GAMS/LGO Nonlinear Solver Suite: Key Features, Usage, and Numerical Performance

JÁNOS D. PINTÉR

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

Submitted for publication: November 2003

Abstract. The LGO solver system integrates a suite of efficient global and local scope optimization strategies. LGO can handle complex nonlinear models under 'minimal' (continuity) assumptions. The LGO implementation linked to the GAMS modeling environment has several new features, and improved overall functionality. In this article we review the solver options and the usage of GAMS/LGO. We also present reproducible numerical test results, to illustrate the performance of the different LGO solver algorithms on a widely used global optimization test model set available in GAMS.

Key words: Nonlinear (convex and global) optimization; GAMS modeling system; LGO solver suite; GAMS/LGO usage and performance analysis.

1 Introduction

Nonlinear optimization has been of interest, arguably, at least since the beginnings of mathematical programming, and the subject attracts increasing attention. Theoretical advances and progress in computer hardware/software technology enabled the development of sophisticated nonlinear optimization software implementations. Among the currently available professional level software products, one can mention, e.g. LANCELOT (Conn, Gould, and Toint, 1992) that implements an augmented Lagrangian based solution approach. Sequential quadratic programming methods are implemented – and/or are in the state of ongoing development – 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 notable solver class is based on the generalized reduced gradient approach, implemented in MINOS (Murtagh and Saunders, 1995; this system includes also other solvers), LSGRG (consult e.g., Lasdon and Smith, 1992; or Edgar, Himmelblau, and Lasdon, 2001), and CONOPT (Drud, 1996). Interior point methodology is used in LOQO (Vanderbei, 2000) and in KNITRO (Waltz and Nocedal, 2003). Finally – but, obviously, without aiming at completeness – one can mention also gradient-free solver implementations such as UOBYQA (Powell, 2002); consult also the review of 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 local solvers only). The COCONUT Project Report (Bliek et al., 2001) authored by 11 researchers provides a more detailed and comprehensive review; see especially the topical Chapters 2 and 4.

The methods and solvers listed above are essentially of *local* scope, and hence – at best – can only guarantee local optimality. (This is certainly not a dismissive comment, just a valid observation.) In lack of valuable *a priori* information about a 'suitable' starting point, in solving general nonlinear models even the best local methods may (will) encounter difficulties: that is, such methods may find only a local solution, or 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, and/or is unable to provide a good starting point – that will lead to the 'absolutely best' solution – then the usage of a global scope search strategy becomes necessary.

Dictated by theoretical interest and practical demands, the field of global optimization (GO) has been gaining increasing attention, and in recent years it reached a certain level of maturity. The number of textbooks focused on GO is well over one hundred worldwide, as of 2003. For illustration, the broad scope *Handbook of Global Optimization* volumes – edited respectively by Horst and Pardalos (1995), and Pardalos and Romeijn (2002) – are mentioned.

The theoretical developments in GO have been rapidly followed by algorithm implementations. While the majority of solvers reviewed in our earlier survey (Pintér, 1996b) has been perhaps more 'academic' than 'professional', these days a number of developers and software companies offer professionally developed and maintained GO solvers. To illustrate this point, it suffices to visit e.g. the web sites of Frontline Systems (2003), the GAMS Development Corporation (2003), LINDO Systems (2003), or Wolfram Research (2003), and check out the corresponding global solver information. (Examples of pertinent LGO solver implementations will be mentioned later on.)

One should mention here also at least a few highly informative, non-commercial web sites that discuss GO models, algorithms, and technology. For instance, the outstanding web site of Neumaier (2003a) is devoted to GO in its entirety; Fourer (2003) and Mittelmann and Spellucci (2003) also provide valuable discussions of nonlinear programming methods and software, with numerous further links and pointers.

In this article, we introduce the recently completed GAMS/LGO solver implementation for global and convex 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 usage within the framework of the GAMS modeling environment. We also present *reproducible* numerical results that illustrate the performance of the LGO component algorithms on a widely used standard nonlinear optimization test model set using performance profiles (Dolan and Moré, 2002; Mittelmann and Pruessner, 2003). The concluding notes are followed by a fairly extensive list of references that provide further details.

2 Model Statement and Specifications

We shall consider the general continuous global optimization (CGO) model stated as

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

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

• $x \in \boldsymbol{R}^n$	real n-vector of decision variables,
• $f : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^1$	continuous (scalar-valued) objective function,
• $D \subset \boldsymbol{R}^n$	non-empty set of feasible solutions (a proper subset of $\boldsymbol{R}^n$).

The feasible set D is defined by

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

Note that the inequality constraints in (1) could have arbitrary signs, including also equalities, and that explicit bounds on the constraint function values can also be imposed. Such – formally more general and/or specialized – models are directly reducible to the form (1).

By the classical theorem of Weierstrass, the above listed basic analytical conditions guarantee that the optimal solution set of the CGO model is non-empty. ((Various generalizations would allow mild relaxations of the analytical conditions listed above, and the existence result would still hold.) At the same time, without further specific structural assumptions, (1) can be a very difficult *numerical* problem. For instance, the feasible set D could well be disconnected, and some of its components could be non-convex; furthermore, the objective function f itself could be highly multiextremal. Since under such circumstances, there could be an unknown number of global and local solutions to (1), there is no 'straightforward' (generally applicable, algebraic) characterization of global optimality. By contrast, in 'traditional' (convex) nonlinear programming, most of the exact algorithms aim at solving the Karush-Kuhn-Tucker system of (necessary) optimality conditions: the corresponding system of equations and inequalities in the present framework becomes another GO problem, often at least as complex as the original model. Neumaier (2003c) presents an interesting discussion that also indicates that the number of KKT cases to check for global optimality may grow exponentially as the model size increases.

To illustrate the potential difficulty of GO models, a relatively simple, two-dimensional box-constrained model instance is displayed in Figure 1. In spite of its low dimensionality – even when visualized – this model has no apparent structure that could be easily exploited by an algorithmic search procedure. To indicate some practical motivation, let us remark passing by that the GO model shown in the figure corresponds to the least-square error function in solving the pair of equations

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

This system has multiple numerical solutions in the variable range $x \in [-2,3]$, $y \in [-2.5,1.5]$: one of these solutions is $x^* \sim 0.793181$, $y^* \sim 0.138698$: consult Pintér and Kampas (2003) for further details.

