Nonlinear Optimization with GAMS /LGO

JÁNOS D. PINTÉR

Pintér Consulting Services, Inc.
129 Glenforest Drive, Halifax, NS, Canada B3M 1J2
jdpinter@hfx.eastlink.ca www.pinterconsulting.com

Abstract. The Lipschitz Global Optimizer (LGO) software integrates global and local scope search methods, to handle nonlinear optimization models. This paper introduces 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.

Key words: Nonlinear (global and local) optimization; LGO solver suite; GAMS modeling system; GAMS /LGO solver option; numerical performance; 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 towards various professional audiences) consult for instance Aris (1999), Bracken and McCormick (1968), Dörner (1996), Gershenfeld (1999), 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 e.g. the topical chapters in Bartholomew-Biggs (2005), Chong and Zak (2001), Diwekar (2003), Edgar, Himmelblau, and Lasdon (2001), Pardalos and Resende (2002), Hillier and Lieberman (2005).

Algorithmic advances and progress in computer technology have enabled the development of sophisticated nonlinear optimization software implementations. Among the currently available software products, one can mention, e.g. LANCELOT (Conn, Gould, and Toint, 1992) which implements an augmented Lagrangian based solution approach. Sequential quadratic programming methods are implemented in EASY-FIT (Schittkowski, 2002), filterSQP (Fletcher and Leyffer, 1998, 2002), GALAHAD (Gould, Orban, and Toint, 2002), and SNOPT (Gill, Murray, and Saunders, 2003). Another prominent class of methods is based on the reduced gradient (RG) approach and its generalization (GRG). 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, Himmelblau, and Lasdon, 2001), and in CONOPT (Drud, 1996). Interior point methodology is used in LOQO (Vanderbei, 2000) 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 prior information that provides a suitable starting point, even the best local search methods encounter difficulties in solving general nonlinear models. 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 i) does not have sufficient insight to guarantee an essentially convex model structure, and ii) 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 books cover the most frequently used GO model types and solution strategies, with information on software and various application areas.

The key theoretical developments have been followed by 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 e.g. 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 programming methods and software, with numerous further links and pointers.

In this article, we introduce the GAMS /LGO solver engine for nonlinear optimization. First we formulate and briefly discuss the general GO model, then review the key features of the LGO solver suite, and discuss its GAMS-specific implementation. We also present reproducible numerical results, to illustrate the performance of GAMS /LGO.

2 Global Optimization: Model Statement and Specifications

Consider the continuous global optimization (CGO) model stated as

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

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

- $x \in \boldsymbol{R}^n$ n-dimensional real-valued vector of decision variables
- $f: \boldsymbol{R}^n \rightarrow \boldsymbol{R}$ 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
- $g: \boldsymbol{R}^n \rightarrow \boldsymbol{R}^m$ 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 (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 theorem of Weierstrass) guarantee that the global solution set X^* of the CGO model is non-empty. In many cases, X^* consists of a unique point x^*. However, it is easy to show GO model instances in which X^* is finite (with cardinality greater than one), countable, non-countable, and 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, certain model instances of (1) lead to very difficult numerical problems. For instance, the feasible set D could be disconnected, and some of its components could be non-convex; furthermore, the objective function f could be multi-extremal over D. In such cases, (1) could have an unknown number of global (as well as local) solutions. Let us point out 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 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 can grow exponentially as the model size (number of variables n and/or constraints m) increases.

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

$$(2) \qquad eqn1 := x - sin(2x+3y) - cos(3x-5y) = 0 \qquad eqn2 := y - sin(x-2y) + cos(x+3y) = 0.$$

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 $(eqn1)^2 + (eqn2)^2$ which vanishes at the solution(s). Although this reformulation leads to a rather simple (two-variable, box-constrained) model instance of (1), the resulting model has no apparent structure that could be easily exploited by an algorithmic search procedure.

