Model Development, Solution, and Analysis in Global Optimization

János D. Pintér

Pintér Consulting Services, Inc. (PCS) & Dalhousie University
PCS: 129 Glenforest Drive, Halifax, NS, Canada B3M 1J2
jdpinter@is.dal.ca http://is.dal.ca/~jdpinter/

To appear in:
Global Optimization — Selected Case Studies
(János D. Pintér, Editor)
Kluwer Academic Publishers, Dordrecht / Boston / London.

Abstract. The primary objective of this chapter is to summarize the key features and usage of the LGO modeling and solver system, as it is applied to several global optimization (GO) case studies described elsewhere in the present volume. The discussion is extended by providing more general comments on GO models, solution strategies, software, test problems, and by pointing towards a broad range of existing and potential applications.

Keywords: GO model types and solution approaches; software development; LGO program system; tests and illustrative applications.

AMS Subject Classification: 65K30, 90C05, 90C31.

1. Introduction

The principal objective of this chapter is to summarize the key features and usage of the LGO modeling and solver system, since it is applied to several case studies described elsewhere in this book. Please see the chapters written by Isenor, Cada and Pintér (2001); Tervo, Kolmonen, Jaatinen and Pintér (2001).

We shall first present a general GO model form with additional notes on key analytical assumptions, and its most frequently studied special cases. This is followed by a concise list of the most frequently used GO solution methods. Next, the principal stages of model development, solution, and result analysis by using the LGO software system are described. The discussion is concluded by pointing towards an extensive set of existing and prospective applications of GO.

2. The Continuous Global Optimization Model

The practical objective of global optimization is to find the 'absolutely best' solution of nonlinear decision models, in the (known or assumed) presence of multiple local optima. A detailed exposition of the most prominent GO model types and (mostly exact, deterministic) solution approaches can be found, for instance, in the Handbook edited by Horst and Pardalos (1995). Several dozens of other textbooks and quite a few informative WWW sites are also devoted to this emerging subject. The list of references provided at the end of this paper gives a sample of the significant amount of GO information available.

Let us consider the general continuous GO model defined by:

- x decision vector, an element of the real n-space $\boldsymbol{R}^n$
- $f(x)$ continuous objective function
- D non-empty set of admissible decisions (D is described by l, u, g below)
- l, u explicit, finite bounds of x which define an embedding 'box' in $\boldsymbol{R}^n$
- $g(x)$ m-vector of continuous constraint functions ($m \geq 0$).

Applying this notation, the continuous global optimization model is stated formally as

(1) $\min f(x)$ subject to $x \in D := \{x: \; l \leq x \leq u \;\; g(x) \leq 0\} \subseteq \boldsymbol{R}^n$.

Explicit bounds on the constraint function values can also be imposed; however, such models are easily rewritten in the form (1).

Let us observe first that, since the functions f and g are all continuous in D, model (1) evidently has a non-empty set X^* of globally optimal solutions. At the same time, one can also immediately realize that, in its full generality, instances of model (1) can pose an extreme numerical challenge. Since the usual convexity assumptions are absent, D may be disconnected and/or non-convex, and the objective function f may well be multiextremal. Therefore the number of local (pseudo-)solutions is typically unknown and, in principle, it can be arbitrarily large; furthermore, the quality of the various local solutions and the best one may also differ to an arbitrary degree.

To illustrate this point, see Figure 1 that displays a rather 'hilly landscape'. (The picture shows the surface plot of a relatively simple composition of trigonometric functions with embedded low-order polynomial arguments, in just two variables). The function displayed could be the objective f in (1) defined on the corresponding interval feasible region $[l, u]$, without the added constraints g. Note that adding the latter could make the corresponding GO problem (even) more complicated, as the components of g could cut out pieces of complicated geometry from the embedding rectangle $[l, u]$.

In many practical cases, it is not too difficult to provide the bounds $[l, u]$, while g may express (simple or complex) interactions among the components of the decision vector x.

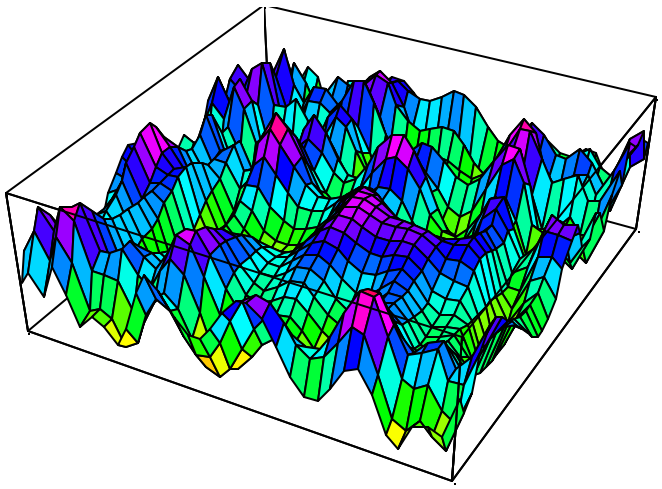


Figure 1. A two-variable multiextremal function, as an instance of model (1).

