with integrated scientific-technical computing systems. Currently, these include the following (in alphabetical order, the user manuals and technical reports indicate the year of release):

- AIMMS /LGO solver option (Paragon Decision Technology 2005, Pintér 2005c)
- GAMS /LGO solver option (GAMS Development Corporation, 2005; Pintér 2003a)
- Global Optimization Toolbox for Maple (Maplesoft 2005, Pintér 2004, Pintér and Purcell 2006)
- MathOptimizer Professional for Mathematica (Wolfram Research 2005, Pintér and Kampas 2003)
- MPL /LGO solver option (Maximal Software 2005, Pintér 2005d)
- TOMLAB /LGO solver option to use with MATLAB (MathWorks 2005; TOMLAB 2005; Pintér et al. 2004)

Peer reviews discussing several of these implementations are also available: consult Benson and Sun (2000), Cogan (2003), Castillo (2005), Henrion (2006), and Wass (2006).

4 GAMS and the GAMS /LGO solver option

The General Algebraic Modeling System (GAMS) is a high-level optimization modeling environment that—making use of its solver options—supports the development, analysis and solution of a broad range of optimization problems. GAMS is capable of handling advanced "real-world" applications, by allowing users to build prototypes as well as large-scale models. Models can be developed, solved and

documented simultaneously, maintaining the same GAMS model file. The GAMS system has been available since 1987, and it has a significant world-wide user base. The first edition of the *GAMS User's Guide* (Brooke et al., 1988) has been both extended and enhanced by accompanying documentation that includes the extensive, hyperlink-enabled GAMS documentation by McCarl (2004).

The website www.gams.com provides further useful information: therefore only some of the key features and facts are highlighted here. The GAMS modeling language is similar to commonly used procedural programming languages such as C, Fortran, Pascal. GAMS offers interfaces to other development environments, including, e.g., MS Excel, MS Access, and MATLAB. GAMS can also be embedded in various application environments: these include C/C++, Delphi, Java, Visual Basic, and WebSphere. The GAMS Model Library is a large (and growing) collection of models originating from a variety of application areas such as economics, econometrics, engineering, finance, management science, and operations research. The library includes examples for all supported model types. Many of the models are of realistic size and complexity, in addition to collections of "academic" test problems. The model converter program CONVERT transforms a GAMS model-instance into a format used by other modeling and solver systems, and hence provides significant assistance in sharing test models by users of the various modeling and solver systems. A further valuable service is GAMS World (http://www.gamsworld.org) with the objective of bridging the gap between academic research and the practice of optimization. The site includes a large additional set of documented models and performance analysis tools.

All modeling and solver features, including the full documentation, are available through an integrated development environment (GAMS IDE) on MS Windows platforms. Command-line GAMS usage is also supported for Windows/Linux/Unix/Mac environments. In addition to advanced model development features, GAMS offers direct links to a range of solver options. These solvers can handle both general (categorized as linear, nonlinear, pure and mixed integer, and stochastic) and more specialized (such as complementarity, equilibrium, and constrained nonlinear systems) models. LGO has been added to the solvers available in the GAMS modeling environment in 2003. Pintér (2003a) provides a concise GAMS /LGO user documentation: portions of that description are used here, with additional details. (All GAMS solver manuals are available through the GAMS web site.) For the sake of completeness, let us remark that two other global solvers—BARON and OQNLP—are also available for the GAMS platform. BARON (Tawarmalani and Sahinidis 2002) is based on a successive convexification approach by constructing enclosure functions and related bound estimates that drive the global search. In its solution procedure, BARON relies on using other solvers: within GAMS, these are MINOS and CPLEX (and optionally others that are modularly available). OQNLP—similarly to LGO—is a stand-alone solver option: it uses a multi-start global search approach based on the OptTek search engine, in combination with the well-received local solver LSGRG. For a recent discussion of OQNLP and its performance, consult Ugray et al. (2006).

The basic structure of the GAMS /LGO modeling and solution procedure is displayed in Fig. 2.

The steps of model development, verification and preprocessing, solution by LGO, optional further solver calls, and report generation are tightly integrated. As a result, our numerical experiments indicate relatively little runtime overhead associated with the operations of GAMS, when compared to the core (compiler-platform based) LGO implementation.

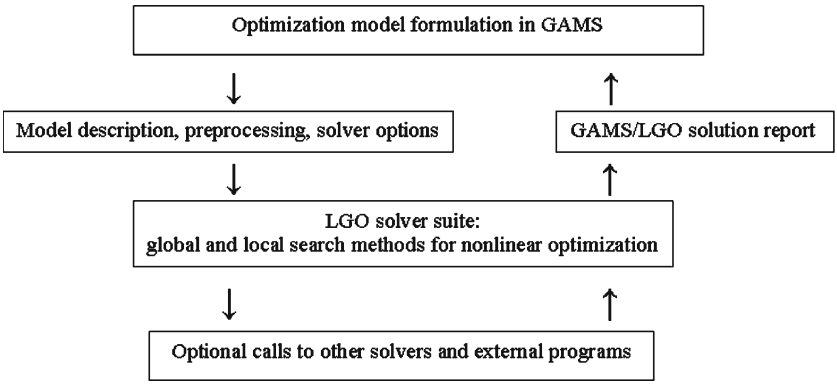

Fig. 2 GAMS /LGO modeling and solution procedure

The GAMS/LGO interface is similar to those of other GAMS solvers, and many options such as resource and iteration limits can be set directly in the GAMS model. To provide LGO-specific options, users can make use of solver option files: consult the solver manuals (GAMS Development Corporation 2005) for more details regarding this point. The list of the current GAMS/LGO options is shown below: see Tables 1 (general options similar to those also for other solvers) and 2 (LGO specific options and parameters). The tables display the option lists, with added brief explanation and the default settings.