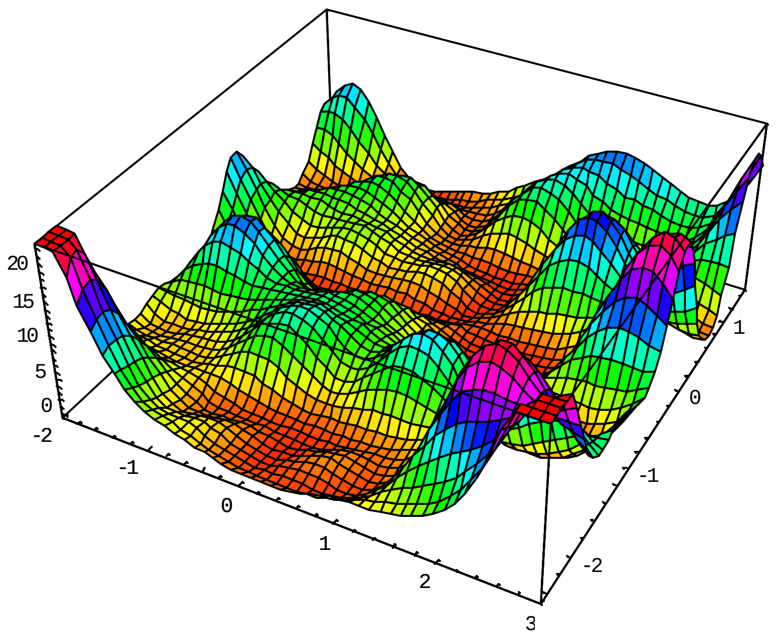

Figure 1.
An illustrative global optimization model: error function related to a pair of equations.

We shall denote by X^* the set of global solutions to the general GO model (1). In many practical cases X^* consists of a single point x^*; however, it is easy to show model instances in which X^* is finite (with cardinality greater than one), countable, non-countable, or it can be even a subset of D which has positive volume. For the sake of algorithmic tractability, we typically assume that X^* is at most countable.

Theoretically, one would like to find *exactly* all global solutions to instances of model (1) by applying a suitable search algorithm. However, this often proves to be impossible: even convex nonlinear programming methods apply, as a rule, an infinite iterative procedure. A more 'pragmatic' goal is to find suitable approximations of points from the set of global optima, and the corresponding optimum value $f^*=f(x^*)$ for $x^*\in X^*$. In practice, this goal needs to be attained on the basis of a finite number of search steps, that is, model function evaluations at algorithmically selected search points. Note that in global search methods there is no essential need to evaluate gradient, Hessian, or higher order *local* information, although such information can be put to good use in an optional local search component.

Formally, one will accept approximate numerical solutions x that satisfy the relation

$$(2)\qquad r(x,X^*) := \min_{x^*\in X^*} r(x,x^*) \le g.$$

In (2) r is a given metric – typically defined by the norm introduced in $\boldsymbol{R}^n$ – and $g>0$ is a fixed tolerance parameter. Similarly, one can accept an approximate solution $x\in D$, if

$$(3)\qquad f(x) \le f^*+e.$$

Here $e>0$ is another tolerance (accuracy) parameter.

To ensure numerical solvability in the sense of (3) – on the basis of a finite sample point sequence from D – we often assume the Lipschitz-continuity of f. (Recall that a function f is Lipschitz-continuous in the set D, if the relation

(4) $$|f(x_1)-f(x_2)|\leq L||x_1-x_2||$$

is valid for all pairs x_1,x_2 from D. The value L is the corresponding Lipschitz-constant.)

Note that the *smallest* $L=L(D,f)$ is typically unknown, although a suitable value is often postulated or estimated in numerical practice: for related discussions, consult e.g. Pintér (1996a), or Strongin and Sergeyev (2000). Similar conditions can be postulated with respect to the component constraint functions in g:

(5) $$|g_j(x_1) - g_j(x_2)|\leq L_j||x_1 - x_2|| \qquad L_j=L(D,g_j) \quad j=1,...,m.$$

Model (1) with the Lipschitz structure (4)-(5) is still very general, and it includes many (in fact, most) more specific GO problem types occurring in practice. As a consequence of this generality, it incorporates also very difficult problem instances that pose a tremendous challenge in any computational environment, today or tomorrow.

For given model instances, the corresponding ‘most suitable’ solution approach may vary to a considerable extent. On one hand, a more general (‘universal’) optimization strategy will work for broad model classes, although its efficiency might be lower for certain, more special GO problem instances. On the other hand, highly tailored algorithms often will not work for GO model classes outside of their intended scope.

3 LGO Solver Suite: Algorithm Components and Key Features

The LGO solver suite (abbreviating a *Lipschitz(-Continuous) Global Optimizer*) has been specifically designed with models in mind that may not, or do not, have an easily identifiable, special structure. (The model depicted in Figure 1 is certainly within the scope of LGO.) This design principle and the corresponding choice of component algorithms makes LGO applicable even to ‘black box’ models. The latter category specifically includes confidential models, and other models that incorporate computationally evaluated or other complicated functions and numerical procedures. At the same time, structured models – for instance, models with convex, concave, or indefinite quadratic objectives over convex domains – also belong to the scope of LGO.

The professional (commercial) LGO algorithm system has been developed, maintained, and gradually extended for more than a decade. LGO is documented in details elsewhere: consult, for instance, Pintér (1996a, 2001a, 2002a, 2003a) and further references therein. Therefore, only a concise review of its features is included here.

The overall solution approach in LGO is based on the seamless combination of rigorous – i.e., theoretically convergent – *global* and efficient *local* search strategies, to support a broad range of possible uses. The solver suite allows the flexible usage of the component solvers: one can select fully automatic (global and/or local search based) and interactive operational modes.

LGO currently offers the following *global search algorithms* (as mutually exclusive, optional choices):

- Branch-and-bound (adaptive partition and sampling) based global search (BB)
- Global adaptive random search (GARS)
- Adaptive random multistart based search (MS)

In addition, LGO includes the following *local solver strategies*:

- Heuristic (continuous) scatter search method (HSS)
- Bound-constrained local search, based on the use of an exact penalty function (EPM)
- Constrained local search, based on a generalized reduced gradient approach (GRG).

As stated earlier, the development of the LGO solver suite has been targeted towards the general class of GO models defined by (1): therefore *global* convergence needs to be assured, under the stated 'minimal' structural assumptions. The adaptive partition and sampling (BB) solver mode is globally convergent for Lipschitz models. (This assumes that one is able to provide valid overestimates of the Lipschitz constant(s) for the model functions: typically, such a condition is only approximated in algorithm implementations, including LGO's BB solver mode.) Furthermore, both GARS and MS are – theoretically – globally convergent, with probability 1. The theoretical background of LGO is discussed in detail by Pintér (1996a), with numerous further references.

In numerical practice, global solvers are often combined with (followed by) local search methods. The local solvers embedded in LGO implement dense nonlinear algorithms (again, without postulating any specific structure): these methods *per se* enable the solution of convex or mildly non-convex models. The EPM and GRG solvers are based on classical nonlinear optimization techniques discussed e.g. by Edgar, Himmelblau, and Lasdon (2001). The HSS heuristics is based our own algorithm development work and – similarly to the other search methods outlined above – includes proprietary elements and features.