To solve the problem (1) in the strict mathematical sense means to find the complete set of globally optimal solutions X^*, and the associated global optimum value $f^*=f(x^*)$, $x^* \in X^*$. In most cases, at least in the realm of continuous GO, we need to replace this ambitious objective by finding a verified, or perhaps statistical estimate of f^*, and the corresponding approximation of a point (points) from the set X^*. Naturally, such estimates are to be determined on the basis of a finite number of algorithmically generated sample points from D, or from the embedding interval $[l,u]$.

For reasons of better analytical and numerical tractability, usually the following additional assumptions are made:

- D is a full-dimensional subset ('body') in $\boldsymbol{R}^n$
- X^* is at most countable
- f and g (the latter component-wise) are Lipschitz-continuous functions on $[l,u]$.

Observe that the first assumption (i.e., the fact that D is the closure of its non-empty interior) makes algorithmic search possible within the set D. (For instance, this requirement implies that nonlinear equality constraints need to be incorporated into the objective function as done e.g., in Pintér (1996a), Chapter 4.1.)

With respect to the second assumption, let us note that in most practical contexts (at least in most well-posed problems) the set of global optimizers consists only of a single point, or at most of several points. (Note at the same time that practicioners applying GO may formulate models whose solution set is a manifold: in such cases, proper numerical GO software implementations will find a single solution, or several of them.)

Finally, the Lipschitz assumption is a sufficient condition for estimating f^* on the basis of a finite set of search points. (The real-valued function h is Lipschitz-continuous on its domain $D \subset \boldsymbol{R}^n$, if $|h(x_1)-h(x_2)| \le L\|x_1-x_2\|$ holds for all pairs x_1,x_2 in D; here $L=L(D,h)$ is a suitable Lipschitz-constant of h on the set D.) We emphasize that the factual knowledge

of the smallest suitable Lipschitz-constant is not required, and in practice it is quite typically unknown indeed. The Lipschitz-criterion is met, e.g., by all continuously differentiable functions defined on [*l*,*u*]. However, their class is even broader.

One should also mention at this point that optimization models including binary (or more general integer) variables can also be stated in the general form (1). Although this may not be the most suitable representation in many concrete cases, it certainly shows the close connection between (and related complexity of) general integer and continuous GO models. Quite a few of the solution approaches listed in the next section, *mutatis mutandis*, can be applied to integer models, as well as to continuous GO problems. This point is well illustrated in several chapters of the present volume.

3. Solution Strategies

Due to the very general model structure postulated above, classical numerical approaches are generally not directly applicable to solve GO model instances (even much simpler ones than the one in Figure 1). Instead, a truly global scope algorithmic search methodology is needed that, in principle, enables us to 'visit around' the entire feasible region *D*. The model (1) covers many specially structured problem-classes (such as e.g., the concave minimization problem under linear or convex constraints), as well as far more general ones (such as differential convex, Lipschitz or general continuous problems). Hence, one can reasonably expect that the GO methods suitably tailored to certain model-types within (1) will also vary to a considerable extent. Very general search strategies can be expected to work for most GO models, albeit their efficiency might be (relatively) low for specially structured problems.

To illustrate this point by an (in fact, not too extreme) example: one can rightfully expect to solve linear programming or convex nonlinear models by using GO tools, since such models certainly belong to the model-class (1). However, these are not the typically envisioned model-instances when designing GO methods. On the other hand, strictly specialized solvers may not work at all for problem-types outside of their scope. LP or convex programming solvers are incapable of handling GO models, except under highly specific circumstances. Furthermore, even specially structured nonconvex optimization methods may not necessarily be applicable to certain more general GO models. (For example, recall Figure 1, and compare it to the concave minimization model under linear constraints).

In the past decades (at least since the sixties), a considerable variety of GO models and solution approaches have been proposed, analyzed, and applied. Most GO software implementations are based upon one of the approaches listed below, quite possibly combining ideas from several strategies. (Many of these approaches are applied to the case studies and problems discussed in the present volume.)

Exact Methods

For these methods—under postulated analytical conditions—rigorous global convergence properties can be proven. (This, however, says very little regarding their practicality, especially since implementation depth and sophistication may vary.)

- adaptive stochastic search algorithms
- branch and bound algorithms (including also Lipschitz optimization and interval methods)
- complete (enumerative) search strategies
- homotopy (parameter continuation) methods
- 'naïve' exact approaches (such as uniform grid or pure random searches)
- statistically based search algorithms (including Bayesian strategies)
- successive outer approximation (relaxation) methods
- trajectory (path-following) methods

For concise or more detailed discussions of exact methods, consult for instance the following books: Horst and Pardalos (1995), Horst and Tuy (1996), Kearfott (1996), Neumaier (1990), Pintér (1996a), Törn and Žilinskas (1989), Zhigljavsky (1991), as well as issues of the *Journal of Global Optimization*. Several books mentioned later in connection with GO applications also contain chapters that describe exact approaches.

Heuristic Strategies

These methods are most frequently based on attractive analogies from nature or from the sciences. However, rigorous global convergence properties are (in general) unknown. (Again, efficiency and practicality significantly depends on their implementation.)