Clearly, there is no "universal recipe" to provide options and switches to a general purpose nonlinear solver like LGO that will be adequate to handle all possible models. (Recall here the notes related to global search based solutions.) However, according to fairly extensive numerical experience, the default option settings shown above are suitable to solve many small to moderate size GO problems without any "tweaking" of the parameters. This general observation has been validated for many of the standard academic tests known from the GO literature, using GAMS /LGO or other LGO implementations. In the next section we will illustrate this point, by presenting model examples formulated in GAMS and solved by LGO. The current GAMS (release 22.0) and the current (February 2006) LGO versions are used in the calculations. All other solvers mentioned are used in their default mode, unless specifically noted otherwise. The illustrative runs reported here have been done on an AMD Athlon 64 3200+ 2.00 GHz single processor based desktop computer, running under the Microsoft Windows XP Professional (Version 2002, Service Pack 2) operating system.

5 Using GAMS /LGO: illustrative examples

Example 1 Solving a pair of transcendental equations

We assume that not all readers are familiar with GAMS: therefore first an easy-to-follow example is presented. This model is based on problem (2) introduced earlier. All GAMS language elements are denoted by **`Courier boldface`** fonts. Explanatory comments are given between the rows denoted by **`$ontext`** and **`$offtext`**, and in rows started by the symbol *. The GAMS output details shown are only slightly

Table 1 GAMS /LGO general options

Option	Description	Default
tlimit	Runtime limit in seconds. This is equivalent to the general GAMS option reslim. If specified, then it overrides the reslim option.	1,000
log_time	Iteration log time interval in seconds. GAMS log output is generated at every log_time seconds.	0.5
log_iter	Iteration log time interval. Log output is written after every log_iter iteration.	10
log_err	Iteration log error output. Error reported (if necessary) after every log_err iterations.	10
debug	Debug option. Prints out complete LGO status report to listing file. 0 No 1 Yes	0
callConopt	Number of seconds given for an optional secondary optimization phase using CONOPT (when available). CONOPT terminates after at most CallConopt seconds. This solver phase also determines duals for the final solution point.	5
help	Prints all available GAMS and GAMS /LGO solver options in the log and listing files.	No printout

formatted for the purposes of this article (to fit the available space better, when necessary).

```
$title Global Optimization Model Example GO_test_2v_2c
$ontext
Find a solution to the system of nonlinear equations
x-sin(2x+3y)-cos(3x-5y)=0
y-sin(x-2y)+cos(x+3y)=0.
This is a 2-variable, 2-constraint global optimization test
problem in itself that could have (in fact, it has) multiple
solutions. Therefore we will determine the minimal norm solution.
$offtext
* Define optimization model
variables obj, x, y;
equations defobj, con1, con2;
* Define an objective function as the squared norm of the
solution to the equations.
defobj.. obj =e= x*x+y*y;
```

Table 2 GAMS/LGO specific options

Option	Description	Default
opmode	Specifies the LGO search mode used. 0 Local search started from the given nominal solution, without a preceding global search (LS) 1 Global branch-and-bound search and local search (BB+LS) 2 Global adaptive random search and local search (GARS+LS) 3 Global multistart random search and local search (MS+LS)	3
G_maxfct	Maximum number of merit (model) function evaluations before termination of global search phase (BB, GARS, or MS). In the default setting, n is the number of variables and m is the number of constraints. The difficulty of global optimization models varies greatly: for difficult models, g_maxfct can be increased as deemed necessary.	500(n+m)
max_nosuc	Maximum number of merit function evaluations in global search phase (BB, GARS, or MS) where no improvement is made. Global search phase terminates upon reaching this limit. For difficult models, max_nosuc can be increased as deemed necessary.	100(n+m)
penmult	Constraint penalty multiplier. Global merit function is defined as objective + the constraint violations weighted by penmult.	100
acc_tr	Global search termination criterion parameter (acceptability threshold). The global search phase (BB, GARS, or MS) ends if an overall merit function value is found in the global search phase that is less than or equal to acc_tr.	−1.0E10
fct_trg	Target objective function value; a partial stopping criterion in the local search phase.	-1.0E10
fi_tol	Local search (merit function improvement) tolerance; a stopping criterion in the local search phase.	1.0E-6
con_tol	Maximal constraint violation tolerance in local search.	1.0E-6
kt_tol	Kuhn-Tucker local optimality condition violation tolerance.	1.0E-6
irngs	Random number seed.	0
Var_lo	Smallest (default) lower bound, unless set by user.	−1.0E+6
Var_up	Largest (default) upper bound, unless set by user.	1.0E+6
Bad_obj	Default value for objective function, if evaluation errors occur.	1.0E+8

```
* Define the constraints.
con1.. x-sin(2*x+3*y)-cos(3*x-5*y) =e= 0 ;
con2.. y-sin(x-2*y)+cos(x+3*y) =e= 0;
* Define bounds and nominal values.
* See the corresponding .lo, .l and .up index notation.
x.lo = -2; x.l = 2.5; x.up = 3;
y.lo = -2.5; y.l = 1.3; y.up = 1.5;
* The model m is defined by the information given above.
model m / all /;
* Invoke the LGO solver option for solving this nonlinear
* programming (NLP) model.
option nlp=lgo;
solve m minimizing obj using nlp;
* Set precision for the display of results.
option decimals = 8;
* Display the solution found.
display obj.l, x.l, y.l;
```

The summary of the GAMS /LGO run and its results – cited directly from the corresponding GAMS log file – are displayed below:

```
LGO 1.0 Aug 1, 2005 WIN.LG.NA 22.0 003.000.000.VIS
Lib005-060224

  LGO Lipschitz Global Optimization
  Copyright (C) Pinter Consulting Services, Inc.
  129 Glenforest Drive, Halifax, NS, Canada B3M 1J2
  E-mail : jdpinter@hfx.eastlink.ca
  Website: www.pinterconsulting.com

  1 defined, 0 fixed, 0 free
  2 LGO equations and 2 LGO variables

  Iter    Objective    SumInf    MaxInf  Seconds  Errors
  2409  9.563139E-02  0.00E+00  0.0E+00  0.015
--- LGO Exit: Normal completion - Global solution
```

The solution arguments and the optimum value (to the maximal 8 decimals precision supported by GAMS) are $x \approx -0.17334605$, $y \approx -0.25609087$, $obj \approx 0.09563139$.

