

CIAO-GO

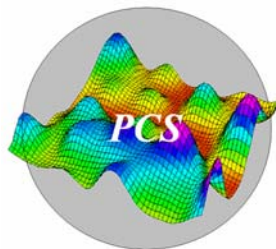
**A Model Development and Solver System for
Global and Local Nonlinear Optimization**

User Guide



Centre for Informatics and Applied Optimization
School of Information Technology and Mathematical Sciences
University of Ballarat
Ballarat, Victoria, Australia

and



Pintér Consulting Services, Inc.
Halifax, Nova Scotia, Canada

Ballarat, March 2004

CIAO-GO

A Model Development and Solver System for Global and Local Nonlinear Optimization

User Guide

Adil M. Bagirov¹, János D. Pintér², and Alexander M. Rubinov¹

¹ Centre for Informatics and Applied Optimization
School of Information Technology and Mathematical Sciences
University of Ballarat
Ballarat, Victoria, Australia

² Pintér Consulting Services, Inc.
Halifax, Nova Scotia, Canada

First Edition: March 2004.

Published by the University of Ballarat, Victoria, Australia.

Copyright Notice

Copyright © University of Ballarat, Victoria, Australia, and Pintér Consulting Services, Inc. Halifax, Nova Scotia, Canada.

All rights reserved.

No part of this documentation and/or of the CIAO-GO software product may be stored, reproduced, transmitted, transferred, translated, or used in any form or by any means, except as authorized by the License Agreement.

Trademarks

The CIAO-GO software product and all accompanying documentation are a joint trademark of the University of Ballarat and Pintér Consulting Services, Inc.

All other software products referred to in this document are trademarks of their respective developers.

Contents

Summary	7
Acknowledgements	8
About the Authors	9
Licensing Agreement, Disclaimer, and Technical Support	10
1. A Brief Introduction to Global Optimization and to CIAO-GO	14
3.1. Global Optimization: Basic Concepts	14
3.2. Non-smooth Optimization	17
3.3. Derivative-free Optimization	18
2. Problem Statement and Structural Assumptions	20
3. The CIAO-GO Program System	22
3.1. Cutting Angle Method	22
3.2. Bagirov's Method	22
3.3. Solver Options	23
3.4. Application Program Structure	24
3.5. Connectivity to Other Modeling Environments	26
3.6. Problem Size and Complexity Considerations	26
4. System Requirements, Portability and Installation	28
4.1. Hardware	28
4.2. Software	28
4.3. Installation	29
5. The CIAO-GO Integrated Development Environment	30
5.1. Summary of Menu Options	30
5.2. Model Formulation	30
5.3. Model Solution	32
5.4. Result Analysis	32
5.5. Help	32
5.6. Exit	32
6. Application Development Steps	34
6.1. Model Formulation	34
6.2. Model Solution	35
6.3. Result Analysis	36

7.	Model Development Tips	38
	7.1. Simplicity and Modular Development	38
	7.2. Decomposition	38
	7.3. Model Scaling	39
	7.4. Search Domain	39
	7.5. Constraint Handling	39
8.	CIAO-GO Program Versions	40
	8.1. Professional, Research and Educational Versions	40
	8.2. Demonstration Programs	40
9.	An Illustrative List of CIAO-GO Applications	41
	References	42
	Appendices	
	A1. Main Program File	48
	A2. Model Function File	50
	A3. Input Parameter File	51
	A4. Summary Output File	52
	A5. Connectivity to Other Application Development Platforms	53
	A6. Workstation and Linux PC Implementations	54

CIAO-GO

A Model Development and Solver System for Global and Local Nonlinear Optimization

Adil M. Bagirov¹, János D. Pintér², and Alexander M. Rubinov¹

¹ Centre for Informatics and Applied Optimization
School of Information Technology and Mathematical Sciences
University of Ballarat
Ballarat, Victoria, Australia

² Pintér Consulting Services Inc.
Halifax, NS, Canada

Summary

Decision problems arise in the context of various scientific, engineering and economic applications. Such problems are frequently modelled by minimizing (or maximizing) the value of an objective function under certain (given) feasibility constraints. Global (nonlinear) optimization is aimed at finding the ‘absolutely best’ solution of such decision models, in the (possible or known) presence of multiple locally optimal solutions.

This document presents and discusses the CIAO-GO modelling and solver environment for continuous global and local optimization. The primary purpose of the CIAO-GO optimization software is to assist in the formulation and solution of **highly nonlinear models, including non-smooth models** under general structural assumptions.

CIAO-GO integrates global and local scope solver modules and is based on completely new approaches. CIAO-GO can also be embedded under a menu-driven user interface, to assist the application development process.

This User Guide primarily discusses the CIAO-GO implementation targeted for personal computers. CIAO-GO is also available for workstations; furthermore, it can be connected to a broad variety of advanced numerical modelling environments, and to other customized user applications.

Keywords: nonlinear decision models; global and local optimization; non-smooth optimization; CIAO-GO optimization software system and its usage; scientific, engineering, and economic applications.

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

Acknowledgements

This report has been prepared as part of a joint research and software development project by the Centre for Informatics and Applied Optimization, School of Information Technology & Mathematical Sciences (CIAO-ITMS), University of Ballarat, Ballarat, Victoria, Australia; and PCS Inc., Halifax, Nova Scotia, Canada.

We wish to thank Professor Sidney Morris, the Head of ITMS, for supporting our project, including financial support for the JDP's work at the University. Numerous faculty and staff members of CIAO-ITMS provided logistics support and helpful advice to us. We wish to specifically acknowledge Associate Professor John Yearwood for strategic discussions, Mr. Graeme Cowling and Ms. Maxine Kingston for overall assistance, and Ms. Rachel Naus for website design.

About the Authors

Adil Bagirov is a Research Fellow at the Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Australia. He graduated from Baku State University (Azerbaijan) in Applied Mathematics / Operations Research, and holds PhD degrees in Mathematics from the Academy of Sciences of Azerbaijan, and from the University of Ballarat.

Dr. Bagirov has written some 50 research articles that appeared in professional journals. His web site is <http://www.ballarat.edu.au/ard/itms/staff/abagirov.shtml> .

János D. Pintér is President and Research Scientist (owner of Pintér Consulting Services, Inc.), as well as an Adjunct Professor at Dalhousie University, both in Halifax, Nova Scotia, Canada. He holds an MSc in Applied Mathematics / Operations Research from the University of Sciences, Budapest (ELTE); obtained his PhD in Probability Theory / Stochastic Optimization from Moscow State University; and also holds a DSc in Mathematics from the Hungarian Academy of Sciences.

Dr. Pintér has written and edited books, and authored numerous articles and technical documents related to nonlinear, global, and stochastic optimization. He is the developer of the LGO, Excel/LGO, GAMS/LGO, TOMLAB/LGO, MathOptimizer and MathOptimizer Professional software products. He has lectured and worked in over 25 countries of the world (in Europe, North America, Asia, Australia, and New Zealand).

Dr. Pintér serves on the editorial board of the *Journal of Global Optimization*, the *Journal of Applied Mathematics and Decision Sciences*, and of the web forums *GAMS Global World* and *GAMS Performance World*.

Dr Pinter's website is <http://www.dal.ca/~jdpinter> .

Alexander M. Rubinov is Professor and Director of the Centre for Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Australia. He graduated from Leningrad State University, Russia. He holds a PhD and a DSc in Mathematics from the Academy of Sciences of the (former) Soviet Union.

Professor Rubinov has published extensively: he wrote and edited books, and wrote numerous articles that appeared in professional journals worldwide. He is an associate editor of the *Journal of Global Optimization*, *Optimization*, *Optimization Methods and Software*, and an editor of the *Pacific Journal on Optimization*. He has presented lectures and worked in Europe, the former Soviet Union, Asia, and the Pacific Rim, including Australia and New Zealand.

His website is

<http://uob-community.ballarat.edu.au/~arubinov/index.html> .

Licensing Agreement, Disclaimer, and Technical Support

Thank you for your interest in the CIAO-GO program system. Before using the software and its technical documentation, please take a few moments to read the following legal and contact information.

Licensing Agreement

The CIAO-GO software product and its entire documentation are developed and maintained by the Centre of Informatics and Applied Optimization, School of Information Technology and Mathematical Sciences, University of Ballarat, Australia; in cooperation with Pintér Consulting Services, Inc., Halifax, NS, Canada. The developer partners will be abbreviated below as CIAO or CIAO-ITMS, and PCS, respectively.

The CIAO-GO program system—including all files distributed as part of the product delivery—and its documentation may not be changed or modified in any way, except by CIAO and PCS, or following the written permission of CIAO and PCS. All proposed changes should be requested by contacting CIAO and PCS.

Without a proper license issued to users individually by CIAO and PCS, no part of the CIAO-GO documentation and/or of the software may be stored, reproduced, transmitted, transferred, or used in any form or by any means, by any information storage and retrieval system or process.

CIAO-GO is provided to registered users in compiled (object, executable, or dynamic link library) form. It is specifically prohibited to apply reverse engineering or any other form of internal program structure analysis to the CIAO-GO software product.

Registered CIAO-GO users are granted permission to use all information summarized by the User Guide, and to apply all CIAO-GO software components and test examples in their own work, without any restriction. A reference to the software and its documentation will be appreciated in published work in which CIAO-GO is used.

If CIAO-GO is used as part of a new software application, then it is requested to maintain all CIAO-GO copyright and licensee information in that application, including all reports generated by the CIAO-GO software.

The CIAO-GO software cannot be made part of a stand-alone user application that is distributed, without making proper (further specific licensing) arrangements with CIAO and PCS.

Regardless of how a copy of CIAO-GO is obtained, it is requested that all users comply with the licensing and registration provisions if they continue to use the software.

Limited Warranty and Disclaimer

CIAO and PCS guarantee that all shipped CIAO-GO products are free from defects in materials and workmanship, under normal use and service for a period of 90 days.

The functionality of the product is also guaranteed to the extent of specific details and illustrative application examples, exactly as described by this document.

CIAO and PCS reserve the right to revise the CIAO-GO software and its documentation, with no obligations to notify any person or organization of such revision. Information updates will be provided upon request.

The CIAO-GO software product is distributed ‘as is’. CIAO and PCS specifically disclaim all warranties—express or implied—related to the merchantability and fitness of the CIAO-GO software for a particular purpose or application use.

In no event shall CIAO and PCS be liable for loss of profit, commercial, business related, or any other damage—including, but not limited to special, incidental, consequential, or any other explicit and implied damages—as a consequence of using, or inability to use, the CIAO-GO software product.

Product Replacement and Returns

CIAO and PCS offer replacement of the CIAO-GO product, if it arrives in unusable form (due to damage caused during shipment). Only shipping and handling charges apply in such cases of replacement. Please send such product replacement requests in writing, and return the damaged product to the sender’s shipment address.