- ant colony systems
- adaptive dynamic control (exact versions are also known)
- approximation (response surface, convex global underestimation,…) methods
- continuation methods (based on smoothing model transformations)
- direct search (flexible polyhedron, scatter search,…) approaches
- fuzzy logic-based search methods
- genetic algorithms, evolution strategies
- 'globalized' extensions of local search methods
- neighbourhood search approaches
- neural networks
- sequential improvement of local optima (tunneling, deflation, filled functions,…)
- simulated annealing
- tabu search

For discussions of (mostly) heuristic strategies, consult for instance the following books: Aarts and Lenstra (1997), Falkenauer (1998), Goldberg (1989), Glover and Laguna (1997), Kosko (1993), Laguna and González-Velarde (2000), Michalewicz (1996),

Michalewicz and Fogel (1999), Mockus, Eddy, Mockus, Mockus, and Reklaitis (1996), Mohammadian (1999), Osman and Kelly (1996), Van Laarhoven and Aarts (1987), Voss, Martello, Osman and Roucairol (1999), as well as some articles in *the Journal of Global Optimization*, and many more in the topical *Journal of Heuristics*.

Brief annotated discussions of predominantly combinatorial algorithms are collected in Skiena (1998); see also Press, Teukolsky, Vetterling and Flannery (1992) on several heuristic methods. Even more concise reviews of GO approaches, available through the World Wide Web, are provided, for instance, by Neumaier (2000), Pintér (1999), and Gray, Hart, Painton, Phillips, Trahan, and Wagner (1997).

4. Software Development

In spite of significant theoretical advances, GO software development and 'standardized' use lag behind. This can be largely explained by the potential numerical difficulty of GO problems, as illustrated by the 'simple' two-dimensional box-constrained optimization problem depicted by Figure 1. Even much simpler problem-instances, such as e.g., indefinite quadratic programming, belong to the hardest class of mathematical programming problems. Among others, several chapters in Horst and Pardalos (1995) offer related discussions of GO model complexity issues.

There exist several broad classes of exact GO approaches (see above) that possess strong theoretical convergence properties, and—at least, in principle—are straightforward to implement and apply. However, all such rigorous approaches involve a computational demand that increases exponentially as a function of problem-size, even in case of the simplest GO problem instances. Therefore—and also in view of the related comments made in Section 3—many practical GO strategies are supplemented (completed) by a 'traditional' local optimization phase(s). Global convergence, however, is only guaranteed by the global scope algorithm-component(s): the latter theoretically should be used in a complete, 'exhaustive' fashion. The above remarks indicate the inherent theoretical and practical difficulty of developing robust, yet efficient GO software.

Since the computational demand of rigorous deterministic strategies can be proved to be some exponential function of the problem dimensionality, GO problems in $\boldsymbol{R}^n$ (n being just 5, 10, 20, 50, or 100,...) can have rapidly increasing and practically prohibitive complexity. The complexity issue will remain valid, in spite of the fact that computational power seems to grow at an unbelievable pace: the so-called 'curse of dimensionality' is here to stay. This observation makes a strong case for applying (also) randomized search components, if efficiency is a significant issue. Obviously, this will happen at the expense of losing the rigorous convergence guarantee of exact deterministic approaches (such as e.g., branch-and-bound strategies).

A few years ago, a survey on continuous GO software was prepared for the newsletter of the Mathematical Programming Society (Pintér, 1996b). Drawing on the responses of software developers and some additional information available, over 50 software

products were annotated in that review. By now the number of solvers aimed at solving GO models is probably in the order of a few hundreds. However, the general impression is that many of the known software products are still at an experimental development stage, and of dominantly ‘academic’ character, as opposed to ‘industrial strength’ tools. (Of course, it is entirely possible that software products used by industry and private companies are not announced publicly.)

Some key aspects that should be addressed by professional quality GO software development are listed below:

- well-specified hardware and software environments (supported development platforms, operating systems and modeling/programming languages)
- quality user guidance: clearly outlined model development procedure, sensible modeling and troubleshooting tips, user file samples (templates), simpler and also non-trivial(!) numerical examples
- fully functional and friendly user interface
- 'fool-proof’ solver selection and execution procedures
- good communication and documentation: clear system output for all foreseeable program compilation, linking, execution situations, including proper error messages, and result file(s)
- visualization features which are especially desirable in nonlinear systems modeling, to avoid problem misrepresentation, and to assist in finding alternative models and solution procedures
- reliable, quality user support
- continuous product maintenance and development (since not only science progresses, but hardware devices, operating systems, as well as target development platforms are in permanent change).

This tentative ‘wish-list’ of requirements indicates that although the task is not impossible, it is and remains a challenge—especially in the context of GO.

5. LGO: An Integrated Model Development and Solver System

The Lipschitz(-Continuous) Global Optimizer (LGO) program system analyzes and numerically solves optimization problems of the general form (1), under the structural assumptions listed in Section 2. Therefore it is particularly suitable to handle GO problems related to models that are supported by limited (or difficult to use) analytical information.

Below a summary of the principal LGO features is presented; followed (in Section 6) by a short description of the recommended application development steps. For theoretical background, including also some key implementation details, consult Pintér (1996a). Additional aspects related to more recent developments, current LGO features, and detailed user guidance are discussed by Pintér (1998, 2000).

Application Program Structure

The interdependence of the key program components is shown on Figure 2. MAIN symbolizes the driver program that invokes LGO; it may have also other functions such as communicating with external programs. FCT describes the application program developed by the LGO user. The input parameters are additional model descriptors and a few key optimization parameters. This structure enables repeated runs, making use of the same executable program that combines the MAIN and FCT source code with the LGO file system. The output consists of automatically generated (summary and detailed) result text files; additional options for result analysis will also be discussed later.

