Comparison of public-domain software for black box global optimization[1]

M. Mongeau[*2], H. Karsenty[†], V. Rouzé[*], J.-B. Hiriart-Urruty[*]

[*]Laboratoire MIP, Université Paul Sabatier, 31062 Toulouse cedex 04, France

[†] École Nationale Supérieure de l'Aéronautique et de l'Espace, BP 4032, 31055 Toulouse cedex, France

Abstract

We instance our experience with six public-domain global optimization software products and report comparative computational results obtained on a set of eleven test problems. The techniques used by the software under study include integral global optimization, genetic algorithms, simulated annealing, clustering, random search, continuation, Bayesian, tunneling, and multi-level methods. The test set contains practical problems: least median of squares regression, protein folding, and multidimensional scaling. These include non-differentiable, and also discontinuous objective functions, some with an exponential number of local minima. The dimension of the search space ranges from 1 to 20. We evaluate the software in view of engineers addressing black box global optimization problems, i.e. problems with an objective function whose explicit form is unknown and whose evaluation is costly. Such an objective function is common in industry. It is for instance given under the form of computer programmes involving a simulation.

Key words: Global optimization, software, test problems, black box, direct methods, genetic algorithms, simulated annealing, clustering, non-convex optimization.

Abbreviated title for use as running head: Public-domain global optimization software

1 Introduction

This paper aims at reporting the results of our computational experiments on "continuous" (as opposed to discrete—we consider also discontinuous objective functions) global optimization software as well as the difficulties encountered with such work in a non-academic application context.

The starting point of our experience is a demand from a space industry for transfer of knowledge. Some specific test problems were provided. The problems of interest are

[1]Revised version of [11]. To appear in *Optimization Methods & Software*. This research was prompted and supported by the Centre National d'Études Spatiales under contract N° 871/94/CNES/1439.

[2]Corresponding author. Email: mongeau@cict.fr

typical of many continuous optimization problems from the industry: small (in terms of number of variables) specific problems involving very expensive objective-function evaluations. We are concerned with minimizing the number of objective-function evaluations needed to obtain a good approximation of a global optimum value. The aim is thus to reach a (subjective) balance between the quality of the solution, and the computational cost required to obtain it. Moreover, it is often the case in industry that there is no explicit form of the objective function. The objective function is provided under the form of a *black box*: it can be evaluated at a given point through either

- a complex amalgam of (sometimes old) computer programmes (often not allowed to be modified) requiring sometimes the interaction of some experienced druid, and/or
- a (physical or computational) simulation.

This prevents the use of automatic differentiation techniques to obtain any derivative. On the other hand, relying on finite differences is dismissed by the high cost of single objective-function evaluations. Until recently, practically no software product could deal with such problems. Note that for the *local* optimization of a *differentiable* function, we refer the reader to [8] which presents an overview of derivative-free optimization methods and references.

Let us now briefly describe our methodology, as we proceeded differently from one would do for experimenting more classical *local* optimization software. We first collected some available global optimization software products through personal contacts, and a call to the optimization community via the networks Opt-Net [14] and NA-Net [12]. We then sorted the (not numerous) replies according to the imposed constraint, namely the fact that the evaluation of the objective function has to be given by a black box.

Most of available software indeed aim at specially-structured problems, or at least at problems whose objective function has an explicit form. We do not consider such specialized software in the present paper. A brief review on continuous global optimization software was recently prepared by Pintr [15].

Difficulties which are characteristics of the use of (general) continuous global optimization software are:

- such software is not available in common optimization software packages (whence the above mentionned call for software);
- user's manuals are often (at least for the case of public-domain software) either brief or obscure;
- such software has rarely been subject to systematic testing, and is not as mature as local optimization software.

In fact, as we shall see later, the inherent difficulty of the problem under consideration is such that a black box global optimization software performance depends mostly on the adjustment of parameters whose value is to be set by the user.

The techniques used by the software products we shall study include integral global optimization, genetic algorithms, simulated annealing, clustering, random search, continuation, Bayesian, tunneling, and multi-level methods.

We chose test problems which are rather non-academic, arising from applications (no instance of polynomials or usual functions etc. too frequently considered in global optimization computational experimentation). We decided to restrict our attention to *unconstrained* problems, in order to compare the performance of the different software products, leaving aside the additional difficulty, often non-negligible, which consists in generating feasible points. The eleven test problems we selected are of three types. Five problems are of the type *least median of squares* regression, with a non-differentiable objective function. The next four problems are often used as tests for methods attempting to minimize (globally) energy in order to find the configuration of a protein (*protein folding*). Their objective functions are discontinuous (presence of poles), and are known to have an exponential number of local minima. Finally, two test problems are *multidimensional scaling* problems. The dimension of the search space of the test problems ranges from 1 to 20.

We shall evaluate the different software products according to:

- efficiency (in terms of number of objective-function evaluations required to obtain a "good enough" objective-function value),
- whether it is user friendly (how easy is it to use for an engineer, for example?—important in order to evaluate the actual impact of a software product in an application context), the difficulty to tune input parameters to be set by the user.

Finally, we shall attempt at establishing the type of problems for which each given software product is efficient (or simply usable).

The remaining of the paper is divided into five sections. Section 2 first enumerates the six public-domain global optimization software products under study with references. We then present in Section 3 the eleven test problems. We report in Section 4 our experience with the practical usage of each of the software products (ease of use, parameter tuning, numerical experimentation). Section 5 exhibits the comparative performance of the software on each problem via graphs displaying the decrease of the objective function in terms of the number of objective-function evaluations. We present some conclusions in Section 6.

2 The Software Products

The six public-domain global optimization software products we are testing in this paper are:

- **ASA**: *Adaptive Simulated Annealing.*

  Author: L. Ingber (ingber@alumni.caltech.edu).

  Software and papers describing the method are available by anonymous FTP: ftp://ftp.ingber.com. See also [5].

- **GLOBAL**: is a modification of the clustering algorithm of Boender et al. [2].

  Author: T. Csendes (csendes@sol.cc.u-szeged.hu).

  Available by anonymous FTP: [3]

- **GAS**: Genetic algorithm looking for all local optima in $I\!R^n$ or in $\{0,1\}^n$.

  Authors: M. Jelasity (jelasity@inf.u-szeged.hu) and J. Dombi.

  Available by anonymous FTP: [6].

- **GOT**: *Fortran-77 and Fortran-90 Global Optimization Toolbox*. Multi-level random search (combination of sampling and local search techniques) looking for the best local optima.

  Author: A.V. Kuntsevich (kun@d120.icyb.kiev.ua).

  Documentation: [7].

- **INTGLOB**: Integral global optimization algorithm. A Monte-Carlo based technique is used to compute a sequence of mean values and modified variance, and a sequence of level sets [18].

  Authors: Q. Zheng and D. Zhuang (Deming.Zhuang@MSVU.Ca)

- **UFO** : *Interactive System for Universal Functional Optimization.* Interactive modular system for solving optimization problems and development optimization algorithms. With respect to global optimization only, four types of methods can be used: random search methods, continuation methods, clustering methods, and multi-level methods (combination of sampling and local search techniques).

  Authors: L. Lukšan (luksan@uivt.cas.cz), M. Tuma, M. Šiška, J. Vlček and N. Ramešová.

  Available by anonymous FTP: ftp.uivt.cas.cz in the directory pub/msdos/ufo.

  Documentation: [9]

3 The Test Problems

The eleven test problems we selected are of three types: least median of squares regression, protein folding, and multidimensional scaling. Table 1 displays the source of each of these test problems. As required by the referees and the editor, we provide the data in an appendix. We also give the search domain in the last column (in a given test problem, the box constraint is the same for every variable).

3.1 Least Median of Squares

The computation of a robust linear-regression estimator called *least median of squares* (LMS) [16] consists in solving the following global optimization problem:

$$\min_{\hat{\theta},b} f(\hat{\theta},b),$$

where $f(\hat{\theta},b) := \text{median } r_i^2(\hat{\theta},b)$,

$$r_i(\hat{\theta},b) := y_i - x_{i1}\hat{\theta}_1 - \ldots - x_{ip}\hat{\theta}_p - b,$$

Problem	dimension	source	application	box constraints
lms1a	1	[16, chap.2, table 2]		[0,1]
lms1b	1	[16, chap.2, table 3]	least median	[-2,6]
lms2	2	[16, chap.3, table 23]	of squares	[0,3]
lms3	3	[16, chap.3, table 1]	regression	[-2,2]
lms5	5	[16, chap.3, table 2]		[-2.5,2.5]
pf3	9			
pf4	12	[17]	protein	[-1,1]
pf5	15		folding	
pf6	18			
ms1	20	[10]	multidimensional	[-2.67,2.67]
ms2	20		scaling	[-3.27,3.27]

Table 1: Test Problems

and the points $(y_i, x_{i1}, \ldots, x_{ip})$ are given.

In other words, this problem is that of finding a hyperplane in $\mathbb{R}^{p+1}$ from which (at least) half of the given points lie within as small a (vertical) distance as possible.

This is a non-differentiable optimization problem. It involves in fact only p degrees of freedom, since for a given $\hat{\theta}$ one can easily compute [16] the value b^* such that

$$f(\hat{\theta}, b^*) = \min_{b \in \mathbb{R}} f(\hat{\theta}, b).$$

Our smallest-sized test problems are of the type LMS: Problems lms1a, lms1b, lms2 lms3, and lms5. They are of dimension $p = 1$, 1, 2, 3, and 5 respectively.

3.2 Protein Folding

A test problem commonly used as test for methods attempting to minimize (globally) energy in order to find the configuration of a protein (*protein folding*) [13] (PF) is the following global optimization problem:

$$\min_x \sum_{i<j}^{n} r(||x_i - x_j||_2),$$

where $x_1, \ldots, x_n \in \mathbb{R}^3$ are the coordinates of n molecules or atoms in $\mathbb{R}^3$, and the function

$$r(s) := s^{-12} - 2s^{-6}$$

is the Lennard-Jones energetic model (identical spherical atoms). The function r is not convex but has a unique minimum, whereas the objective function has an exponential number of local minima. The objective function is moreover discontinuous as it tends to infinity each time a molecule approaches another molecule. Finding a global minimum for the instances where $n \geq 5$ is an open problem. We shall consider Problems pf3, pf4, pf5, and pf6 (Problem pf$i$ corresponds to the $3i$-dimensional instance).

