Paper

A Software Platform for Global Optimization

Ewa Niewiadomska-Szynkiewicz and Michał Marks

Abstract—This paper addresses issues associated with the global optimization algorithms, which are methods to find optimal solutions for given problems. It focuses on an integrated software environment – global optimization object-oriented library (GOOL), which provides the graphical user interface together with the library of solvers for convex and nonconvex, unconstrained and constrained problems. We describe the design, performance and possible applications of the GOOL system. The practical example – price management problem – is provided to illustrate the effectiveness and range of applications of our software tool.

Keywords—global optimization, integrated software systems, nonconvex optimization, numerical libraries, price management.

1. Introduction

Many decision problems are formulated as optimization tasks in which the objective function is nonconvex and has multiple extrema in the region of interest. In addition, in many practical contexts, the optimization problem cannot be described analytically due to the natural complexity and uncertainty of real-life systems. In such cases the simulation experiment is usually used to evaluate the expected performance of the system for each set of decision variables. It involves of simulation-based optimization that is the merging of optimization and simulation techniques [1], [2].

The usage of traditional optimization methods is usually inefficient for solving multimodal or simulation-based problems. Therefore, methods designed for global optimization are important from a practical point of view. The problem of designing algorithms to compute global solutions is very difficult. In general there are no local criteria in deciding whether a local solution is a global one. During last decades, however, many theoretical and computational contributions helped to solve multiextreme problems arising from real-world applications [3]–[5].

In our paper we will present an integrated software environment, called global optimization object-oriented library (GOOL), which can be used to solve complex optimization problems. GOOL supplies the library of optimization algorithms for convex and nonconvex, unconstrained and constrained problems together with the graphical environment for supporting the considered problem definition and tools for dynamic, on-line monitoring of the computed results. The GOOL system integrates various functionalities, and can be successfully used in research, education and commercial applications. The preliminary version of the system was described in [6]. The currently available version is more advanced and has wider range of applications.

This paper is organized as follows. In Section 2 we will discuss the principle features of the global optimization algorithms. Next, we will describe organization, implementation and usage of our software platform GOOL, and numerical algorithms supplied in GOOL. Finally, the results of the application of solvers from the GOOL library to a price management problem will be presented and discussed.

2. Global Optimization Algorithms

Global optimization algorithms can be categorized into two groups: deterministic and stochastic, with respect to their implementation.

Deterministic algorithms are typically based on approximation techniques, approaches that adaptively perform partition, search and bounding, chaotic movement and tabu search. These methods usually require more or less access to global information about the problem. Many of them are guaranteed to find the global minimum (within some tolerance). A unified and insightful treatment of deterministic global optimization is provided in [3], [7], [8].

Stochastic algorithms are typically based on random search, adaptive search, biological inspired heuristics and metaheuristics. Heuristic stochastic methods are widely used in many industrial and scientific applications. These approaches are flexible, robust and less demanding of the problem properties. The main methodological and theoretical developments in stochastic global optimization, the basic principles and methods of global random search, Markovian and population-based random search and methods based on statistical models of multimodal functions are discussed in [9]. The evolutionary algorithms, genetic algorithms, genetic programming, learning classifier systems, evolution strategy, differential evolution, particle swarm optimization, and ant colony optimization, and other metaheuristics, such as simulated annealing, hill climbing, tabu search, and random optimization are elaborated in [4], [5], [10], [11].

Global optimization is generally complex and usually involves cumbersome calculations, especially when consider simulation-optimization case when we have to perform simulation experiment in every iteration of the algorithm. The restrictions are caused by demands on computer re-

sources – central processing unit (CPU) and memory. The directions, which should bring benefits are:

- hybrid techniques that combines global and local algorithms, to solve the optimization problem;
- parallel computing where the whole task is partitioned between several cores, processors or computers.

Hybrid approaches can speed up the convergence to the solution. Parallel implementation allows to reduce the computation time, improve the accuracy of the solution, and to execute large program which cannot be put on a single processor [1], [12], [13].

3. GOOL: Software Environment for Global Optimization

In this section we present the design and implementation of GOOL and comparison of our project to the other existing tools for global optimization.

3.1. Related Works

Most of the existing libraries of optimization techniques focus on the problem of computing locally optimal solutions. However, recently a number of software packages with numerical solvers for global optimization have been developed, and can be find in the Internet. They support sequential and parallel programming. Publicly available implementations of interval analysis and branch-and-bound schemes are discussed in [14]–[16].