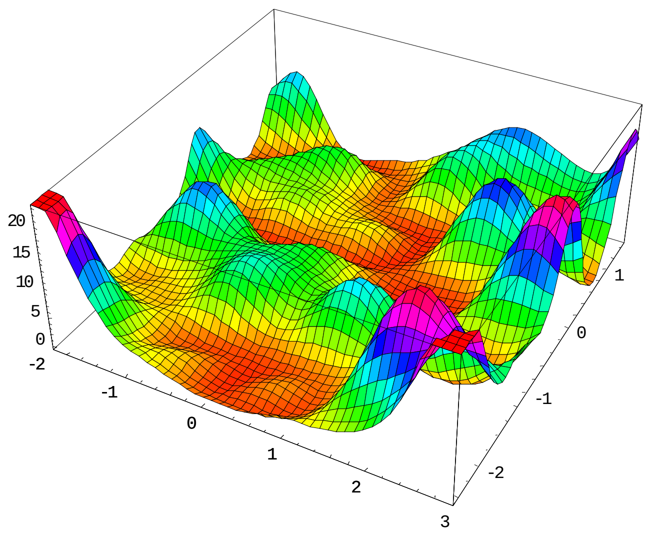

Figure 1.
An illustrative global optimization model.

In line with the discussion above, problem (2) could have multiple global and local solutions. For example, one of the numerical solutions is $x^*\approx 0.8388353863$, $y^*\approx 0.5371194096$; the residual absolute errors of the equations are $eqn1\approx 2.12628\cdot 10^{-10}$, $eqn2\approx -5.21127\cdot 10^{-11}$. This solution has been produced using the 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 points. Formally, one could accept an approximate numerical solution x_a^* that satisfies the relation

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

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

(4) $$f(x_e^*) \leq f^* + \varepsilon,$$

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 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 can require the Lipschitz-continuity of the model functions. Recall that a function h is Lipschitz-continuous in the set D, if the relation

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

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 smooth functions, 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,…,m) is often postulated theoretically, or it is estimated in practice: for related discussions, consult e.g. 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 model (1) with a continuous or Lipschitz structure is still very general, and it includes most GO problem types that occur in practice. As a consequence, it includes also very difficult problem instances – in arbitrarily low model dimension n, and even without added constraints (m=0) – that pose a challenge in any computational environment of today or tomorrow. For example, one can think of optimizing numerically an objective function which includes trigonometric functions with embedded high frequency arguments and high amplitude multipliers. Figure 1 displays a relatively “mild” instance of such a problem; (Pintér, 2002) discusses and solves model instances for n=1,…,10. Let us note additionally that the increase of model dimensionality (n and m) in itself can lead to an exponential increase of model complexity, in terms of the number of local/global optima. For a very simple illustrative example, one can think of maximizing $\|x\|$ over an n-dimensional finite size box region B that includes the origin. In this case each of the 2^n box vertices is a locally optimal solution.

For given CGO model instances, the “most suitable” solution approach could vary to a considerable extent. A “universal” GO strategy can be expected to work for broad model classes, although its efficiency could be lower for certain problem types. 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 Figure 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 e.g. as an object file or a dynamic link library. It also includes models that incorporate computationally evaluated functions (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 specialized methods may be more efficient to handle such models.

LGO has been gradually developed since 1990, with continuing development and maintenance (as a proprietary solver system). 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, Holmström, Göran and Edvall, 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. The selected global solver component is automatically followed by the local solver. 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^*. (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 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.

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-continuity 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 multi-start 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 local search. 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

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

In (6) the index sets *E* and *I* denote the subsets of equality and inequality constraints, respectively. In the LS mode finite difference based gradient approximations are used (tacitly assuming at least local smoothness when needed for convergence). Again, this gradient-free approach supports also the optimization of "black box" systems. Let us note that "black box" handling is not a basic GAMS feature: however, it may become handy e.g. when interfacing GAMS with other (Excel, MATLAB, C, VB, Java, 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 global solver modes is automatically followed by local search. The local solver embedded in LGO implements a dense nonlinear optimization algorithm, without postulating or exploiting any specific further model structure. This GRG solver is based on classical nonlinear optimization techniques discussed e.g. by Edgar, Himmelblau, and Lasdon (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 region(s) of attraction where this solver mode is used.

As a result of using the above listed solver strategies, LGO will return – barring numerical problems and "unsuitable" option settings – a global search based solution (BB + LS, GARS + LS), several such solutions (MS + LS), or a local search 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), 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 perhaps requires a few 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 programming 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). There is also a compiler based LGO implementation that is enhanced by a simple Windows menu interface, providing an integrated development environment that can be used in conjunction with C and Fortran compilers. These implementations are discussed e.g. in (Pintér, 1997, 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, indicating also the year of product release):

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

Peer reviews discussing several of these implementations are also available: consult Benson and Sun (2000), Cogan (2003), and Castillo (2005), two further software reviews (Henrion, 2006; Wass, 2006) are to appear in 2006.