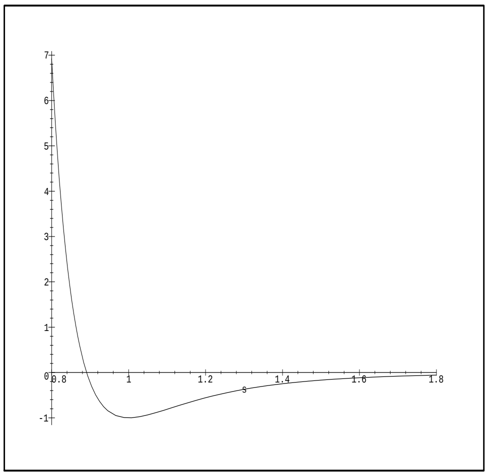

Figure 1: The function r in protein folding

3.3 Multidimensional Scaling

The problem of finding n points of $I\!R^2$ fitting (in the least-squares sense) given dissimilarities, δ_{ij}, between each pair of points is called the multidimensional scaling (MS) problem. It can be formulated as the following global optimization problem, given weights w_{ij}:

$$\min_x \sum_{i<j}^{n} w_{ij}(\delta_{ij} - ||x_i - x_j||)^2,$$

where $x_1, \ldots, x_n \in I\!R^2$.

We consider two problems, ms1 and ms2, for both of which $n = 10$ (dimension 20).

We refer the interested reader to [1] (and references therein) where a global optimum of the following variant of the multidimensional scaling problem:

$$\min_{r,x} \sum_{i<j}^{n} w_{ij}(\delta_{ij}^2 - ||x_i - x_j||^2)^2, \quad r \in I\!N \setminus \{0\}, \ x_1, \ldots, x_n \in I\!R^r, \tag{1}$$

is efficiently computed. A semi-definite programming formulation of (1) indeed exhibits that it has some hidden convexity feature, and can therefore be solved via an interior-point method.

4 Usage of Each Software Product

We report in this section our experience with the practical usage of each of the software products. The results of the computational experimentation mentioned in the current section are displayed in a comparative fashion in Section 5.

4.1 ASA

The Adaptive Simulated Annealing software is provided as a source code written in C that can be run for example under Unix. It comes with an extensive user's manual which cannot easily be exploited. It describes numerous options under which one can use the software. We chose using ASA with the default options. The only remaining parameter one can set is then the starting point.

Numerical experiments revealed that the quality of the results obtained is highly sensitive to variations of this parameter. We experimented 5 different starting points for each of the test problems. The results plotted in the figures of Section 5 correspond to the best run of the programme.

4.2 GLOBAL

The GLOBAL subroutine is provided as a source code written in Fortran available for MS-Dos and Unix environments. The user can set the value of two parameters : `NSAMPL`, the number of sample points generated, and `NSEL`, the number of points selected as starting points for the local search. The default (or rather, recommended) values of these parameters are 100 and 2 respectively. When using GLOBAL, two options are possible : `local` and `unirandi`. The user's manual recommends the former in the case of a smooth objective function and the latter for non-smooth objective functions.

In fact, computational experimentation reveals that, for problems involving fewer than 5 parameters, best results are obtained with `unirandi`, whereas `local` yields better results for problems in 5 dimensions and above, and this, irrespective of whether the objective function is smooth or not. Furthermore, for dimensions above 12, `unirandi` does not even terminate. We can also confirm that the default values for the parameters `NSAMPL` and `NSEL` yield good results. For the PF- and MS-type test problems, the results obtained are moreover not sensitive to variations of the values of these two parameters.

Note that the number of objective-function evaluations cannot be specified a priori. Moreover, due to its random components, results obtained with GLOBAL (consequently) differ from one execution to another. Also, we performed ten trials for each test problem in order to generate the results displayed in Section 5. Note also that better values of the objective functions are obtained for Problems lms3 and lms5 with values of `NSAMPL` and `NSEL` different from the default values (we report separately in Section 5 results obtained with `unirandi`, `NSAMPL`=300, and `NSEL`=10 for Problem lms3, and also with `local`, `NSAMPL`=200, and `NSEL`=10 for Problem lms5—cf. Figures 6 and 8, respectively).

4.3 GAS

This genetic algorithm is written in C++ and is available for MS-Dos and Unix environments. It aims at separating an initial *population* into *subpopulations*, each of which will ultimately be centered around a local optimum. Six input parameters can be set by the user. These parameters are however not independent: a randomly-chosen set of parameter values may well yield an empty set of final solutions. In such a case, one has to increase the value of the `EVALS` parameter (number of objective-function evaluations), or to decrease the value of the `MAXLEVEL` parameter (an element of the interval [1,8], proportional to the *diameter* of the greatest subpopulation).

A disadvantage of this software is that for each specific dimension, N, of the search space, one must modify some files, and obtain an executable file that can then be used for optimizing objective functions in N dimensions. In our computational experimentation, we first used the default values of the input parameters, but we often had to choose other values so as to generate results involving a moderate number of objective-function evaluations. Again, due to its random components, results obtained with GAS differ from one execution to another.

4.4 GOT

The source code of GOT has been provided to us by the author. It is available for Fortran-77 and Fortran-90 HP UX compilers, and also for Microsoft Fortran 5.0/5.1 and Microsoft Fortran Power Station 1.0 compilers.

For GOT, the number of objective-function evaluations used is automatically determined, strictly based on the dimension of the problem (and cannot be otherwise imposed). In fact, in order to prevent the programme from running for too long, we limited the parameter `ks` (maximal number of iterations in subroutine `rmin`) to 10, instead of its default value 500. Setting this parameter to its default value makes the programme run much too long, and without improving the results. Again, due to its random components, results obtained with GOT differ from one execution to another.

4.5 INTGLOB

This software product is written in Fortran. It was provided to us by its authors under the form of a diskette containing executable files that can be run on MS-DOS environment with a math co-processor. The version we have is restricted to problems of dimension less than or equal to 20. It is the easiest software to use among the six under study: the user has no input parameter value to provide, except for a stopping criterion value which controls the number of objective-function evaluations.

The best x found so far, and the corresponding value of the objective function are progressively displayed as the programme runs.

4.6 UFO

UFO is written in Microsoft Fortran 5.0 for PC 386/486/586. We shall report in Section 5 only results corresponding to the following three variants/methods proposed by UFO: the so-called *class-1 type-1* method (random search method in which a local search is started from the point with lowest objective-function value) to which we shall henceforward refer as UFO-a; *class-3 type-2* method (single-linkage clustering) which we call UFO-b; and *class-4 type-3* method (multi-level single-linkage), the default option of UFO, referred to as UFO-c in the current paper. We discarded the other variants of UFO. They either bugged or did not yield as good results.

Apart from the various types of global optimization methods that can be selected, the user must specify the values of three input parameters. Firstly, `MNLMIN`, the maximal number of local minima one wishes to obtain, secondly, `MFV`, the maximal number of objective-function evaluations per iteration, and finally, `MNRND`, the number of randomly-generated points. For a problem of dimension n, the default values of these three parameters are respectively $50 + 20n$, 1000, and $100 + 20n$. In the numerical results we are reporting in Section 5, we set these parameters to the following values: `MNLMIN`=20, as in GLOBAL, otherwise the number of objective-function evaluations is excessive (in fact, we make this value vary in order to obtain the different results plotted in the figures of Section 5 for the UFO-b and UFO-c variants); `MFV`=3000, so as avoiding this bound to be reached, even for high-dimensional problems (if this bound is attained, the algorithm restarts from scratch while cumulating the total number of objective-function evaluations); and `MNRND`=100, as for GLOBAL, since with such a value we obtained better results than with the default value (better or as good final objective-function values with fewer function evaluations). In fact, for the LMS-type problems (lower dimensional), better results were obtain with the default value of `MNRND`, so we shall report numerical results with `MNRND`$=100+20n$ for the LMS-type test problems.

5 Comparative Results

Each of Figures 2 to 14 compares the performance of the six software products on a single test problem. It displays the objective-function value found by every software product in terms of the number of objective-function evaluations required. One can hence appreciate, for a given number of objective-function evaluations, which software product obtains the best (lowest) value of the objective function.

We use a broken line to show the results obtained with ASA (not reported for Problem pf4, due to the great number of objective-function evaluations required [out of the range of the figure]).

Crosses display results obtained by Csendes' GLOBAL subroutine (and marked "C"). The result of one trial of GLOBAL (out of ten) cannot be represented on the figure for Problem lms3, because of the great number of objective-function evaluations required (out of the range of the figure).

INTGLOB is represented by a plain line.

Squares are used for the results obtained by GAS.

We represent the results of GOT with circles. No results from GOT are reported for the MS-type test problems, since this software product requires too many objective-function evaluations for these instances (out of the range of the figure).

Finally, the diamonds corresponds to specific options of the UFO software. "Ua" corresponds to UFO-a (random search method in which a local search is started from the point with lowest objective-function value). No results from this variant of UFO are reported for Problem lms2, since it required too many objective-function evaluations for this instance (out of the range of the figure). "Ub" corresponds to UFO-b (single-linkage clustering). "Uc" means UFO-c (multi-level single-linkage). When the results obtained via UFO-b and UFO-c are not single points on the figures, we use a broken line to display the results (and the numbers above the broken line correspond to values of the parameter `MNLMIN`).