Note that all LGO component solvers are *direct* (gradient-free): the global solvers genuinely need only model function values, and the local solvers apply finite difference gradient approximations. This approach corresponds to the broad applicability objectives of LGO stated earlier.

LGO has been implemented, tested and supported across several programming languages and environments. These currently include native Fortran (Lahey Fortran 77/90, Lahey-Fujitsu Fortran 95, Digital/Compaq Visual Fortran 95, g77, and some other) implementations, with direct connectivity e.g. to Borland C/C++, Microsoft Visual C/C++, gcc, lcc-win32 and to Fortran 77/90/95 models.

There is also a range of LGO (and related) implementations available for prominent modeling and scientific computing environments. In addition to the GAMS version discussed here, there is an Excel LGO solver engine implementation (a joint development with Frontline Systems, 2003); and a *Mathematica* (Wolfram, 1999) external solver implementation (Pintér and Kampas, 2003), in addition to the native *Mathematica* implementation described in (Pintér, 2002b). Further development work is in progress. Peer reviews discussing several of these solver system implementations are available: see e.g. Benson and Sun (2000), and Cogan (2003).

Concluding the brief overview presented in this section, let us remark that LGO has been successfully applied to solve a broad range of models and problems, both in educational/research and professional/commercial contexts. The standard LGO shipment version can be configured to handle models formulated with thousands of variables and constrains. Application areas include engineering design, chemical and process industries, econometrics and finance, medical research, biotechnology, and scientific modeling.

Several recent applications and case studies are:

- Laser equipment design (Isenor, Pintér, and Cada, 2003)
- Object configuration analysis and design (Kampas and Pintér, 2003)
- Potential energy models (Pintér, 2001b; Stortelder, de Swart, and Pintér, 2001)
- Radiotherapy design (Tervo, Kolmonen, Lyyra-Laitinen, Pintér, and Lahtinen, 2003)
- Model calibration (Pintér, 2003b)
- Acoustic equipment design (Pintér, 2002b)

Various other current and potential GO applications are reviewed by (Pintér, 2002a), with numerous references.

4 GAMS/LGO Solver Implementation

The General Algebraic Modeling System (GAMS) is a high-level model development environment that supports the analysis and solution of a broad range of mathematical programming problems. GAMS includes a model (language) compiler and intelligent pre-processing facilities, with links to an integrated collection of high-performance solvers. The latter serve to solve both general (linear, convex and global nonlinear, pure and mixed integer, stochastic) and more specialized (complementarity, equilibrium, and constrained nonlinear systems) models.

GAMS is capable of handling advanced, scalable modeling applications: it allows the user to build prototypes, as well as large, maintainable models that can be adapted flexibly and quickly as needed. Models can be developed, solved and documented simultaneously, maintaining the same GAMS model file.

The GAMS language is similar to commonly used procedural programming languages (like C, Fortran, Pascal, etc.), and hence it is especially easy to apply for users familiar with such languages. GAMS also supports interfacing models to other development environments, including MS Excel and Access, Matlab, and Gnuplot; furthermore, it can be embedded in various application environments (C/C++, Delphi, Java, Visual Basic, WebSphere, and so on).

GAMS has been commercially available since 1987 and it has significant world-wide user base. The original *GAMS User Guide* (Brooke, Kendrick, and Meeraus, 1988) has been significantly extended and enhanced by accompanying documentation: consult the web site www.gams.com for current information that includes e.g. the extensive, hyperlink-enabled GAMS documentation by McCarl (2002).

The GAMS Model Library is a dynamic collection of hundreds of models, from a variety of application areas: 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 a collection of more ‘academic’ test problems).

A further valuable service is GAMS World: this web site aims at bridging the gap between academia and practicioners. The site includes (an additional set of) hundreds of documented models and performance analysis tools: some of these will be discussed and used later on.

The modeling and solver features and the full documentation of GAMS are available through an integrated development environment (GAMS IDE) on MS Windows platforms; command-line GAMS usage is also supported when using Windows, as well as Linux and various Unix platforms.

Recently, LGO has been added to the portfolio of solvers available in the GAMS modeling environment. The web link http://www.gams.com/solvers/lgo.pdf provides a concise GAMS/LGO user documentation: portions of that description will be used here, with added notes.

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

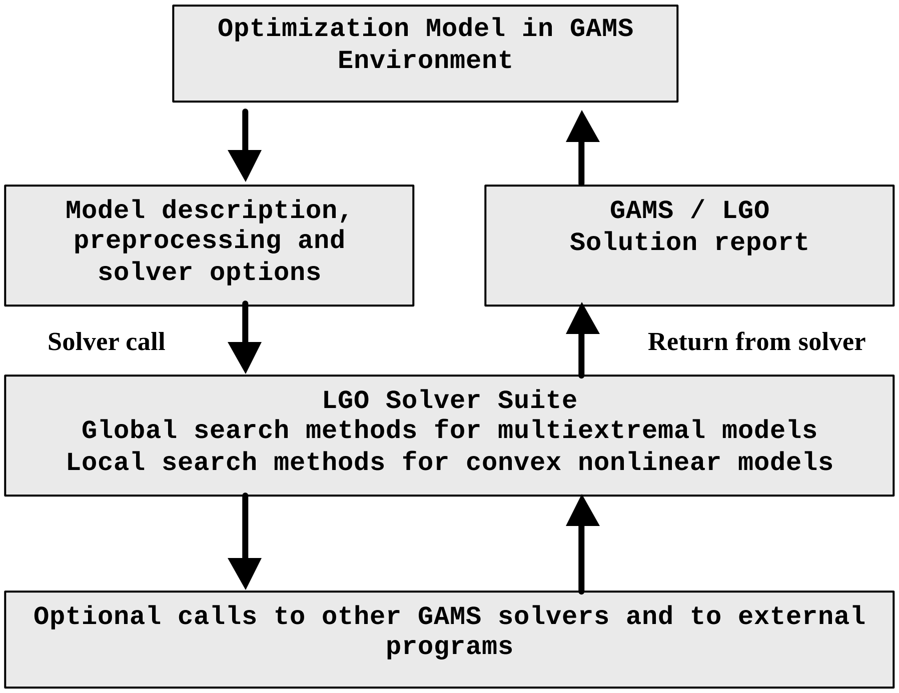
Optimization Model in GAMS Environment
Model description, preprocessing and solver options
GAMS / LGO Solution report
Solver call
Return from solver
LGO Solver Suite
Global search methods for multiextremal models
Local search methods for convex nonlinear models
Optional calls to other GAMS solvers and to external programs

Figure 2. GAMS/LGO operations: basic structure.