4 GAMS and the GAMS /LGO Solver Option

The General Algebraic Modeling System (GAMS) is a high-level model development environment that supports the analysis and solution of a broad range of optimization problems. GAMS is capable of handling advanced modeling 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. GAMS has been available since 1987, and it has a significant world-wide user base. The first edition of the user's guide (Brooke, Kendrick, and Meeraus, 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 a wealth of information: some of the key points are highlighted below.

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, MATLAB, and Gnuplot. 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 collected 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 uses 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 discussion of OQNLP and its performance, consult e.g., Ugray et al. (2006).

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

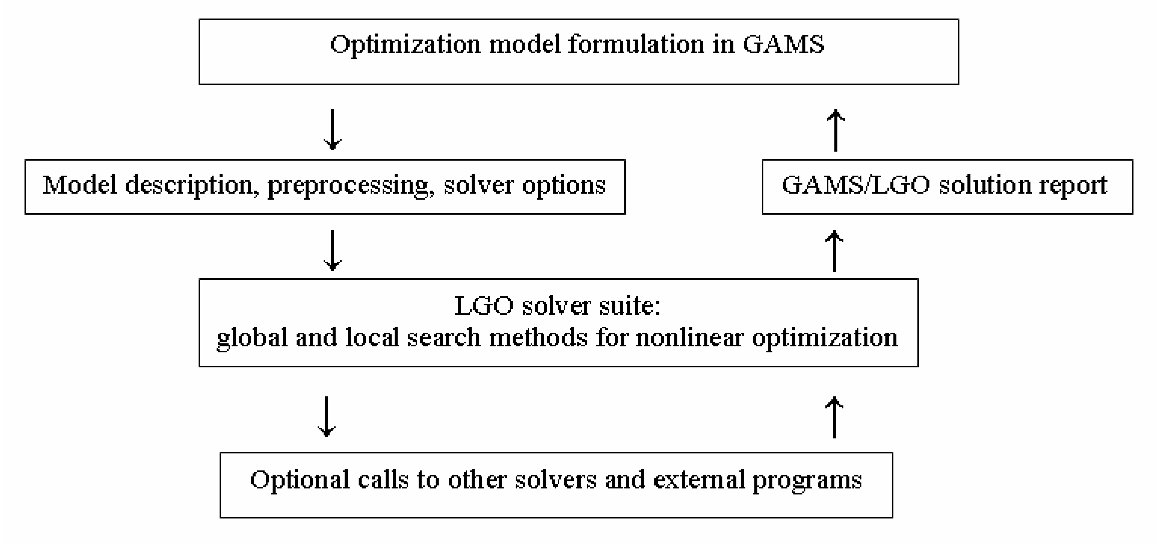
Optimization model formulation in GAMS
Model description, preprocessing, solver options
GAMS/LGO solution report
LGO solver suite:
global and local search methods for nonlinear optimization
Optional calls to other solvers and external programs

Figure 2.
GAMS /LGO modeling and solution procedure.

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.

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, 2003) 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.

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.	1000
log_time	Iteration log time interval in seconds. GAMS log output is generated every log time seconds.	0.5
log_iter	Iteration log time interval. Log output is written every log_iter iteration.	10
log_err	Iteration log error output. Error reported (if necessary) 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

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

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. According to our fairly extensive numerical experience, the default option settings shown above are suitable to solve at least small to moderate size GO problems without any “tweaking” of the parameters. (Let us also recall here the notes related to global search based solutions.) This 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 have been done on an AMD64 3.2 GHz processor based desktop machine, running under the Windows XP Professional 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). All GAMS language elements are denoted by **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 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;

* 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; (very small runtimes are sometimes indicated as `0.000` solution time by GAMS).