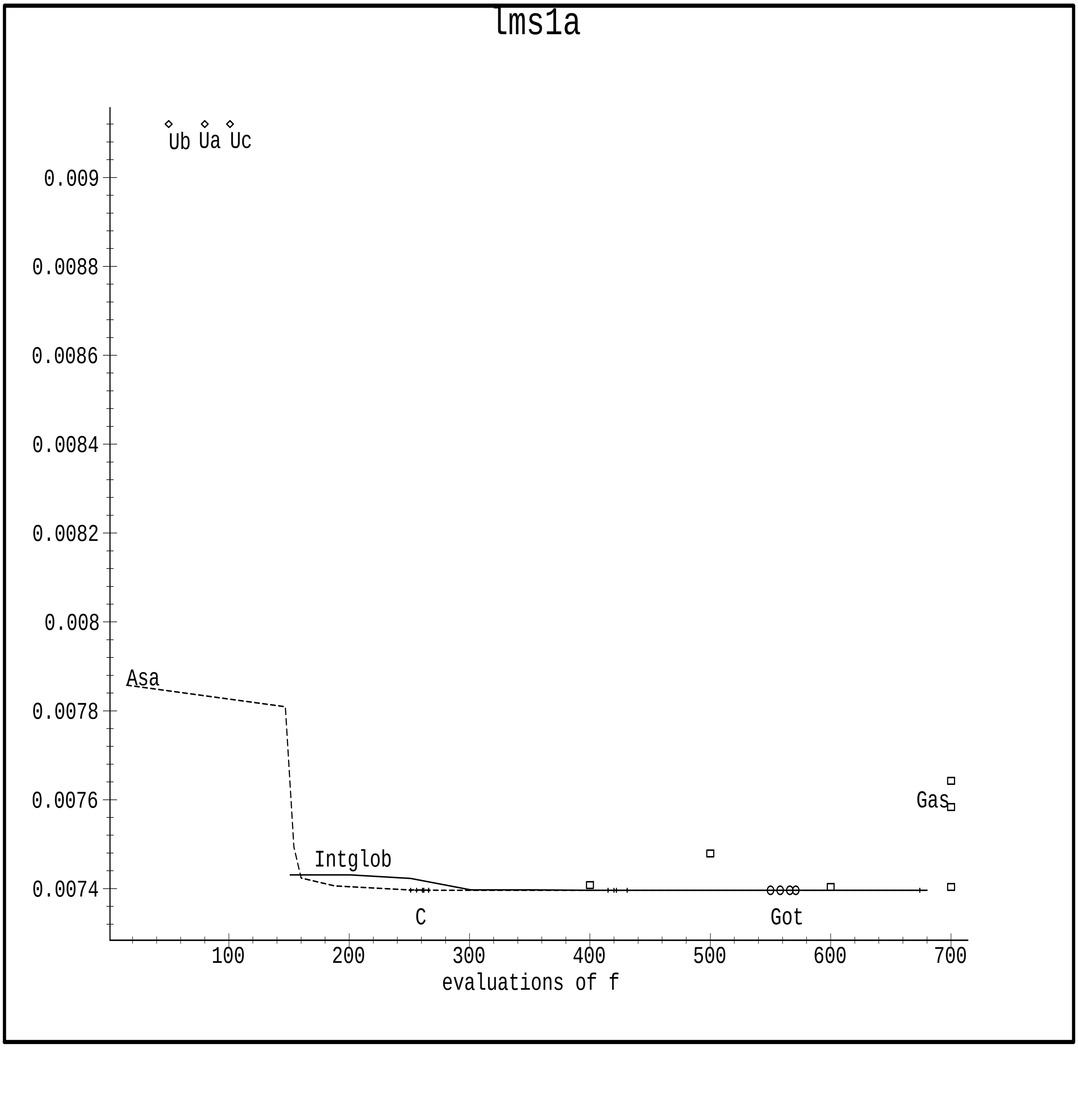

Figure 2: Comparative progress on Problem lms1a

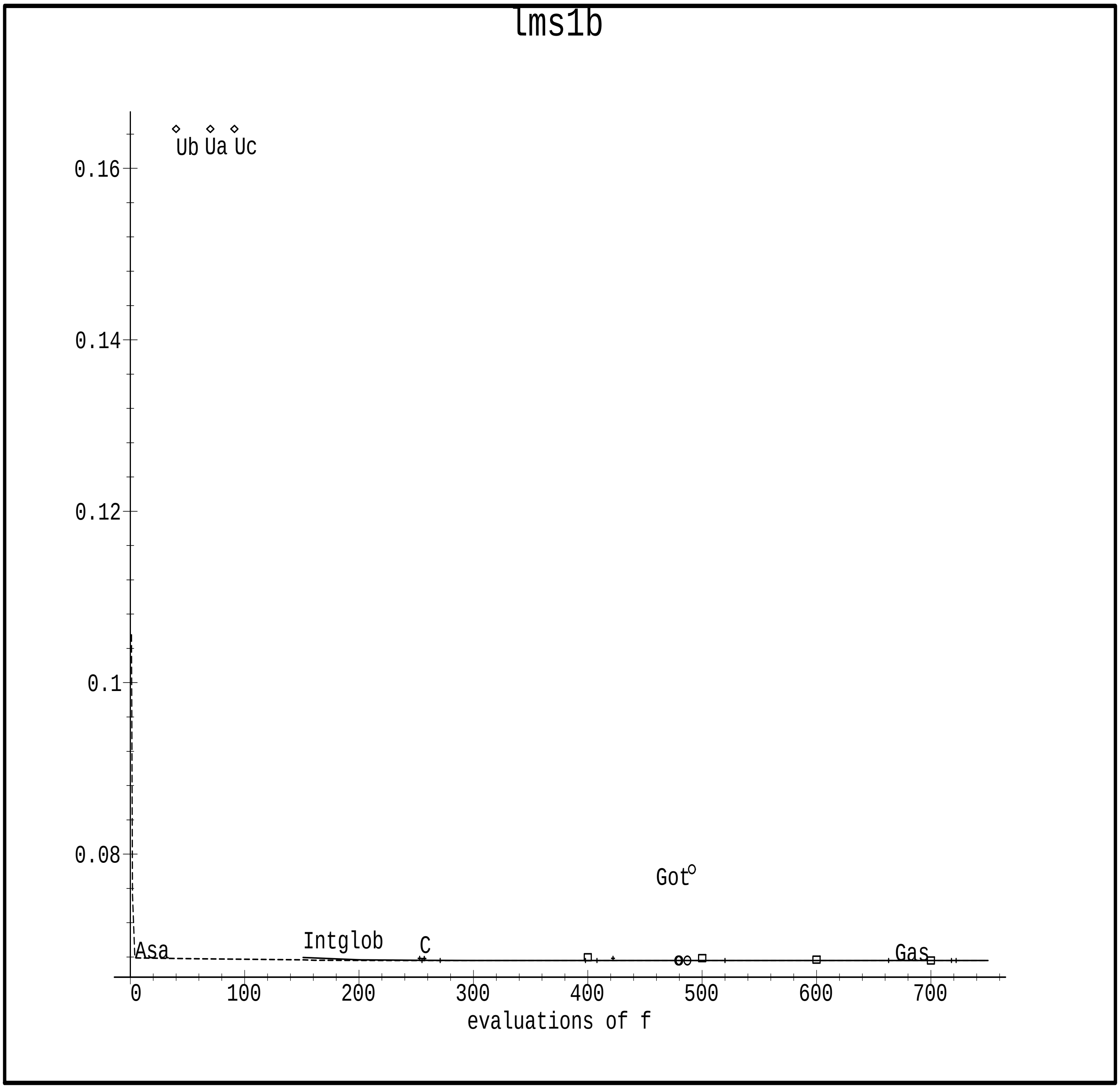

Figure 3: Comparative progress on Problem lms1b

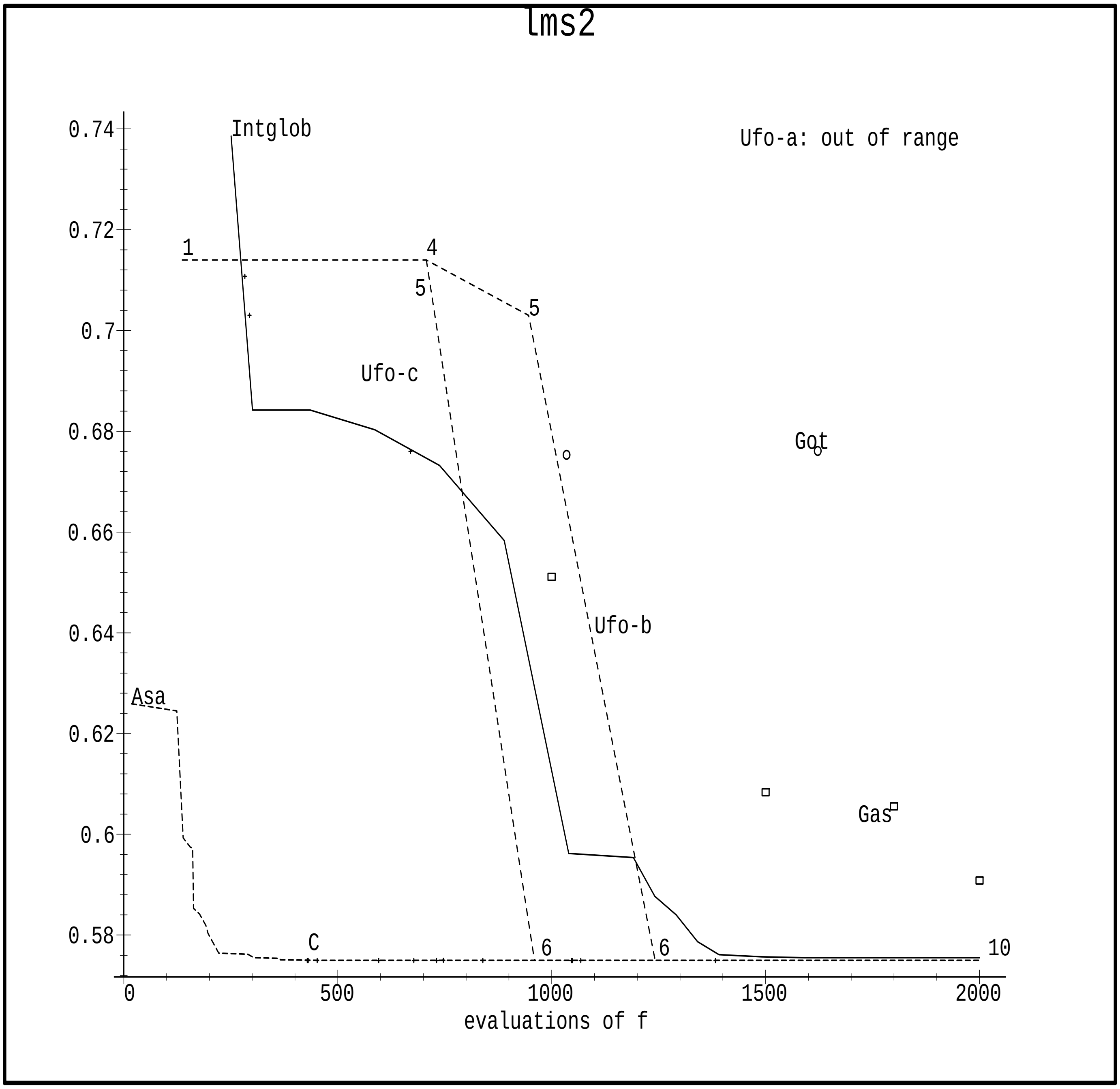

Figure 4: Comparative progress on Problem lms2