Product returns are accepted within 30 days from the day of shipment to User, with a full refund, except shipping and handling charges that apply also in such cases. Return expenses are to be covered by the user. All returned items have to be in saleable (as if brand new) condition.

A brief written explanation regarding the reason for return will be much appreciated: we will make every reasonable effort to keep all CIAO-GO software users satisfied.

Product Upgrades

We continuously add features to CIAO-GO. Current licensed users will be offered the opportunity to purchase upgrades, at a full deduction of the price already paid. Shipping and handling charges will typically be payable also in such cases.

Product Maintenance Services

Licensed users will also be offered the benefits of a maintenance program: this will allow users to receive automatically all software and documentation upgrades.

Technical Support

Registered users of CIAO-GO are entitled to technical support, provided jointly by CIAO and PCS. Please note that e-mail is the preferred way of communication, except in cases of extreme urgency. Consulting fees may be charged for direct assistance via telephone/fax.

Present and prospective CIAO-GO users are encouraged to contact CIAO and PCS, should they wish to discuss specific applicability issues, and possibilities of research and/or commercial co-operation. All constructive suggestions and comments that could lead to improvements of CIAO-GO, its documentation, and its applicability are welcome and appreciated. We are much interested to hear user comments and suggestions; test models and application examples are also welcome.

Workshops, training courses, and consulting services are also offered in relation to optimization, specifically including the CIAO-GO software, as well as other modeling and optimization software tools developed by CIAO and PCS. Please contact us for further information.

Thank you for your interest in our software.

Sincerely,

Adil M. Bagirov, Research Fellow

Centre for Informatics and Applied Optimization, School of ITMS, University of Ballarat
University Drive, Mount Helen, PO Box 663, Ballarat, Victoria, 3353 Australia

Telephone: +61-(03)-53279330

Fax: +61-(03)-53279289

E-mail: a.bagirov@ballarat.edu.au

Web: <http://www.ballarat.edu.au/~abaghirov>

János D. Pintér, President and Research Scientist

Pintér Consulting Services, Inc.

129 Glenforest Drive, Halifax, NS, Canada B3M 1J2

Telephone: +1-(902)-443-5910

Fax: +1-(902)-431-5100

E-mail: jdpinter@hfx.eastlink.ca

Web: <http://www.dal.ca/~jdpinter> <http://www.pinterconsulting.com>

Alexander M. Rubinov, Director

Centre for Informatics and Applied Optimization, School of ITMS, University of Ballarat
University Drive, Mount Helen, PO Box 663, Ballarat, Victoria, 3353 Australia

Telephone: +61-(03)-53279281

Fax: +61-(03)-53279289

E-mail: a.rubinov@ballarat.edu.au

Web: <http://www.ballarat.edu.au/arubinov>

1. A Brief Introduction to Global Optimization and to CIAO-GO

1.1. Global Optimization: Basic Concepts

Quantitative decisions in the sciences, engineering, and economics are frequently modeled by formulating a corresponding constrained optimization problem. Most typically, the best decision—expressed by a real vector x —is sought that satisfies all stated feasibility constraints, and minimizes (or maximizes) the value of a given objective function. Alternatives and extensions (feasibility models, multi-objective, dynamic or stochastic models, etc.) can also be formulated and applied, to reflect the key issues under study.

To formalize the above mental paradigm, optimization (specifically, mathematical programming) is related to the analysis and solution of problems stated in the form

$$(1.1) \quad \min f(x) \quad \text{subject to} \quad x \in D \subset \mathbb{R}^n.$$

The function f expresses the objective (criterion, such as cost or loss) function in the decision problem, and D denotes the set of possibly considered (admissible) solutions.

Literally, thousands of textbooks, research monographs, and edited volumes (worldwide) have been devoted to the subject of optimization and its multitude of applications. For detailed surveys and discussions of optimization models, algorithms, and software, consult e.g. Bachem, Grötschel and Korte (1983), Hillier and Lieberman (1986), Winston (1992), Moré and Wright (1993), Birge and Murty (1994), Liebling and de Werra (1997).

Global optimization (GO) is an emerging field within mathematical programming, in which the ‘absolutely best’ solution of multi-extremal optimization problems is sought. For illustration, see Figure 1 that displays a multi-extremal function in just one variable (in a given segment); Figure 2 directly extends the same example to two variables.

One can immediately see that even the one-dimensional model shown is far from trivial; furthermore, that the difficulty in higher dimensions may increase quite ‘dramatically’. In mathematical terminology, the difficulty of GO models can be expected to increase exponentially with increasing model size, thus even ‘much simpler’ instances from the GO problem class are, as a rule, NP-hard. This implies that one can not expect to find algorithmic procedures that will work applying ‘only’ a computational effort that is proportional to a polynomial function of the ‘model size’. (The latter is primarily characterized by the number of model variables and constraints.)

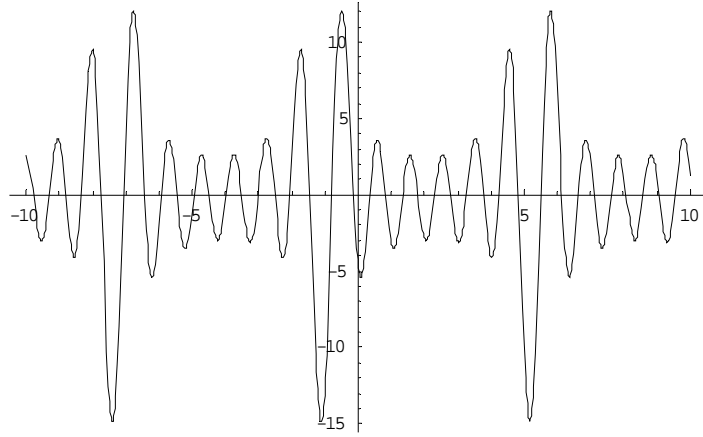


Figure 1. A multi- modal function in one variable.

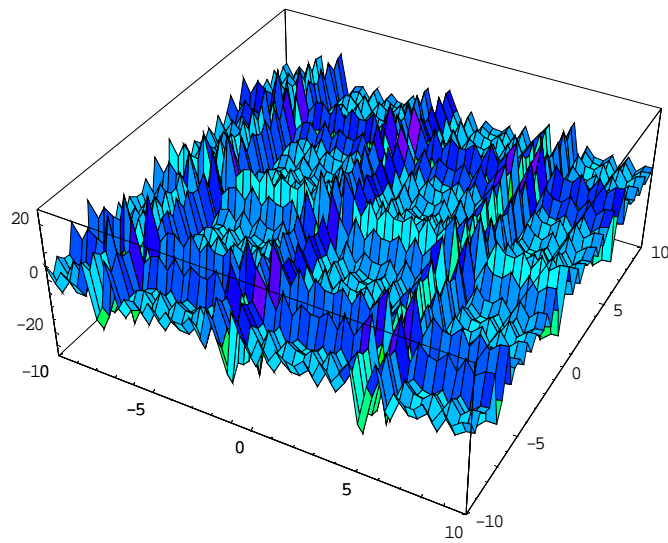


Figure 2. A multi-modal function in two variables.

Let us mention here another (truly broad) area of GO applications. In many practically important cases, the system performance function f and/or the constraints g are determined by a computationally intensive numerical model, or by performing a sequence of stochastic simulation cycles, and so on. That is, the quality of each decision alternative studied is evaluated by activating certain ‘oracle’ (or ‘black box’) operations. One may refer, for instance, to the calibration of complex descriptive (physical, chemical, biological, ecological, etc.) system models, the modeling of stochastic systems, the optimized design and tuning of equipment or processes, and so on.

Global optimization (GO) is an important and rapidly growing area of research. At the time of preparing this manual, well over one hundred textbooks, thousands of articles, and numerous informative web sites have been devoted to this important subject. For in-depth and reasonably up-to-date reviews of the most important GO models and solution approaches, consult e.g. the *Handbook of Global Optimization* volumes edited by Horst and Pardalos (1995), and by Pardalos and Romeijn (2002). Regarding web sites, one can recommend e.g. Neumaier (2003), or relevant sections in the pages of Fourer (2003), Mittelman and Spelucci (2003), with extensive links. Our own web sites (mentioned on page F – 3) also contain extensive related information: we will refer to some of our research work later on.

From a numerical point of view, GO problems can be extremely difficult, and most ‘standard’ optimization approaches are, generally speaking, not directly applicable to solve them. The variety of local optimization strategies—for example, to solve linear programming, convex quadratic programming, or general nonlinear (convex) programming problems—is essentially based on assumptions related to the underlying problem structure. Similarly, there exists a broad range of global optimization problem-types. Some of the most important, broad GO problem-classes are the following:

- Concave minimization under linear or (nonlinear) convex constraints
- General (indefinite) quadratic programming
- Difference convex (DC) programming
- Lipschitz-continuous GO
- Continuous GO

One can observe that the items in the list above, as a rule, correspond to increasingly more general—and thus, typically, also more difficult—problem types. Note also that one could add to the list above the entire category of combinatorial optimization problems, since pure (and mixed) integer programming problems can be directly transformed into continuous GO models.

As it can be expected, the specific theory and the solution approaches to such diverse GO problems also will vary to a considerable extent. Again, we refer to the *Handbook of Global Optimization* volumes as well as to the entire *Nonconvex Optimization and its Applications* (Kluwer Academic Publishers) book series, and other topical books that

present a discussion of the most prominent methods. Selected references are provided at the end of this manual.

In the next two sections, we present an informal discussion of the key concepts and approaches embedded in the CIAO-GO software system. More details will follow in Section 3.

1.2. Non-smooth Local Optimization

There are many practically relevant mathematical models that involve non-smooth functions, i.e. continuous functions that have a discontinuous gradient. Specifically, numerous optimization problems arising in applications can be presented both in smooth and non-smooth form, and the non-smooth version often provides a more compact formulation of such problems. For example, the problem of unsupervised data classification (clustering) can be formulated as a smooth optimization problem that involves both integer and continuous variables: in such formulations the number of variables can be very large. The non-smooth optimization setting of the same problem can be given without integer variables, and thereby can reduce the number of variables significantly.

Within the broad class of non-smooth functions, the set of locally Lipschitz-continuous functions, and in particular the class of convex functions, is of special interest. The notion of subdifferential (see Rockafellar (1970)) is a generalization of the gradient for non-smooth convex functions. Different approaches to the generalization of this notion have been proposed subsequently: the Clarke subdifferential (see Clarke (1983)) and the quasidifferential (see Demyanov and Rubinov (1986, 1995)) are the most important among these from the numerical points of view

All non-smooth optimization methods use some sub-differential information, in each iteration step: a number of corresponding algorithms have been developed in the last forty years. These algorithms can be divided into two main groups: namely, monotonic and non-monotonic.