As this figure indicates, the steps of model development, preprocessing, solution by LGO, optional other solver calls, and reporting are seamlessly integrated. In fact, our numerical experiments clearly show very little (runtime) overhead associated with the operations of GAMS, when compared to a ‘silent mode’ core LGO implementation that includes only the solver operations and terse, text-file based I/O communication.

For completeness, let us remark at this point that two other global solvers – BARON and OQNLP – have also become available recently for the GAMS platform. BARON (consult Tawarmalani and Sahinidis, 2002) is based on a successive convexification approach by constructing enclosure functions and related bound estimates that drive the search. In its solution procedure, BARON relies also on other solvers (MINOS, CPLEX, and possibly others) that are modularly available under GAMS.

OQNLP – similarly to LGO – is a stand-alone solver system: it uses a multi-start global search approach based on the OptTek search engine, in combination with the well-received local solver LSGRG: consult e.g., Ugray, Lasdon, Plummer, Glover, Kelly, and Marti (2002).

Both BARON and OQNLP are documented also at the GAMS web site.

5 Using GAMS/LGO: An Illustrative Example

Since we assume that not all readers are familiar with the workings of GAMS, we shall present a small-scale illustrative example below. The model is taken from the GAMS Model Library (contributed by the author), slightly edited for the present purposes. All GAMS language elements are denoted by boldface letters; added comments are provided between the rows denoted by **`$ontext`** and **`$offtext`**, and in rows started by `*`.

Example

```
$title Global Optimization Test Example

$ontext

This example is taken from the LGO User Guide. It is a 5-variable, 3-constraint
test problem; the global solution is the zero vector, the optimum value is 0.

Source: Janos D. Pinter (2003) LGO Users Guide. Pinter Consulting Services,
Halifax, Canada.

Note that changing the scalar parameter scaleaux (see below) to larger values
makes the model more difficult, by enhancing its multiextremality 'ridges'.

$offtext

* Define optimization model

scalar scaleaux / 2 /;

variables obj, aux1, aux1a, aux2, aux2a, aux, fct, x1, x2, x3, x4, x5;

equations defobj, deffct, defaux, defaux1, defaux1a, defaux2, defaux2a,
con1, con2, con3;

defobj..   obj   =e= fct + scaleaux*aux;

deffct..   fct   =e= sqr(x1) + sqr(x2) + sqr(x3) + sqr(x4) + sqr(x5);
```

```
defaux..   aux   =e= aux1a+aux2a;

defaux1..  aux1  =e= sqr(sqr(x1)-x2) + sqr(x3) + 2*sqr(x4) + sqr(x5-x2);

defaux1a.. aux1a =e= abs(sin(4*mod(aux1,pi)));

defaux2..  aux2  =e= sqr(x1+x2-x3+x4-x5) + 2*sqr(-x1+x2+x3-x4+x5);

defaux2a.. aux2a =e= abs(sin(3*mod(aux2,pi)));

con1..     x1 + 3*sqr(x2) + sqr(x3) - 2*sqr(x4) + sqr(x5) =e= 0 ;

con2..     x1 + 4*x2 - x3 + x4 - 3*x5 =e= 0;

con3..     sqr(x1) - sqr(x3) + 2*sqr(x2) - sqr(x4) - sqr(x5) =e= 0 ;

model m / all /;

* Define bounds and (optional) nominal values
* (see the corresponding lo, up and l notation)

x1.lo = -10; x1.l = 2; x1.up = 5;

x2.lo = -10; x2.l = 2; x2.up = 5;

x3.lo = -10; x3.l = 2; x3.up = 5;

x4.lo = -10; x4.l = 2; x4.up = 5;

x5.lo = -10; x5.l = 2; x5.up = 5;

* Invoke the LGO solver option

option dnlp=lgo;

solve m minimizing obj using dnlp;
```

The summary results – directly cited from the corresponding GAMS log output file – are displayed below:

```
LGO 1.0       Sep  3, 2003 WIN.LG.NA 21.2 001.000.000.VIS

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

    7 defined, 0 fixed, 0 free
    3 LGO equations and 5 LGO variables

   Iter      Objective     SumInf   MaxInf     Seconds  Errors
  28380   7.634679E-18   0.00E+00  0.0E+00       0.281

--- LGO Exit: Normal completion - Global solution
    0.281 LGO Secs (0.201 Eval Secs, 0.007 ms/eval)
```

As shown above, the objective function value returned equals `7.634679E-18; and` the solution is feasible (see the `MaxInf` column of `0.0E+00` that serves to show the maximal infeasiblity). Furthermore, the runtime is `0.281` LGO seconds. (The above report was generated on a P4 1.6 GHz processor based desktop machine, running under the Windows XP Professional operating system; GAMS/LGO release 21.2 is used.)

Note that GAMS/LGO can also be used in conjunction with other available solvers. For instance, an LGO run may be directly followed by a call to the local NLP solver CONOPT (Drud, 1996) from the best solution point found. Such 'polishing' steps may be especially useful in difficult models, since model re-scaling and restart may (in general) improve the precision of the solution found.

6 LGO Solver Options

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 and 2. Options are divided by category as 1) general options and 2) limits and tolerances. The tables (cited from the documentation) display the option lists, with added brief explanation and the default settings.

Table 1. GAMS/LGO General Options

Option	Description	Default
opmode	Specifies the search mode used. 0 Local search 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
tlimit	Time limit in seconds. This is equivalent to the GAMS option reslim. If specified, this overrides the GAMS reslim option.	1000
log_time	Iteration log time interval in seconds. Log output occurs every log_time seconds.	0.5
log_iter	Iteration log time interval. Log output occurs every log_iter iterations.	10
log_err	Iteration log error output. Error reported (if applicable) 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 cleanup phase using CONOPT. CONOPT terminates after at most CallConopt seconds. The cleanup phase determines duals for final solution point.	5
help	Prints out all available GAMS/LGO solver options in the log and listing files.	

Table 2. GAMS/LGO Limits and Tolerances

Option	Description	Default
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. Algorithm phase terminates upon reaching this limit. (See comment above.)	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 not greater than acc_tr.	-1.0E10
fct_trg	Target objective function value (partial stopping criterion in internal local search phase).	-1.0E10
fi_tol	Local search (merit function improvement) tolerance.	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

Although there is clearly no ‘universal recipe’ to provide options and switches to a nonlinear (let alone global) solver, in our experience the default settings shown above are typically adequate to solve small to moderate dimensional standard test problems known from the GO literature. In the next section, we will illustrate this point, by *automatically* solving a set of such models formulated in the GAMS modeling language.

Let us also note at this point that the developers of GAMS made available a 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: see http://www.gams.com/dd/docs/bigdocs/gams2002/convert.pdf for details.