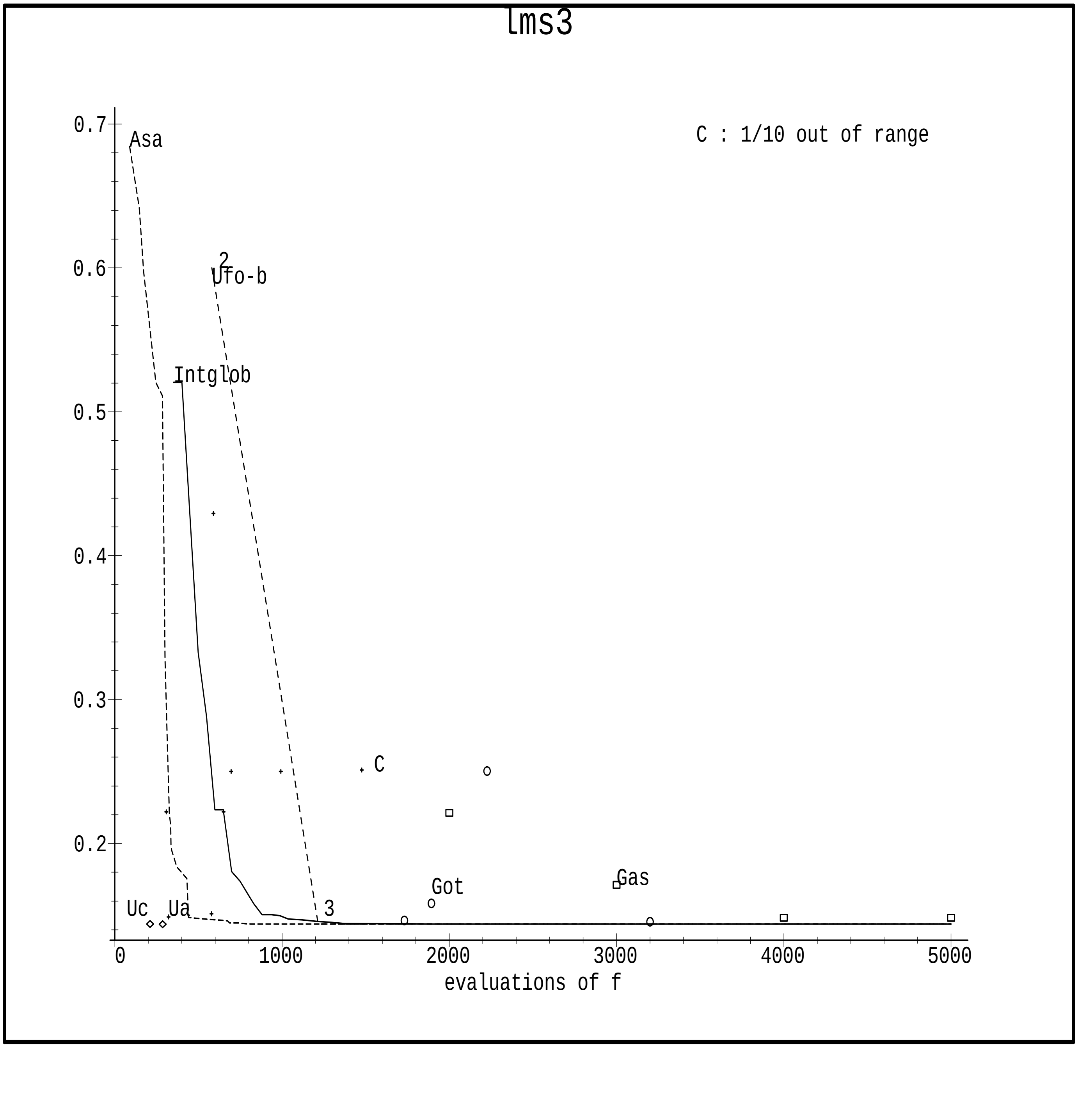

Figure 5: Comparative progress on Problem lms3

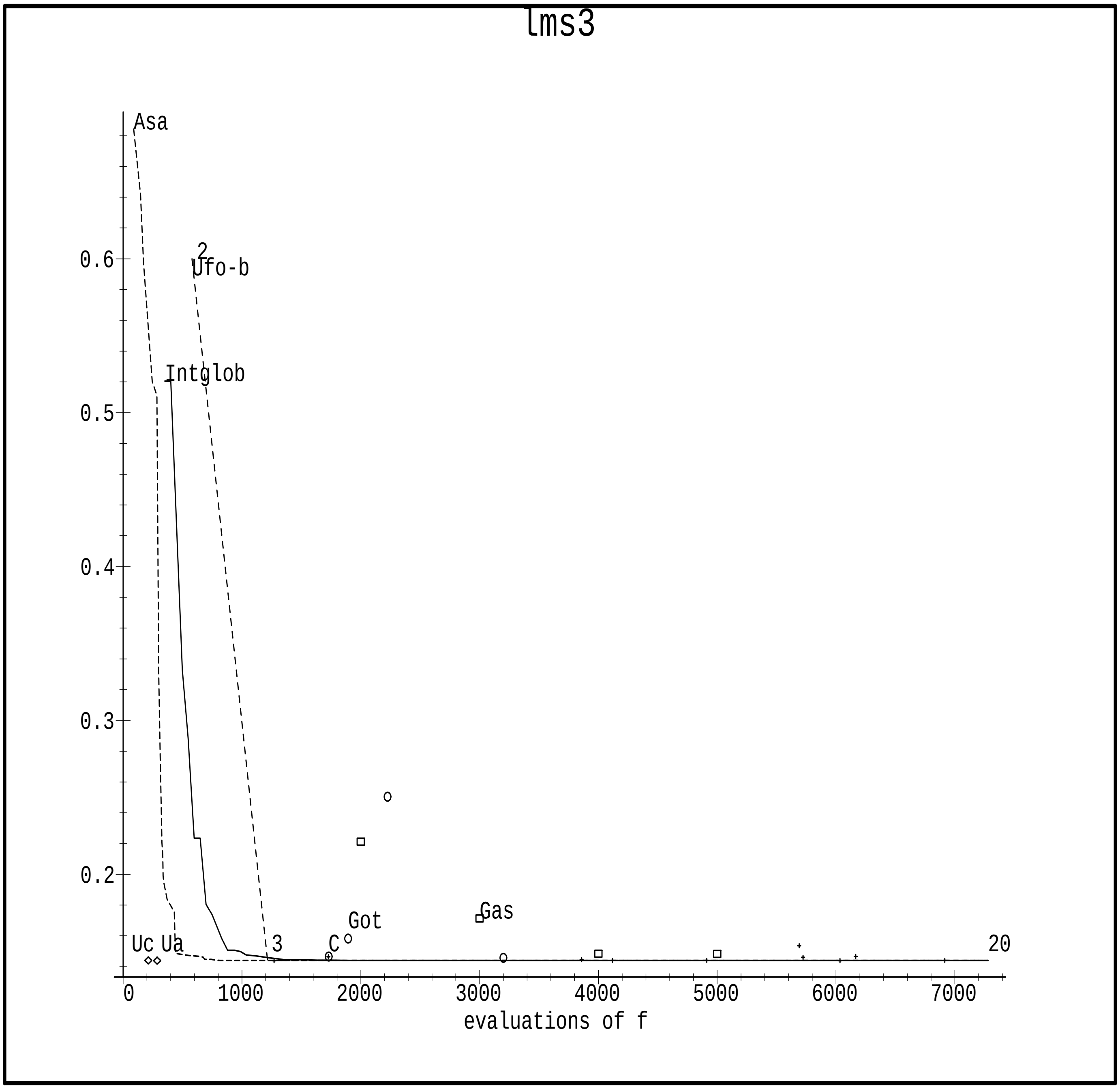

Figure 6: Comparative progress on Problem lms3 with non-default parameter values for GLOBAL