The so-called bundle method and its variants are arguably the best representatives of the class of descent methods in non-smooth optimization. This approach was first proposed by Lemarechal (1975, 1978), Wolfe (1976), and Mifflin (1976), for solving non-smooth convex problems. In order to calculate descent directions, this method uses interior approximations of the ε -sub-differential of a convex function. This way, the problem of calculating a descent direction is reduced to a corresponding quadratic programming problem. The latter then can be solved by Wolfe's or Mifflin's algorithm.

Subsequently, variants of the bundle method have been developed to solve more general Lipschitz programming problems using an interior approximation of Goldstein's ε -sub-differential (Kiwiel, 1985) of locally Lipschitz-continuous functions.

The CIAO-GO solver system includes Bagirov’s method that is a derivative-free version of the bundle method.

1.3. Derivative-free Optimization

In the last decade, derivative-free or direct search methods have attracted increasing attention. These methods are especially useful, when Newton-type searches are inapplicable. They are effective tools for the minimization of a nonlinear function f with one or more of the following properties:

- The calculation of the model function(s) f is expensive or time consuming. For example, each value of f may be obtained by solving a costly numerical sub-problem, or by performing a sequence of laboratory experiments.
- The first (and higher order) partial derivatives of f cannot be calculated exactly.
- The values of f are ‘noisy’. For example, the calculated value of f may depend on a numerical discretization scheme, inaccurate data.

To minimize of an objective function f with the properties listed above, the best choice is often a direct search method that requires and uses only function values. Essentially the same argument applies, if some of the component constraint functions in g have similar features to those outlined above.

It should be emphasized that many well known (smooth convex) optimization techniques—including quasi-Newton methods, sequential quadratic programming, etc.—cannot be applied directly to the above considered type of problems. The main reasons are that these methods require gradient (or higher order) information regarding the objective function.

Heuristic direct search methods were first suggested several decades ago. One of these classical approaches is the simple Hooke-Jeeves (1961) sequential directional search algorithm. Another search approach, the so-called simplex method, was proposed by Nelder and Mead (1965). (Incidentally, this simplex algorithm has nothing to do with Dantzig’s method for linear programming.) Another, more sophisticated class of direct search methods is based on the (overall) quadratic interpolation of the objective function (see Conn and Toint (1995) and Powell (2002)).

Following this introduction, the main body of this User Guide describes the CIAO-GO software system, and its usage in details. This includes the discussion of the following aspects:

- Mathematical model form (scope of applicability)
- Algorithmic solution options, solver modules
- Application program structure
- Hardware and software platforms, portability
- Software installation
- Application development steps: model formulation, solution and result analysis

- User input files and CIAO-GO output
- Program versions, connectivity to other software platforms
- Several potential application areas.

The appendices present additional information, including sample CIAO-GO input and output files, and notes on various implementations.

2. Problem Statement and Structural Assumptions

In this section, we will add some structural details and assumptions to the concise model description of model (1.1).

We shall consider the following general global optimization problem (GOP). Given a bounded, robust set $D \subset R^n$ (i.e., D is the closure of a non-empty open set, and it is a proper subset of the Euclidean n -space) that represents all feasible solutions. Furthermore, we have a (by assumption) continuous real-valued objective function $f: D \rightarrow R^l$. Our objective is to analyze and solve the GOP

$$(2.1) \quad \min f(x) \quad \text{subject to} \quad x \in D.$$

By the classical theorem of Weierstrass, the set of global solutions to the model (2.1) is non-empty. Let X^* be the global solution set; for reasons of algorithmic tractability, it will be assumed that X^* is finite.

Introducing the notation $f^* = f(x^*)$, our objective is verbally described by

$$(2.2) \quad \text{'Find all elements of } X^*, \text{ and the function value } f^* \text{'}$$

In numerous cases, it is very difficult—if not impossible—to solve model (2.1) in the exact sense of (2.2). Consequently, diverse numerical approximations of the requirement (2.2) need to be used. For instance, one may attempt to find at least one $x \in D$ such that the distance between x and X^* is ‘acceptably small’, i.e.

$$(2.3) \quad \min_{x^* \in X^*} \|x - x^*\| \leq \delta.$$

In (2.3) x^* is a suitable element of X^* , $\|\cdot\|$ is the Euclidean norm in R^n , and $\delta > 0$ is a given tolerance parameter.

An alternative numerical solution requirement is expressed by the following: find an $x \in D$ such that

$$(2.4) \quad f(x) \leq f^* + \varepsilon.$$

In (2.4) $\varepsilon > 0$ is another given tolerance parameter.

To guarantee the solvability of (2.1) in the sense of (2.4)—on the basis of a suitable finite collection of sample points from D —the Lipschitz-continuity of the objective function f is frequently postulated, whenever appropriate. That is, we shall assume the validity of the relation

$$(2.5) \quad |f(x_1) - f(x_2)| \leq L \|x_1 - x_2\| \quad \text{for all pairs } x_1, x_2 \in D.$$

In (2.5), $L=L(f;D)$ is the—typically unknown—Lipschitz constant of f on the set D . Inequality (2.5) guarantees that the possible variations of function f are uniformly ‘controlled’ by the respective changes of its argument. (The Lipschitz assumption is met, for instance, by all continuously differentiable functions defined over the compact set D .)

In many practical applications, the feasible set D is given by explicit, finite lower and upper bounds a and b on $x \in R^n$, and a finite number of additional Lipschitz-continuous constraints:

$$(2.6) \quad D = \{a \leq x \leq b: g_i(x) \leq 0 \quad i=1, \dots, m\}.$$

Problem (2.1)—with the added specifications (2.5) and (2.6)—is still very general: in fact, it subsumes most continuous GOP’s of practical significance. For instance, the illustrative models in Figures 1 and 2 (defined by respective composite trigonometric objective functions f_i in finite intervals $D_i \subset R^n$ for $n=1,2$) obviously belong to the scope of the GOP discussed here. Specifically, the functions f_i are Lipschitz-continuous.

As a direct consequence of this generality, (2.1) can be a very difficult numerical problem that represents a challenge in any computational environment of today (as well as of tomorrow). For detailed surveys of global optimization approaches under Lipschitz-continuity assumptions, consult Hansen and Jaumard (1995), or Pintér (1996a.)

Note furthermore that even far more special GOP instances—such as, e.g., the general (indefinite) quadratic programming problem—are known to be NP-hard: consult the surveys of Benson (1995) and Vavasis (1995), with additional related references therein.

The program system described below can be applied to solve a very broad class of optimization problems, including Lipschitz continuous global optimization, non-smooth optimization and even ‘black-box’ applications (in which the model functions are evaluated by implicit computational procedures).

3. The CIAO-GO Program System

In this main section, we present a concise, but more specific discussion of the algorithmic approaches embedded in the CIAO-GO software system. During this discussion, some added specific notation will also be used, as needed. Further details (including formal proofs) can be found in the topical references provided.

3.1. Cutting Angle Method

The cutting angle method can be used for the global minimization of a Lipschitz function defined on the unit simplex S :

$$S = \left\{ x \in \mathbb{R}_+^n : \sum_{i=1}^n x_i = 1 \right\}.$$

This method can be considered as a generalization of the well-known cutting plane method in convex minimization. The cutting angle method was introduced in Andramonov, Rubinov, Glover (1999). We use a version of this method presented in Bagirov and Rubinov (2001). See also Rubinov(2000), Batten and Beliakov (2002), Bagirov and Rubinov (2003) and references therein.

For a given Lipschitz function f the cutting angle constructs a sequence of underestimates: $h_m, m > n$ of f , where

$$h_m(x) = \left\{ \max_{k \leq n} l_k^k x_k, \max_{k > n} \min_{i \in I(x^k)} l_i^k x_i \right\}.$$

Here l_1^1, \dots, l_n^n are positive numbers and l^{n+1}, \dots, l^m are positive vectors.

The sequence $\{h_m\}$ uniformly converges to the function f and a global minimizer of h_m can be considered as an approximation of a global minimizer of a function f . A set of local minimizers of h_m can be explicitly described and then a global minimizer can be found by sorting out local minimizers. The cutting angle method is fast enough in small dimensions. Various types of combinations of the cutting angle and a local search are useful in applications. In particular the cutting angle method can be used in order to escape from a stationary point that is found by a local search. The cutting angle method can also be used for a search of a descent direction at a given point. We use a hybrid of the Bagirov's method and the Cutting Angle method in CIAO-GO.

3.2 Bagirov's Method

The Bagirov's method is a version of the bundle method (see Hiriart-Urruty and Lemarechal, 1993; Kiwiel, 1985) where discrete gradients are used instead of subgradients. It is defined with respect to a given direction and only values of the function f

considered are used for its computation. In this case an auxiliary problem for finding descent directions is solved by a very effective classical algorithm by Wolfe (1976).

Some versions of these method were introduced and studied by Bagirov (1992, 1995, 2002): it is proved that the limit of the set of discrete gradients is a subset of the Clarke sub-differential of semi-smooth functions. The limit of the set of discrete gradients coincides with the sub-differential of proper convex functions.

The use of discrete gradients allows one to construct continuous approximations to the Clarke sub-differential of semi-smooth functions. It is also proved the set of discrete gradients can be used in a continuous approximation of the Demyanov-Rubinov quasi-differential of functions presented as the smooth composition of Clarke regular semi-smooth functions (Bagirov, 1998, 1999).

The use of the discrete gradients allows one to propose an algorithm for the computation of descent directions of locally Lipschitz continuous functions.

3.3. Solver Options

CIAO-GO serves to find the global solution of the GOP stated by (2.1)-(2.5)-(2.6). Note that this problem is often considered in its simplest, box-constrained form: introducing

$$D_0 = \{a \leq x \leq b\}, \quad \text{find } \min f(x) \text{ subject to } x \in D_0.$$

The more general GOP, which includes also the constraint set (2.6), can be approximated in the form (3.1), following a penalty transformation based incorporation of the constraint functions into the objective function. The penalty function approach in nonlinear optimization was proposed by Fiacco and McCormick, (1968); with subsequent studies by many others: consult, e.g., the survey of Fletcher (1983). In other words—as one of the possible approximations—the objective function $f(x)$ can be augmented by terms of the form $c_i \max [g_i(x), 0]$ for inequality constraints and by terms of the form $r_j |g_j(x)|$ for equality constraints. The corresponding—still Lipschitz model formulation

$$(3.2) \quad \min f(x) + \sum_i c_i \max[0, g_i(x)] + \sum_j r_j |g_j(x)| \quad \text{subject to} \quad x \in D_0$$

effectively replaces the constraint set (2.6) by D_0 . This function is non-smooth.

The aggregated function (3.2) attempts to balance the aims of reducing the original objective function and of satisfying all implicit constraints. The penalty multipliers c_i associated with the constraints g_i $i=1, \dots, I$ can be adaptively selected, to enforce the (approximate) feasibility and optimality of the solution found.