7 Illustrative Numerical Results

The purpose of our numerical experiments summarized below is twofold. The first is to illustrate the numerical performance of the available LGO solver options in an objective, reproducible setting; the second is to demonstrate that in nonlinear models the quality of solution found by a global solver is often improved, when compared to results attained by ‘traditional’ solvers. Again, this is not a dismissive comment regarding convex optimization methodology which, in fact, is an integral part of LGO itself: it merely indicates the limitations of local scope solution approaches.

In our numerical experiments described here, we have used LGO to solve a set of GAMS models based on the *Handbook of Test Problems in Local and Global Optimization* by Floudas et al. (1999). For brevity, we shall refer to the model collection studied as HTPLGO. The set of models considered is available from GLOBALLib (GAMS Global World, 2003). GLOBALLib is a collection of nonlinear models that provides GO solver developers with a large and varied set of theoretical and practical test models. Since the given model formulations often include non-default starting values which may not be available in practical GO – and they may influence results of local solvers significantly – such initial values were not considered in our experiments. (LGO *per se* does not need – although it can use – initial values; if such information is absent, then a random point or the centre of the embedding box $[l,u]$ $l \le x \le u$, etc. can be used as a starting nominal solution.)

We ran GAMS/LGO with all algorithmic options available; recall these as follows:

- operational mode (for brevity we shall use opmode) 0: local search from a given nominal solution, without a preceding global search mode (LS)
- opmode 1: global branch-and-bound search and local search (BB+LS)
- opmode 2: global adaptive random search and local search (GARS+LS)
- opmode 3: global multi-start random search and local search (MS+LS)

Furthermore, we compared LGO to three state-of-art local NLP solvers: CONOPT3 (version 3.12A-015-049), MINOS 5.51, and SNOPT (version 6.2-1(1)). In our analysis, we made use of the PAVER Server (Mittelmann and Pruessner, 2003; GAMS Performance World, 2003) for a fully automated performance assessment.

We emphasize that all of our numerical experiments are *completely reproducible* (assuming, of course, access to the solvers used). The necessary testing instructions, all test models and related scripts can be found at http://www.gamsworld.org/global/reproduce/.

Design of the Numerical Experiments

The selection process of GO models for benchmarking experiments is unavoidably somewhat subjective, and it is not possible to find test model sets that are 'completely unbiased'. For recent discussions regarding GO test methodology and criteria, see e.g. Pintér (2002a), Ali, Khompatraporn, and Zabinsky (2003), Khompatraporn, Pintér, and Zabinsky (2003), Pruessner and Mittelmann (2003), with detailed lists of further topical references. The web sites maintained by Mittelmann and Spelucci (2003), Neumaier (2003a, b) also provide useful related information and extensive collections of test models.

In the numerical tests reported here, we have chosen a widely used test set that covers a variety of optimization applications: consult Floudas et al. (1999) for short model descriptions. As mentioned, the actual model implementations used come from GLOBALLib; 12 models were not included, since these resulted in (division by zero) runtime errors without given starting level values. (The reason for this is that, within GAMS, the default initial variable value is 0, if no such value is specified). Two further models from the HTPLGO set were removed in our experiments, because the number of variables was quite large (greater than the 1000 variables). Thus, the entire test set used consists of 117 models. We omit here detailed model size information, as this is readily available from the GLOBALLib website: let us mention only that the test models included have up to 142 variables, 109 constraints, 729 non-zero and 567 nonlinear-non-zero model terms. Since GO models can be difficult even in a few variables, the model set can be considered – and indeed proves to be – a non-trivial challenge for nonlinear solvers.

The experiments summarized below were run on a computer (owned by the GAMS Development Corporation) with a dual 1GHz processor, and with 1GB of physical memory. The Linux operating system and GAMS/LGO from the GAMS distribution version 21.2 has been used. We specified time limits of 60 seconds per solve with unlimited iteration limits. We also verified that our experiments are completely reproducible, by repeating the test runs several times. In particular, we verified that all solver and model return statuses were completely identical, and that solver resource (CPU) times varied by no more than 10%. (Obviously, the latter could be due to circumstantial machine operations.)

Note here that GAMS can automatically capture model size and solver summary report (model and solver status codes, number of iterations, objective value, runtime) information in *trace files.* These are comma delimited text files with a specified format; consult Bussieck, Drud, Meeraus, and Pruessner (2003). The trace files capture all benchmarking data required for our current experiments. We generated a single trace file for each LGO solver method (that is for opmode=0, 1, 2, and 3), as well as for the three local solvers mentioned, resulting in a total of 7 different trace files.

In order to summarize the numerical results, we use *performance profiles* introduced recently by Dolan and Moré (2002). Performance profiles are cumulative distribution functions over a given performance metric and provide condensed information in terms of robustness, efficiency and solution quality information in a single plot. In the next subsection, we will give a brief overview of performance profiles: most of this description is directly based on the work of Dolan and Moré cited above. Furthermore, we discuss how performance profiles can be used to include quality of solution information: this point is based on (Mittelmann and Pruessner, 2003).

Performance Profiles

Let us suppose that we wish to analyze the performance of solvers $s=1,\ldots,n_s$: to this end, each of these is run to solve a given set P of problems $p=1,\ldots,n_p$. Define $t_{p,s}$ as the solver resource (computational time) spent by solver s on problem p, assuming here that solver s indeed solves problem p according to a preset numerical accuracy criterion. Then we can define the corresponding performance ratio as

(6)
$$r_{p,s} = \frac{t_{p,s}}{\min\left\{t_{p,s} : 1 \le s \le n_s\right\}}$$

In (6) it is tacitly assumed that at least one of the solvers can handle problem p (otherwise it should perhaps be excluded from consideration, in the given context). For a solver s that can not handle p, one may choose a parameter $r_M \geq max\ r_{p,s}$, in which the maximum is taken for all successful solvers. (The choice of a value $r_M > r_{p,s}$ arguably makes more sense, but this would introduce some added subjectivity.) Next, for $1 \le t \le \infty$, consider the empirical probability distribution

(7)
$$p_s(t) = \frac{1}{n_p} C\left[\left\{p \in P : r_{p,s} \le t\right\}\right]$$

Here $C[\{.\}]$ denotes the cardinality (size) of the set $\{.\}$.

Note that by (7) $p_s(t)$:$[1,\infty]$®$[0,1]$ is the probability that a given performance ratio $r_{p,s}$ is within t of the best ratio. The function in (7) is called a *performance profile*: it is the cumulative distribution function of the performance ratio defined by (6). Let us remark that the performance ratio is not necessarily confined to the one defined above; rather, it can be chosen based on the purpose of the analysis.

As discussed by Dolan and Moré, the performance profile gives useful comparative summary information about both solver robustness and efficiency. If we are only interested in the 'winning' solver efficiency, then one can examine profile values $p_s(t)$ for $t=1$ of different solvers s. For each solver, the corresponding value $p_s(1)$ specifies the probability that the solver will 'win' over all other solvers considered. For the profile defined above, the 'winning time' is determined by the solver that finds the optimal solution in the least amount of time. Obviously, in the GO context distinction needs to be made between local or global solutions, and the corresponding global solution time should be used when calculating (6) and (7).