Let us note here that the same model happens to be solvable also by LGO's local search mode: this is invoked by a suitable setting in the options file (discussed later on) as shown below.

```
--- Using option file C:\...\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: $x \approx 0.83883539$, $y \approx 0.53711941$, $obj \approx 0.99214207$.

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 (small, box-constrained GO) model. (Notice 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 Aug 1, 2005, available to the author).

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

We will consider a well-known circle packing model that has been intensively studied at least for several decades, mostly by "pure" mathematicians (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,…,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 the visualized by using MathOptimizer Professional (LGO in conjunction with the Mathematica platform (Pintér and Kampas, 2003).

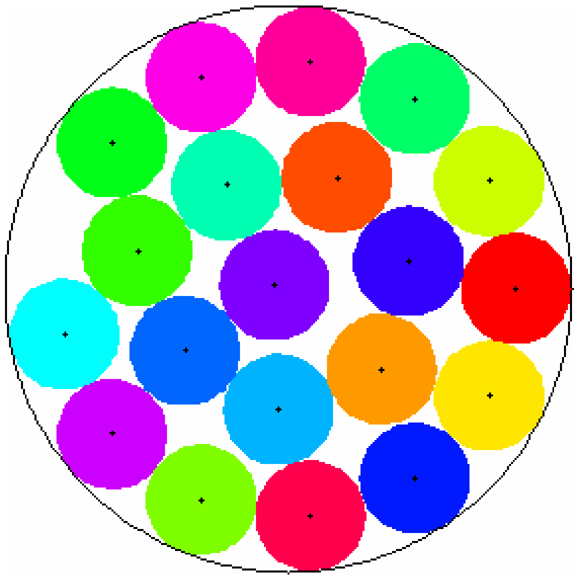

Figure 3.
Packing 20 uniform size circles in the unit circle by global optimization.
Source: Pintér and Kampas (2006).

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 such packing model statements 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): his website also provides further references to related work.

To formulate the mathematical model of the circle packing problem, we will assume that the unit circle is centered at the origin. For a given k, denote the optimized circle radius by r, and the centre of circle i by c_i=(x_i, y_i) i=1,…,k. Then we want to solve the following optimization problem:

(7) maximize r

$$\begin{array}{ll} 2r \le \| c_i - c_j \| & 1 \le i < j \le k \\ \| c_i \| + r \le 1 & 1 \le i \le k \\ 0 \le x_i \le 1 & 1 \le i \le k \\ 0 \le y_i \le 1 & 1 \le i \le k \\ 0 \le r \le 1. & \end{array}$$

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

The model (7) has $2k$+1 decision variables, $k(k$-1)/2 non-convex (reverse convex) constraints that represent the "no overlap" condition, and k convex nonlinear constraints that represent the "container" condition. Observe that the number of non-convex constraints increases essentially at a quadratic rate as k grows.

Let us also point out that this model has obvious structural symmetry which could be exploited in a modeling and solution procedure. (Figure 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 added constraints. Instead of following this modeling path, we will use GAMS /LGO (and several other solvers) in a completely "blind" manner, since we are interested in their generic solver capabilities. (Again, all solvers are 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,…,60. (These bounds could also be made tighter.) The current standard GAMS /LGO solver is set up to handle maximally 3000 variables and 2000 constraints (the latter in addition to the variable lower/upper bounds): hence, the k=65 instance would exceed the constraint limitation.

A possible GAMS model formulation of the circle packing problem (7) is displayed below. Notice (see the related in-code note) the easy scalability of the model by changing a single value. Let us also remark that instead of the nonlinear constraints stated in (7) their equivalent forms

$$\begin{array}{ll} 4r^2 \le \| c_i - c_j \|^2 & 1 \le i < j \le k \\ \| c_i \|^2 \le (1 - r)^2 & 1 \le i \le k \end{array}$$

are used below.