Another solution approach to handle the constraint functions g_i is discussed in Rubinov et. al. (2003) where the problem (2.1), (2.6) is reduced to the following unconstrained optimization problem:

$$\min(f(x) + c)^k + dg^+(x), \quad g^+(x) = \max\{0, g_i(x), i = 1, \dots, I\}, c > 0, d > 0, 0 < k < 1.$$

In summary, CIAO-GO combines derivative-free global and local optimization strategies in an integrated manner. The derivative-free approach is of particular relevance with respect to ‘black box’ type applications as discussed earlier.

The current CIAO-GO implementation incorporates the following solver modules:

- Bagirov’s method (BM)
- Bagirov-Rubinov’s method (BRM) --Hybrid of the cutting angle and Bagirov’s methods

The local search phase (LS) is based on Bagirov’s algorithm.

Note also that the global solver options all sample the values of an aggregated merit (exact penalty) function as outlined above. The aggregation of constraints and objective function is automatically supported by CIAO-GO.

The fully automatic solver mode is governed by an external input parameter file. This file (prepared or adapted by the user) will be discussed later on.

Additional search strategies and solver options are also considered for future inclusion.

3.4. Application Program Structure

CIAO-GO has been originally developed in professional Fortran environments: specifically, using Lahey Fortran (LF) 90, and 95; and then it was ported also to other environments. For recent LF documentation, see Lahey Computer Systems (2002a, b). LF90 and LF95 directly support links to other programming languages under Windows such as for instance, Borland C++ and Microsoft Visual C++.

CIAO-GO can also be used on workstations and Unix/Linux platforms; more generally, it runs on all platforms for which a professional Fortran compiler is available. (Digital/Compaq Visual Fortran, g77, Salford Fortran 77 are further examples of compatible platforms.)

CIAO-GO can be delivered as an object file, as an executable program, or as a dynamic link library (dll). There are secondary—and somewhat compiler-dependent—differences among these versions, and we will not discuss all details and versions here. The exposition below will cover mainly the object file based version. The changes between versions are straightforward, and the core CIAO-GO solver functionality is identical in all cases.

CIAO-GO is optionally delivered within the framework of an integrated development environment (IDE). The features of the CIAO-GO IDE will be discussed in a separate

section; in the other (remaining) sections of this Guide we will concentrate on key features available in all implementation versions.

To establish a standard application program structure, the following template files—reflecting a Fortran development environment—are provided to the user:

- **MAIN.FOR** serves essentially provide a shell for calling CIAO-GO. It may also invoke additional user actions (such as, e.g., provision of links to other program files, external applications, and/or to report generation and further use of the results).
- **USERFCT.FOR** serves to define the model (objective and constraint) functions that describe the optimization problem. Again, this file may include calls to other application programs, in order to evaluate the model functions.
- **CIAO-GO.IN** is the input parameter file; the corresponding template also includes a brief description of each parameter.

The user source code files mentioned above are to be linked to the CIAO-GO object file system: this step obviously needs one of the compatible compiler options mentioned above. CIAO-GO.OBJ integrates the solvers and all other modules, which are needed to operate the solver variants and to report results.

During runtime, CIAO-GO reads the file CIAO-GO.IN that controls its operations. This structure facilitates repeated executions of CIAO-GO under various model specifications and/or solver parameterizations.

During its execution, the solver system iteratively calls the model function file, in order to find the best solution. Upon program termination, two result (output) files are automatically generated.

One of these files, CIAO-GO.SUM, presents the summary results. It is most useful in ‘routine’ CIAO-GO applications, since this way one can avoid browsing through (a possibly large amount of) runtime details. The second text file, called CIAO-GO.OUT,

provides a more detailed (logbook style) information related to the complete optimization process. Therefore its analysis can be very useful, e.g., during the gradual development of a new application.

The interdependence structure of these essential program components is shown below, see Figure 3.

CIAO-GO.IN
↓

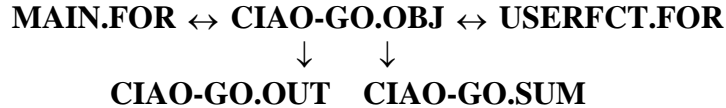


Figure 3. CIAO-GO application program: basic structure

Note that both MAIN.FOR and USERFCT.FOR can be provided as compiled object files, optionally in several languages. Furthermore, they can also be connected to additional program (source code, object code, or dll) files, or even to executables. In such cases, CIAO-GO will perform generic ‘black box’ optimization, assuming only that proper input/output communication is established between CIAO-GO and the attached user files.

3.5. Connectivity to Other Modeling Environments

Let us note at this point that CIAO-GO can be directly connected to various modeling environments. This enables model formulations given in those environments, typically even in the absence of a corresponding ‘low level’ programming language platform.

Upon request, CIAO-GO can be made available as a customized solver option in various model development environments. In the simplest case, this needs only a launching call from such environments, and then proper iterative communication between the modeling platform (i.e., the model setup) and the solver. Optimization related modeling systems are typically equipped not only with a tailored interpreter capability, but also with an extensive range of other features (such as model development, error checking, visualization, report generation, full project documentation, and so on).

Several prominent scientific / technical / business modeling environments are

- EXCEL Solver (Frontline Systems, 1999)
- MAPLE (Waterloo Maple / Maplesoft, 2002)
- MATHEMATICA (Wolfram, 1999)
- MATLAB (MathWorks, 2000)

From the list of modeling environments focused on optimization one can mention, for instance

- AIMMS (Paragon Decision Technology, 1995)
- AMPL (Fourer, Gay and Kernighan, 1993)
- EASY-FIT (Schittkowski, 1996)
- GAMS (Brooke, Kendrick and Meeraus, 1988)
- LINDO Solver Suite (LINDO Systems, 1997)
- MPL (Maximal Software, 2002)

Obviously, any corresponding development needs more or less significant amount of additional work. As an example, for a discussion of connecting solver options to AMPL, consult Gay (1997).

3.6. Problem Size and Complexity Considerations

For reasons of keeping the program storage (and most likely also the corresponding program execution time) requirements within reasonable bounds, version-specific limitations are imposed in relation to the number of decision variables and constraints, and to that of the simultaneously stored vectors (in the cutting angle solver mode). Of course, these or similar limitations can be easily resolved (by resetting them within the CIAO-GO program system), but the expected memory requirements and runtimes will correspondingly increase.

The currently delivered (largest) standard versions of CIAO-GO enable the handling of models with up to a few thousand variables and constraints. We have solved such model instances in a few minutes on today's Pentium (III or higher) processor based personal computer, equipped with at least 256 Megabytes of RAM. For small problems, the solution time is often just a few seconds or less. Longer runtimes can be expected for more complex, numerically intensive models.

4. System Requirements, Portability and Installation

As mentioned earlier, the core CIAO-GO solver system is written in Fortran, and can be used without any significant changes on professional level Fortran 77 / 90 / 95 platforms. The standard PC version of CIAO-GO has been developed making use of the Lahey Fortran (LF 90 and LF95) implementations. The CIAO-GO IDE version for MS Windows features a simple-to-use menu system, with built-in user support features (help, external system call) that will be discussed subsequently. This version can also be used in conjunction with several other major language platforms (such as e.g. Borland C/C++ and MS Visual C/C++).

For distinction, the CIAO-GO solver engine with a text I/O interface, but without the added IDE features, will be optionally referred to as the 'silent mode' implementation. This version is directly available for all professional Fortran platforms.

4.1. Hardware

In order to run CIAO-GO with appropriate efficiency, the following (minimal) hardware configuration is recommended:

- Intel Pentium processor based PC.
- 256 Megabytes of RAM; naturally, more RAM is better, especially when large size configuration CIAO-GO software versions (discussed later) are used.
- 100 Megabytes available hard drive space. Note that the CIAO-GO program system itself takes up much less space than that; however, several large arrays may be generated during runtime.
- SVGA monitor (or, in fact, any monitor which supports at least 640x480 resolution). Note that the current standard CIAO-GO IDE shipment version (interface) is targeted for 800x600 resolution and 256 colors. (Other video modes can also be supported, of course, as requested by user.

In workstation environments, there is no need to explicitly set similar requirements, since the available hardware is typically more than adequate to handle the CIAO-GO system, in reasonably sized configurations.

4.2. Software

- Appropriate operating system: at the time of this writing, CIAO-GO has been installed and tested on personal computers operated under currently used Windows systems. (For instance, MS Windows 2000 or XP Professional platforms are recommended.)
- Compatibility with all currently used Microsoft PC operating systems will be guaranteed; additional development may also take place.
- Professional C or Fortran development platform—for instance, Lahey Fortran, Digital or Compaq Visual Fortran, g77, Salford FTN77—or one of the LF compatible platforms. At present, the latter include Borland C/C++ and Microsoft Visual C/C++.

Note that a standard professional Fortran 77/90/95 environment available on most Unix/Linux workstations and operating systems provides a fully adequate platform for

using CIAO-GO (in ‘silent mode’). The graphics and some other interactive capabilities, however, need to be properly adjusted (replaced or developed), since they are platform (software and/or hardware) dependent.

4.3. Installation

The installation procedure is simple and straightforward: below we shall outline it for a PC environment. Assuming that the (Fortran or other compatible) development platform is on the path of the PC in question, the CIAO-GO file system can be placed—e.g, copied from the master CD or diskette(s)—into any directory of the user’s choice. For instance, C:\CIAO-GO can be created and used. For concreteness, in the subsequent discussion, we shall suppose that CIAO-GO is installed under C:\CIAO-GO, and that the actual development work also takes place in this directory, or in its suitable subdirectories (see note below).

Note also at this point that—making use of standard Windows tools—the CIAO-GO integrated development environment can be directly launched from the Start menu and/or from the desktop screen, by creating a shortcut to the CIAO-GO shell. (This program system shell is named CIAO-GO_IDE.EXE, and it will be described in the next section).

Let us mention (as a simple side-note for novice users) that it is not recommended to place the CIAO-GO system directly into the Fortran or C executable subdirectory, since, in general, it is not too salient practice to ‘dump’ there application systems. For similar reasons, we suggest to keep CIAO-GO applications elsewhere than C:\CIAO-GO. For instance, they could be placed into corresponding directories under C:\CIAO-GO, or into any other directory which suits the CIAO-GO user. The current application can be always kept in C:\CIAO-GO for development, then moved or copied (for backup and archivation purposes) into its own directory.

5. The CIAO-GO Integrated Development Environment

The following description (in Sections 5 and 6) refers primarily to the CIAO-GO IDE under Windows. Users of the 'silent version' may skip these sections, if they wish, although the recommended use of CIAO-GO should typically follow similar guidelines.

Note here that the menu-driven user interface (CIAO-GO IDE executable program shell) can be made readily available for all current MS Windows operating systems, as a standard Windows application. Similar menu development for other platforms should be fairly straightforward, and can be customized depending on user demands.