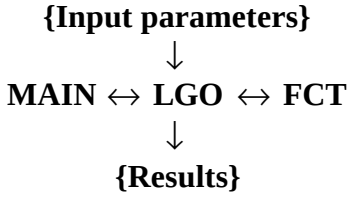


Figure 2. LGO application program: basic structure.

LGO supports a range of flexible options to use MAIN and FCT. The latter files can be provided in any of several compatible programming languages. They can be connected to additional program (source or object code) files; can be provided as compiled object files, or even as executables. Simpler structures are also possible. For instance, LGO can be provided as an executable program that calls only FCT, so that MAIN is absent (being built into LGO); or MAIN and FCT combined can call LGO in a 'silent subroutine' mode. The listed options include genuine ‘black box’ optimization, assuming that proper input/ output communication is established between LGO and the external user files.

Solver Options

Motivated by the related discussion in Sections 3 and 4, the core of LGO is a suite of robust and efficient nonlinear (global and local) optimization methods. Currently, the following component algorithms are offered:

- global adaptive partition and search (branch-and-bound method, enhanced with random sampling)
- global adaptive random search
- unconstrained local search, applying an exact penalty function approach
- constrained local search, based upon sequentially generated local approximations of the model functions

Note that the component algorithms themselves are made up by several (sub)algorithms, but these are 'hidden' from the user, in order to make the usage of LGO easier. The built-in global scope algorithms are also equipped with statistical tools. These generate a statistical lower bound estimate, based upon the search points and corresponding function values sampled in the global phase of LGO. Both global scope algorithms are gradient-

free procedures. This fact makes their application easy in many practical modeling situations in which higher order information is difficult (or computationally expensive) to obtain. The global solvers, when activated with suitable parameterization, enable a robust and theoretically established approximation of the solution.

In most practical cases, numerical efficiency requires the additional use of local search strategies. This is the main reason for using also built-in local solvers. As emphasized earlier, the exclusive use of local solvers *per se* may miss the global optimum, unless the problem is postulated to have a convex structure. In the local solver options, gradient information is numerically approximated, by finite differences. (Automatic differentiation options can also be used, in the presence of suitable program libraries supporting such operations: however, currently this is not offered as a standard feature.)

Solver Operational Modes

The LGO solvers can be activated automatically or interactively. The automatic option only asks the LGO user to select one of the global search modes; which will be followed automatically by the local search procedures.

The interactive option makes possible to select solvers one by one, followed by the explicit allocation of computational effort to each solver mode invoked (by setting the maximal number of objective and constraint function evaluations). This approach makes possible to provide globally established optimum estimates, even in case of a limited search effort (forced e.g., by expensive function evaluations and a limited solution time).

LGO also offers an automatic local search procedure (that includes both local solvers) launched from a user-supplied initial solution. This feature supports the solution of convex models, while it also enables the fast use of pre-specified expert estimates of the global solution. LGO can also be used to solve linear programming models, although this can not be expected to be its main virtue. Rather, it can be considered and used also as an effective dense nonlinear solver.

Development Environments and Connectivity Issues

The current standard version of LGO has been developed in Fortran 90 (LF90, by Lahey Computer Systems, 1998) for personal computers. In addition to immediate LF90 and LF95 connectivity, Dynamic Link Library (DLL) connection is also supported by LF90 (as of September 2000) with respect to the following development environments: Borland C/C++ and Delphi; Microsoft Visual Basic and Visual C/C++. Connection to other applications via DLLs (and in certain cases, via static link options) is also possible.

LGO versions can be run in 'plain' DOS sessions, in a DOS 'box' of any of the currently used Windows (95/98/2000/NT) versions, or as a fully Windows-style application. LGO sessions can also be called directly from other Windows applications that support external program calls. As mentioned above, LGO can invoke external application

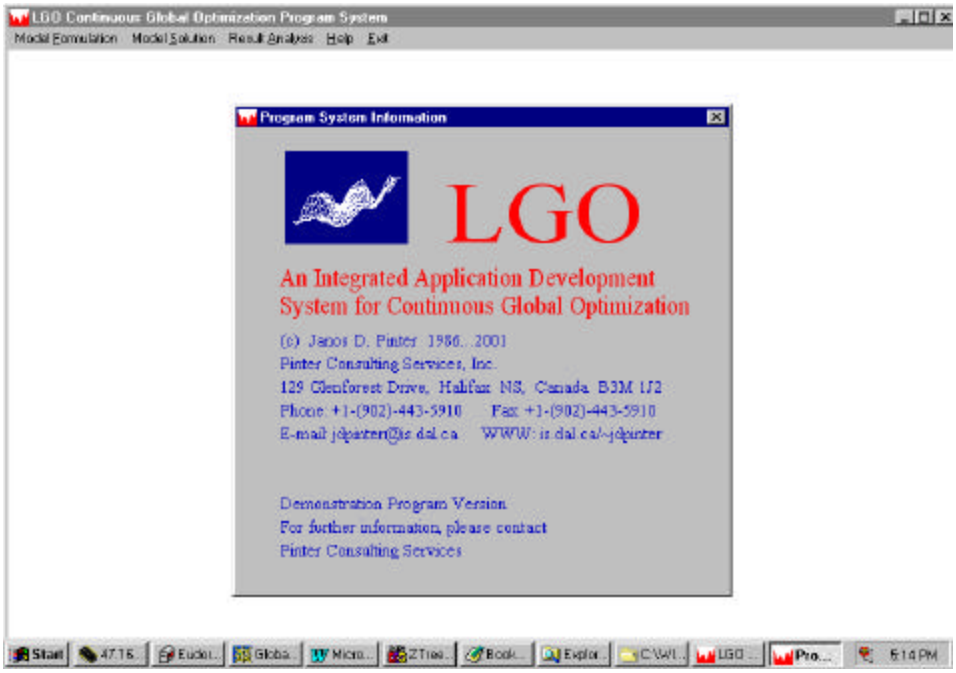
programs, in order to evaluate functions, execute subroutines or auxiliary calculations that may contribute to modules of the GO problem analyzed.