It is possible to choose different definitions of 'winning' based on different performance ratios. If we are only interested in the eventual probability of success of solver s for the problem set P, then we may like to estimate

(8)
$$\lim_{t\to\infty} p_s(t)$$

since for very large values of t we are looking at the estimated 'probability of success' of a solver, given 'unlimited' resource time. Again, in the GO context, 'success' is defined by finding the global solution: more precisely, by providing a sufficiently close estimate of at least one element of the non-empty global solution set X^*.

Solution Quality Considerations

The performance ratio defined in (6) can be directly modified, if the user is also interested in the quality of the solution returned by a solver. This aspect is of particular interest for multi-extremal (including discrete and mixed integer) models. Recall that in the performance ratio definition we can assign a suitably large ratio r_M to those problem/solver pairs p,s in which solver s fails to solve problem p. If we are interested also in the quality of solution, then we can modify the definition accordingly.

Let $o_{p,s}$ denotes the solution value found by solver s for problem p and b_p is the best solution value found when applying all solvers to problem p. Then we can define a new performance ratio as

(9)
$$r_{p,s} = \left\{ \begin{array}{ll} \dfrac{t_{p,s}}{\min\left\{t_{p,s} : 1 \le s \le n_s\right\}} & if \quad \left|\dfrac{o_{p,s}-b_p}{b_p}\right| \le d \\ \\ r_M & if \quad \left|\dfrac{o_{p,s}-b_p}{b_p}\right| > d \end{array} \right\}$$

Here $d>0$ is a user-defined relative objective value difference threshold and r_M is again an upper bound on $r_{p,s}$ over all problems p and solvers s.

The performance ratio in (9) is similar to (6), but more realistic: we consider a solver successful only, if the solution returned by a solver is within d of the best solution found. For the ratio (9), the corresponding performance profile can be defined analogously to (7).

Notice that the expression in (9) uses the relative error and hence it is valid only for nonzero values $b_{p.}$ In the current numerical experiments, for optimum values (estimates) $|b_p|< 0.1$ we shall use the absolute error $|o_{p,s} - b_p|$ in the condition embedded in (9).

The interpretation of the entire profile is similar as before, but now it also depends on the tolerance d. This way, we can generate a family of performance profiles, for selected values of d >0.

Efficiency Results for LGO Solver Options

In this section we present and discuss results using the performance profiles obtained via the PAVER Server (GAMS Performance World, 2003). The first profile plot (see Figure 3) shows efficiency information only, using benchmark data based on the set of 117 GAMS models. Here we compare the four different LGO operational modes (and underlying methods). Since our goal is not to show competitiveness (in terms of efficiency) with local solvers, we omit comparisons to the other (local) nonlinear solvers here. Not surprisingly, if we are interested in efficiency only, then the local search option of LGO (opmode=0, denoted by LGO-0 in the figure) is the fastest option ‘winning’ roughly 70% of the time: see Figure 3 at time factor 1. This result also indicates that about 70% of the models considered can be solved by LGO’s local solver mode alone.

Since we are also interested in the probability of overall success, next we examine the profiles as $t\rightarrow\infty$. In this case – for the given set of test models – solver modes 2 and 3 (GARS and MS) have the highest probability of success in finding a solution, while opmode 1 (BB) seems to lag a bit behind. Obviously, these findings are not only dependent on the test models selected, but also on the current LGO solver implementations. In our numerous other tests (not discussed here), BB sometimes works very fast – especially for certain structured models – while both GARS and MS will often work better for seemingly ‘structureless’ problems. Furthermore, while MS is typically slowest (due to its repeated local searches), it frequently leads to the best solution found in difficult models.

The lower probability of overall success for the local solver option (LS) can be directly attributed to the need for a ‘sufficiently good’ starting point (from the typically unknown region of attraction of the global solution). Recall that the LGO global solver options are not affected by starting points, except possible numerical anomalies.

Notice that in Figure 3 we also display a profile named CAN_SOLVE: this can be interpreted as the ‘overall success rate’ of the solvers in the LGO solver suite. With all solvers available within LGO, the probability of success is almost 85%, while all solvers perform individually between 70% and 80%. This may sound perhaps a bit disappointing – obviously, one would like to see a ‘perfect’ 100% success rate – but can probably be considered as ‘respectable’ performance. Note in this context that all numerical experiments were done in a totally automated fashion (in fact, on GAMS premises, without any interaction with the author): as usual, one could expect improvement by solver ‘tuning’, but this was not part of the current evaluation exercise. Let us also note that some of the GLOBALLib models are definitely more difficult, than the most often used illustrative GO test examples.

For the sole purpose of a qualitative comparison, let us also refer to the similar performance profiles presented by Dolan and Moré (2002). The best results for the full COPS test model set cited therein are characterized by performance profiles that end up between 80% and 90%, and the best partial profiles reach 90%. Although we are partly 'comparing apples and oranges' here, GO models are – as a rule – arguably more difficult than convex nonlinear models. At the same time, our own test results also indicate room for improvements, of course.