5.1. Summary of Menu Options

Upon launching the CIAO-GO program system (from the desktop, from the taskbar, or by any other means activating C:\CIAO-GO\CIAO-GO_IDE.EXE) the application development menu frame is displayed. The CIAO-GO front page—including license information—also appears briefly. The main menu makes available the following options (see Figure 4, the corresponding submenu items are also shown):

<u>M</u>odel <u>F</u>ormulation	<u>M</u>odel <u>S</u>olution	<u>R</u>esult <u>A</u>nalysis	<u>H</u>elp	<u>E</u>xit
Project File Names	Input Parameter File	Summary Output File	Inf. Summary	Quit
User Main File	Run Model	Detailed Output File	Development Steps	
User Function File		External System Call	About CIAO-GO	
Compile and Link Model				
External System Call				

Figure 4. The CIAO-GO IDE application development menu

Note that—following standard Windows conventions—all main menu options shown can be invoked by pressing Alt and the (case-insensitive) corresponding underlined key combination. (For instance, Alt-h opens up the Help submenu.) On completion of each chosen action, the CIAO-GO IDE main menu window is displayed again.

Below we shall briefly describe the functionality of each menu option.

5.2. Model Formulation

The Project File Names option prompts the user to provide names for the following program items:

- Project (i.e., the current CIAO-GO project, fully described by the files named below)
- User Main File (provides main application program frame)
- User Function File (provides model description)
- Project Executable File (combines the user files defined above with the CIAO-GO system, and subsequently generates CIAO-GO run results)

- Input Parameter File (used by the project executable file)
- Summary Output File (provides concise information regarding run results)
- Detailed Output (Log) File (provides more detailed information regarding run results).

Default file names are given in the menu: these correspond to a test problem file system provided to CIAO-GO users. The defaults can (and should), of course, be overwritten as per actual user application needs.

The User Main File option invokes the application frame file (with the name chosen in the previous menu option). In standard CIAO-GO shipments, the simple Notepad program (a Windows accessory) can be activated to display and manipulate this file, as well as all other CIAO-GO text files discussed later on. (If CIAO-GO users wish to use a different text editor, then that can also be arranged, upon request.)

The application program frame file—and all other file templates provided—are commented in sufficient detail, to assist their straightforward adjustment to actual user needs. In particular, the main file opens the user input/output files, and calls the CIAO-GO solver system. Optional, application-dependent user actions can be also launched here, before and/or after executing the CIAO-GO run. (See Appendix 1 for an example file and additional details.)

The User Function File option serves to open a file, in order to describe the model. The objective function and the constraints are defined here, following again a commented template. (See Appendix 2 for an example file and additional details.)

The Compile and Link Model option generates object files on the basis of the user source files discussed above, then links these to the CIAO-GO object file system and libraries. The actual compilation depends on the development environment used. LF90 or 95 compilation options are provided in the standard CIAO-GO shipment, but other compatible environments (listed above) can also be supported, as per user request.

The External System Call option allows the activation of other programs from within the CIAO-GO menu shell. Specifically, it opens a dialog in which the user can provide the name of any Windows-executable program available on the computer (assuming proper path and/or directory settings). If the program directory or name needs to be looked up, then the 'Browse' option (dialog button) opens Windows Explorer, to assist in this action. The purpose of external system call options is to enable the usage of additional tools (such as model libraries, text editors, other computational or visualization programs, etc.) thereby possibly assisting the model development work within the CIAO-GO system.

5.3. Model Solution

Following the actions described above, the CIAO-GO 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 template is provided to assist users, so that the preparation (or adjustment) of this file should be straightforward. In particular, the number of constraints and variables, their names, explicit lower/upper bounds and nominal values for the variables, and several key optimization procedure parameters are defined in this file. (See Appendix 3 for an example file and additional details.)

The Run CIAO-GO menu option launches the executable application program. CIAO-GO executables run in fully automatic operational mode.

5.4. Result Analysis

This main menu option is to be used following a CIAO-GO program run—includes three choices. The Summary Output File option displays a brief report of the results that has been automatically generated by CIAO-GO. (See Appendix 4 for an example file.) The Detailed Output File option displays another, more detailed ‘logbook’ text file generated by CIAO-GO. The External System Call option—analogously to the similar menu option described above—allows the activation of other programs from within the CIAO-GO shell.

Note that returning to CIAO-GO from this point—immediately after a completed run—will overwrite the currently generated output files. Therefore in sequentially executed runs with changing model, input parameters, or runtime operations it is advisable to save the results. This can be simply done by making use of a standard Windows dialog that can be enabled from the text editor used for I/O file management.

5.5. Help

Concise and detailed on-line help are directly available to users.

The Information Summary menu option serves to introduce novice users to the program system. The mathematical model form encompassed by CIAO-GO, main system features, hardware and operating system requirements, notes on installation and usage, connectivity to other development platforms, available CIAO-GO versions, and a list of existing applications are summarized in the corresponding help file.

The Development Steps menu option provides brief step-by-step instructions to users, regarding the essential stages of application development using the CIAO-GO IDE. (This description is closely related in spirit to the forthcoming discussion in Section 6.)

The About CIAO-GO option displays (again) the CIAO-GO front screen that contains license information.

5.6. Exit

Upon selecting the Exit option, all current project files are closed and saved; furthermore all temporary files are deleted. The CIAO-GO IDE program shell itself is closed.

Summing up this concise review of the CIAO-GO IDE menu options, it can be seen that both brief and detailed background information—related to the subject of global optimization, to CIAO-GO, and to the model structure required—is permanently available. In addition, the essential stages of user application (code) development—editing, compilation, linking, execution, visual and text result analysis—can be directly activated from the CIAO-GO menu.

The templates of all user source code files are provided in the form of a sample problem (or problems) and corresponding input/output file system(s). Subsequently, these files are to be modified by users, in order to build new models. The CIAO-GO menu environment supports rapid prototyping, making the application development process faster and easier.

For novice users, it is highly recommended to spend some time (possibly just a few minutes) on browsing through the help options, and the user file templates provided. This will help to gain fast and sufficient familiarity with the essentials of using CIAO-GO.

6. Application Development Steps

In this section we shall briefly discuss the principal stages of model development within the CIAO-GO development environment. Note again that the development stages outlined below should, in essence, be followed also in 'silent' (text file I/O based) application development.

6.1. Model Formulation

In order to apply CIAO-GO, the user first has to define the optimization problem, according to standard specifications. This consists of the following steps.

- Preparation of User Main File

The main program file can be simply an 'empty' program shell to activate CIAO-GO. Additional user actions—activated before and/or after applying CIAO-GO—can also be included here: these may be related, e.g., to calling external programs. In addition, in the CIAO-GO IDE version the input and output files are also opened in the user main program file (with optional filenames).

In the 'silent' program version, the I/O filenames are fixed as CIAO-GO.IN, CIAO-GO.OUT and CIAO-GO.SUM; therefore these are manipulated internally, without user interaction.

- Preparation of User Function File

The user function file defines the objective and constraint functions of the optimization model. Note that—for enhanced numerical accuracy—all user file real variables should be defined applying double precision, as indicated by the file templates provided.

Sample files—corresponding to both MAIN.FOR and USERFCT.FOR in Figure 3—are provided: for examples, please see Appendices 1 and 2. These sample files can assist users in setting up their own applications easily. For instance, a new project can be defined by copying the content of the existing template files, and then directly modifying them as needed to obtain (say) NEW_MAIN.FOR and NEW_FCT.FOR.

Following the necessary project (model) definitions, the next step is its combination with the CIAO-GO object file system and associated libraries, within a compatible environment. Again, the compilation steps are somewhat specific to the CIAO-GO structure and the compiler available to the user. However, the essentials are the same: the (optional) main program and the (obligatory) model function file(s) is/are combined with the CIAO-GO solver file to create an application-specific executable program file system.

- **Compilation and Linking**

If all compilation and linking operations are successful, then after their completion control is returned to the CIAO-GO IDE main menu level.

In 'silent mode', a short DOS style message may appear on the screen, and the CIAO-GO application run is started. (This message may be simply suppressed in the compile/link batch file provided with 'silent mode' software deliveries.)

In case of compilation and/or linking errors, corresponding messages appear on the screen (in a DOS 'box' under Windows), and the CIAO-GO IDE also indicates the errors. A message prompts the user to return to editing mode. Syntax and linking errors need to be corrected as this point, before proceeding further.

- **External System Call**

In the CIAO-GO IDE, this option makes available callable external applications. The user either directly calls some (executable) application program or Windows Explorer can be activated at this point, to select and run an application. This option can be useful, to gain additional information, and/or to assist in the model formulation.

In 'silent mode', such external calls are not part of the CIAO-GO system; they are handled separately, if needed (using standard Windows tools such as Explorer).

6.2. Model Solution

Given a properly compiled and linked CIAO-GO application, the next step is program execution. To run the generated model, a corresponding input parameter file is needed.

- **Input Parameter File**

Again, a commented sample is provided for the input parameter file that defines the key model information (size, variable and function names, variable bounds and nominal values) as well as some program execution options. For an example, please consult Appendix 3.

- **Program Execution**

CIAO-GO executable files operate in a fairly obvious and self-explanatory manner, since (in the CIAO-GO IDE) all user input is prompted by pop-up dialogs.

In the IDE framework, during program execution, basic information related to algorithmic progress, and—if needed—runtime troubleshooting and error messages are communicated to the user. Such information can be provided in the form of corresponding messages to the screen during execution, and by generating the two output files referred to earlier.

The CIAO-GO IDE solver system can be activated in several modes (automatic global and local search modes, as well as local search mode only). The search procedure selection is fully controlled by the input parameter file. During the corresponding CIAO-GO run, only short messages—iteration count and improving results obtained by the sequentially invoked solver modules—may appear on the screen. The local search (only) mode is launched from the nominal solution provided by the user in the input parameter file.

Upon terminating the run, a screen output provides summary information. Specifically, solution feasibility, optimum estimate, the total number of (objective and constraint) function evaluations, and the total program runtime are reported.

The 'silent mode' runs are fully automatic; there is no communication with the user during runtime (that's why such versions are called 'silent'). Upon completing the run, the user has access to the two (OUT and SUM) text result files referred to earlier: these can be inspected by any simple text editor (such as e.g., the Windows accessory Notepad).

Note again that repeated runs can be immediately launched, without leaving the CIAO-GO IDE (or by restarting the application-specific executable); the result text files will be overwritten using this option. (Thus one should save them, unless restart is done on purpose, using a modified input file(s).)

After the optimization run is terminated (successfully, or possibly with runtime error messages due to model flaws), control is returned again to the applications menu level. Accordingly, repeated editing, compilation and linking, and/or result analysis may follow.