The goal of the COCONUT (continuous constraints – updating the technology) project [17] was to integrate the currently available techniques from mathematical programming, constraint programming, and interval analysis into a single discipline, to get algorithms for global constrained optimization. The authors of [18] report the results of testing a number of existing state of the art solvers using COCONUT routines on a set of over 1000 test problems collected from the literature. Solvers implementing various types of techniques for global optimization (deterministic and stochastic), i.e., interval methods, continuous branch and bound, multistart, genetic and evolutionary, tabu search and scatter search are provided in [15]. The Global World [19] is a forum for discussion and dissemination of all aspects of global optimization problems. It provides links to libraries of solvers and a library of academic and practical test problems.

3.2. GOOL Overview

The GOOL provides an integrated graphical software framework that can be used to solve the following very general problem:

$$\min_{x\in\Re^n} f(x) \tag{1}$$

$$g_i(x) \le 0, \quad i = 1,\ldots,m,$$

where f and g_i are real-valued functions.

The GOOL supplies a library of deterministic and stochastic optimization solvers. When most of the available libraries for calculating the optimal solution provide tools only for commerce, research or educational purposes, the GOOL system integrates all these functionalities. The process of implementing a given application for GOOL is quite straightforward and convenient especially thanks to graphical user interface (GUI). The system provides tools for on-line monitoring of computation process and various presentation techniques.

Two different versions implementing two approaches to user-system interactions: GOOL/COM (batch) and GOOL/GUI (interactive) are supplied. GOOL/COM is dedicated to the complex optimization problems, where values of the objective function are calculated based on simulation (simulation-optimization scheme, [2]). In the case of simulation optimization the user's task is to provide the simulation model to evaluate the expected performance of the system to be optimized. It is assumed that solvers from the GOOL library provide decision variables and receive values of the objective function f and constraints g in Eq. (1) from the user application. Let the input files be called `task_file.tsk` and `methods_file.met`. Then, writing the command `gool_con task_file [methods_file]` at the command line, we call GOOL to solve the optimization problem defined in the file `task_file` using the optimization algorithm pointed in the file `methods_file`. The input file `task_file` contains the information related to the particular problem to be solved (problem dimension, objective function definition, its gradient and constraints) or the name of the user application (simulator). The selection of the solver is optional.

Fig. 1. GOOL/GUI: the main window.

The GOOL/GUI is the software framework for educational purposes and research (see Figs. 1 and 2). It supplies the graphical environment for optimization problem definition and calculation results presentation. The optimization prob-

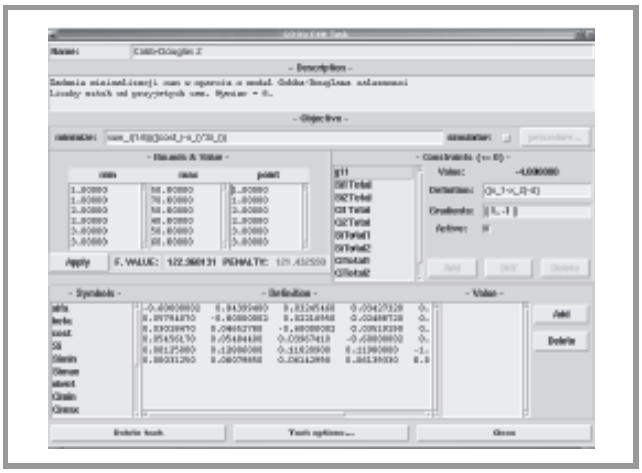

Fig. 2. GOOL editor: optimization problem implementation.

lem is defined using the GOOL editor. The GOOL symbolic expressions analyzer allows to enter quite complicated functions concerning such expressions like: pow, sin, sum, etc., and iterative expressions. The gradient is calculated if necessary. After starting the calculations the user can on-line employ the monitoring of the results.

3.3. System Architecture

The system consists of three components (Fig. 3):

– library of numerical methods,

– system kernel,

– graphical user interface.

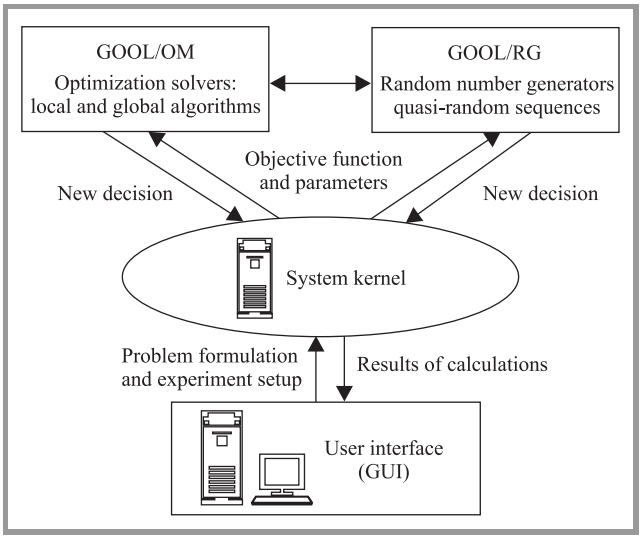

Fig. 3. The GOOL system architecture.