Recall that the LGO iteration count (2409 in this example) is based on the total number of model function evaluations in the (default) multi-start global and local search phases, without using analytical gradient or higher order information. The total runtime is approximately 0.015 seconds (note that very small runtimes are sometimes indicated as 0.000 solution time by GAMS).

As shown below, the same model happens to be solvable also by LGO's local search mode: this is invoked by the setting opmode = 0 in the options file lgo.opt.

```
--- Using option file I:\...\gamsdir\lgo.opt
  > opmode=0

  1 defined, 0 fixed, 0 free
  2 LGO equations and 2 LGO variables
```

```
 Iter     Objective     SumInf     MaxInf     Seconds     Errors
  291  9.921421E-01  0.00E+00   0.0E+00     0.000

— LGO Exit: Normal completion - Local solution
```

Here LGO uses only 291 model function evaluations, and finds a local solution with a greater norm than the global search based solution: $x \approx 0.83883539$, $y \approx 0.53711941$, obj ≈ 0.99214207.

As indicated earlier, GAMS /LGO can also be used in conjunction with other available solvers. For instance, an LGO solver run could be directly followed by a call to the local NLP solver CONOPT (Drud 1996) from the best solution point found, assuming the availability of that solver. Such polishing steps may be especially useful in difficult models, since model re-scaling and restart invoked by using another solver could, in general, improve the precision of the solution found. If all went well, then CONOPT will essentially just confirm the solution found by LGO as optimal (without distinguishing between global or local solutions). This is the case also in the example above. At the same time, the local solvers MINOS, CONOPT, and SNOPT all report infeasible results for this model when used on their own, indicating the genuine need for a global solver to handle this very small (two-variable, box-constrained GO) model. Let us note also that LGO has found a local solution in its LS mode that CONOPT could not improve. The OQNLP solver returns the same global solution as LGO (while cautiously stating that it is a local solution). The BARON solver can not handle trigonometric functions (as of the version BARON 7.2.5 August 1, 2005, available to the author), and thus it could not be used to solve this illustrative example.

This numerical example illustrates that nonlinear—especially, global—optimization models can be truly challenging, even in relatively simple, small-scale instances. This fact in itself motivates the use of several solver options whenever available.

Example 2 Packing identical size circles in the unit circle

Next, we consider a well-known circle packing model that has been intensively studied at least for several decades, mostly by "pure" mathematicians (up to recent times, with no or modest use of computers). Given the unit circle C (of radius 1), and a positive integer k, find a set of k identical size circles C_i $i = 1, \ldots, k$ with the maximal possible radius $r = r(k)$ so that all C_i are contained by C, in a non-overlapping arrangement. For illustration, an optimized configuration for $k = 20$ is displayed below. This arrangement has been found and visualized by using the MathOptimizer Professional (LGO linked to the Mathematica platform) software implementation (Pintér and Kampas 2003).

There exists a significant body of literature (books, articles, dissertations, and web sites) discussing various packings of identical size circles. For example, Melissen (1997) provides a detailed review of packing model variants and related analytical results, with more than 350 topical references. For the problem stated above, analytical proofs are known only for $k \leq 11$ (as per Melissen's cited work), although putative arrangements are known (as of today) for up to about 500 circles. With respect to the best known configurations, we will use the information presented by Specht (2005): this website also provides further topical references.

We will assume that the unit circle "container" is centered at the origin. For a given k, denote the optimized circle radius by $r = r(k)$, and the centre of circle i by $c_i = (x_i, y_i)$ $i = 1, \ldots, k$. Then we can formulate the circle packing problem as shown below.

$$
\begin{array}{l}
\text{maximize } r \\
2r \le ||c_i - c_j||, \quad 1 \le i < j \le k, \\
\|c_i\| + r \le 1, \quad 1 \le i \le k, \\
0 \le x_i \le 1, \quad 1 \le i \le k, \\
0 \le y_i \le 1, \quad 1 \le i \le k, \\
0 \le r \le 1.
\end{array} \tag{7}
$$

Here $\|c_i - c_j|| = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$ and $\|c_i\| := (x_i^2 + y_i^2)^{1/2}$.

Model (7) has $2k + 1$ decision variables, $k(k-1)/2$ non-convex (reverse convex) constraints that represent the "circles do not overlap" condition, and k convex nonlinear constraints that represent the "all circles in container" condition. Observe that the number of non-convex constraints increases essentially at a quadratic rate as k grows: this fact makes model (7) and similar problems rather tough to solve by generic GO software. Let us also point out that this model has obvious structural symmetry which could be exploited in a modeling and solution procedure. (Fig. 3 shows essentially the same configuration as Specht's website for the $k = 20$ case, except that it is rotated, and that the innermost circle can be moved around to some extent.) For example, a lexicographic arrangement of the circle centers could be required: this would narrow the search domain, but also would lead to additional model constraints. Instead of following such a modeling path, we will use GAMS /LGO and several other solvers in a completely "blind" manner, since we are interested here only in their generic solver capabilities. (Again, all solvers will be used with their default settings.) Notice additionally that the variable bounds could be made a bit tighter as a function of k, but again—in this illustrative example—we take the "easy road to modeling" on purpose. The only tighter bound that we shall use is $0.05 \le r \le 0.4$, based on the fact that we will solve model-instances with $k = 5, 10, 15, \ldots, 60$. (These bounds could also