```
$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).
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;
```

This example also illustrates the compact and transparent nature of the GAMS model formulation.

In all program runs, first we use LGO, automatically followed by CONOPT: as discussed earlier, this combination could lead (and in some cases, does lead) to some result improvements. The results for k=5,10,15,…,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 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. "NFE" denotes the number of model function evaluations done by LGO. The "Runtimes" 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 these are approximate.

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

k	LGO	LGO+CONOPT	Best known result	NFE	Runtimes (sec)
5	0.37019190816	0.37019190816	0.37019190816	17383	0.125+0.016
10	0.26077981223	0.26077981216	0.26225892419	42678	0.907+0.094
15	0.22088519921	0.22088519954	0.22117253909	81849	3.844+0.109
20	0.19522401104	0.19522401102	0.19522401102	131203	10.344+0.016
25	0.17352441376	0.17352441371	0.17382766142	203233	24.672+0.031
30	0.15137590832	0.16080454102	0.16134910906	295572	50.500+0.125
35	0.14656680267	0.14866214852	0.14931677664	356579	85.031+0.109
40	0.13781012238	0.13857348501	0.14037360420	534093	165.391+0.188
45	0.13177203692	0.13177203691	0.13204959425	580471	221.079+0.109
50	0.12569303835	0.12569303835	0.12582548953	739523	376.484+0.078
55	0.11486959821	0.11871318638	0.12178632453	907129	530.906+0.625
60	0.10731691701	0.11469879969	0.11565748013	979374	669.016+1.234

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% to 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 (started from the LGO solution) requires a relatively very modest extra computational effort.
- LGO (or LGO+CONOPT) require a computational effort that apparently scales well with model size; it is also fairly reasonable considering today's computers. (The 60-circle model in its GAMS formulation has 121 decision variables and 1830 – mostly non-convex – constraints.)

To put these findings 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 minutes, 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 1000000 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 we wish to report results using the 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 findings is reported below. CONOPT, MINOS, and SNOPT – being high-quality local solvers – have difficulties in finding feasible solutions when used in a stand-alone mode. Specifically, CONOPT and SNOPT report model infeasibility for all cases considered here. MINOS finds a local solution for *k*=5, then for larger models it reports model 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 1000 seconds 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 seconds. 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 seconds. (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 seconds.

The illustrative test results summarized above indicate that LGO in its default operational mode (with or without CONOPT) produces good quality solutions with a reasonable effort, when solving models from an arguably non-trivial model-class. The results also demonstrate the need for using global solvers to handle such general nonlinear models. We are convinced that these findings are valid, in spite of unavoidable biases in model selection, solver settings, and benchmarking methodology. At the same time, we strongly believe that it remains impossible to draw far-reaching conclusions based on a limited set of examples. Our illustrative results certainly 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, Scherbina, Huyer and Vinkó (2005).

We will not go into further details on benchmarking here which is a substantial subject in itself. From the related literature we refer only to some recent work with GO relevance by Dolan and Moré (2002), Pintér (2002), Ali, Khompatraporn and Zabinsky (2005), Khompatraporn, Pintér and Zabinsky (2005). GAMS specific studies and numerical results are discussed e.g., by Bussieck, Drud, Meeraus and Pruessner (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.

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 substantial book chapters) that include also detailed test results, application examples and case studies, consult e.g. Grossmann (1996), Pintér

(1996a), Floudas (1999), Floudas et al. (1999), Papalambros and Wilde (2000), Pintér (2002), Tawarmalani and Sahinidis (2002), Zabinsky (2003), Neumaier (2004), Liberti and Maculan (2006), and Pintér (2006).

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, de Swart, and Pintér, 2001)
- Laser design (Isenor, Pintér, and Cada, 2003)
- Model calibration (Pintér, 2003b)
- A detailed LGO benchmarking study using several GAMS model libraries (Pintér, 2003c)