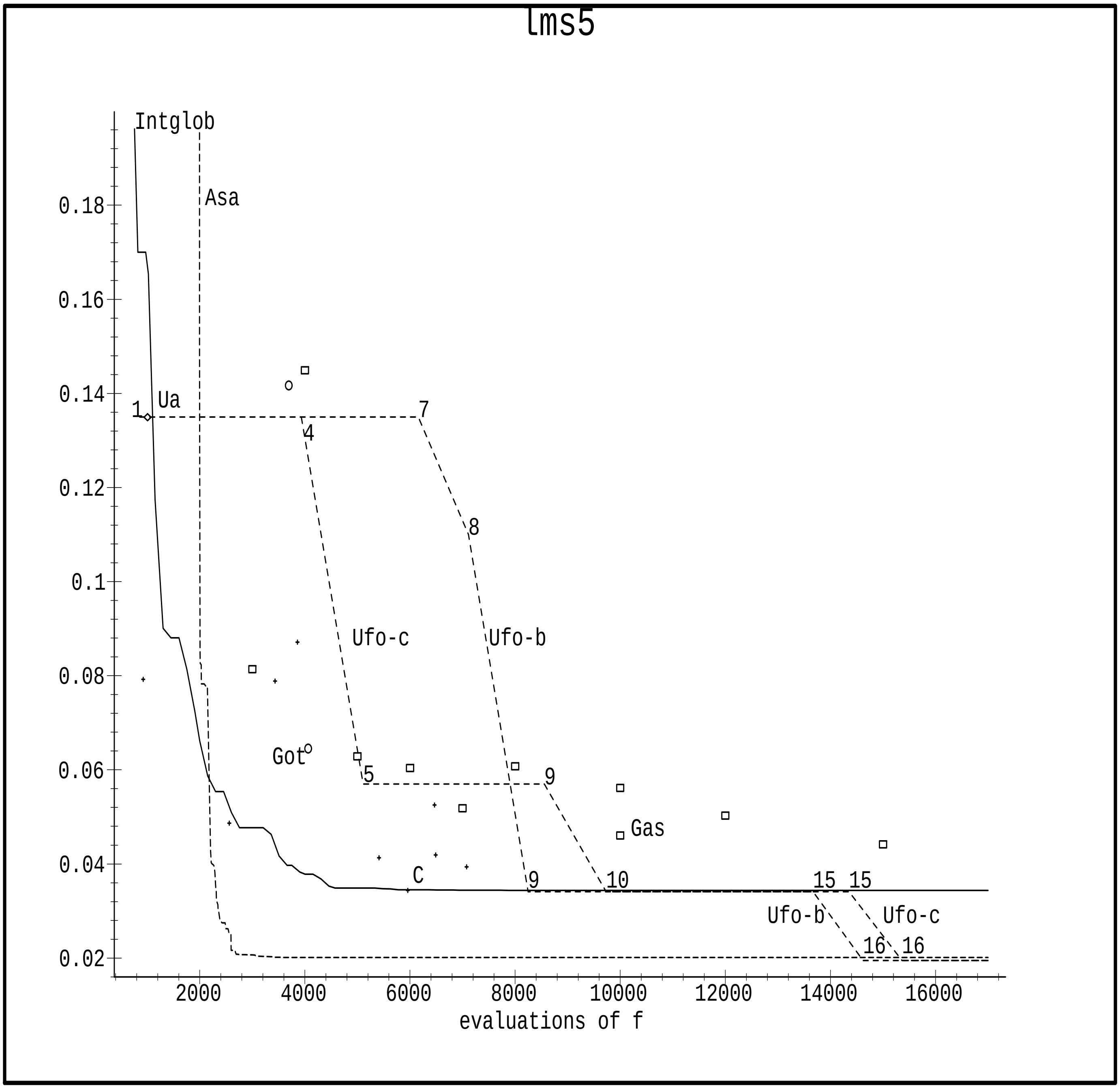

Figure 7: Comparative progress on Problem lms5

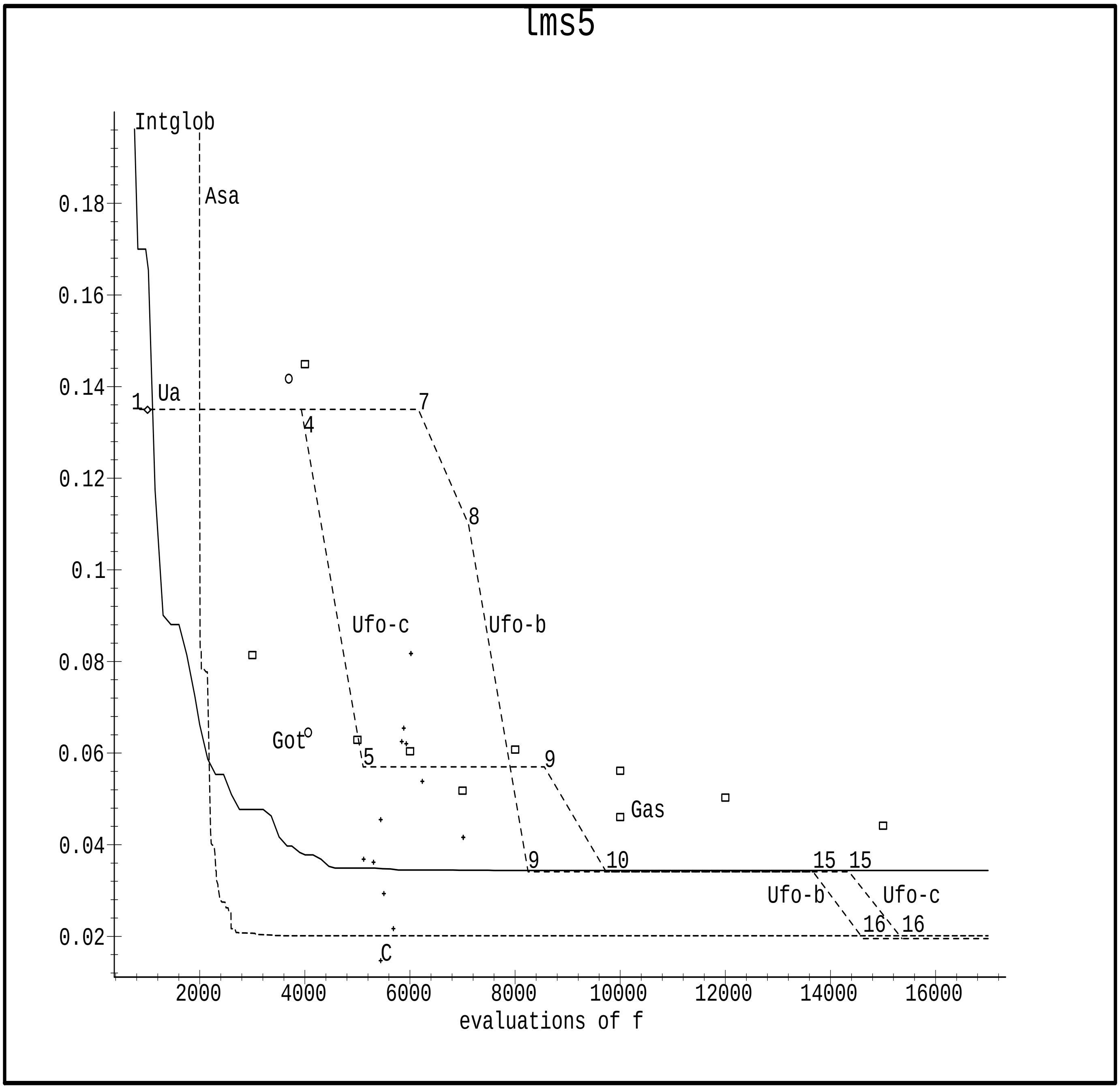

Figure 8: Comparative progress on Problem lms5 with non-default parameter values for GLOBAL