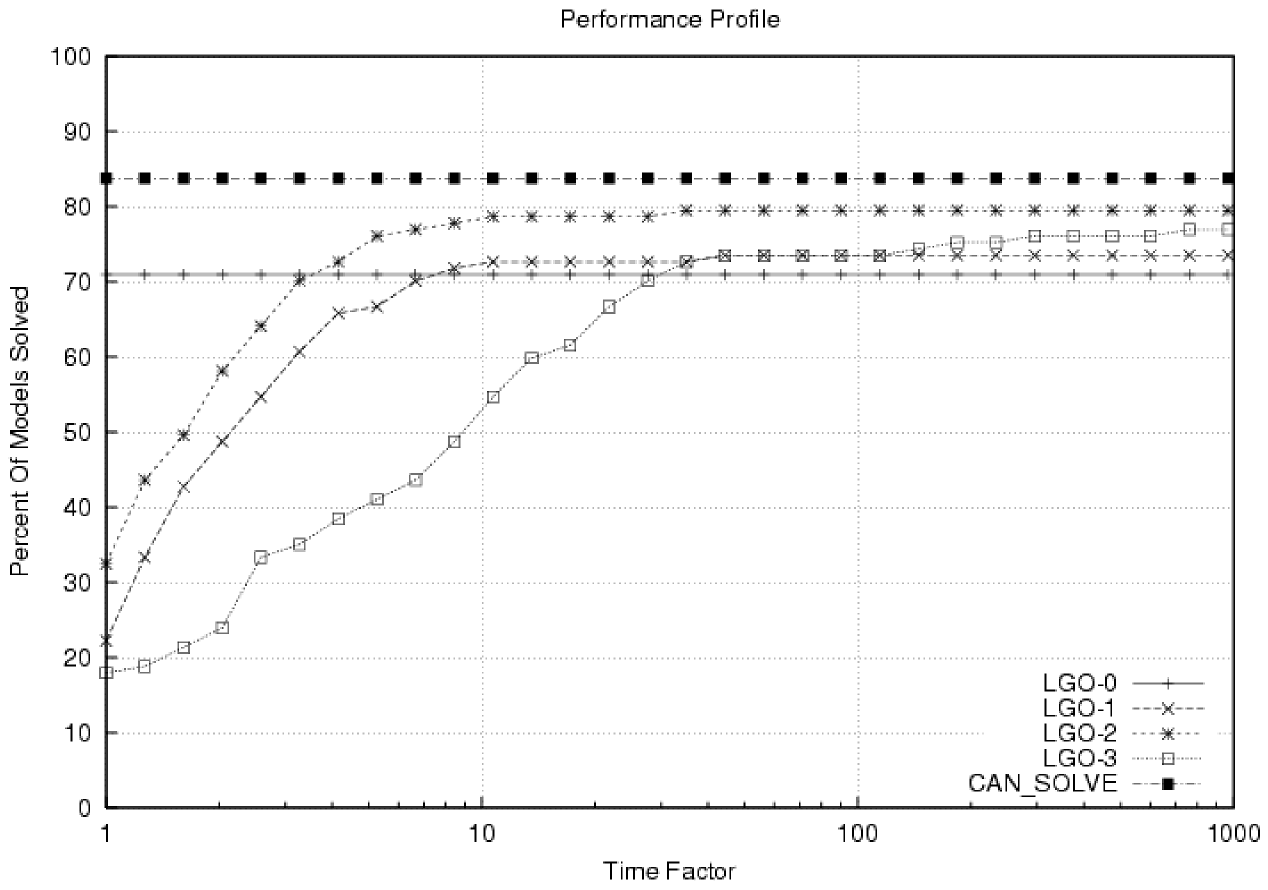

Figure 3. Efficiency profiles: all LGO solver modes are applied to GLOBALLib models.

A Comparison of Solution Quality Results for Nonlinear Solvers

In order to assess also the significance of the global search modes, next we compare the four LGO algorithmic options to three well-established, high quality local solvers: CONOPT, MINOS and SNOPT. Again, we show results using performance profile plots, this time using the profiles with the performance ratio defined in (8) and with $d = 10^{-5}$. (Recall the note related to the use of relative vs. absolute error measures in (9).)

To illustrate the value of global scope search, now we use only a subset of the GLOBALLib models in which the effect of non-default starting points may be significant: therefore the use of global optimization techniques would be of interest. This subset (selected by colleagues at the GAMS Corporation) consists of 63 models.

Figure 4 shows the profile plots for all four LGO solver modes and the three local nonlinear solvers, on the subset of models mentioned above. While for 60% of the models MINOS is the fastest in finding the best solution (see the plot for a time factor 1), this seems to be due to the fact that the default starting point in some local solvers works well for many models.

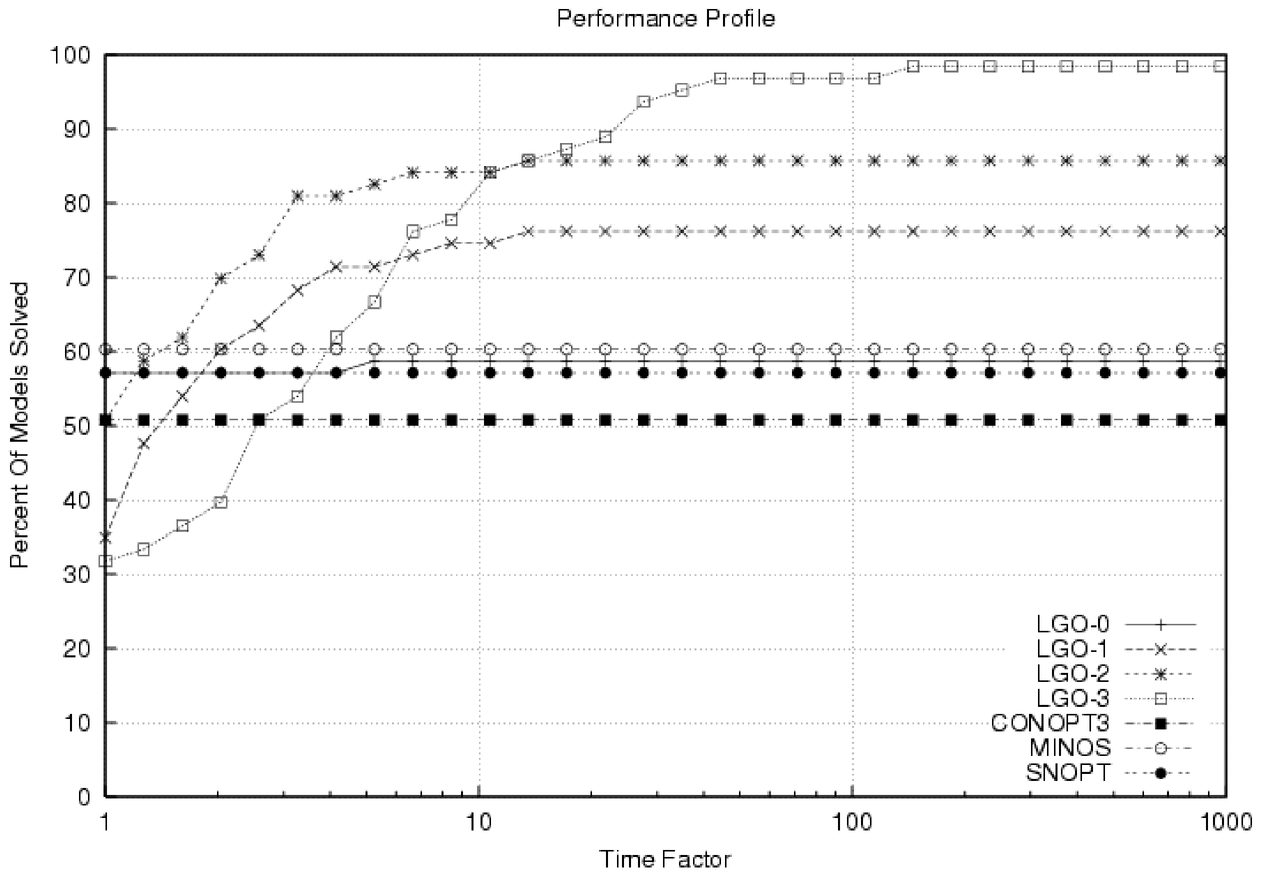

Figure 4. Quality of solution profiles: nonlinear solvers applied to a subset of GLOBALLib models.