- Radiotherapy planning (Tervo, Kolmonen, Lyyra-Laitinen, Pintér, and Lahtinen, 2003)
- Design optimization in acoustic engineering (Pintér and Purcell, 2003)
- Various application examples and case studies developed for the Maple platform (Maplesoft, 2004; Pintér, Linder and Chin, 2005)
- 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, Kampas, and Pintér, 2005)
- Numerous further applications that are part of proprietary studies.

The listed books and examples show the broad applicability of global optimization technology (and of LGO, as an instance) across an increasing range of professional studies as well as in actual process design and product development.

Acknowledgements

I wish to express my thanks to Alexander Meeraus, Steven Dirkse, Armin Pruessner, and other colleagues at the GAMS Development Corporation, for their contributions to the GAMS /LGO implementation. Thanks are also due to my co-authors for cooperation as indicated by the references. Finally, thanks are due to two anonymous referees for their careful reading of the paper and for constructive suggestions.

References

Ali, M. M., Khompatraporn, Ch. and Zabinsky, Z.B. (2005) A numerical evaluation of several global optimization algorithms on selected benchmark test problems. *Journal of Global Optimization* 31, 635-672.

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

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

Benson, H.P. and Sun, E. (2000) LGO – Versatile tool for global optimization. *OR/MS Today* 27 (5), 52-55. 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., and Faltings, B. (2001) *Algorithms for Solving Nonlinear Constrained and Optimization Problems: The State of the Art.* COCONUT Project Report. 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. and McCormick, G.P. (1968) *Selected Applications of Nonlinear Programming.* Wiley, New York.

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

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

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

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

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

Cogan, B. (2003) How to get the best out of optimisation software. *Scientific Computing World* 71, 67-68. See also http://www.scientific-computing.com/scwjulaug03review_optimisation.html.

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

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

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

Dolan, E.D. and Moré, J.J. (2002) Benchmarking optimization software with performance profiles. *Mathematical Programming* 91, 201-213.

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

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

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

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

Fletcher, R. and Leyffer, S. (2002) Nonlinear programming without a penalty function. *Mathematical Programming* 91, 239-269.

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

Floudas, C.A., Pardalos, P.M., Adjiman, C.S., 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.

Fourer, R. (2005) *Nonlinear Programming Frequently Asked Questions.* Maintained by the Optimization Technology Center of Northwestern University and Argonne National Laboratory. See http://www-unix.mcs.anl.gov/otc/Guide/faq/nonlinear-programming-faq.html.

Frontline Systems (2005) *Premium Solver Platform – Field-Installable Solver Engines.* Frontline Systems, Inc., Incline Village, NV. See also http://www.solver.com.

GAMS Development Corporation (2005) *GAMS.* GAMS Development Corporation, Washington, DC. See http://www.gams.com.

GAMS Global World (2005) *GLOBAL Library: A Collection of Nonlinear Programming Models.* See http://www.gamsworld.org/global/globallib.htm.

GAMS Performance World (2005) *PAVER – Automated Performance Analysis & Visualization.* See http://www.gamsworld.org/performance/paver.

Gershenfeld, N. (1999) *The Nature of Mathematical Modeling.* Cambridge University Press, Cambridge.

Gill, P.E., Murray, W. and Saunders, M.A. (2002) SNOPT: An algorithm for large-scale constrained optimization. *SIAM Journal on Optimization* 12, 979-1006.

Gould, N.I.M., Orban, D., and Toint, Ph.L. (2002) 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.

Hansen, P.E. and Jørgensen, S.E., Eds. (1991) *Introduction to Environmental Management.* Elsevier, Amsterdam.

Henrion, D. (2006) A review of the Global Optimization Toolbox for Maple. *IEEE Computer Science Magazine* (To appear).

Hillier, F.J. and Lieberman, G.J. (2005) *Introduction to Operations Research*. (8th Edition.) McGraw-Hill, New York.

Horst, R. and Pardalos, P.M., Eds. (1995) *Handbook of Global Optimization, Volume 1*. Kluwer Academic Publishers, Dordrecht.