The core component is numerical library consisting of two sets of modules:

– GOOL/OM: optimization solvers,

– GOOL/RG: random number generators.

In addition the system facilities are provided in the form of four groups of services. These are:

1. User interface services, which provide a consistent user interface. The most important tasks of the user interface are as follows: supporting the process of defining a considered optimization problem, results visualization, providing communication with the user.

2. Calculation management services, which manage execution of a given solver.

3. Communication management services, which manage communications between calculation process and user interface.

4. Data repository services, which provide a store for all data objects: all defined options, parameters and calculation results.

3.4. Algorithmic and System Options

Various algorithmic and system options are available to the user, all come with a default value so it is not necessary to modify any options. The ability to modify them, however, provides a great deal of flexibility. It is possible to change all parameters of the chosen solver, type of random generator or local search using graphical interface in GOOL/GUI or text file `methods_file` in GOOL/COM. Different termination criteria are provided: typical to each algorithm (if exists), convergence tolerance, maximum number of iterations or function evaluations. The results can be displayed every iteration or recorded into the disc file and displayed at any time.

3.5. Graphical User Interface

The GUI is organized in a set of windows. The setting windows are used to facilitate the configuration phase. The optimization problem is defined, an objective function and all constraints are entered.
The GOOL provides tools for dynamic, on-line monitoring of the computed results. The following graphical presentation techniques are available: 2D, 3D graphs, leaves of the function and a table of numbers (Fig. 4). The visualization of a multidimensional problem is achieved by displaying in the separate windows the leaves for each pair of variables, under the assumption that all other variables are fixed. The results presentation is organized in different ways, and is fitted to the optimization method (points, lines, grids). Multiple windows with the results for different range of data can be active during one run of the program. The changes of values of parameters typical to each algorithm can be graphically displayed too. The user can chose options of presentation (zoom, colors, results of many optimizations in one window, etc.). The detailed report of the results including the problem solution, number of iterations, number