6.3. Result Analysis

The solution procedure is typically followed by the inspection of the result files, automatically generated by CIAO-GO. This includes the following options:

- View Summary Results
- View Detailed Results
- External System Call

As mentioned already, the summary file provides information (only) regarding the final results obtained. For an example, consult Appendix 4. The detailed output (runtime log) file contains additional information. Specifically, it verifies (echoes) the input parameters, and traces the optimization process, reporting the solution option(s) used, the improving sequence of solutions generated, and the stopping criteria met during execution.

The usage of external system calls was highlighted above, see section 6.1.

Again, in the 'silent mode' delivery, the result files should be opened in a simple text editor, and all external system calls should be handled outside of CIAO-GO (as needed).

Obviously, the result files may have different importance at different stages of model development. The summary file is most useful in 'routine' CIAO-GO runs, while the additional information provided by the detailed output file can assist both model development and the choice of CIAO-GO solver parameter settings.

Note also at this point that the results obtained may be directly passed over to other programs. These external programs can be launched from the user programs (the MAIN and USERFCT components) of CIAO-GO, but can also be started independently of them. After the analysis of results is completed, control is returned again to the CIAO-GO IDE applications menu level. In a typical model development process, the stages outlined are applied in an iterative fashion.

Note finally that all I/O files prepared or analyzed by the user should be handled (generated, modified, or simply kept) as a text file. Users should always retain a copy of the sample files for further reference.

7. Model Development Tips

From the previous discussion (recall especially Section 2) it is clear that the general problem-class encompassed by the global optimization model contains also very complicated problem instances. It is not difficult to construct GO problems in just, say, ten variables that would most probably pose a serious challenge to the available repertoire of numerical algorithms (again, recall Figures 1 and 2). Fortunately, in many cases, practical problems are less ‘tricky’, than such mathematical ‘constructions’, but certainly there exist numerous exceptionally hard—while still practically relevant—GO problems.

The inherent complexity of continuous or Lipschitz problems is unavoidable: therefore a few comments on suggested model formulation and solution strategies are in order. These notes should serve the software user, to obtain meaningful results in an efficient manner.

7.1. Simplicity and Modular Development

It is always good practice to start with relatively small problem instances that are often simpler to analyze and understand. This suggestion refers to possibly ‘minimal’ problem dimensionality, (initially) simplified analytic description, and to modular application program development. By applying a well planned, incremental model building strategy, a tremendous amount of often almost completely useless, ‘blindly automated’ computational work can be avoided.

In particular, it is most advisable to evaluate all model (objective and constraint function) values at least at a few reasonably chosen arguments beforehand, to verify that the corresponding model components indeed work properly. To find out numerical flaws in the problem definition from runtime error messages—or from ‘very surprising’ results—is not impossible, but it is certainly more complicated.

7.2. Decomposition

Problem decomposition can be of great assistance in many areas of mathematical programming—especially so in nonlinear / global optimization. Let us illustrate this point in simple terms: to solve fifty individual 20-variable problems—even when their solution needs to be repeated, say, 10 times—could be simpler, more reliable and faster, than to solve a corresponding aggregated (and complex) GOP in 1000 variables. This, of course, does not imply that decomposition should be applied at all cost, especially when it leads to ‘forcing’ less realistic modeling. However, if there are clearly more and less important decision variables, then it may be well justifiable—from a practical point of view—to optimize first the primary variables, and then turn to the determination of the secondary ones. Such procedures may lead to perhaps somewhat sub-optimal results, but will attain those results fast, or at least in an acceptable time-frame. A partially heuristic trial and error procedure can be better, than ‘blind’ optimization that often leads to a waste of computational effort, and to disappointing results.

7.3. Model Scaling

It is always good practice to formulate an optimization model in a ‘well-scaled’ manner. As a simple rule of thumb, such prior scaling should guarantee—if at all possible—that all numerical values used in the model (function arguments and output) fall, say, between -1000 and 1000 (or even more ideally, between 0 and 1). The analysis of the detailed output file can also assist in a proper scaling of the model: this, in turn, often has a remarkably positive effect on the numerical solution process.

7.4. Search Domain

Because of the inherent exponential complexity of GOPs, a reasonable choice of the search region (i.e., of the embedding ‘box’ $[a,b]$) is very important. Again, let us mention a simple example: if the range (stated initial uncertainty) of just 20 decision variables can be reduced by a factor of 0.5 each, then the overall search region size has been effectively reduced by a factor greater than one million...

Of course, such domain reduction should be practiced with caution, to avoid problem misrepresentation, and the unjustifiable exclusion of valid search regions from consideration. The possibilities of intelligent search domain reduction may largely vary, depending on the actual problem, and the available application-specific expertise.

However—especially in difficult high-dimensional models—a reasonable effort spent on defining the search region ‘as precisely as possible’ is a worthy exercise. In such problems, one can also consider the execution of several (interactive or automatic) runs, on gradually decreasing search regions.

7.5. Constraint Handling

Non-convex constraints (especially in the general framework presented) can be very complicated. Since the objective function f is often already ‘difficult’, and global search on a box domain is more straightforward to implement than on complicated sets, it is typically reasonable to keep the search area as simple as possible.

For instance, if there is a rather complicated constraint, then it may be a good idea to add this (as a penalty term) to the objective function—when it is possible (justifiable) to do so. Note that the penalty approach is applied also automatically by CIAO-GO, but a conscious modeling effort may assist the search, since mathematically equivalent model forms may be more or less easy to solve.

The solution of complicated optimization problems will typically benefit from intelligent preliminary analysis, in order to find a model form amenable to solution procedures. As a final note on good modeling practice, active contact with the users of the model and results is always important. Mathematically—that is, formally—correct results derived from an apparently wrong model are of little use. After all, the practical objective of optimization is to assist actual decision-making.

8. CIAO-GO Program Versions

8.1. Professional, Research and Educational Versions

CIAO-GO is available in educational, research and professional versions. The core solver capabilities of these versions are identical; only the problem sizes solvable are different.

The standard educational program version is limited (at present) to handling models with 20 variables and 20 constraints. Note that although this may sound rather small, GO models of such size can already be truly difficult, and that most textbook problems in nonlinear/global optimization are of even smaller size.

The CIAO-GO (non-profit) research version is typically limited to 1000 variables and 1000 constraints. Such versions may assist in solving truly complex, significant optimization problems.

The professional versions can be configured and adjusted to individual demands. Practical solver limitations are caused only by the available hardware and the ‘acceptable’ program execution time. (Recall the inherent exponential complexity of GO problems.)

Educational or research versions are typically licensed to university departments, research groups, or individual researchers for non-profit use. In case of educational group licenses, technical support is provided to a designated contact person representing the licensee group.

Research and professional CIAO-GO version licensees are fully supported, also on an individual basis. Support requests should be directed to the Centre of Informatics and Applied Optimization, University of Ballarat (preferably by e-mail, providing also code information when necessary).

8.2. Demonstration Programs

Several demonstration program versions are readily available: these are typically self-running executable files. Some of these solve a sequence of randomly generated illustrative low-dimensional—yet visibly non-trivial—test problems, or a specific application. These programs demonstrate the full (runtime) functionality of CIAO-GO.

New demonstration programs can also be provided upon request, in order to test the suitability of CIAO-GO to handle particular problem-types.

9. An Illustrative List of CIAO-GO Applications

Decision problems that require the use of advanced GO solution approaches are prevalent in applications described by nonlinear system models.

The list of existing and potential application areas includes, for instance, the following broad problem-classes (illustrative references that reflect our related joint expertise are also provided):

- Computational physics and chemistry, potential energy models (Pintér, 2000, 2001b; Stortelder, de Swart, and Pintér, 2001)
- Continuous facility location (Pintér, 1996a)
- Data classification / cluster analysis / pattern recognition (Bagirov, Rubinov, Soukhoroukova, and Yearwood (2003); Bagirov, Rubinov, and Yearwood (2002); Pintér and Pesti, 1991; Pintér, 1996a)

References

- Andramonov, M.Y., Rubinov, A.M. and Glover, B.M. (1999) Cutting angle method in global optimization, *Applied Mathematics Letters*, vol. 12 , 95-100.
- Bachem, A., Grötschel, M. and Korte, B., Eds. (1983) *Mathematical Programming: The State of the Art*. Springer, Berlin.
- Bagirov, A.M. (1992) A method of approximating a sub-differential. *Russian Journal of Computational Mathematics and Mathematical Physics* 32(4), 561 - 566.
- Bagirov, A.M. and Gasanov, A.A. (1995) A method of approximating a quasi-differential. *Russian Journal of Computational Mathematics and Mathematical Physics* 35(4), 403-409.
- Bagirov, A.M. (1998) Continuous sub-differential approximation and its construction. *Indian Journal of Pure and Applied Mathematics* 1, 17-29.
- Bagirov, A.M. (1999) Minimization methods for one class of non-smooth functions and calculation of semi-equilibrium prices. In: *Progress in Optimization: Contributions from Australasia*. Eberhard, A. et al. (Eds.), *Applied Optimization*, Volume 30, Kluwer Academic Publishers, 147-175.
- Bagirov, A.M. (1999) Derivative-free methods for unconstrained non-smooth optimization and its numerical analysis. *Investigacao Operacional* 19, 75-93.
- Bagirov, A.M. (2000) Numerical methods for minimizing quasidifferentiable functions: a survey and comparison. In: *Quasidifferentiability and Related Topics*, Demyanov, V.F. and Rubinov, A.M. (Eds.), *Nonconvex Optimization and Its Applications*, Volume 43, 33-71, Kluwer Academic Publishers, Dordrecht.
- Bagirov, A.M., Rubinov, A.M. and Yearwood, J. (2002) A heuristic algorithm for feature selection based on optimisation techniques, In: *Heuristic and Optimization for Knowledge Discovery*, C. Newton, H. Abbas and R. Sarker (eds.), Idea Group Publishing, 13-26.
- Bagirov, A.M. (2002) A method for minimization of quasidifferentiable functions. *Optimization Methods and Software*, Vol. 17, No. 1, 31-60.
- Bagirov, A.M. (2003) Continuous subdifferential approximations and their applications. *Journal of Mathematical Sciences*, Vol. 115, Issue 5, 2567-2609.
- Bagirov, A.M. Rubinov, A.M. (2001) Modified versions of the cutting angle method. In: *Nonconvex Optimization and Its Applications. Vol. 54: Advances in Convex Analysis and Global Optimization*, N. Hadjisavvas and P.M. Pardalos (eds.), Kluwer Academic Publishers, Dordrecht, 245-268.
- Bagirov, A.M. and Rubinov, A.M. (2003) Cutting angle method and a local search, *Journal of Global Optimization*, vol. 27, 193-213.
- Bagirov, A.M., Rubinov, A.M. and J. Yearwood (2002) A global optimisation approach to classification. *Optimization and Engineering*, Vol. 3, No. 2, 129-155.
- Bagirov, A.M., Rubinov, A.M., Soukhoroukova, N.V. and Yearwood, J. (2003) Supervised and unsupervised data classification via nonsmooth and global optimization. *TOP: Spanish Operations Research Journal*, Vol. 11, No. 1, 1-93.
- Batten, L.M. and Beliakov, G. (2002) Fast algorithm for the cutting angle method of global optimization, *Journal of Global Optimization*, Vol. 24, 149-161.
- Benson, H.P. (1995) Concave minimization: Theory, applications and algorithms. In: Horst and Pardalos, Eds. (1995); pp. 43-148.