Horst, R. and Tuy, H. (1996) *Global Optimization: Deterministic Approaches* (3rd Edition). Springer-Verlag, Berlin.

Isenor, G., Pintér, J.D., and Cada, M. (2003) A global optimization approach to laser design. *Optimization and Engineering* 4, 177-196.

Kalwelagen, E. (2005) Interfacing GAMS with other applications. Tutorial and examples. GAMS Development Corporation, Washington, DC. See http://www.gams.com/~erwin, with numerous other useful notes.

Kampas, F.J. and Pintér, J.D. (2005) Global optimization in *Mathematica*: a comparative numerical study. *Proceedings of the 2005 Wolfram Technology Conference* (Champaign, IL). http://library.wolfram.com/infocenter/Conferences/5824/.

Kampas, F.J. and Pintér, J.D. (2006) Configuration analysis and design by using optimization tools in Mathematica. *The Mathematica Journal* 10 (1).

Khompatraporn, Ch., Pintér, J.D. and Zabinsky, Z.B. (2005) Comparative assessment of algorithms and software for global optimization. *Journal of Global Optimization* 31, 613-633.

Lasdon, L.S. and Smith, S. (1992) Solving large sparse nonlinear programs using GRG. *ORSA Journal on Computing* 4, 2-15.

Leyffer, S. and Nocedal, J., Eds. (2003) Large Scale Nonconvex Optimization. *SIAG/OPT Views and News* 14 (1).

Liberti, L., and Maculan, N., Eds. (2006) *Global Optimization: From Theory to Implementation.* Springer Science + Business Media, New York.

LINDO Systems (2005) *LINDO Solver Suite*. LINDO Systems, Inc., Chicago, IL. See also www.lindo.com.

Lopez, R.J. (2005) *Advanced Engineering Mathematics with Maple.* Electronic book edition published by Maplesoft, Inc., Waterloo, ON.

Maplesoft (2004) *Global Optimization Toolbox for Maple.* Maplesoft, Inc., Waterloo, ON. See http://www.maplesoft.com/products/toolboxes/globaloptimization.

Maplesoft (2005) *Maple.* Maplesoft, Inc., Waterloo, ON. See http://www.maplesoft.com.

(The) MathWorks (2004) *MATLAB*. The MathWorks, Inc., Natick, MA. See http:/www. mathworks.com.

Maximal Software (2005) *MPL*. Distributed by Maximal Software, Inc. Arlington, VA. See http://www.maximal-usa.com.

McCarl, B.A. (2004) *McCarl's GAMS User Guide*. See http://www.gams.com/dd/docs/bigdocs/ gams2002/. (Current edition dated 2004.)

Melissen, J.B.M. (1997) *Packing and Covering with Circles.* Ph.D. Dissertation, University of Utrecht.

Mittelmann, H.D. and Spellucci, P. (2005) *Decision Tree for Optimization Software.* See http:// plato.asu.edu/guide.html.

Mittelmann, H.D. and Pruessner, A. (2006) A server for automated performance analysis of benchmarking data. *Optimization Methods and Software* 21, 105-120.

Murray, J.D. (1983) *Mathematical Biology*. Springer-Verlag, Berlin.

Murtagh, B.A. and Saunders, M.A. (1995) MINOS 5.4 User's Guide. *Technical Report SOL 83-20R* (Revised edition.) Department of Operations Research, Stanford University, Stanford, CA.

Neumaier, A. (2004) Complete search in continuous optimization and constraint satisfaction. In: Iserles, A., Ed. *Acta Numerica 2004*, 271-369. Cambridge University Press.

Neumaier, A. (2005a) *Global Optimization*. See http://www.mat.univie.ac.at/~neum/ glopt.html.

Neumaier, A. (2005b) *COCONUT Project*. See http://www.mat.univie.ac.at/~neum/ glopt/coconut/ index.html.

Neumaier, A., Scherbina, O., Huyer, W. And Vinkó, T. (2005) A comparison of complete global optimization solvers. *Mathematical Programming* 103, 335-356.

Paragon Decision Technology (2005) *AIMMS*. Paragon Decision Technology BV, Haarlem, The Netherlands. See http://www.aimms.com.

Pardalos, P.M., and Resende, M.G.H., Eds. (2002) *Handbook of Applied Optimization.* Oxford University Press, Oxford.

Pardalos, P.M., and Romeijn, H.E., Eds. (2002) *Handbook of Global Optimization, Volume 2*. Kluwer Academic Publishers, Dordrecht.

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

Pintér, J.D. (1996b) Continuous global optimization software: A brief review. *Optima* 52, 1-8. See also http://plato.la.asu.edu/gom.html.

Pintér, J.D. (1997) LGO — A program system for continuous and Lipschitz optimization. In: Bomze, I.M., Csendes, T., Horst, R. and Pardalos, P.M., Eds. *Developments in Global Optimization*, pp. 183-197. Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (2001a) *Computational Global Optimization in Nonlinear Systems – An Interactive Tutorial*. Lionheart Publishing, Inc. Atlanta, GA. See also http://www.lionhrtpub.com/books/ globaloptimization.html.

Pintér, J.D. (2001b) Globally optimized spherical point arrangements: model variants and illustrative results. *Annals of Operations Research* 104, 213-230.

Pintér, J.D. (2002) Global optimization: software, test problems, and applications. In: Pardalos, P. M. and Romeijn, H. E., Eds., *Handbook of Global Optimization, Volume 2*, pp. 515-569. Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (2003a) GAMS /LGO (User Guide). http://www.gams.com/solvers/lgo.pdf.