The LGO 'command-line' mode is available also for workstation and mainframe platforms, or for other PC Fortran platforms, without the interactive graphics capabilities to be discussed later. This mode provides plain text output to the screen and/or to files. The description presented in the next section refers to the current menu-driven Windows version, but model development using any other LGO version follows essentially the same guidelines.

6. Model Development Using LGO

LGO Menu Interface

In the Windows 95/98/NT/2000 implementation, the solver system is embedded under a menu-driven interface, to assist LGO users. Upon launching the program system, the following menu options can be activated (the principal menu options indicated by boldface letters lead to the selection of corresponding submenu items):



Model Formulation	**Model Solution**	**Result Analysis**	**Help**	**Exit**
Project File Names	Input Parameter File	Summary Output File	LGO Information Summary	Quit
User Main File	Run LGO	Detailed Output File	Model Development Summary	
User Function File		External System Call	About LGO	
Compile LGO Model				
External System Call				

Figure 3. LGO menu options.

The functionality of the menu options is described below.

Model Formulation

In order to apply LGO, the user first needs to define the optimization problem, according to standard specifications. The Project File Names option prompts the user to give names to the following program items (their functionality is shown in brackets, in accordance with the basic structure shown by Figure 2):

- user main file (see MAIN in Figure 2; provides main application program frame)
- user function file (see FCT in Figure 2; provides model description)
- project executable file (combines the user files named above with the LGO system, and subsequently generates LGO output files)
- input parameter file (provides runtime information for the project executable file)
- summary output file (contains principal information regarding run results)
- detailed output file (contains more detailed information regarding run results).

Upon invoking this menu option, default file names are provided in a Windows dialog. The given names correspond to a test problem file system available to LGO users, as an example to set up their own applications. The defaults can—and, of course, should—be overwritten following the user application needs, while keeping the sample user files. The sample files are commented in sufficient detail to assist their easy usage.

Selecting next the User Main File option invokes the application frame program file, under the name chosen in the previous menu option. In standard LGO shipments the simple Notepad text editor (a Windows accessory) is activated to display and manipulate this file, as well as all other LGO text files discussed later on. (If LGO users prefer to use a different text editor, then this can also be simply arranged upon request.) In the main user file the input/output files are opened, and the call to the LGO solver system is made. Optional, application-dependent user actions can be also included here, before and/or after executing the LGO run.

The User Function File option serves to open a file, in order to describe the optimization model that will be submitted to LGO. The objective function and the constraints are defined here, following again a commented sample file.

The Compile LGO Model option invokes the generation of object files on the basis of the user source files discussed above; then links these to the LGO object file system and the auxiliary files (modules, libraries, and resource file) needed. The actual compilation obviously depends on the (Lahey Fortran 90 or 95, Borland C/C++ or Delphi, Microsoft C/C++ or Visual Basic) development environment used. If all compilation and linking operations are successful, then control is returned to the main menu level, and the user may proceed to the model solution stage. In case of compilation and/or linking errors, corresponding messages appear on the screen (in a DOS ‘box’ under Windows). The LGO system also indicates the errors: a message prompts the user to return to editing mode. (All errors need to be corrected as this point, before proceeding further.)

The External System Call option can be used to launch direct calls to arbitrary program systems available to the user, either by simply typing the name of an executable program (available from the path), or by invoking Windows Explorer. This option—available also under the Result Analysis main menu item—can be helpful in describing and analyzing models, and/or in exporting LGO results.

Model Solution

Following successful model compilation and linking, the LGO executable program is ready to run, but it still needs certain input parameters. These parameters are set by selecting the Input Parameter File option. Again, a commented sample of this file is provided to assist users, thus its preparation is straightforward. In particular, the number of variables and constraints, explicit lower and upper bounds, nominal values (to support local search) and a few key optimization parameters are defined in this file. This structure allows for repeated runs on different feasible interval sets, launched from different starting poins, and/or using modified optimization parameters, using the same executable program.

The Run LGO menu option launches the application. As mentioned earlier, LGO executable programs can be run in automatic global and local search modes, as well as in interactive operational modes. All user actions are prompted by the system, with added brief on-line explanations. During program execution, LGO generates troubleshooting messages as needed and terse progress reports (current solver mode, incumbent optimum estimate, and the current number of function calls) to the screen. In interactive mode, after completing each solver option the user is prompted to select another (recommended) solver or to terminate the optimization procedure. Upon successful completion, two automatically generated output files are available for inspection and further work. In case of runtime errors caught by LGO or by the operating system the user needs to return to the model formulation stage. (Such errors are caused most typically by modeling flaws.)