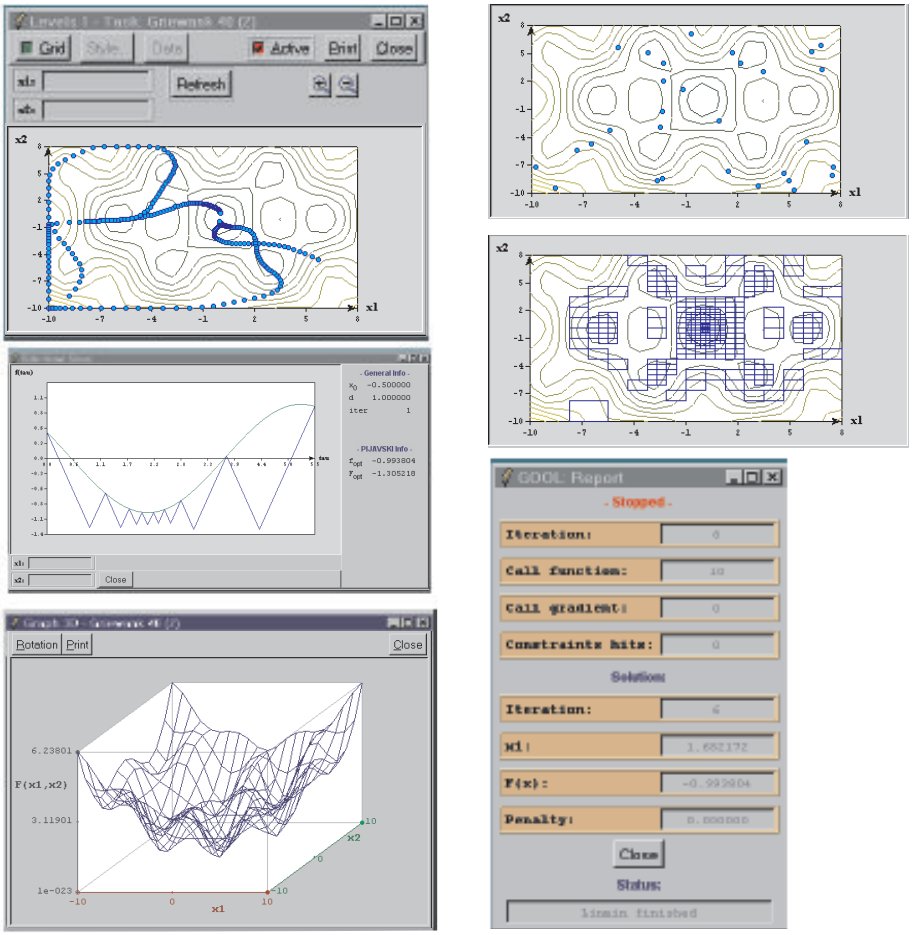

Fig. 4. Results visualization.

of function and gradient evaluation and number of constraints violation is displayed after finishing the calculations.

3.6. GOOL Operation

The interaction with GOOL/GUI is organized as follows. At the beginning the user is asked to define the problem to be solved and select the optimization algorithm. Within the next step the user is asked to provide some information related to the considered method and calculation process if necessary. This information includes: parameters typical for the chosen algorithm, type of the stop criterion, maximal number of iterations, type of results visualization, etc. After completing the initial settings, GOOL starts the calculation engine. The user employs monitoring of the current situation. It helps him to assess the effectiveness of the chosen optimization algorithm. The calculations may be interrupted.

3.7. Implementation

The GOOL system is completely based on C++. All numerical methods – the optimization engine – and the higher-level activities, i.e., problem definition, parameters setting, results presentation, managing calculations and communication between the optimization engine and the user interface are implemented in uniform form as C++ classes. Two functionalities of GOOL, i.e., user interface and calculations are separated and can be easily modified. The hierarchy of classes implementing numerical solvers is natural and well defined (Fig. 5). Three fundamental clas-

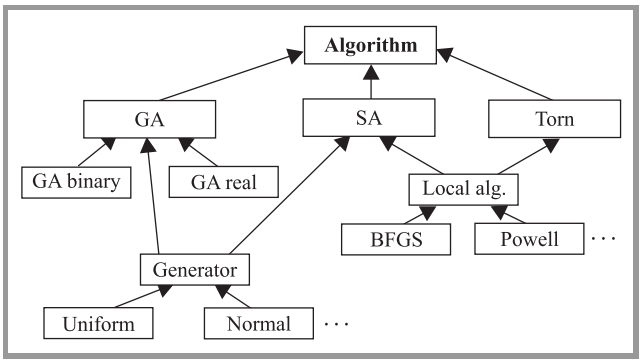

Fig. 5. GOOL: hierarchy of classes. Explanations: GA – genetic algorithm, SA – simulated annealing, BFGS – Broyden-Fletcher-Goldfarb-Shanno.

ses: `Task`, inserting the considered optimization problem to be solved, `Algorithm`, the basic class of all optimization methods and `Generator`, for random numbers generation are provided. The library can be extended by new methods developed by the user. Available software may be easily adopted, new techniques can be implemented applying classes defined in GOOL. The open design of the system architecture, and its extensibility to include other open source modules, was chosen in the hope that the system will be a useful platform for research and education in global optimization. The code is currently available for MS-Windows and Linux operating systems. The software is free available for researchers and students.

4. Library of Solvers

The numerical library consists of two parts: GOOL/OM and GOOL/RG. The GOOL/OM is a collection of different optimization solvers for calculating local and global minimum. GOOL/RG provides several random numbers generators.

4.1. Local Optimization

Several techniques for calculating local minimum of the performance index were implemented in GOOL. The following methods for one-dimensional search are available: golden section search, parabolic interpolation, one-dimensional search with first derivatives (Goldstein test). Two, well known nongradient methods in multidimensions [20] are available too: downhill simplex algorithm (Nelder-Mead) and direction set (Powell's) method.

4.2. Global Optimization

Deterministic and stochastic techniques are provided. Currently implemented are methods based on the approximation and branch-and-bound techniques, deterministic chaotic movement, clustering techniques, random search, heuristics and metaheuristics. The following variants are provided:

- Branch-and-bound (BB) for Lipschitz problems: uniform grid, few versions of non-uniform grids [3], [21]: Galperin's, Gourdin-Hansen-Jeaumard's, Meewella-Mayne's, and Pijavskij's algorithm of linear sub-approximations of the performance function, developed for one-dimensional problems.
- Chaotic movement: Griewank's algorithm (trajectory method) [22], [23].
- Clustering method developed by Torn [21], with different grouping techniques.
- Pure random search and three variants of population set based direct search methods controlled random search (CRS): CRS2, CRS3 and CRS6 as described in [13], [24].
- Simulated annealing (SA) as described in [25].
- Genetic algorithm (GA) using fixed-length binary strings for its individuals and evolutionary strategy (ES) with real-valued individuals [10], [11].