On the other hand, if we are interested only in the eventual probability of success in finding the best solution, then the global solver modes in LGO all have a higher success rate than any of the local solvers (as can be expected). Specifically, LS (LGO-0) solves nearly as many models as MINOS (about 60%); BB (LGO-1) solves nearly 80%; GARS (LGO-2) solves about 86%, and MS (LGO-3) solves nearly 100% of the models (within the time frame considered).

The results show that – at least in the given test model subset – there is a large percentage of models where the global optimization techniques in LGO yield a better solution than high-quality local solvers. For almost 40% of the models considered in this subset (i.e., for almost 25% of the entire model library used), LGO finds a better objective value than the local solvers. The difference in solution quality already becomes quite significant, when the allocated solution time is about 10 times larger than the time needed by the fastest (local) solver. This finding indicates that efficient GO solver implementations can be applied to find improved solutions in nonlinear decision models with unknown convexity vs. multiextremality structure. Consequently, in many practically motivated decision models – specifically including e.g. ‘black box’ optimization, response surface and experimental design models, as well as many stochastic and dynamic optimization models – there is a definite need to apply GO methodology.

8 Conclusions

Computational global optimization is coming of age, and – in addition to already existing applications – it has significant potentials. Recently, several global optimization solvers have been implemented, to use within the framework of modeling environments. This article introduced the GAMS/LGO solver implementation that can be used to obtain (numerical) global solutions in merely continuous (and frequently even in discontinuous) GO models.

The test results summarized here illustrate the overall quality and efficiency of the various LGO solver options across a range of models; the results also demonstrate the need for using global solvers to handle general nonlinear models. We are convinced that these findings are valid, in spite of unavoidable biases in model selection, solver settings, and benchmarking methodology.

Acknowledgements

I wish to express my thanks to Armin Pruessner of the GAMS Development Corporation for insightful comments and significant assistance with respect to conducting the GAMS/LGO numerical experiments; he also provided Figures 2 to 4 shown in the paper. Thanks are also due to Alexander Meeraus, Steven Dirkse, Michael Bussieck and other colleagues at the GAMS Development Corporation, for numerous discussions, contributions, and advice related to the GAMS/LGO implementation.

The research and development work related to LGO has been partially supported by the National Research Council of Canada (NRC IRAP Project No. 362093), and by the Hungarian Scientific Research Fund (OTKA Grant No. T 034350).

References

Ali, M. M., Khompatraporn, Ch. and Zabinsky, Z.B. (2003) A numerical evaluation of several global optimization algorithms on selected benchmark test problems. *Journal of Global Optimization*. (To appear.)

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., Spelucci, 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. (Manuscript.) For further information on the COCONUT Project (including this report), see http://www.mat.univie.ac.at/~neum/glopt/coconut/index.html.

Brooke, A, Kendrick, D. and Meeraus, A. (1988) *GAMS: A User's Guide*. The Scientific Press, Redwood City, CA. For extensive current information, see also www.gams.com.

Bussieck, M.R., Drud, A.S., Meeraus, A. and Pruessner, A. (2003) Quality assurance and global optimization. In: *1st International Workshop on Global Constrained Optimization and Constraint Satisfaction (COCOS 2002*, Valbonne, France). Lecture Notes in Computer Science, Springer Verlag, Berlin. (To appear.)

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.

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.

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., 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. (2003) *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 (2003) *Premium Solver Platform – Field-Installable Solver Engines: User Guide.* Frontline Systems, Inc., Incline Village, NV. See also http://www.solver.com.

GAMS Development Corporation (2003) *GAMS – The Solver Manuals.* GAMS Development Corporation, Washington, DC. See http://www.gams.com/solvers/allsolvers.pdf.

GAMS Global World (2003) *GLOBAL Library: A Collection of Nonlinear Programming Models.* See http://www.gamsworld.org/global/globallib.htm.

GAMS Performance World (2003) *PAVER – Automated Performance Analysis & Visualization.* See http://www.gamsworld.org/performance/paver.

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.

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

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

Kampas, F.J. and Pintér, J.D. (2003) Configuration analysis and design by using optimization tools in *Mathematica*. *The Mathematica Journal.* (To appear.)

Khompatraporn, Ch., Pintér, J.D. and Zabinsky, Z.B. (2003) Comparative assessment of algorithms and software for global optimization. *Journal of Global Optimization.* (To appear.)

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).

LINDO Systems (1996) *Solver Suite.* LINDO Systems, Inc., Chicago, IL. See also www.lindo.com.

McCarl, B.A. (2002) *McCarl's GAMS User Guide*.
See http://www.gams.com/dd/docs/bigdocs/gams2002/.

Mittelmann, H.D. and Spellucci, P. (2003) *Decision Tree for Optimization Software.*
See http://plato.la.asu.edu/guide.html.

Mittelmann, H.D. and Pruessner, A. (2003) A server for automated performance analysis of benchmarking data. (Submitted for publication.)

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. (2003a) *Global Optimization Web Page*.
See http://www.mat.univie.ac.at/~neum/glopt.html.

Neumaier, A. (2003b) *COCONUT Project Web Page*.
See http://www.mat.univie.ac.at/~neum/glopt/coconut/index.html.

Neumaier, A. (2003c) Complete search in continuous optimization and constraint satisfaction. In: Iserles, A., Ed. *Acta Numerica 2004.* Cambridge University Press. (To appear.)
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. (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. (2002a) Global optimization: software, test problems, and applications. Chapter 15 (pp. 515-569) in: *Handbook of Global Optimization, Volume 2* (Pardalos, P. M. and Romeijn, H. E., Eds.) Kluwer Academic Publishers, Dordrecht.

Pintér, J.D. (2002b) *MathOptimizer – An Advanced Modeling and Optimization System for Mathematica Users. User Guide*. Pintér Consulting Services, Inc., Halifax, NS, Canada.

Pintér, J.D. (2003a) *LGO – A Model Development System for Continuous Global Optimization. User's Guide*. (Current revised edition.) Pintér Consulting Services, Inc., Halifax, NS.

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

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

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

Pruessner, A. and Mittelmann, H.D. (2003) Automated performance analysis in the evaluation of nonlinear programming solvers. Lecture presented at the International Symposium on Mathematical Programming, Copenhagen, August 2003.
See http://www.gams.com/presentations/present_nlpII.pdf.

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

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.

Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J. and Marti, R. (2002) A multistart scatter search heuristic for smooth NLP and MINLP problems, *INFORMS Journal on Computing.* (To appear.)

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.

Wolfram, S. (1999) *The Mathematica Book*. Wolfram Media and Cambridge University Press. (4th Edition.)

Wolfram Research (2003) *Mathematica* products home page 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.