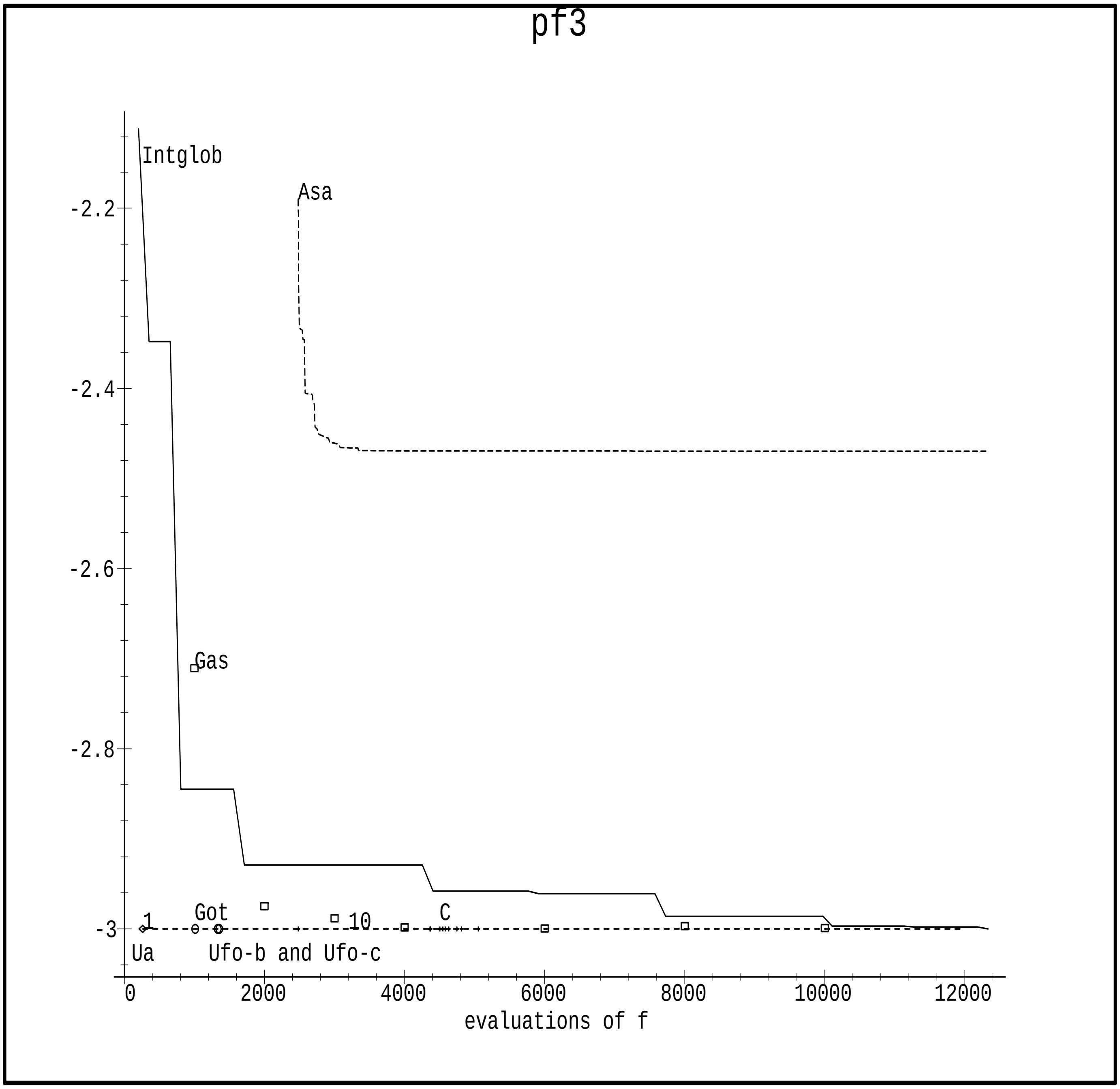

Figure 9: Comparative progress on Problem pf3

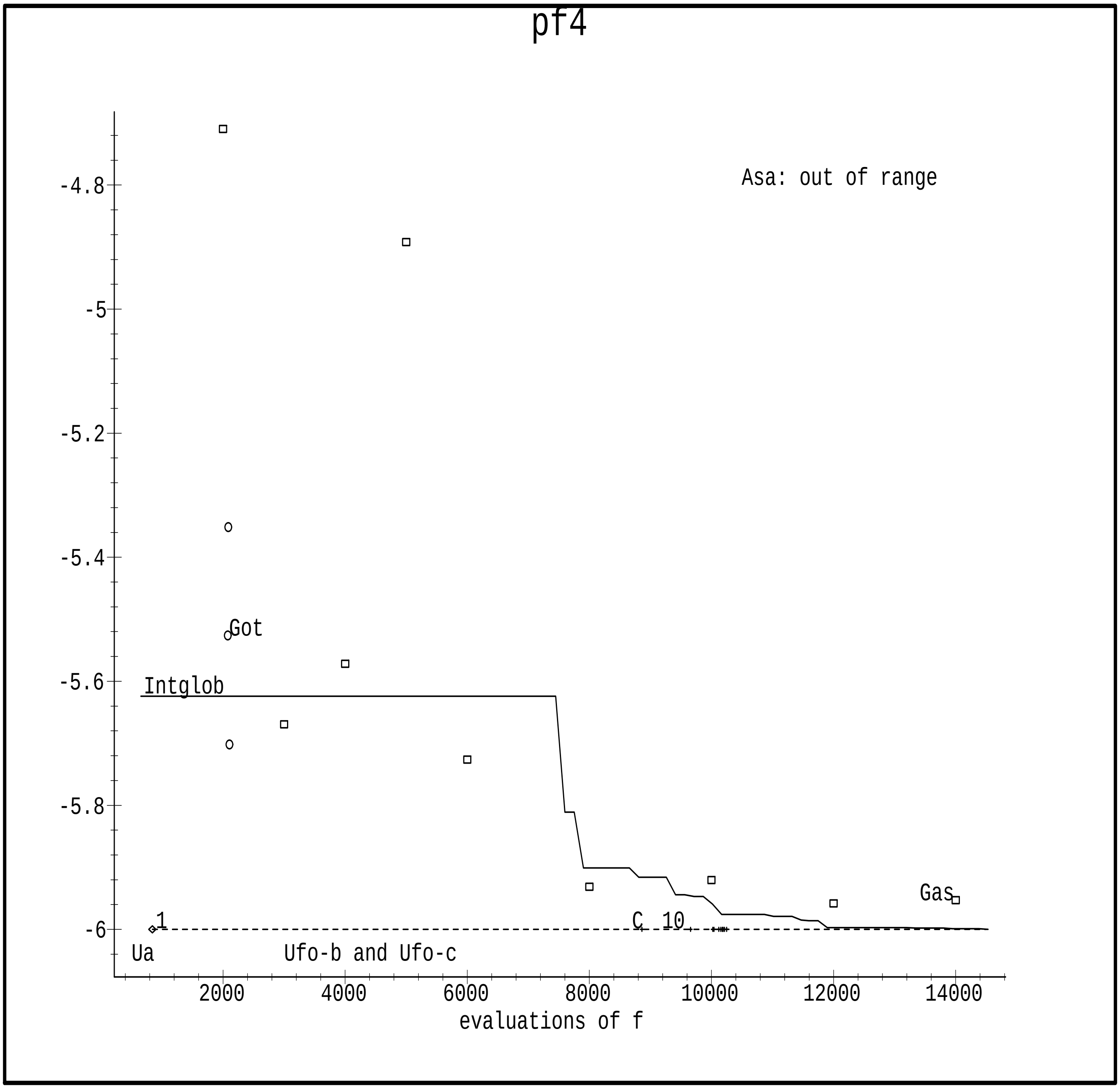

Figure 10: Comparative progress on Problem pf4

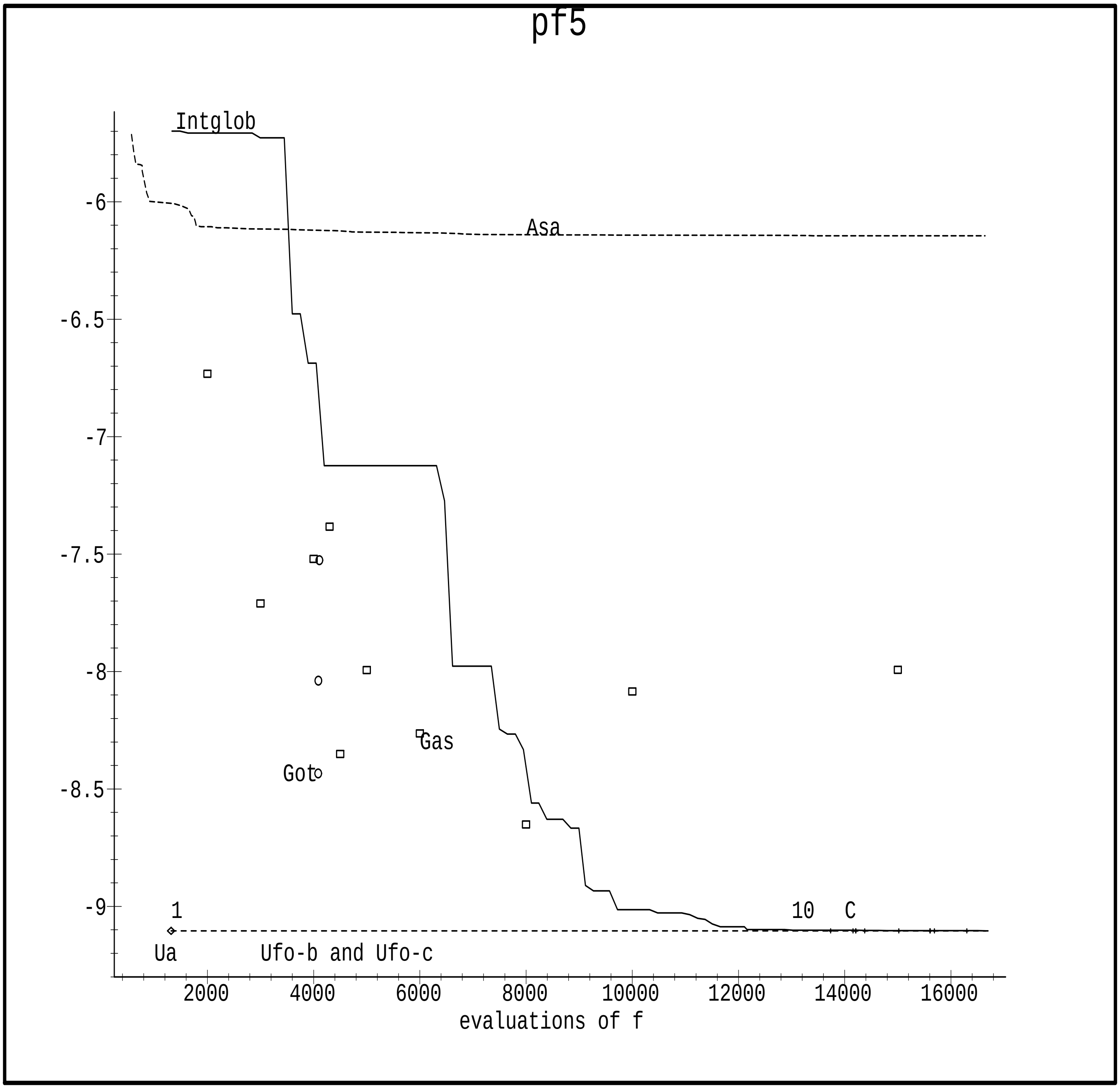

Figure 11: Comparative progress on Problem pf5

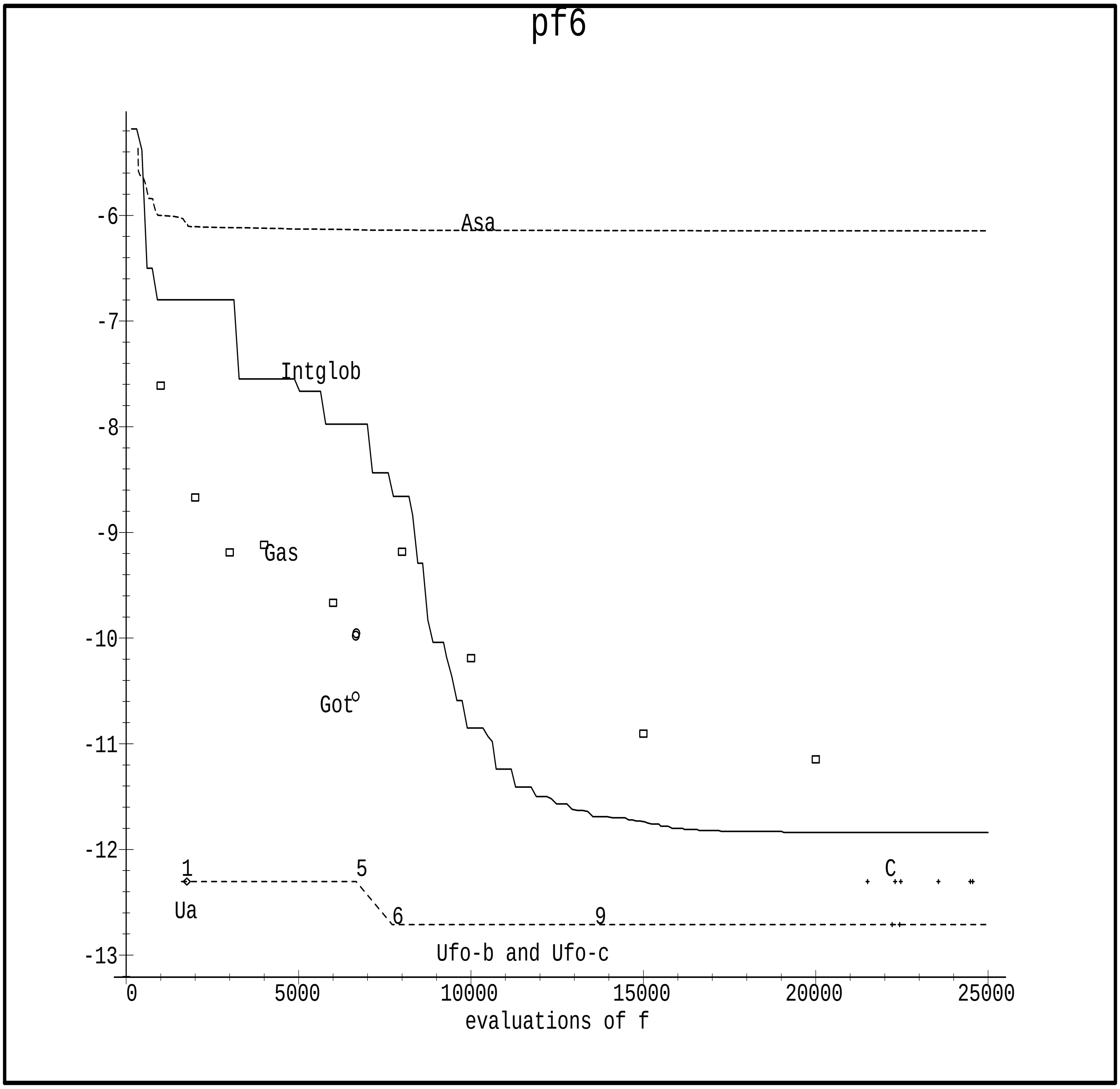

Figure 12: Comparative progress on Problem pf6

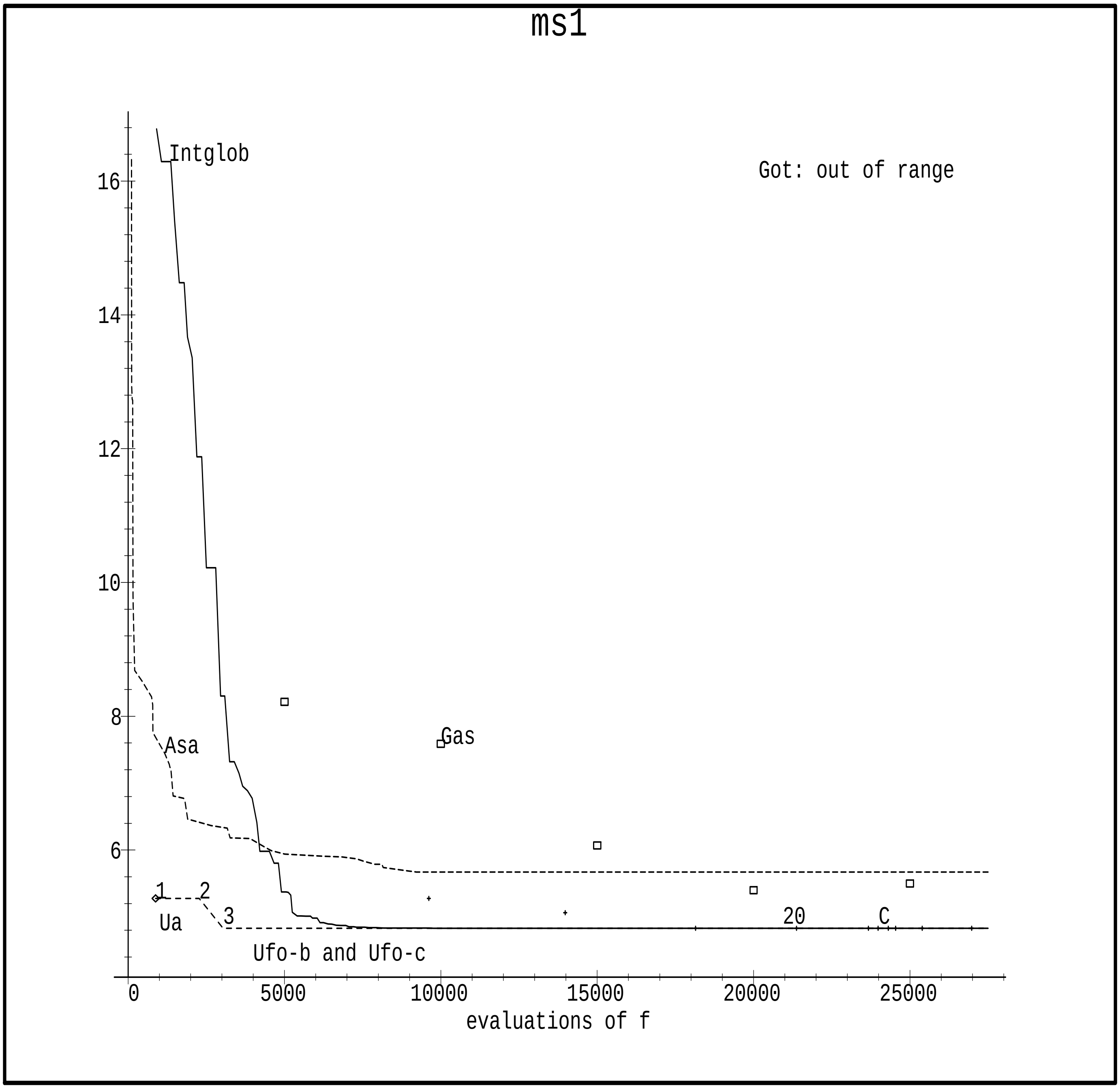

Figure 13: Comparative progress on Problem ms1

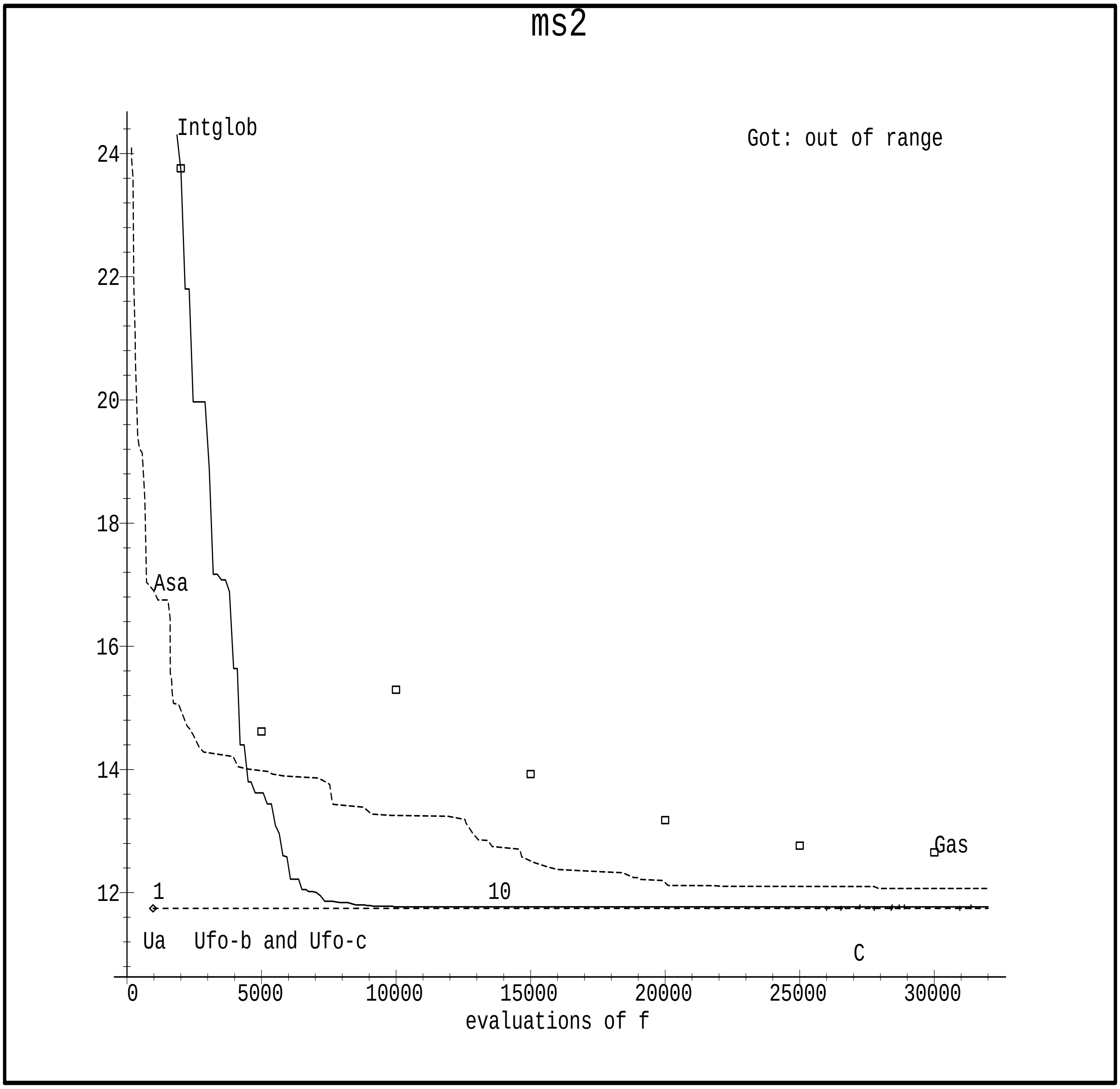

Figure 14: Comparative progress on Problem ms2

Inspection of Figures 2 to 8 reveals that for our test problems of dimension from 1 to 5 (the LMS-type problems), ASA yields globally the best results with respect to both the value of the function to be minimized, and to the number of objective-function evaluations. UFO finds the best value of the objective function for Problem lms5 (5 dimensions) but only after 14 000 evaluations. GLOBAL obtains *sometimes* this value, and within 5 500 evaluations. For this problem, note that INTGLOB yields a much better objective-function value than that found by ASA, if one does not allow more than 2 000 objective-function evaluations. Note finally that Figures 6 and 8 differ from Figures 5 and 7 only with respect to the results of GLOBAL. Indeed, these latter figures display the same results, except for the fact that GLOBAL was run with parameter values different from their default values, as mentioned in Section 4.2. With such parameter values, one notes for example that GLOBAL is the only software product to reach an objective-function value as low as 0.01472 (within 5442 evaluations) for Problem lms5. The second best objective-function value obtained for this problem is 0.01948 found by UFO (within 14 621 evaluations !).

For higher-dimensional problems (PF- and MS-type test problems), ASA completely looses its supremacy. Figures 9 to 14 show indeed that UFO obtains, by far, the best results: it systematically finds the best objective-function value, and within a number of evaluations between 2 to 17 times smaller than its closest competitor.

GAS can find objective-function values similar to the other software products only in the low-dimensional instances, and this, at the expense of substantial additional effort (it requires much more evaluations). GOT obtains results roughly comparable to the other software products only for Problems lms1a, lms1b and pf3 (1-, 1-, and 12-dimensional problems, respectively). GAS nonetheless reached objective-function values as good as the other software products for Problems ms1 and ms2 (dimension 20) but within an *excessive* (at least relative to the performance of the other methods) number of function evaluations, respectively 70 000 and 72 000 objective-function evaluations! (that is why these results could not fit on Figures 13 and 14).

6 Conclusions

We first proposed in this paper a set of practice-oriented test problems for “continuous” (i.e. non-discrete) unconstrained global optimization, which are less academic than the ones often used in the literature for evaluating new global optimization methods. The eleven test problems are of dimension ranging from 1 to 20, involve smooth, non-differentiable, and even discontinuous objective functions, and some of these objective functions have an exponential number of local minima.