The available algorithms can be used to solve general constrained optimization problems. The constraints that cannot be handled explicitly are accounted for in the objective function using simple penalty terms for constraints violation. The reformulation of Eq. (1) is made inside the GOOL system:

$$\min_{x\in\Re^n}\left[f(x)+\Psi(x)\right], \qquad \Psi(x)=\mu\sum_{i=1}^{m}\max(0,g_i(x))^p. \quad (2)$$

The user can insert the value of parameters μ and p in Eq. (2).

4.3. Random Numbers Generation

Many heuristic algorithms provided in GOOL use random number generators to calculate a new decision. The large number of random generators have been developed over the last decades. Several procedures representing different types of generators are available in the library: uniform (two variants), normal (three variants), beta distribution, Cauchy distribution. Sequences of n-tuples that fill n-space more uniformly, than uncorrelated random points are called the quasi-random sequences [20]. That term is somewhat of a misnomer, since there is nothing "random" about quasi-random sequences – they are cleverly crafted to be, in fact sub-random. Three such sequences are available in GOOL: Halton, Sobol and Faure.

5. Case Study Results

5.1. Formulation of Price Management Problem

Several stochastic algorithms from GOOL library were compared. In this section we present the computational results obtained for prices optimization problem.
The considered case study was to calculate the optimal prices for products that are sold in the market. The goal was to maximize the total profit *PR*:

$$\max_{x}\left[PR=\sum_{i=1}^{n}\left(\frac{x_i}{(1+v_i)}-d_i\right)S_i(x)\right], \quad (3)$$

where n denotes number of products exist (corresponding to n price decisions x_i), v_i and d_i are given constants corresponding to the market entities of VAT (value added tax) and cost per product, S_i are expected sales of product i within the considered period, assuming that prices of all products are fixed over this period. Several sales models can be found in the literature [26]. All these models describe market response on the price of jth product. We considered three of them.

Cobb-Douglas model. This model is following:

$$S_i(x) = \alpha_i \prod_{j=1}^{n} x_j^{\beta_{ij}}, \tag{4}$$

where x_j denotes the price of product j, α_i is the scaling factor for sales of product i, β_{ij} is the elasticity of sales of product i with respect to the price of product j (β_{ii} is referred to as the direct elasticity and β_{ij}, $i \neq j$ is the cross elasticity).

Gutenberg model. The response function Eq. (4) is widely used but it does not capture some important effects, such as different market sensitivities to small and large price changes. Another sales model is formulated:

$$S_i(x) = a_i - bx_i + c_{1i} \sinh\left(c_{2i}(x_i - \overline{x}_i)\right), \tag{5}$$

where a_i, b_i, c_{1i} and c_{2i} denote model parameters and $\overline{x}_i$ the average competitive price, i.e., price computed as the average of competitor prices taking into account their respective market share. The additional term can be added to this expression: $c_{3i} \sinh\left(c_{4i}(x_i - x_{i_0})\right)$, where x_{i_0} denotes the current price of the product i. The response function Eq. (5) belongs to the group of s-shaped models. The major difficulty is in fact that for some values of parameters such market response can involve multiextreme profit PR in Eq. (3), as presented in Fig. 6 (see [27] for details).

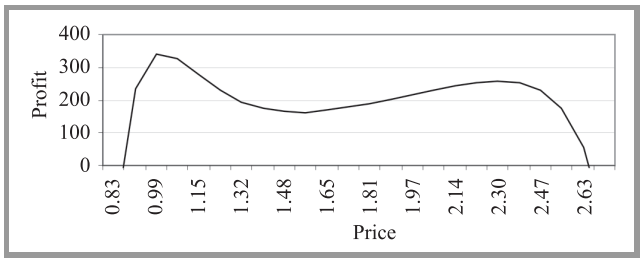

Fig. 6. Values of profit for different prices of a given product (model Eq. (5)).