- Birge, J.R. and Murty, K.G., Eds. (1994) *Mathematical Programming: State of the Art*. University of Michigan Press, Ann Arbor, MI.
- Bomze, I.M., Csendes, T., Horst, R. and Pardalos, P.M., Eds. (1996) *Developments in Global Optimization*. Kluwer Academic Publishers, Dordrecht.
- Brooke, A., Kendrick, D. and Meeraus, A. (1988) *GAMS — A User's Guide*. The Scientific Press, Palo Alto, CA.
- Clarke, F.H. (1983) *Optimization and Non-smooth Analysis*, New York: John Wiley.
- Conn, A.R. and Toint, Ph.L. (1995) Algorithm using quadratic interpolation for unconstrained derivative-free optimisation. In G. di Pillo and F. Gianessi (Eds.) *Nonlinear Optimization and Applications*. Plenum Press, New York..
- Demyanov, V.F. and Rubinov, A.M. (1986) *Quasi-differential Calculus*. Optimization Software, Inc. New York.
- Demyanov, V.F. and Rubinov, A.M. (1995) *Constructive Non-smooth Analysis*. Peter Lang, Frankfurt am Main.
- Dixon, L.C.W. and Szegö, G.P., Eds. (1975, 1978) *Towards Global Optimisation*. Volumes 1-2. North-Holland, Amsterdam.
- Edgar, T.F., Himmelblau, D.M. and Lasdon, L.S. (2001) *Optimization of Chemical Processes*. (2nd Edn.) McGraw-Hill, New York.
- Fedorov, V.V., ed. (1985) *Problems of Cybernetics: Models and Methods in Global Optimization*. USSR Academy of Sciences, Moscow. (In Russian.)
- Fiacco, A.V. and McCormick, G.P. (1968) *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Wiley, New York.
- Fletcher, R. (1983) Penalty functions. In: Bachem, Grötschel, and Korte, Eds. (1983), pp. 87-114.
- Forgó, F. (1988) *Nonconvex Programming*. Akadémiai Kiadó (Academic Publishers, Hungarian Academy of Sciences), Budapest.
- Floudas, C.A. and Pardalos, P.M., Eds. (1992) *Recent Advances in Global Optimization*. Princeton University Press, Princeton, NJ.
- Floudas, C.A., Pardalos, P.M., Adjiman, C.S., Esposito, W.R., Gümüs, 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>.
- Fourer, R., Gay, D.M. and Kernighan, B.W. (1993) *AMPL — A Modelling Language for Mathematical Programming*. The Scientific Press, San Francisco, CA. (Reprinted by Boyd and Fraser, Danvers, MA, 1996.)
- Frontline Systems (1999) *Solver User's Guide*. Frontline Systems, Inc., Incline Village, NV.
- Frontline Systems and Pintér Consulting Services (2001) *LGO Global Solver Engine for Excel – Premium Solver Platform*. Frontline Systems, Inc., Incline Village, NV. See <http://www.solver.com/xlslgoeng.htm>
- Gay, D.M. (1997) Hooking your solver to AMPL. *Working Paper*, Bell Laboratories, Lucent Technologies, Murray Hill, NJ.

- Grossmann, I.E., ed. (1996) *Global Optimization in Engineering Design*. Kluwer Academic Publishers, Dordrecht.
- Hansen, E.R. (1992) *Global Optimization Using Interval Analysis*. Marcel Dekker, New York.
- Hansen, P. and Jaumard, B. (1995) Lipschitz optimization. In: Horst and Pardalos, Eds. (1995), pp. 407-493.
- Hillier, F.S. and Lieberman, G.J. (1986) *Introduction to Operations Research*. (4th Edn.) Holden Day, Oakland, CA.
- Hiriart-Urruty, J.B. and Lemarechal, C. (1993) *Convex Analysis and Minimization Algorithms*. Springer Verlag, Heidelberg, New York, Vol. 1 and 2.
- Hooke R. and Jeeves, T.A. (1961) Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8, 212-229.
- Horst, R. and Pardalos, P.M. Eds. (1995) *Handbook of Global Optimization, Volume 1*. Kluwer Academic Publishers, Dordrecht.
- Horst, R., Pardalos, P.M. and Thoai, N.V. (1995) *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht.
- Horst, R. and Tuy, H. (1996) *Global Optimization — Deterministic Approaches*. (3rd Edn.) Springer-Verlag, Berlin.
- Journal of Global Optimization* (published since 1991). Kluwer Academic Publishers, Dordrecht.
- Kearfott, R. B. (1996) *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht.
- Kiwiel, K.C. (1985) *Methods of Descent for Nondifferentiable Optimization, Lecture Notes in Mathematics*, Springer-Verlag, Berlin, Vol. 1133.
- Lahey Computer Systems (2002a) *Fortran 95 — User's Guide*. Lahey Computer Systems, Inc., Incline Village, NV.
- Lahey Computer Systems (2002b) *Fortran 95 — Language Reference*. Lahey Computer Systems, Inc., Incline Village, NV.
- Lemarechal, C. (1975) An extension of Davidon methods to nondifferentiable problems. In: M.L. Balinski and P. Wolfe (Eds.), *Nondifferentiable Optimization. Mathematical Programming Study 3*, North-Holland, Amsterdam, 95-109.
- Lemarechal, C. (1978) Bundle methods in non-smooth optimization. In: C. Lemarechal and R. Mifflin (Eds.), *Non-smooth Optimization*. Pergamon Press, Oxford, 78-102.
- Liebling, T.M. and de Werra, D., Eds. (1997) *Lectures on Mathematical Programming ISMP97*. Elsevier, Amsterdam.
- LINDO Systems (1997) *Solver Suite*. LINDO Systems, Inc. Chicago, IL.
- Locatelli, M. (2000) Simulated annealing algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and Applications*, Vol. 104, (1), 121-134.
- MathWorks (2000) *Using MATLAB*. The MathWorks, Inc. Natick, MA.
- Maximal Software (2002) *MPL Modeling System*. Maximal Software, Inc., Arlington, VA.
- Michalewicz, Z. (1986) *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin.
- Mifflin, R. (1976) An algorithm for constrained optimisation with semismooth functions. *Mathematics of Operations Research 2*, 191-207.

- Mittelmann, H.D. and Spelucci, P. (2003) *Decision Tree for Optimization Software*. <http://plato.la.asu.edu/guide.html>
- Mockus, J. (1989) *Bayesian Approach to Global Optimization*. Kluwer Academic Publishers, Dordrecht.
- Moré, J.J. and Wright, S.J. (1993) *Optimization Software Guide*. SIAM, Philadelphia.
- Nelder, J.A. and Mead, R. (1965) A simplex method for function minimization. *Computer Journal* 7, 1965, 308-313.
- Neumaier, A. (1990) *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge.
- Neumaier, A. (2003) *Global Optimization*. <http://www.mat.univie.ac.at/~neum/glopt.htm>.
- Paragon Decision Technology B.V. (1995) *AIMMS*. Haarlem.
- Pardalos, P.M. and Rosen, J.B. (1987) *Constrained Global Optimization: Algorithms and Applications*. Springer-Verlag, Berlin.
- 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. (1997) LGO — A program system for continuous and Lipschitz global optimization. In: Bomze, Csendes, Horst and Pardalos, Eds. (1997), pp. 183-197.
- Pintér, J.D. (2001) *Computational Global Optimization in Nonlinear Systems. An Interactive Tutorial*. Lionheart Publishing, Inc., Atlanta, GA. <http://www.lionhrtpub.com/books/globaloptimization.html>.
- Pintér, J.D. (2002a) Global optimization: software, test problems, and applications. In: Pardalos and Romeijn, Eds. (2002), pp. 515-569.
- Pintér, J.D. (2002b) *MathOptimizer – An Advanced Modeling and Optimization System for Mathematica Users. User Guide*. Pintér Consulting Services, Inc., Halifax, NS, Canada. <http://www.dal.ca/~jdpinter/>, and <http://www.wolfram.com/products/applications/mathoptimizer/>
- Pintér, J.D. (2003) *GAMS/LGO User Guide*. GAMS Development Corporation, Washington, DC. See <http://www.gams.com/solvers/LGO.pdf>
- Pintér, J.D. (2004a) *Applied Nonlinear Optimization in Modeling Environments*. CRC Press, Boca Raton, FL. (To appear.)
- Pintér, J.D., ed. (2004b) *Global Optimization: Selected Case Studies*. Kluwer Academic Publishers, Dordrecht. (To appear.)
- 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, Canada.
- Powell, M.J.D. (2002) UOBYQA: unconstrained optimisation by quadratic approximation. *Mathematical Programming Series B*, Vol. 92, No. 3, 555-582.
- Ratschek, H. and Rokne, J. (1988) *New Computer Methods for Global Optimization*. Ellis Horwood, Chichester.
- Rockafellar, R.T. (1970) *Convex Analysis*. Princeton University Press, Princeton, NJ.

- Rubinov, A. (2000) *Abstract Convexity and Global Optimization*. Kluwer Academic Publishers, Dordrecht.
- Rubinov, A.M., Yang, X.Q. and Bagirov, A.M. (2002) Penalty functions with a small penalty parameter. *Optimization Methods and Software*, Vol. 17, No. 5, 931-964.
- Schittkowski, K. (1996) *Parameter Estimation in Dynamical Systems with EASY-FIT. User's Guide*. Mathematical Institute, University of Bayreuth.
- Shor, N.Z. (1985) *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, Heidelberg.
- Strongin, R.G. (1978) *Numerical Methods for Multiextremal Problems*. Nauka, Moscow. (In Russian.)
- 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*. Kluwer Academic Publishers, Dordrecht.
- Törn, A.A. and Zilinskas, A. (1989) *Global Optimization*. Springer-Verlag, Berlin.
- Van Laarhoven, P.J.M. and Aarts, E.H.L. (1987) *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Dordrecht.
- Vavasis, S.A. (1995) Complexity issues in global optimization: A survey. In: Horst and Pardalos, Eds. (1995); pp. 27-41.
- Waterloo Maple (2002) *MAPLE User Guide*. Waterloo Maple, Inc., (Current company name: Maplesoft, Inc.) Waterloo, ON.
- Wilde, D.J. (1978) *Globally Optimal Design*. Wiley, New York.
- Winston, W.L. (1992) *Operations Research: Applications and Algorithms*. (3rd Edn.) Wadsworth Publishing Company, Belmont.
- Wolfe, P.H. (1975) A method of conjugate sub-gradients of minimizing nondifferentiable convex functions. *Mathematical Programming Study* Vol. 3, 145 - 173.
- Wolfram, S. (1999) *The MATHEMATICA Book*. (4th Edn.) Wolfram Media, Champaign, IL and Cambridge University Press.
- Wright, M.H. (1996) Direct search methods: once scorned, now respectable. In: D.F. Griffiths and G.A. Watson, Eds. *Numerical Analysis 1995: Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis*, pp. 191-208, Addison Wesley Longman, Harlow, United Kingdom.
- Zhigljavsky, A.A. (1991) *Theory of Global Random Search*. Kluwer Academic Publishers, Dordrecht.