We then selected six public-domain software products: ASA, GLOBAL, INTGLOB, GAS, GOT, and UFO, which are using techniques including integral global optimization, genetic algorithms, simulated annealing, clustering, random search, continuation, Bayesian, tunneling, and multi-level methods, i.e. covering most global optimization techniques that can be applied to solve *black box* type problems (neither the structure nor the analytic form of the objective function can be exploited).

We performed numerical comparisons of these software products on each of the test problems in view of determining which ones are more appropriate for solving global optimization problems involving costly evaluations of the objective function. For this purpose, we monitored the progress made by each software product on each single test problem in terms of the number of objective-function evaluations required.

Based on our (limited) computational experiment, one would draw the following conclusions. First, for solving low-dimensional problems (up to 5 dimensions), if one is interested in an easy software (without any parameter tuning), then INTGLOB seems appropriate. Otherwise, ASA would be recommended. However, if it is crucial to obtain the *best* possible optimal value, *and* if one can afford to run several times a software product for the same problem (yielding thereby more function evaluations), then GLOBAL appears to be interesting. Secondly, for higher-dimensional problems (up to 20 dimensions), UFO proved to be the most efficient, at least according to our modest amount of experience.

We must indeed emphasize again the fact that our study is not exhaustive: we did not systematically optimize the various (often numerous) input parameters specifying the different techniques, the inherent structure of the specific test problems selected might have favoured some software product, etc. Clearly, for a specific class of problems, one may well attempt to optimize the value of the input parameters of a specific software product on a sample set of problems of this class. We would then expect this software product, with the optimal parameter values, to perform well on a new problem of this specific class of problems. This is particularly crucial for a software product, such as ASA, involving a lot of degrees of freedom in its parameter setting, and whose performance is highly dependent upon such input-parameter values. This a priori input-parameter optimization problem is thereby extensive, since it is a non-trivial global optimization problem by itself! (Furthermore, it appears from our computational experience that such input-parameter tuning often needs not only to be performed for each class of problems but in fact also for each specific instance!)

Our comparative study nonetheless provides some preliminary comparison which is, above all, independent (we did not contribute to any of the software products considered in the current paper).

Not surprisingly, considering the difficulty of the general continuous global optimization problem, the numbers of objective-function evaluations required by any of the software products remain far too high for most industrial applications. In such non-academic contexts, the optimization problem is indeed often only a *subproblem*, to be solved within a more general system.