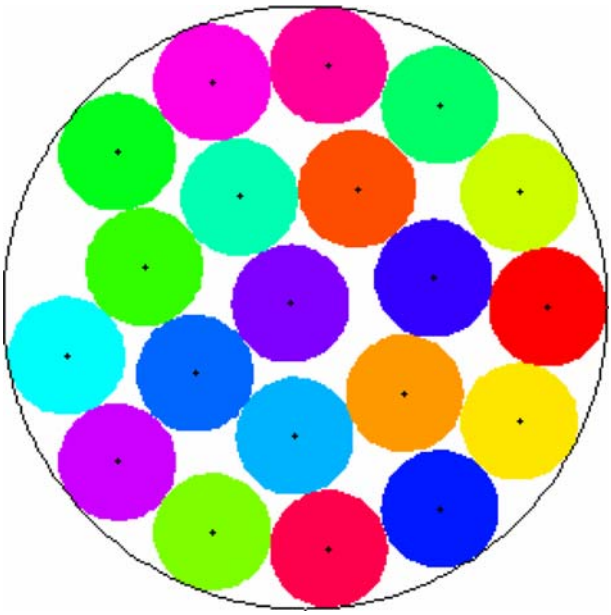

Fig. 3 Packing 20 uniform size circles in the unit circle by global optimization

be made tighter.) The current standard GAMS /LGO solver is set up to handle maximally 3,000 variables and 2,000 constraints (the general constraint handling limit is in addition to the explicitly handled variable lower/upper bounds): hence, the $k = 65$ instance would exceed the constraint limitation.

A possible GAMS formulation of the circle packing problem (7) is displayed below. The example also illustrates the compact and transparent nature of the GAMS model formulation. In particular, notice (see the related in-code note) the immediate scalability of the model by changing a single value. Let us also remark that instead of the non-linear constraints stated in (7) their equivalent forms

$$4r^2 \le ||c_i - c_j||^2 \quad 1 \le i < j \le k,$$
$$\|c_i\|^2 \le (1 - r)^2 \quad 1 \le i \le k$$

are used in our GAMS model.

```
$title Packing identical size circles in the unit circle
$ontext
Given the unit circle (of radius 1), find a set of identical
size circles with an optimized (maximal) radius r so that all
such circles are contained by the unit circle, in a non-overlapping
arrangement.

One can set up model-instances simply by changing the second
index below: that is, i1*ik will define the k-circle model-instance.
$offtext
Sets
         i / i1 * i5 / ;
* The alias command gives more than one name to a set.
alias (i, j) ;
* Here we define the set ij(i,j) of ordered pairs i,j i<j.
set ij(i,j);
ij(i,j)$ (ord(i) < ord(j)) = yes;

variables
           r
         x(i)
         y(i);
* Note that the equations keyword is interpreted as
* constraints (hence, it also covers inequalities). (sqr(z)
* denotes z*z.)
equations
         circumscribe(i)
         nooverlap(i,j);

circumscribe(i).. sqr(1.-r) =g= sqr(x(i)) + sqr(y(i));
nooverlap(ij(i,j)).. sqr(x(i)-x(j))+sqr(y(i)-y(j))
=g= 4*sqr(r);
```

```
x.lo(i) = -1.; x.up(i) = 1.;
y.lo(i) = -1.; y.up(i) = 1.;
r.lo = 0.05; r.up = 0.4;

model m /all/;

solve m using nlp maximizing r;
```

In all program runs, first we use LGO, automatically followed by CONOPT: as discussed earlier, this combination could (and in some cases, does) lead to improved results. The results for $k = 5, 10, 15, \ldots, 60$ packed circles are summarized by Table 3. In the table heading "k" denotes the number of circles; the column entries under "LGO" are the optimized circle radii $r = r(k)$ found by LGO in itself; while "LGO + CONOPT" heads the column of radii found by the sequential combination of the two solvers. The "Best known result" column is cited from (Specht 2005), rounded to the 11-decimal precision reported by GAMS / LGO + CONOPT. The "NFE" denotes the number of model function evaluations done by LGO. The "Runtime" column shows the LGO + CONOPT times in seconds, separated by a + sign. Again, all runtimes (especially the very small ones) depend also on the state of the computer and operating system used, and hence the runtimes reported are approximate.

Without going into a detailed numerical analysis of these illustrative results, one can draw a few key conclusions.

- LGO in itself finds solutions (in its default operational mode, with standard option settings) for up to 60 circles that are within 93–99.9999% relative precision to the best solution known.
- All results found by LGO + CONOPT are within 97.5% of the putative optimum, typically within 99.5% or much higher precision. The addition of CONOPT—when started from the solution found by LGO—requires a relatively very modest extra computational effort.
- LGO (or LGO + CONOPT) require a computational effort that apparently scales well in relation to the model-instance size; the runtimes are also fairly reasonable considering today's computers. For example the 60-circle model-instance has

Table 3 Packing identical size circles in the unit circle: summary of results

k	LGO	LGO + CONOPT	Best known result	NFE	Runtime (sec)
5	0.37019190816	0.37019190816	0.37019190816	17,383	0.125 + 0.016
10	0.26077981223	0.26077981216	0.26225892419	42,678	0.907 + 0.094
15	0.22088519921	0.22088519954	0.22117253909	81,849	3.844 + 0.109
20	0.19522401104	0.19522401102	0.19522401102	131,203	10.344 + 0.016
25	0.17352441376	0.17352441371	0.17382766142	203,233	24.672 + 0.031
30	0.15137590832	0.16080454102	0.16134910906	295,572	50.500 + 0.125
35	0.14656680267	0.14866214852	0.14931677664	356,579	85.031 + 0.109
40	0.13781012238	0.13857348501	0.14037360420	534,093	165.391 + 0.188
45	0.13177203692	0.13177203691	0.13204959425	580,471	221.079 + 0.109
50	0.12569303835	0.12569303835	0.12582548953	739,523	376.484 + 0.078
55	0.11486959821	0.11871318638	0.12178632453	907,129	530.906 + 0.625
60	0.10731691701	0.11469879969	0.11565748013	979,374	669.016 + 1.234

121 decision variables and 1,830 mostly non-convex constraints: the combined LGO + CONOPT runtime is a little over 11 minutes.