Pintér, J.D. (2003b) Globally optimized calibration of nonlinear models: techniques, software, and applications. *Optimization Methods and Software* 18, 335-355.

Pintér, J.D. (2003c) GAMS /LGO nonlinear solver suite: key features, usage, and numerical performance. http://www.gams.com/solvers/GAMS_LGO_paper.pdf

Pintér, J.D. (2004) The Maple Global Optimization Toolbox. Downloadable from the Maplesoft product page http://www.maplesoft.com/products/toolboxes/globaloptimization.

Pintér, J.D. (2005a) LGO – A Model Development System for Continuous Global Optimization. User's Guide. (Current revised edition.) Pintér Consulting Services, Inc., Halifax, NS.

Pintér, J.D. (2005b) Nonlinear optimization in modeling environments: software implementations for compilers, spreadsheets, modeling languages, and integrated computing systems. In: Jeyakumar, V. and Rubinov, A.M., Eds., *Continuous Optimization: Current Trends and Applications*, pp. 147-173. Springer Science + Business Media, New York.

Pintér, J.D. (2005c) AIMMS/LGO solver engine: a brief introduction and user's guide. http://www.aimms.com/aimms/download/solvers/aimms_lgo_solver_engine_introduction_and_user_guide.pdf.

Pintér, J.D. (2005d) Nonlinear optimization with MPL /LGO: introduction and user's guide. Distributed by Maximal Software, Inc., Arlington, VA, USA. www.maximal-usa.com.

Pintér, J.D., Ed. (2006) *Global Optimization – Scientific and Engineering Case Studies.* Springer Science + Business Media, New York.

Pintér, J.D., Holmström, K., Göran, A.O., and Edvall, M.M. (2004) User's Guide for TOMLAB /LGO. TOMLAB Optimization AB, Västerås, Sweden.
See http://tomlab.biz/docs/ TOMLAB_LGO.pdf.

Pintér, J.D. and Kampas, F.J. (2003) MathOptimizer Professional – An Advanced Modeling and Optimization System for Mathematica, Using the LGO Solver Engine. User's Guide. Pintér Consulting Services, Inc., Halifax, NS.

Pintér, J.D. and Kampas, F.J. (2005) Nonlinear optimization in Mathematica with MathOptimizer Professional. *Mathematica in Education and Research* 10, 1-18.

Pintér, J.D. and Kampas, F.J. (2006) MathOptimizer Professional: key features and illustrative applications. In: Liberti and Maculan, Eds., *Global Optimization: From Theory to Implementation,* pp. 263-280. Springer Science + Business Media, New York.

Pintér, J.D., Linder, D. and Chin, P. (2005) Global Optimization Toolbox for Maple: An introduction with illustrative applications. *Optimization Methods and Software* (To appear.)

Powell, M.J.D. (2002) UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming* 92, 555-582.

Schittkowski, K. (2002) *Numerical Data Fitting in Dynamical Systems*. Kluwer Academic Publishers, Dordrecht.

Specht, E. (2005) www.packomania.com.

Steeb, W-H. (2005) *The Nonlinear Workbook* (3rd Edition). World Scientific, Singapore.

Stojanovic, S. (2003) *Computational Financial Mathematics Using Mathematica.* Birkhäuser, Boston.

Stortelder, W.J.H., de Swart, J.J.B. and Pintér, J.D. (2001) Finding elliptic Fekete points sets: two numerical solution approaches. *Journal of Computational and Applied Mathematics* 130, 205-216.

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

Tawarmalani, M. and Sahinidis, N.V. (2002) Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Kluwer Academic Publishers, Dordrecht.

Tervo, J., Kolmonen, P., Lyyra-Laitinen, T., Pintér, J.D., and Lahtinen, T. (2003) An optimization-based approach to the multiple static delivery technique in radiation therapy. Annals of Operations Research 119, 205-227.

TOMLAB Optimization (2005) *TOMLAB.* TOMLAB Optimization AB, Västerås, Sweden. http://www.tomlab.biz.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J. and Marti, R. (2006) A multistart scatter search heuristic for smooth NLP and MINLP problems, *INFORMS Journal on Computing.* (To appear.) See http://www.utexas.edu/courses/lasdon/ijocmultistart5.htm.

Vanderbei, R.J. (1999) LOQO User's manual – version 3.10. *Optimization Methods and Software* 12, 485-514.

Waltz, R. and Nocedal, J. (2003) KNITRO 2.0 User's Manual. *Technical Report OTC 2003/05*, Optimization Technology Center, Northwestern University, Evanston, IL. See also Ziena Optimization, Inc. http://www.ziena.com/knitro/manual.htm.

Wass, J.A. (2006) Global Optimization with Maple. *Scientific Computing & Instrumentation* (To appear).

Wolfram, S. (2003) *The Mathematica Book* (4th Edition). Wolfram Media and Cambridge University Press.

Wolfram Research (2005) *Mathematica.* See http://www.wolfram.com/.

Wright, M.H. (1996) Direct search methods: once scorned, now respectable. In: *Numerical Analysis 1995: Proceedings of the 1995 Biennial Conference on Numerical Analysis* (D. F. Griffiths and G. A. Watson. Eds.), pp. 191-208. Addison Wesley Longman Ltd, April 1996.