Appendix 1.

Main Program File

Let us note first of all that there are some differences between the various CIAO-GO implementations. In some of these, the driver (main) program segment is directly included with the CIAO-GO solver system; other versions supply also a driver program segment that can be adapted by users. Therefore example files will be provided separately for both cases. Fortran and C code examples are included.

Please observe the comments included in the subsequent source code files, and also recall the earlier discussion. The main program—see PROGRAM MAIN—serves to call CIAO-GO. In addition—see SUBROUTINE USER_FILES—the I/O user filenames are defined here. Note that other, application-specific user actions can also be invoked in these two program segments: this provides a flexible option to connect various external application programs to CIAO-GO.

CIAO-GO IDE Version: Fortran Main Program File and User Files Subroutine

```
! TESTM.FOR --- A test / demo problem
! -----
! This is a program template, to be prepared (adapted) by CIAO-GO users.
! The use of all included statements is (minimally) necessary.
! Additional user operations - related to problem description, further
! report options, additional program/system calls, etc. - can also
! be included or connected, at corresponding points (segments) below.
! Please retain an original copy of this file for further reference
! Save your own MAIN.FOR file in text format
! -----

PROGRAM MAIN

USE WINTERACTER
! WINTER.F90 contains numerous Windows features used by IDE

! --- User filename definitions ---
! Please modify the file names given below, to suit user model

OPEN(10,FILE='TEST.IN',STATUS='OLD')
! This file serves for parameterizing the CIAO-GO program system.
! Note that the input parameterization is problem-dependent;
! please see the *.IN file, and check the comments on pm settings.

OPEN(11,FILE='TEST.OUT',STATUS='UNKNOWN')
! This file serves for (optional) post-run analysis: it provides the
! principal information describing the problem, and the process log
! obtained during the optimization procedure.

OPEN(12,FILE='TEST.SUM',STATUS='UNKNOWN')
! This file contains a brief summary of the run results.

! Activate CIAO-GO optimization (solver) system
CALL CIAOGO
STOP
END
```

CIAO-GO 'Silent' Version: Fortran Main Program File

```
PROGRAM MAIN  
CALL CIAOGO  
STOP  
END
```

Recall that in the 'silent mode' implementation the I/O files are automatically (that is, internally) named CIAO-GO.IN, CIAO-GO.OUT, and CIAO-GO.SUM. Note further that in some implementations the main program is embedded, since CIAO-GO is delivered as an exe file that calls USER_FCT.DLL.

Appendix 2.

Model Function File

The purpose of this file is to define the objective and constraint functions that describe the global optimization problem. Recall that all such functions can be evaluated—if necessary—by calling additional (external) user subroutines and functions.

CIAO-GO IDE Version: Fortran User Function File

```
!      TESTF.FOR --- A 2-variable, 2-constraint test/demo problem

!
!      -----
!      This is the function segment template to be prepared (adapted) by CIAO-GO
!      users. The use of all included statements is (minimally) necessary
!      Additional user statements - related to problem description, further
!      reporting options, calls, etc. - can also be included or connected
!      Please retain an original copy of this segment for further reference
!      Save your own file in text format.
!      -----

      SUBROUTINE FV(x)
      implicit real*8 (a-h,o-z)
!      Please define all (real) variables and operations in double precision,
!      in order to attain higher numerical accuracy
!      The following statement is needed only in Windows DLL program versions
      DLL_EXPORT FV
!      Notation:
!      x      - vector of decision variables (maximal dimension: maxdim)
!      objf   - scalar objective function value (to be minimized)
!              (maximization problems are standardized, multiplying objf by -1)
!      con    - vector of constraint function values (maximal dimension: maxcon)
!      Please note that the dimension settings given below are specific to
!      your present CIAO-GO version, and they can not be modified. Please
!      contact CIAO, if a larger CIAO-GO configuration is needed.
!      Maximal (supported) number of variables and constraints
      PARAMETER(maxdim=1000, maxcon=1000)
      DOUBLE PRECISION x(maxdim), con(maxcon)
      COMMON /csize/m,/cobjf/objf,/cnumcon/con,numcon

!      ----- User problem definitions begin -----

!      --- Objective function ---
      objf=dmax1(x(1)*x(1)+x(2)*x(2)*x(2)*x(2),(2.d0-x(1))*(2.d0-x(1))
1      +(2.d0-x(2))*(2.d0-x(2)),2.d0*dexp(-x(1)+x(2)))

!      --- Constraint types and functions ---
!      Note: explicit box constraint handling is done separately by CIAO-GO;
!      therefore only the additional constraints need to be declared here.
      con(1)=x(1)*x(2)-5.d0
      con(2)=-x(1)**2+x(2)-1.d+01
!      ----- User problem definitions end -----

      return
      end
```

Appendix 3.

Input Parameter File

This file serves for defining the variable bounds and suggested nominal values, as well as several important input parameters, in addition to those given internally in the CIAO-GO.OBJ file system. Note that this structure allows for repeated CIAO-GO runs on different regions, and/or the use of different starting points, when using CIAO-GO in a convex programming (local search) mode. We present three input file examples, corresponding to the models shown above.

CIAO-GO IDE Version

```
TEST.IN FILE --- A 2-variable, 2-constraint demo/test problem.
NOTE TO USERS: Please retain a copy of this file for further reference;
save your own file in text format. The given structure is to be followed
exactly; keep all comment lines and the given *** input data formats.
Numerous parameters are automatically defined in the CIAO-GO object files;
the parameter values provided below can be changed by the user.

--- MODEL DESCRIPTORS ---
Simple Test 1      | Model name (character string, positions 1-20)
2                 | Number of variables (positive integer, positions 1-20)
2                 | Number of constraints non-negative integer, positions 1-20
Variable names    | Lower bounds      | Nominal values    | Upper bounds
Variable 1        | -10.              | -1.               | 10.
Variable 2        | -10.              | 1.                | 12.
Simple TestFct    | Objective function name (character string, positions 1-20)
Constraint names  | Constraint types: equations 0, <= inequalities -1
Equation 1        | -1
Equation 2        | -1

--- SOLVER OPTIONS AND PARAMETERS ---
1                 | Operational modes 1: BM; 2: BRM.
100000           | Maximal no. of fct evals in global search phase
100.             | Constraint penalty multiplier
300              | Program execution time limit (seconds)
```

Appendix 4.

Summary Output File

The files displayed below present all essential information related to a completed CIAO-GO run. The optimization models are defined in Appendix 2, and the corresponding input parameters given in Appendix 3 are used.

CIAO-GO IDE Version

**** CIAO - GO ****

A Model Development and Solver System for
Continuous Global and Local Optimization

(c) CIAO-ITMS, University of Ballarat
and
(c) Pinter Consulting Services, Inc.

Summary of optimisation

Value of the objective at initial point	14.778112198
Solution time without I/O operations	0.000000
Final value of the objective function	1.9522244939
Number of BM iterations	75
Total number of model function evaluations	617
Final solution	
Variable 1	1.1390351777
Variable 2	0.8995618743
Equation 1	-3.9753673779
Equation 2	-10.3978402962
Solution time including I/O operations (seconds)	0.00

*** CIAO-GO *** operations completed.

Note that the total runtime shown above was measured using an Intel Pentium IV processor based personal computer running under MS Windows XP Professional.

Appendix 5.

Connectivity to Other Application Development Platforms

CIAO-GO has been mainly developed using Lahey Fortran (LF90 and LF95). It can be directly connected to applications written in Fortran 77/90/95. Supported platforms include e.g., Digital/Compaq Visual Fortran, g77, and Salford Fortran FTN 77.

Mixed language development (using dll connections) is supported by LF, e.g., with respect to Borland C/C++ and Microsoft Visual C/C++. Recent versions of LF also support static links to the C/C++ environments mentioned above. General Windows API connection is also possible.

The LF documentation includes information and examples related to mixed language platform development. Users may obtain the latest information related to such connectivity issues, by contacting Lahey Computer Systems or by visiting their WWW site (<http://www.lahey.com>).

Among other direct connectivity options, CIAO-GO can also be called from any application that enables external (system) calls. Conversely, CIAO-GO can call external numerical procedures to evaluate functions that are components of the optimization problem in question.

Finally, as mentioned earlier, CIAO-GO can be (and it has been) made available as a solver engine for a number of advanced modeling and computing platforms. Please contact us, if you are interested in the existing solver implementations, or in similar customized development for other platforms.

Appendix 6.

Workstation and Linux PC Implementations

The following remarks are related to the installation and use of CIAO-GO on a typical Fortran 77 (90, or 95) platform, available on workstations under one of the Unix operating system versions. These notes are relevant also with respect to personal computers operated under some Linux version.

For instance, on a Unix OS based system, the CIAO-GO object file can be placed in a public access directory such as, e.g., `opt/lib/CIAO-GO`. As a matter of course, direct access to a Fortran or compatible C/C++ development environment is also necessary. Such a compiler—named e.g. `f77` or `g77/gcc`—is typically available on workstations. (Some Linux and Windows implementations of such compilers are also available for free, or for a small fee, from their developers or from the GNU software development organization). We shall use the name `f77` in the example below, as well as the typical notation `.F` and `.O` for Fortran source and for object files, respectively.

Assume that the application source code files `MAIN.F` and `USER_FCT.F` (corresponding to the user main and function files discussed earlier) are located in the user's home directory. Then compilation is invoked by the command line

```
f77 MAIN.F USERFCT.F opt/lib/CIAO-GO/CIAO-GO.O -o CIAO-GO
```

As a result, the executable file `CIAO-GO` is generated.

This file can be run using the input parameterization given (by default) in the file `CIAO-GO.IN`; results will be generated in `CIAO-GO.OUT` and `CIAO-GO.SUM`.

At present, the typically suggested problem size on workstation platforms (as well as on up-to-date personal computers) is at most a few thousand variables and constraints, to avoid possibly excessive runtimes and resource (memory space) demands.

The most straightforward workstation/Linux installations of CIAO-GO are command line style ('silent mode'), as outlined above. In order to implement new graphics features similar to those described earlier in relation to the CIAO-GO IDE, additional program development work is necessary. Such work and other customizations can be provided upon request.