To put these numerical results in perspective, let us also mention that in many cases the best known result has been found by a significant effort both in terms of modeling (research time) and computational resources—as opposed to the effort reported here (from a fraction of a second to about 11 min, on a desktop PC). Note furthermore that if we "tweak" the model and/or the solvers, then the default results shown above can be improved. Solver option settings could allow LGO to apply more global search effort, and/or to apply its other solver modes. For example, increasing the global search effort in LGO to 1,000,000 steps, for $k = 10$, we obtain the LGO + CONOPT solution 0.26225892419 in less than 22 seconds: this value coincides with the best known solution to at least 11 decimals. The modeling procedure itself could also be refined, e.g., by adding randomized or grid-like initial circle arrangements, using more tight bounds on r, etc. However, all such "tweaking" has been avoided, since our present objective is to report results using all GAMS solvers in their default mode.

We have attempted to use also several other solvers to handle this model-class: a brief summary of our numerical experiments is reported below. CONOPT, MINOS, and SNOPT—all being high-quality local solvers—have had difficulties in finding feasible solutions when used in a stand-alone mode. Specifically, CONOPT and SNOPT report model infeasibility for all model-instances considered here. MINOS finds a local solution for $k = 5$, then for all other model-instances it reports infeasibility or too many iterations, without finding a solution.

For the smaller model-instances, the global solver OQNLP (in conjunction with the local solver LSGRG) finds good quality solutions in a time frame similar to that of LGO, although the summary report typically states an "Iteration Count Exceeded" message. Unfortunately, OQNLP does not report program execution timings: however, in our experiments it could not complete the solution process for $k = 40$ circles within the preset 1,000 s time limit.

The global solver BARON solves the $k = 5$ model-instance in its preprocessing phase (that uses MINOS), and reports that solution. However, the $k = 10$ case is not solved (that is, BARON does not terminate) in 1,200 s. Therefore no further attempts were made to use BARON in solving larger model-instances. To be fair, let us remark here that the solution found during preprocessing for the case $k = 10$ by BARON+MINOS is, in fact, close to the best known solution. However, thereafter BARON seems to require a significant amount of time to narrow the gap between the best solution (lower bound) and the stated upper bound, since the lower bound found during preprocessing did not improve at all in 1,200 s. (Recall here our earlier related comment.) As an added note, BARON could not complete even its parser and preprocessing phase for the $k = 60$ case in 300 s.

The illustrative test results summarized above indicate that LGO in its default operational mode—with or without CONOPT—produces good quality numerical solutions with a reasonable effort, when solving models from an arguably non-trivial test model-class. The results also demonstrate the need for using global solvers to handle such general nonlinear models. We are convinced that these general observations are valid, in spite of the unavoidable "bias" aspects of test model selection, solver settings, and benchmarking methodology. We also strongly believe that it remains impossible to draw far-reaching conclusions based on a limited set of examples. Our illustrative numerical results (as well as furthur comparative test results that are not

presented here) do not justify the claim "Among the currently available global solvers, BARON is the fastest and the most robust one. . ." cited from a recent benchmarking study by Neumaier et al. (2005).

We will not go into further details on benchmarking here which is a substantial topic in itself. From the related literature we refer only to some recent work with GO relevance by Dolan and Moré (2002), Pintér (2002), Ali et al. (2005), Khompatraporn et al. (2005), Neumaier (2005b). GAMS specific studies and numerical results are discussed e.g., by Bussieck et al. (2003), GAMS Performance World (2003), Pintér (2003c), Mittelmann and Pruessner (2006), and Ugray et al. (2006). The computational study Pintér (2003c) includes also many of the standard academic tests known from the GO literature collected in chapters of Floudas et al. (1999) and available in GAMS format (GAMS Global World, 2005).

6 Concluding remarks

Computational global optimization is coming of age. Recently, several global optimization solvers have been implemented for use within the framework of prominent modeling and optimization environments. As a result, global optimization methodology and software is increasingly used worldwide, and it already has significant applications. In addition to its obvious educational perspectives, prominent research and commercial application areas include biotechnology, chemical and process industries, econometrics and finance, engineering design, medical research, and scientific modeling. For a selection of books (and two substantial book chapters) that include also detailed test results, application examples and case studies, consult e.g. Floudas (1999), Floudas et al. (1999), Grossmann (1996), Liberti and Maculan (2006), Neumaier (2004), Papalambros and Wilde (2000), Pintér (1996a, 2002, 2006), Tawarmalani and Sahinidis (2002), and Zabinsky (2003).

Regarding LGO implementations and their applications, Pintér (2005b) presents an overview of several of these with numerical examples. More detailed numerical studies and specific applications are discussed, e.g., in the following works:

- Minimal potential energy models in computational physics and chemistry (Pintér 2001b, Stortelder et al. 2001)
- Laser design (Isenor et al. 2003)
- Model calibration (Pintér 2003b)
- A detailed benchmarking study using several GAMS model libraries (Pintér 2003c)
- Radiotherapy planning (Tervo et al. 2003)
- Design optimization in acoustic engineering (Pintér and Purcell 2003)
- Various application examples and case studies developed for the Maple platform (Maplesoft 2004, Pintér and Purcell 2006)
- A comparative numerical study of global optimization tools in Mathematica (Kampas and Pintér 2005)
- Generalized (non-uniform) circle packings (Pintér and Kampas 2005) and other object configuration analysis problems (Kampas and Pintér 2006)
- Circle packing models, with industrial application perspectives (Castillo et al. 2005)
- Numerous further applications that are part of proprietary studies.