Visualization

LGO has built-in visualization capabilities to assist the model development procedure. Specifically, upon completing the optimization process the user can view projections of an aggregated merit function (objective plus constraint-induced penalties, as needed) in interactively selected variable subspaces. In these projections, all other coordinates are held at their optimized estimate. The search progress is also summarized in these figures by displaying the projected scatterplot of improving search points (all other coordinates of the points correspond to the actually found sequence of improving solution vectors).

Figure 4 displays a projection of the merit function, using a standard LGO test function as an example. (This particular, fairly complex test problem has 10 variables and 6 constraints, some of the latter being nonconvex: the code can be provided upon request to interested colleagues.) Feasible/infeasible set projections with respect to selected constraint functions can also be generated interactively: see the example shown in Figure

5. (Both Figures 4 and 5 are based on original pictures generated by LGO, saved as color bitmaps; please see the explanatory notes provided in the pictures.)

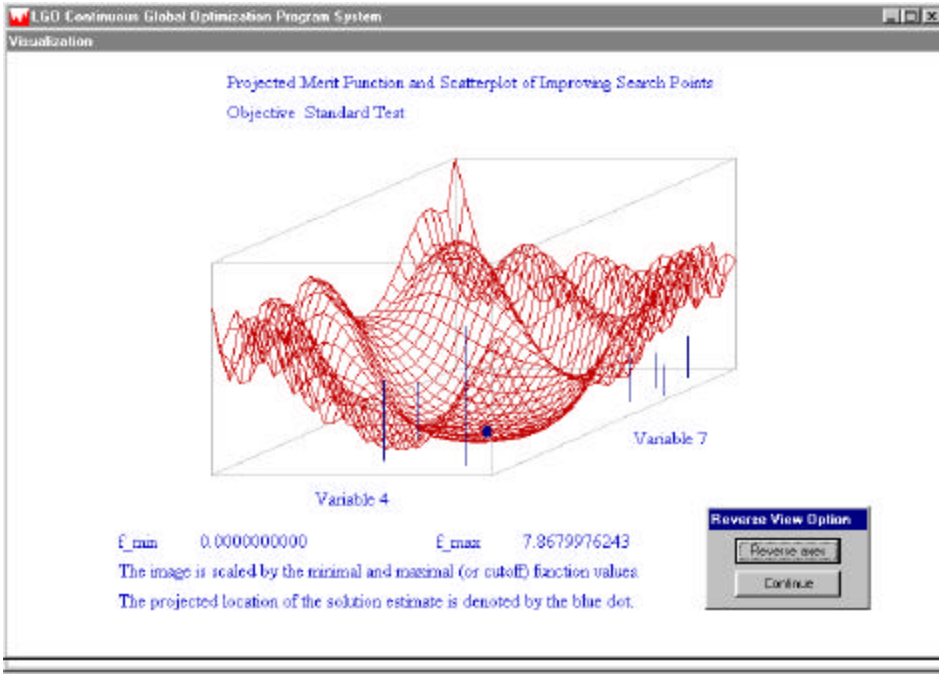


Figure 4. Aggregated merit function in a standard LGO test problem

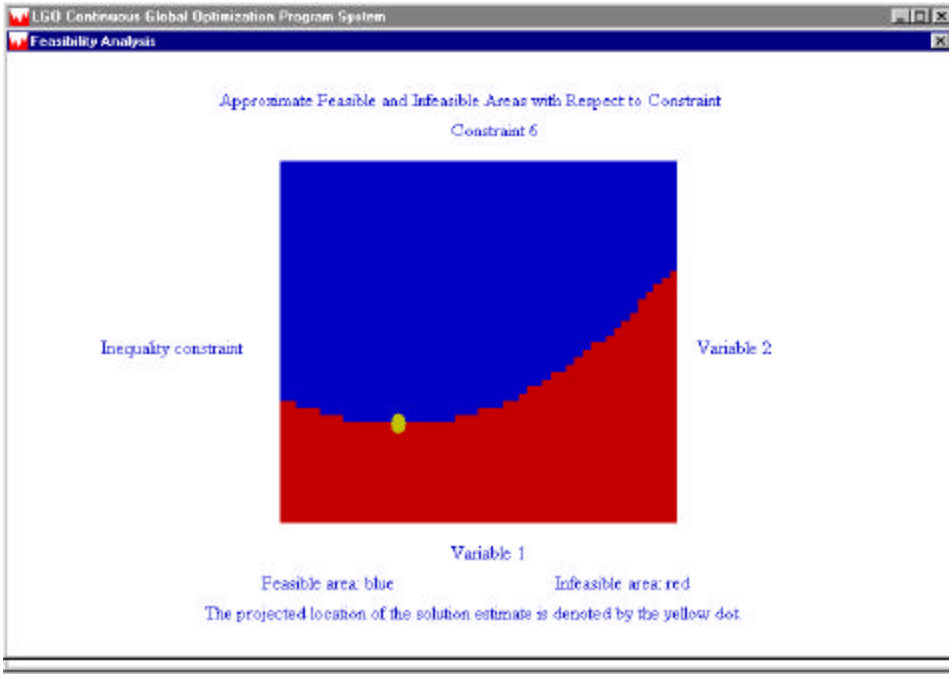


Figure 5. Feasibility plot in a standard LGO test problem

Although the use of these visualization options can be simply skipped, in most cases it is a salient idea to check the graphical information, at least in subspaces of special interest. By visual analysis, the user can verify a hypothesis regarding the expected shape and complexity of the objective and constraints. This can effectively help in finding modified problem formulations and/or LGO input parameters to be applied in subsequent LGO runs. Such pictures may also be used to suggest tentative reductions of the search region, in order to enable a more detailed 'exploration' (in a subsequent run) around the current estimated optimum, or somewhere else. As mentioned earlier, runs on modified box regions, or from various starting points can be launched without recompiling the LGO executable program.