Hybrid model. In the model Eq. (5) the cross-effects with other substitute or complementary own products are not included. The next considered model formulated in [28] combines functions Eqs. (4) and (5):

$$\begin{aligned} S_i(x) \quad = \quad & a_i - bx_i + \alpha_i \prod_{j=1}^{n} x_j^{\beta_{ij}} \\ & + c_{1i} \sinh\left(c_{2i}(x_i - \overline{x}_i)\right) \\ & + c_{3i} \sinh\left(c_{4i}(x_i - x_{i_0})\right), \end{aligned} \tag{6}$$

where α_i, β_{ij}, $\overline{x}_i$ and x_{i_0} the same like in Eqs. (4) and (5), a_i, b, c_{1i}, c_{2i}, c_{3i}, c_{4i} model parameters. This model exhibits an s-shape and includes cross-effects.

Constraints. The following constraints for price, sale and cash of each product and for total sale and cash can be considered: $x_{i_{\min}} \leq x_i \leq x_{i_{\max}}$, $S_{i_{\min}} \leq S_i \leq S_{i_{\max}}$, $C_{i_{\min}} \leq x_i S_i \leq C_{i_{\max}}$, $TS_{\min} \leq \sum_{i=1}^{n} x_i \leq TS_{\max}$, $TC_{\min} \leq \sum_{i=1}^{n} x_i S_i \leq TC_{\max}$. In listed constraints $x_{i_{\min}}$ and $x_{i_{\max}}$ denote minimal and maximal prices of product i, $S_{i_{\min}}$, $S_{i_{\max}}$ minimal and maximal sale, $C_{i_{\min}}$, $C_{i_{\max}}$ minimal and maximal cash, $TS_{\min}$, $TS_{\max}$ minimal and maximal total sale, and $TC_{\min}$, $TC_{\max}$ minimal and maximal total cash. In practice, usually prices of only some products are changed at anyone time. The following constraint restricts the number of prices, which can be modified

$$\sum_{i=1}^{n} \frac{\gamma(x_i - x_{i_0})^2}{1 + \gamma(x_i - x_{i_0})^2} \leq w, \tag{7}$$

where γ and w are assumed parameters, x_{i_0} the current price of the product i.

5.2. Comparison of Market Response Models

The comparative study of all presented market response models was performed. The goal of the experiments was to calculate the optimal prices for fifteen products ($n = 15$ in Eq. (3)). The optimization model was defined using the GOOL graphical editor (see Fig. 2). All models parameters were randomly generated in ranges determined based on real historical data. The evolutionary strategy solver supplied in the GOOL library was used to solve the task.

The results – suggested prices of fifteen products – obtained for three presented market response models, taking into account only price bounds are depicted in Fig. 7. We can observe that the model Eq. (4) suggests the highest prices while the model Eq. (5) expresses less optimism suggesting lower prices. The results for Eq. (6) are between values obtained using Eqs. (4) and (5).

5.3. Comparison of Solvers

The goal of the second series of tests was to compare the efficiency of selected solvers from the GOOL library. The experiments were performed for historical data. Calculations were terminated after 100 iterations of each algorithm. The results obtained for 15 products, market response function Eq. (6) w.r.t. all listed constraints are compared in Table 1. The values collected in the adequate columns denote: PR – the total profit defined in Eq. (3), time – time of calculations in seconds.

Table 1
Simulation results for market response function Eq. (6)

Method	PR	Time [s]
CRS2	1252	91
SA	1286	96
ES	1281	11

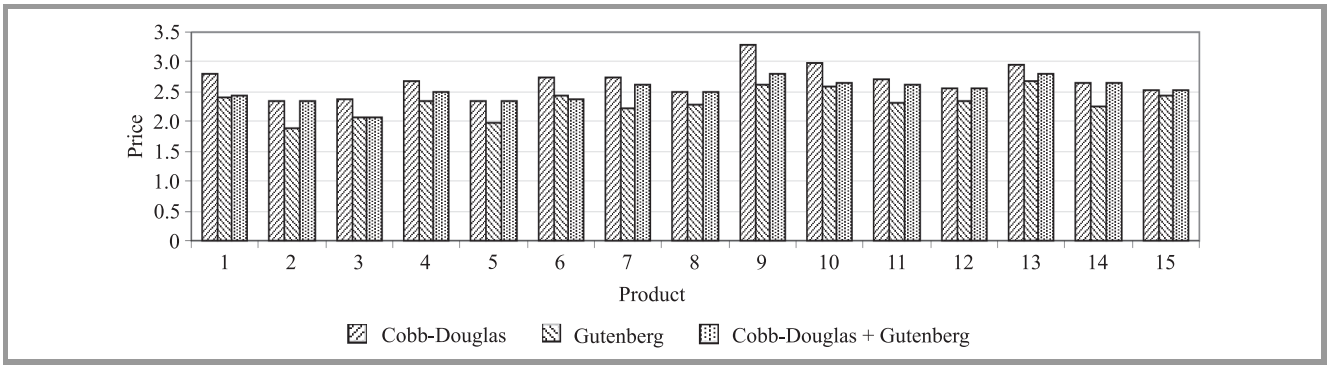

Fig. 7. Prices for different market response models.