The listed examples indicate the usability of global optimization technology across an increasing range of professional studies and real-world applications.

Acknowledgements I wish to thank Alexander Meeraus, Steven Dirkse, Armin Pruessner, and other colleagues at the GAMS Development Corporation, for their contributions to the GAMS/LGO solver implementation and its tests. Thanks are due also to two anonymous referees for their careful reading of the paper and for constructive suggestions.

References

Ali, M.M., Khompatraporn, Ch., Zabinsky, Z.B.: A numerical evaluation of several global optimization algorithms on selected benchmark test problems. J. Global Optim. **31**, 635–672 (2005)

Aris, R.: Mathematical Modeling: A Chemical Engineer's Perspective. Academic Press, San Diego, CA (1999)

Bartholomew-Biggs, M.: Nonlinear Optimization with Financial Applications. Kluwer Academic Publishers, Dordrecht (2005)

Benson, H.P., Sun, E.: LGO—Versatile tool for global optimization. ORMS Today **27**(5), 52–55 (2000) see http://www.lionhrtpub.com/orms/orms-10-00/swr.html

Bliek, Ch., Spellucci, P., Vicente, L.N., Neumaier, A., Granvilliers, L., Monfroy, E., Benhamou, F., Huens, E., Van Hentenryck, P., Sam-Haroud, D., Faltings, B.: Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art. COCONUT Project Report (2001) For further information on the COCONUT Project, including this downloadable report, see http://www.mat. univie.ac.at/~neum/glopt/coconut/index.html.

Bracken, J., McCormick, G.P.: Selected Applications of Nonlinear Programming. Wiley, New York (1968)

Brooke, A., Kendrick, D., Meeraus, A.: GAMS: A User's Guide. The Scientific Press, Redwood City, CA (1988)

Bussieck, M.R., Drud, A.S., Meeraus, A., Pruessner, A.: Quality assurance and global optimization. Presented at the 1st International Workshop on Global Constrained Optimization and Constraint Satisfaction COCOS 2002, Valbonne, France (2003)

Castillo, I.: Maple 10 and the Global Optimization Toolbox. ORMS Today **32**(6), 56–60 (2005) see http://www.lionhrtpub.com/orms/orms-12-05/swr.html

Castillo, I., Kampas, F.J., Pintér, J.D.: Solving circle packing problems by global optimization: numerical results and industrial applications. (Submitted for publication) (2005)

Chong, E.K.P., Zak, S.H.: An Introduction to Optimization, 2nd edn. Wiley, New York (2001)

Cogan, B.: How to get the best out of optimisation software. Sci. Comput. World **71**, 67–68 (2003) see http://www.scientific-computing.com/scwjulaug03review_optimisation.html

Conn, A., Gould, N.I.M., Toint, Ph.L.: LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization. Springer-Verlag, Heidelberg (1992)

Diwekar, U.: Introduction to Applied Optimization. Kluwer Academic Publishers, Dordrecht (2003)

Dörner, D.: The Logic of Failure. Perseus Books, Cambridge, MA (1996)

Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)

Drud, A.S.: CONOPT: A System for Large-Scale Nonlinear Optimization, Reference Manual for CONOPT Subroutine Library. ARKI Consulting and Development A/S, Bagsvaerd, Denmark (1996)

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

Ferris, M.C.: MATLAB and GAMS: Interfacing optimization and visualization software. Mathematical Programming Technical Report 98–19. University of Wisconsin, Wisconsin (2005) see http://www.cs. wisc.edu/math-prog/matlab.html.

Fletcher, R., Leyffer, S.: User Manual for filterSQP. Dundee University Numerical Analysis Report N/A 181 (1998)

Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Math. Program. **91**, 239–269 (2002)