After the LGO optimization run is terminated, control is returned again to the main menu level. Accordingly, repeated model formulation, compilation and linking, input file changes, or result analysis may follow.

Result Analysis

The solution procedure is typically followed by an inspection of the text result files generated by LGO. This includes two options. The Summary Output File option displays a basic report of the results: optimum estimate, (global search phase based) lower bound estimate, solution vector estimate, and the total runtime. The Detailed Output File option displays a more thorough description of the program run. Namely, it echoes the variable ranges, the given nominal values, and the input parameters of all sequentially invoked solver options. Furthermore, it also keeps track of the improving solutions found in the global search phase(s), and the stopping criteria met during execution, in addition to the information presented in the summary output file.

The two result files may have different importance at different stages of model development. The summary file is most useful in well-tested ‘routine’ LGO runs, while the additional information provided by the detailed output file can assist both model development and the choice of LGO solver parameterizations.

The results obtained may be directly communicated to other programs. These external programs can be made available from the user main and function files of LGO, but can also be run independently. Finally, they can also be invoked by using the External System Call options discussed above. (For example, using these options one can directly import text information into the LGO user files, as well as export results from LGO to a report generator or to some advanced visualization environment.)

After the analysis of results is completed, control is returned again to the main menu level. In a typical application development process, the model formulation, run and analysis stages are applied in an iterative fashion.

Help

In addition to the detailed User's Guide (Pintér, 2000), concise on-line help is available. The LGO Information Summary menu option serves to introduce novice users to the program system. The Model Development Summary option provides instructions regarding the essential stages of application development using LGO. The About LGO option displays the LGO front screen, including copyright and licensee information.

Exit

Upon selecting the Exit | Quit option, all current project files are closed and saved; all temporary files are deleted; and the LGO run is terminated.

Summing up the review of LGO menu options, it can be seen that essential background information related to the subject of global optimization, to LGO, and to the model structure used is permanently available. In addition, all principal stages of user application (code) development—editing, compilation, linking, program execution, visual and text result analysis—can be directly activated from the menu. These features effectively support rapid prototyping and testing, thereby making the application development process faster and easier.

During the past decade, LGO has been applied to a most diverse set of GO problems, arising from a variety of disciplines. Several of these are described elsewhere in this volume; in the next section some additional pointers are provided.

7. GO Test Problems, Existing and Potential Applications

The quantitative analysis of natural—physical, chemical, biological, environmental, or even economic and societal—systems and their governing processes naturally involves (often highly) nonlinear functions. Consequently, management/optimization/control models based on such nonlinear systems description frequently possess multiple local optima. For sophisticated examples and general principles, as well as for illuminating and far-reaching discussions consult, for instance, Casti (1990), Eigen and Winkler (1975), Mandelbrot (1983), Murray (1989), Schroeder (1991).

The present discussion is concluded by pointing towards the possibility of applying GO methods to a huge variety of test and real-world challenges.

Extensive sets of nonlinear programming test problems, often derived from real-world applications, have been collected by Moré, Garbow and Hillström (1981); Hock and Schittkowski (1987); Floudas and Pardalos (1990); Jansson and Knüppel (1992); Floudas, Pardalos, Adjiman, Esposito, Gumus, Harding, Klepeis, Meyer and Schweiger (1999). On the WWW, one can visit, e.g., the site of Neumaier (2000), and those of Argonne (1993) and Sandia National Laboratories (Gray, Hart, Painton, Phillips, Trahan, and Wagner, 1997). Especially Neumaier's GO pages provides numerous further links

and pointers, including also discussions of test problems. (There is a growing number of further informative WWW sites devoted to GO and related subjects.)

For additional literature on real-world applications, the reader may like to consult, for example, the following works.

- Bomze, Csendes, Horst and Pardalos (1997) is an edited volume with contributions on decision support systems and techniques for solving GO problems, but also on molecular structures, queuing systems, image reconstruction, location analysis and process network synthesis.
- Corliss and Kearfott (1999) review several industrial and financial applications of rigorous GO: currency trading, portfolio management, finite element analysis, magnetic resonance imaging, gene prediction, computer algebra, and signal processing.
- De Leone, Murli, Pardalos and Toraldo (1998) is an edited volume with applications in graphs, econometry, traveling salesman type problems, inverse problems, astronomy.
- Greenberg (1995) provides an annotated bibliography on the use of mathematical programming in environmental systems modeling and management; a number of the listed (over 350) items are—or clearly should be—tackled by GO techniques.
- Grossmann (1996) is an edited volume on GO algorithms and their applications, primarily in chemical engineering (engineering design, process network synthesis, planning, scheduling, and distribution systems).
- Hendrix (1998) presents a variety of interesting applications, e.g., from the fields of environmental management, geometric design, robust product design, and model parameter estimation.
- Migdalas, Pardalos and Värbrand (1997) is another edited volume on multilevel optimization algorithms and their applications.
- Mistakidis and Stavroulakis (1997) present engineering applications of the finite element method.
- Mockus, Eddy, Mockus, Mockus and Reklaitis (1996) discuss network problems, combinatorial optimization (knapsack, travelling salesman, and flow-shop) and batch process scheduling models.
- Pintér (1996a) discusses a variety of detailed numerical tests and case studies (systems of nonlinear equations, data classification, combination of expert opinions, chemical mixture design, environmental model calibration, industrial wastewater systems design, river and lake water quality management, risk assessment and control of pollution accidents).