Further, our experience leads us to be skeptical about the possibility of developing an efficient general-purpose global optimization method that would neither exploit any structure nor any a priori knowledge of a problem.

Acknowledgement

The authors would like to thank Richard Epenoy and Paul Legendre from the Centre National d'Études Spatiales in Toulouse who prompted this study in 1994-95, and brought to our attention least median of squares regression as a source of test problems.

A Test Problem Data

Problem ms1:

$$(\delta_{ij}) = \begin{pmatrix} 0 & 0.49 & 1.74 & 2.26 & 2.14 & 1.85 & 1.62 & 0.79 & 1.87 & 1.85 \\ 0.49 & 0 & 1.47 & 2.08 & 1.84 & 1.39 & 1.43 & 0.82 & 1.69 & 1.50 \\ 1.74 & 1.47 & 0 & 2.67 & 0.84 & 1.32 & 1.70 & 1.59 & 1.66 & 1.67 \\ 2.26 & 2.08 & 2.67 & 0 & 2.67 & 1.92 & 1.74 & 1.96 & 1.76 & 2.28 \\ 2.14 & 1.84 & 0.84 & 2.67 & 0 & 1.39 & 1.39 & 2.04 & 2.01 & 1.38 \\ 1.85 & 1.39 & 1.32 & 1.92 & 1.39 & 0 & 1.48 & 1.78 & 1.47 & 1.11 \\ 1.62 & 1.43 & 1.70 & 1.74 & 1.39 & 1.48 & 0 & 1.66 & 1.84 & 1.06 \\ 0.79 & 0.82 & 1.59 & 1.96 & 2.04 & 1.78 & 1.66 & 0 & 1.68 & 2.11 \\ 1.87 & 1.69 & 1.66 & 1.76 & 2.01 & 1.47 & 1.84 & 1.68 & 0 & 2.97 \\ 1.85 & 1.50 & 1.67 & 2.28 & 1.38 & 1.11 & 1.06 & 2.11 & 2.97 & 0 \end{pmatrix}$$

Problem ms2:

$$(\delta_{ij}) = \begin{pmatrix} 0 & 1.27 & 1.69 & 2.04 & 3.09 & 3.20 & 2.86 & 3.17 & 3.21 & 2.38 \\ 1.27 & 0 & 1.43 & 2.35 & 3.18 & 3.22 & 2.56 & 3.18 & 3.18 & 2.31 \\ 1.69 & 1.43 & 0 & 2.43 & 3.26 & 3.27 & 2.58 & 3.18 & 3.18 & 2.42 \\ 2.04 & 2.35 & 2.43 & 0 & 2.85 & 2.88 & 2.59 & 3.12 & 3.17 & 1.94 \\ 3.09 & 3.18 & 3.26 & 2.85 & 0 & 1.55 & 3.12 & 1.31 & 1.70 & 2.85 \\ 3.20 & 3.22 & 3.27 & 2.88 & 1.55 & 0 & 3.06 & 1.64 & 1.36 & 2.81 \\ 2.86 & 2.56 & 2.58 & 2.59 & 3.12 & 3.06 & 0 & 3.00 & 2.95 & 2.56 \\ 3.17 & 3.18 & 3.18 & 3.12 & 1.31 & 1.64 & 3.00 & 0 & 1.32 & 2.91 \\ 3.21 & 3.18 & 3.18 & 3.17 & 1.70 & 1.36 & 2.95 & 1.32 & 0 & 2.97 \\ 2.38 & 2.31 & 2.42 & 1.94 & 2.85 & 2.81 & 2.56 & 2.91 & 2.97 & 0 \end{pmatrix}$$

	lms1a		lms2			lms3				lms5					
i	x_{i1}	y_i	x_{i1}	x_{i2}	y_i	x_{i1}	x_{i2}	x_{i3}	y_i	x_{i1}	x_{i2}	x_{i3}	x_{i4}	x_{i5}	y_i
1	50	0.44	7	5.60	16.68	80	27	89	42	3.83	28.87	7.20	26.60	6.19	37.01
2	51	0.47	3	2.20	11.50	80	27	88	37	2.89	20.10	-11.71	24.40	5.17	26.51
3	52	0.47	3	3.40	12.03	75	25	90	37	2.86	69.05	12.32	25.70	7.04	36.51
4	53	0.59	4	0.80	14.88	62	24	87	28	2.92	65.40	14.28	25.70	7.10	40.70
5	54	0.66	6	1.50	13.75	62	22	87	18	3.06	29.59	6.31	25.40	6.15	37.10
6	55	0.73	7	3.30	18.11	62	23	87	18	2.07	44.82	6.16	21.60	6.41	33.90
7	56	0.81	2	1.10	8.00	62	24	93	19	2.52	77.37	12.70	24.90	6.86	41.80
8	57	0.88	7	2.10	17.83	62	24	93	20	2.45	24.67	-0.17	25.01	5.78	33.40
9	58	1.06	30	14.60	79.24	58	23	87	15	3.13	65.01	9.85	26.60	6.51	41.01
10	59	1.20	5	6.05	21.50	58	18	80	14	2.44	9.99	-0.05	28.01	5.57	37.20
11	60	1.35	16	6.88	40.33	58	18	89	14	2.09	12.20	-12.86	23.51	5.62	23.30
12	61	1.49	10	2.15	21.00	58	17	88	13	2.52	22.55	0.92	23.60	5.34	35.20
13	62	1.61	4	2.55	13.50	58	18	82	11	2.22	14.30	4.77	24.51	5.80	34.90
14	63	2.12	6	4.62	19.75	58	19	93	12	2.67	31.79	-0.96	25.80	6.19	33.10
15	64	11.90	9	4.48	24.00	50	18	89	8	2.71	11.60	-16.04	25.20	5.62	22.70
16	65	12.40	10	7.76	29.00	50	18	86	7	3.14	68.47	10.62	25.01	6.94	39.70
17	66	14.20	6	2.00	15.35	50	19	72	8	3.54	42.64	2.66	25.01	6.33	31.80
18	67	15.90	7	1.32	19.00	50	19	79	8	2.52	16.70	-10.99	24.80	6.01	31.70
19	68	18.20	3	0.36	9.50	50	20	80	9	2.68	86.27	15.03	25.51	7.51	43.10
20	69	21.20	17	7.70	35.10	56	20	82	15	2.37	76.73	12.77	24.51	6.96	41.01
21	70	4.30	10	1.40	17.90	70	20	91	15						
22	71	2.40	26	8.10	52.32										
23	72	2.70	9	4.50	18.75										
24	73	2.90	8	6.35	19.83										
25			4	1.50	10.75										

Table 2: Problems lms1a, lms2, lms3 and lms5

i	x_{i1}	y_i	i	x_{i1}	y_i
1	4.37	5.23	25	4.38	5.02
2	4.56	5.74	26	4.42	4.66
3	4.26	4.93	27	4.29	4.66
4	4.56	5.74	28	4.38	4.90
5	4.30	5.19	29	4.22	4.39
6	4.46	5.46	30	3.48	6.05
7	3.84	4.65	31	4.38	4.42
8	4.57	5.27	32	4.56	5.10
9	4.26	5.57	33	4.45	5.22
10	4.37	5.12	34	3.49	6.29
11	3.49	5.73	35	4.23	4.34
12	4.43	5.45	36	4.62	5.62
13	4.48	5.42	37	4.53	5.10
14	4.01	4.05	38	4.45	5.22
15	4.29	4.26	39	4.53	5.18
16	4.42	4.58	40	4.43	5.57
17	4.23	3.94	41	4.38	4.62
18	4.42	4.18	42	4.45	5.06
9	4.23	4.18	43	4.50	5.34
20	3.49	5.89	44	4.45	5.34
21	4.29	4.38	45	4.55	5.54
22	4.29	4.22	46	4.45	4.98
23	4.42	4.42	47	4.42	4.50
24	4.49	4.85			

Table 3: Problem lms1b

References

[1] A. Alfakih, A. Khandani, and H. Wolkowicz. Solving Euclidean distance matrix completion problems via semidefinite programming. *Computational Optimization and Applications*, 12:13–30, 1998.

[2] C. G. Boender, A. H. G. Rinnooy Kan, G. T. Timmer, and L. Stougie. A stochastic method for global optimization. *Mathematical Programming*, 22:125–140, 1982.

[3] T. Csendes. Global. Jozsef Attila University, Szeged, Hungary. ftp://ftp.jate.u-szeged.hu/pub/math/optimization/fortran/.

[4] S. B. Hamma. *Étude de méthodes numériques d'optimisation globale*. PhD thesis, Université Paul Sabatier, Toulouse, France, 1992.

[5] L. Ingber. Adaptive simulated annealing (ASA), 1993. Lester Ingber Research, Chicago, URL http://www.ingber.com/#ASA-CODE.

[6] M. Jelasity and J. Dombi. GAS, a method for global optimization and heuristic search. Jozsef Attila University, Szeged, Hungary. ftp://ftp.jate.u-szeged.hu/pub/math/optimization/GAS/.

[7] A. V. Kuntsevich. Fortran-77 and fortran-90 global optimization toolbox: User's guide. Technical report, Institut für Mathematik, Karl Franzens Universität, A-8010 Graz, Austria, 1995.

[8] S. Lucidi and M. Sciandrone. On the global convergence of derivative free methods for unconstrained optimization. Technical Report 32-96, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1996.

[9] L. Lukšan, M. Tuma, M. Šiška, J. Vlček, and N. Ramešová. Interactive system for universal functional optimization (UFO). Technical Report V-701, Institute of Computer Science, Academy of Science of the Czech Republic, Prague, 1996.

[10] R. Mathar and A. Žilinskas. A class of test functions for global optimization. *Journal of Global Optimization*, 5:195–199, 1994.

[11] M. Mongeau, H. Karsenty, V. Rouzé, and J.-B. Hiriart-Urruty. Comparison of public-domain software for black box global optimization. Technical Report LAO 98-01, Université Paul Sabatier, Toulouse, 1998.

[12] NA Digest. Electronic publication of NA-Net. See web site http://www.netlib.org/na-net/.

[13] A. Neumaier. Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review*, 39(3):407–460, 1997.

[14] Opt-Net Digest. Electronic publication of SIGOPT. See web site http://www.informatik.uni-koeln.de/opt-net.

[15] J. D. Pintér. Continuous global optimization software: A brief review. *Optima*, 52:1–8, 1996.

[16] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* John Wiley, 1987.

[17] S. A. Vavasis. Open problems. *Journal of Global Optimization*, 4:343–344, 1994. Problem 2. Exact optima for the protein-folding test function.

[18] Q. Zheng and D. Zhuang. Integral global optimization : Algorithms, implementations and numerical tests. *Journal of Global Optimization*, 7(4):421–454, 1995.