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

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

Fourer, R.: Nonlinear Programming Frequently Asked Questions. Maintained by the Optimization Technology Center of Northwestern University and Argonne National Laboratory (2005) see http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html
Frontline Systems: Premium Solver Platform—Field-Installable Solver Engines. Frontline Systems Inc., Incline Village, NV (2005) see http://www.solver.com
GAMS Development Corporation: GAMS. GAMS Development Corporation, Washington, DC (2005) see http://www.gams.com.
GAMS Global World: GLOBAL Library: A Collection of Nonlinear Programming Models (2005) see http://www.gamsworld.org/global/globallib.htm.
GAMS Performance World: PAVER—Automated Performance Analysis & Visualization (2005) see http://www.gamsworld.org/performance/paver.
Gershenfeld, N.: The Nature of Mathematical Modeling. Cambridge University Press, Cambridge (1999)
Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An algorithm for large-scale constrained optimization. SIAM J. Optim. **12**, 979–1006 (2002)
Gould, N.I.M., Orban, D., Toint, Ph.L.: GALAHAD—A library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. Technical Report RAL-TR-2002-014. Rutherford Appleton Laboratory, Chilton, Oxfordshire, England (2002)
Grossmann, I.E. (ed.): Global Optimization in Enginnering Design. Kluwer Academic Publishers, Dordrecht/Boston/London (1996)
Hansen, P.E., Jørgensen, S.E. (eds.): Introduction to Environmental Management. Elsevier, Amsterdam (1991)
Henrion, D.: A review of the global optimization toolbox for maple. IEEE Control Syst. Mag. (To appear) (2006) Available online at http://www.laas.fr/∼henrion/papers/mapleglobopt.pdf
Hillier, F.J., Lieberman, G.J.: Introduction to Operations Research, 8th edn. McGraw-Hill, New York (2005)
Horst, R., Pardalos, P.M. (eds.): Handbook of Global Optimization, vol. 1. Kluwer Academic Publishers, Dordrecht (1995)
Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 3rd edn. Springer-Verlag, Berlin (1996)
Isenor, G., Pintér, J.D., Cada, M.: A global optimization approach to laser design. Optim. Eng. **4**, 177–196 (2003)
Kalwelagen, E.: Interfacing GAMS with other applications. Tutorial and examples. GAMS Development Corporation, Washington, DC (2005) See http://www.gams.com/∼erwin/interface/ interface.html
Kampas, F.J., Pintér, J.D.: Global optimization in Mathematica: a comparative numerical study. In: Proceedings of the 2005 Wolfram Technology Conference Champaign, IL, 2005. http://library.wolfram.com/infocenter/Conferences/5824/
Kampas, F.J., Pintér, J.D.: Configuration analysis and design by using optimization tools in Mathematica. The Mathematica Journal **10**, 128–154 (2006)
Khompatraporn, Ch., Pintér, J.D., Zabinsky, Z.B.: Comparative assessment of algorithms and software for global optimization. J. Global Optim. **31**, 613–633 (2005)
Lasdon, L.S., Smith, S.: Solving large sparse nonlinear programs using GRG. ORSA J. Comput. **4**, 2–15 (1992)
Leyffer, S., Nocedal, J. (eds.): Large scale nonconvex optimization. SIAG/OPT Views News **14** (1) (2003) Published by the SIAM Activity Group on Optimization (25 pages.) Available online at http://www.mat.uc.pt/siagopt/.
Liberti, L., Maculan, N. (eds.): Global Optimization: From Theory to Implementation. Springer Science + Business Media, New York (2006)
LINDO Systems: LINDO Solver Suite. LINDO Systems Inc., Chicago, IL (2005) see www.lindo.com.
Lopez, R.J.: Advanced Engineering Mathematics with Maple. Electronic book edition published by Maplesoft Inc., Waterloo, ON (2005)
Maplesoft: Global Optimization Toolbox for Maple. Maplesoft Inc., Waterloo, ON (2004) see http://www.maplesoft.com/products/toolboxes/globaloptimization.
Maplesoft: Maple. Maplesoft Inc., Waterloo, ON (2005) see http://www.maplesoft.com.
(The) MathWorks: MATLAB. The MathWorks, Inc., Natick, MA (2004) see http:/www.mathworks.com
Maximal Software: MPL. Distributed by Maximal Software Inc. Arlington, VA (2005) see http://www.maximal-usa.com.

McCarl, B.A.: McCarl's GAMS User Guide (2004) see http://www.gams.com/dd/docs/ bigdocs/ gams2002/. (Current edition dated 2004.)
Melissen, J.B.M.: Packing and Covering with Circles. Ph.D. Dissertation, University of Utrecht, Utrecht (1997)
Mittelmann, H.D., Spellucci, P.: Decision Tree for Optimization Software (2005) see http:// plato.asu.edu/guide.html
Mittelmann, H.D., Pruessner, A.: A server for automated performance analysis of benchmarking data. Optim. Methods Softw. **21**, 105–120 (2006)
Murray, J.D.: Mathematical Biology. Springer-Verlag, Berlin (1983)
Murtagh, B.A., Saunders, M.A.: MINOS 5.4 User's Guide. Technical Report SOL 83-20R (Revised edn.) Department of Operations Research, Stanford University, Stanford, CA (1995)
Neumaier, A.: Complete search in continuous optimization and constraint satisfaction. In: Iserles, A., (ed.) Acta Numerica 2004, pp. 271–369. Cambridge University Press, Cambridge (2004)
Neumaier, A.: Global Optimization (2005a) see http://www.mat.univie.ac.at/∼neum/ glopt.html.
Neumaier, A.: COCONUT Project (2005b) see http://www.mat.univie.ac.at/∼neum/ glopt/coconut/ index.html.
Neumaier, A., Scherbina, O., Huyer, W., Vinkó, T.: A comparison of complete global optimization solvers. Math. Program. **103**, 335–356 (2005)
Papalambros, P.Y., Wilde, D.J.: Principles of Optimal Design. Cambridge University Press, Cambridge (2000)
Paragon Decision Technology: AIMMS. Paragon Decision Technology BV, Haarlem, The Netherlands (2005) see http://www.aimms.com.
Pardalos, P.M., Resende, M.G.H. (eds.): Handbook of Applied Optimization. Oxford University Press, Oxford (2002)
Pardalos, P.M., Romeijn, H.E. (eds.): Handbook of Global Optimization, vol. 2. Kluwer Academic Publishers, Dordrecht (2002)
Pintér, J.D.: Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications). Kluwer Academic Publishers, Dordrecht (1996a)
Pintér, J.D.: Continuous global optimization software: A brief review. Optima **52**, 1–8 (1996b) see http://plato.la.asu.edu/gom.html.
Pintér, J.D.: LGO—A program system for continuous and Lipschitz optimization. In: Bomze, I.M., Csendes, T., Horst, R., Pardalos, P.M., (eds.) Developments in Global Optimization, pp. 183–197. Kluwer Academic Publishers, Dordrecht (1997)
Pintér, J.D.: Computational Global Optimization in Nonlinear Systems—An Interactive Tutorial. Lionheart Publishing, Inc. Atlanta, GA (2001a) see http://www.lionhrtpub.com/books/globaloptimization.html.
Pintér, J.D.: Globally optimized spherical point arrangements: model variants and illustrative results. Ann. Oper. Res. **104**, 213–230 (2001b)
Pintér, J.D.: Global optimization: software, test problems, and applications. In: Pardalos, P. M., Romeijn, H. E. (eds.) Handbook of Global Optimization, vol. 2, pp. 515–569. Kluwer Academic Publishers, Dordrecht (2002)
Pintér, J.D.: GAMS/LGO (User Guide) (2003a) http://www.gams.com/solvers/lgo.pdf
Pintér, J.D.: Globally optimized calibration of nonlinear models: techniques, software, and applications. Optim. Methods Softw. **18**, 335–355 (2003b)
Pintér, J.D.: GAMS/LGO nonlinear solver suite: key features, usage, and numerical performance (2003c) http://www.gams.com/solvers/GAMS_LGO_paper.pdf
Pintér, J.D.: The Maple Global Optimization Toolbox. Technical report; downloadable from the product page (2004) http://www.maplesoft.com/products/toolboxes/globaloptimization.
Pintér, J.D.: LGO – A Model Development System for Continuous Global Optimization. User's Guide. (Current revised edition.) Pintér Consulting Services, Inc., Halifax, NS (2005a)
Pintér, J.D.: Nonlinear optimization in modeling environments: software implementations for compilers, spreadsheets, modeling languages, and integrated computing systems. In: Jeyakumar, V., Rubinov, A.M. (eds.) Continuous Optimization: Current Trends and Applications, pp. 147–173. Springer Science + Business Media, New York (2005b)
Pintér, J.D.: AIMMS/LGO solver engine: a brief introduction and user's guide (2005c) see http:// www.aimms.com/aimms/download/solvers/aimms_lgo_solver_engine_introduction_and_ user_guide.pdf
Pintér, J.D.: Nonlinear optimization with MPL/LGO: introduction and user's guide. Distributed by Maximal Software, Inc., Arlington, VA, USA (2005d) www.maximal-usa.com.
Pintér, J.D. (ed.): Global Optimization—Scientific and Engineering Case Studies. Springer Science + Business Media, New York (2006)