Although the present chapter is largely focused on continuous GO, one should mention here also the significant number of combinatorial optimization applications described in Glover and Laguna (1997), Laguna and González-Velarde (2000), Michalewicz (1996), Osman and Kelly (1996), Voss, Martello, Osman and Roucairol (1999).

Numerous issues of the *Journal of Global Optimization*, the *Journal of Heuristics*, and *Reliable Computing*—as well as a large number of other professional OR/MS, natural

science and engineering journals—also publish articles describing, or calling for, GO applications.

As the above illustrative list and the present volume itself demonstrate, the application potentials of global optimization are very significant and most diverse indeed.

Acknowledgements

This research work has been partially supported by grants received from the National Research Council of Canada (NRC IRAP Project No. 362093) and from the Hungarian Scientific Research Fund (OTKA Grants No. T 017241 and T 034350).

References

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

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

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

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

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

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

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

Falkenauer, E. (1998) *Genetic Algorithms and Grouping Problems.* John Wiley & Sons, Chichester.

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

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

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.

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

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

Greenberg, H.J. (1995) Mathematical programming models for environmental quality control. *Operations Research* 43, 578-622.

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

Hendrix, E.M.T. (1998) *Global Optimization at Work*. Ph.D. Thesis, LU Wageningen, The Netherlands.

Hock, W. and Schittkowski, K. (1987) *Test Examples for Nonlinear Programming Codes*. Lecture Notes in Economics and Mathematical Systems 187, Springer-Verlag, Berlin / Heidelberg / New York.

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

Horst, R. and Tuy, H. (1996) *Global Optimization—Deterministic Approaches*. (3rd Edn.) Springer-Verlag, Berlin / Heidelberg / New York.

Isenor, G., Cada, M., and Pintér, J.D. (2001) {Title} *In this volume.???*

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

Journal of Global Optimization (published since 1991, by Kluwer Academic Publishers).

Journal of Heuristics (published since 1996, by Kluwer Academic Publishers). date???

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

Kosko, B. (1993) *Fuzzy Thinking (The New Science of Fuzzy Logic).* Hyperion, New York.

Lahey Computer Systems, Inc. (1998) *Fortran 90 User's Guide (Version 4.5).* LCS, Incline Village, NV.

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

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

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

Michalewicz, Z. and Fogel, D.B. (1999) *How to Solve It (Modern Heuristics).* Springer-Verlag, Berlin / Heidelberg / New York.

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

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

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

Mohammadian, M., ed. (1999) *Computational Intelligence for Modelling, Control and Automation*. (Concurrent Systems Engineering Series, Vols. 54-56.) IOS Press and Ohmsha, Amsterdam / Berlin / Oxford /Tokyo / Washington, DC.

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

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

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

Neumaier, A. (2000) *Global Optimization.* (WWW site maintained since 1996.) http://solon.cma.univie.ac.at/~neum/glopt.html.

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

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

Pintér, J.D. (1996b) Continuous global optimization software: A brief review. *Optima 52*, 1-8. (For a WWW version, see e.g., http://plato.la.asu.edu/gom.html.)

Pintér, J.D. (1998) A model development system for global optimization; In: De Leone, R., Murli, A., Pardalos, P.M. and Toraldo, G., eds., *High Performance Software for Nonlinear Optimizaton: Status and Perspectives*, pp. 301-314.. Kluwer Academic Publishers, Dordrecht / Boston / London.

Pintér, J.D. (1999) Continuous global optimization: An introduction to models, solution approaches, tests and applications. *Interactive Transactions in Operations Research and Management Science* 2, No.2. http://catt.bus.okstate.edu/itorms/pinter/.

Pintér, J.D. (2000) *LGO—A Model Development System for Continuous Global Optimization. User's Guide*. Pintér Consulting Services, Inc. Halifax, NS.

Press, W., Teukolsky, S.A., Vetterling, W. and Flannery, B.P. (1992) *Numerical Recipes in Fortran*, and *Numerical Recipes in C.* (2nd Edn.) Cambridge University Press, Cambridge.

Reliable Computing (published since 1998, by Kluwer Academic Publishers). date???

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

Skiena, S.S. (1998) *The Algorithm Design Manual.* Springer-Verlag, New York.

Tervo, J., Kolmonen, P. and Pintér, J.D. (2001) {Title} *In this volume.???*

Törn, A.A. and Žilinskas, A. (1989) *Global Optimization*. Lecture Notes in Computer Science 350, Springer-Verlag, Berlin / Heidelberg / New York.

Van Laarhoven, P.J.M. and Aarts, E.H.L. (1987) *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Dordrecht / Boston / London.

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

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