The available numerical results indicate that ES and SA methods give better solution than CRS method. The best result was obtained using SA but the time required to compute solution was longer than ES method. Attempts to solve the considered problem using other solvers provided in GOOL (clustering method, branch-and-bound and chaotic movement) failed. The feasible solution was not found or the computation time was unacceptable long. The conclusion to be drawn is that heuristics such as ES, although quite simple are efficient and robust for many real-life optimization problems.

Finally, two versions of controlled random search methods described in [13] and [24], i.e., CRS2 and CRS6 were compared. CRS are population set based random search algorithms. The basic random search consists of three main steps: generate the initial set of points, transform the population, and check the assumed stopping condition. Several versions of CRS methods related to different strategies of new trial points calculations were developed. CRS2 is the simplest one. CRS6 is much more advanced – it uses quadratic interpolation and random numbers generation from the β distribution to calculate new trial points. The weakness of all CRS methods is the way in which the constraints of a type $g_i(x) \leq 0$ are handled. The infeasible points are simply rejected from further consideration. The suggested approach is to use penalty terms for constraints violation Eq. (2).

The optimization results of price management problem Eq. (3) with sales model Eq. (4) and all listed constraints, considering CRS2 and CRS6 methods are presented in Tables 2 and 3. The experiments were performed for several sets of historical data, containing various groups of products offered in supermarkets. Prices of 15, 31 and 53 products were calculated. Each solver was executed five times, the assumed accuracy was 10^{-4}. The results obtained for modified objective function Eq. (2) were compared with those obtained for the standard approach that discards infeasible points (Table 2). The values collected in tables denote: n – number of products, PR – average total profit, time – average time of calculations in seconds, f_{call} – average number of the objective function Eq. (3) evaluations.

The results presented in Tables 2 and 3 indicate that the CRS2 algorithm is very fast but only gives an approximate solution, even in the case when the penalty function is used (see Table 2). The CRS6 method provides better results with respect to CRS2 but the time required to compute a solution was longer than the CRS2 method.

Table 2
Total profit PR in case of two approaches to infeasible points (15 products)

Method	Discarding infeasible points	Penalty for constraints violation
CRS2	1215.95	1237.45
CRS6	1241.27	1241.27

Table 3
Comparison of the fastest and the most accurate methods

n	Best PR	Algorithm	f_{call}	PR	Time [s]
15	1241.27	CRS2	24002	1235.41	1.98
		CRS6	39392	1241.27	3.16
31	830.75	CRS2	69562	805.76	23.39
		CRS6	122298	830.71	33.64
53	544.65	CRS2	76169	526.03	23.85
		CRS6	285849	544.65	75.99

As a conclusion the following strategy is proposed: in cases when accuracy of the solution is the crucial the CRS6 method with the discarding of infeasible points are suggested; when it is crucial that the problem is solved quickly the CRS2 method with the penalty function should be used.

6. Summary and Conclusions

In this paper a brief description of the software platform GOOL for complex systems optimization was made. GOOL was design to be powerful, effective, flexible, and easy to use software for optimization. It is suitable to solve

different optimization problems and can be successfully used for global minimum calculating. The user-friendly interface allows to perform the numerical experiments in the effective manner both for research and education. The open design of the system architecture, and its extensibility to include new solvers make GOOL be a useful platform for global optimization. The current version of GOOL can support researchers and engineers during the design and control of real-life complex systems in the sense of decision automation. In our future research we plan to extend our system to multiobjective optimization to provide the tool that will support interactive optimization process.

Acknowledgments

This work was partially supported by Ministry of Science and Higher Education grant NN514 416934.

References

[1] E. Niewiadomska-Szynkiewicz, "Symulacja komputerowa w analizie i projektowaniu złożonych systemów sterowania", Warsaw, Warsaw University of Technology Press, 2005 (in Polish).

[2] J. C. Spall, *Introduction to Stochastic Search and Optimization*. New Jersey: Wiley, 2003.

[3] R. Horst and P. M. Pardalos, *Handbook of Global Optimization*. Dordrecht: Kluwer, 1995.

[4] Z. Michalewicz and D. B. Fogel, *How to Solve it: Modern Heuristcs*. New York: Springer, 2000.