Pintér, J.D., Holmström, K., Göran, A.O., Edvall, M.M.: User's Guide for TOMLAB/LGO. TOMLAB Optimization AB, Västerås, Sweden (2004). see http://tomlab.biz/docs/ TOMLAB_LGO.pdf
Pintér, J.D., Kampas, F.J.: MathOptimizer Professional—An Advanced Modeling and Optimization System for Mathematica, Using the LGO Solver Engine. User's Guide. Pintér Consulting Services Inc., Halifax, NS (2003)
Pintér, J.D., Kampas, F.J.: Nonlinear optimization in Mathematica with MathOptimizer Professional. Math. Educ. Res. **10**(2), 1–18 (2005)
Pintér, J.D., Kampas, F.J.: MathOptimizer Professional: key features and illustrative applications. In: Liberti, L., Maculan, N. (eds.) Global Optimization: From Theory to Implementation, pp. 263–280. Springer Science + Business Media, New York (2006)
Pintér, J.D., Linder, D., Chin, P.: Global Optimization Toolbox for Maple: an introduction with illustrative applications. Optim. Methods Softw. **21**, 565–582 (2006)
Pintér, J.D., Purcell, C.J.: Optimization of finite element models with MathOptimizer and ModelMaker. Proceedings of the 2003 Mathematica Developer Conference, Champaign, IL. (2003) Available at http://library.wolfram.com/infocenter/Articles/5347/
Powell, M.J.D.: UOBYQA: unconstrained optimization by quadratic approximation. Math. Program. **92**, 555–582 (2002)
Schittkowski, K.: Numerical Data Fitting in Dynamical Systems. Kluwer Academic Publishers, Dordrecht (2002)
Specht, E.: www.packomania.com (2005)
Steeb, W-H.: The Nonlinear Workbook, 3rd edn. World Scientific, Singapore (2005)
Stojanovic, S.: Computational Financial Mathematics Using Mathematica. Birkhäuser, Boston (2003)
Stortelder, W.J.H., de Swart, J.J.B., Pintér, J.D.: Finding elliptic Fekete points sets: two numerical solution approaches. J. Comput. Appl. Math. **130**, 205–216 (2001)
Strongin, R.G., Sergeyev, Ya.D.: Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms. Kluwer Academic Publishers, Dordrecht (2000)
Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Kluwer Academic Publishers, Dordrecht (2002)
Tervo, J., Kolmonen, P., Lyyra-Laitinen, T., Pintér, J.D., Lahtinen, T.: An optimization-based approach to the multiple static delivery technique in radiation therapy. Ann. Oper. Res. **119**, 205–227 (2003)
TOMLAB Optimization: TOMLAB. TOMLAB Optimization AB, Västerås, Sweden (2005) http://www.tomlab.biz
Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., Marti, R.: A multistart scatter search heuristic for smooth NLP and MINLP problems. INFORMS J. Comput. (To appear.) (2006). See http://www.utexas.edu/courses/lasdon/ijocmultistart5.htm.
Vanderbei, R.J.: LOQO User's manual—version 3.10. Optim. Methods Softw. **12**, 485–514 (1999)
Waltz, R., Nocedal, J.: KNITRO 2.0 User's Manual. Technical Report OTC 2003/05, Optimization Technology Center, Northwestern University, Evanston, IL (2003) see also Ziena Optimization, Inc. http://www.ziena.com/knitro/manual.htm.
Wass, J.A.: Global optimization with Maple. Sci. Comput. **24**, 16 (2006)
Wolfram Research: *Mathematica.* (2005) see http://www.wolfram.com/
Wright, M.H.: Direct search methods: once scorned, now respectable. In: Griffiths, D.F., Watson, G.A. (eds.) Numerical Analysis 1995: Proceedings of the 1995 Biennial Conference on Numerical Analysis, pp. 191–208. Addison Wesley Longman Ltd, Reading, MA (1996)
Zabinsky, Z.B.: Stochastic Adaptive Search for Global Optimization. Kluwer Academic Publishers, Dordrecht (2003)