[5] T. Weise, *Global Optimization Algorithms: Theory and Application*, e-book, 2009 [Online]. Available: http://www.it-weise.de/projects/book.pdf

[6] M. Publicewicz and E. Niewiadomska-Szynkiewicz, "GOOL – global optimization object-oriented library", in *Proc. KAEiOG'2003, Conf.*, Łagów, Poland, 2003, pp. 173–181.

[7] A. C. Floudas, *Deterministic Global Optimization: Theory, Methods and Applications*. Dordrecht: Kluwer, 1999.

[8] A. Neumaier, *Complete Search in Continuous Global Optimization and Constraint Satisfaction*, Acta Numerica. Cambridge, Cambridge University Press, 2004, pp. 271–369.

[9] A. A. Zhigljavsky and A. Zilinskas, *Stochastic Global Optimization*. Springer Optimization and Its Applications. New York: Springer, 2007.

[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer, 1997.

[11] R. Schaefer, *Foundations of Global Genetic Optimization*. Berlin-Heidelberg: Springer, 2007.

[12] A. Karbowski and E. Niewiadomska-Szynkiewicz, "Obliczenia równoległe i rozproszone", Warsaw, Warsaw University of Technology Press, 2001 (in Polish).

[13] W. L. Price, "Global optimization by controlled random search", *JOTA*, vol. 40, no. 3, pp. 333–348, 1983.

[14] K. Holmqvist and A. Migdalas, "C++ class library for interval arithmetic in global optimization", in *State of the Art in Global Optimization*, C. A. Floudas and P. M. Pardalos, Eds. Dordrecht: Kluwer, 1996.

[15] "Solver technology – global optimization" [Online]. Available: http://www.solver.com/technology5.htm

[16] S. Tschoke and T. Polzer, "Portable parallel branch-and-bound library: PPBB-Lib, user manual, library version 2.0", University of Paderborn, Germany, 1999.

[17] "COCONUT – continuous constraints updating the technology" [Online]. Available: http://www.mat.univie.ac.at/~neum/glopt/coconut

[18] A. Neumaier, O. Shcherbina, W. Huyer, and T. Vinko, "A comparison of complete global optimization solvers", *Math. Programm.*, vol. 103, no. 2, pp. 335–356, 2005.

[19] "Global World Forum" [Online]. Available: http://www.gamsworld.org/global/index.ht

[20] W. H. Press, S. A. Tukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, The Art of Scientific Computing. Cambridge, Cambridge University Press, 1992.

[21] A. Torn and A. Zilinskas, *Global Optimization*, LNCS, vol. 350. Berlin: Springer, 1989.

[22] A. O. Griewank, "Generalized descent for global optimization", *J. Opt. Theory Appl.*, vol. 34, no. 2, pp. 11–39, 1981.

[23] J. W. Rogers and R. A. Donnelly, "A search technique for global optimization in chaotic environment", *JOTA*, vol. 61, no. 1, pp. 111–121, 1989.

[24] M. M. Ali and C. Storey, "Modified controlled random search algorithms", *Int. J. Comput. Math.*, vol. 53, no. 3–4, pp. 229–235, 1994.

[25] A. Dekkers and E. Aarts, "Global optimization and simulated annealing", *Math. Programm.*, vol. 50, no. 1–3, pp. 367–393, 1991.

[26] H. Simon, *Price Management*. North-Holland: Elsevier, 1989.

[27] M. Dygas and E. Niewiadomska-Szynkiewicz, "Optymalna wycena produktów i usług – modele, oprogramowanie i eksperymenty symulacyjne", Int. Rep. ICCE WUT, no. 03-17, Warsaw, 2003 (in Polish).

[28] K. Malinowski, "PriceStrat 4.0 Initial Research Paper", KSS Int. Doc., Manchester, 2000.

Ewa Niewiadomska-Szynkiewicz received her Ph.D. in 1996, D.Sc. in 2006 from the Warsaw University of Technology. She works as a Professor of control and computation engineering at the Warsaw University of Technology. She is the Head of the Complex Systems Group. She is also an associate professor at the Research and Academic Computer Network (NASK), and the Director for Research of NASK since 2009. She is the author or co-author of three books and over 90 journal and conference papers. Her research interests focus on complex systems modeling and control, computer simulation, global optimization, parallel calculations and computer networks. She was involved in a number of research projects including three EU projects, coordinated the Groups activities, managed organisation of a number of national-level and international conferences.

e-mail: ens@ia.pw.edu.pl
Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland

e-mail: ewan@nask.pl
Research Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Michał Marks – for biography, see this issue, p. 41.