A comparison of complete global optimization solvers

Arnold Neumaier Oleg Shcherbina Waltraud Huyer

Institut für Mathematik, Universität Wien Nordbergstr. 15, A-1090 Wien, Austria

Tamás Vinkó

Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary

Please send correspondence to: Arnold.Neumaier@univie.ac.at

Abstract. Results are reported of testing a number of existing state of the art solvers for global constrained optimization and constraint satisfaction on a set of over 1000 test problems in up to 1000 variables.

1 Overview

As the recent survey Neumaier [24] of complete solution techniques in global optimization documents, there are now about a dozen solvers for constrained global optimization that claim to solve global optimization and/or constraint satisfaction problems to global optimality by performing a complete search.

Within the COCONUT project [30, 31], we evaluated many of the existing software packages for global optimization and constraint satisfaction problems. This is the first time that different constrained global optimization and constraint satisfaction algorithms are compared on a systematic basis and with a test set that allows to derive statistically significant conclusions. We tested the global solvers BARON, GlobSol, ICOS, LGO, LINGO, OQNLP, Premium Solver, the local solver MINOS, and a basic combination strategy COCOS implemented in the COCONUT platform.

The testing process turned out to be extremely time-consuming, due to various reasons not initially anticipated. A lot of effort went into creating appropriate interfaces, making the comparison fair and reliable, and making it possible to process a large number of test examples in a semiautomatic fashion.

In a recent paper about testing local optimization software, DOLAN & MORÉ [6, 7] write: We realize that testing optimization software is a notoriously difficult problem and that there may be objections to the testing presented in this report. For example, performance of a particular solver may improve significantly if non-default options are given. Another objection is that we only use one starting point per problem and that the performance of a solver may be sensitive to the choice of starting point. We also have used the default stopping criteria of the solvers. This choice may bias results but should not affect comparisons that rely on large time differences. In spite of these objections, we feel that it is essential that we provide some indication of the performance of optimization solvers on interesting problems.

These difficulties are also present with our benchmarking studies. Section 2 describes our testing methodology. We use a large test set of over 1000 problems from various collections. Our main performance criterium is currently how often the attainment of the global optimum, or the infeasibility of a problem, is correctly or incorrectly claimed (within some time limit). All solvers are tested with the default options suggested by the providers of the codes, with the request to stop at a time limit or after the solver believed that first global solution was obtained.

These are very high standards, much more demanding than what had been done by anyone before. Thorough comparisons are indeed very rare, due to the difficulty of performing extensive and meaningful testing. Indeed, we know of only two comparative studies [17, 22] in global optimization ranging over more than perhaps a dozen examples, and both are limited to bound constrained black box optimization. (See also HUYER [14] for some further tests.)

Only recently rudimentary beginnings were made elsewhere in testing constrained global optimization [11]. On the other hand, there are a number of reports about comparing codes in local optimization [1, 4, 6, 7, 13, 16, 27], and there is an extensive web site [21] with wide-ranging comparative results on local constrained optimization codes.

Section 3 describes the tests done on the most important state of the art global solvers. Shortly expressed, the result is the following:

Among the currently available global solver, BARON is the fastest and most robust one, with LINGO and OQNLP being close. None of the current global solvers is fully reliable, with one exception: For pure constraint satisfaction problems, ICOS, while slower than BARON, has excellent reliability properties when it is able to finish the complete search.

Models in dimension < 100 can currently be solved with a success rate (global optimum found) of about 80% (BARON and OQNLP) while (within half an hour of CPU time) much fewer larger models can be solved. Overall, the stochastic solver OQNLP was most successful, but it is much slower and cannot offer information about when the search is completed.

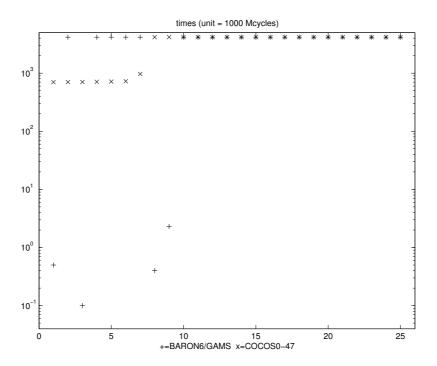


Figure 1: COCOS vs. BARON: Times for lib2s2, hard models with 10–99 variables. Marks at the upper border of the figure indicate inability to find a global minimizer within the alloted time

The best complete solvers (BARON and LINGO) were able to complete the search in about one third of the models with less than 100 variables, but BARON, the most reliable optimizer, lost in about 7 percent of the cases the global minimum.

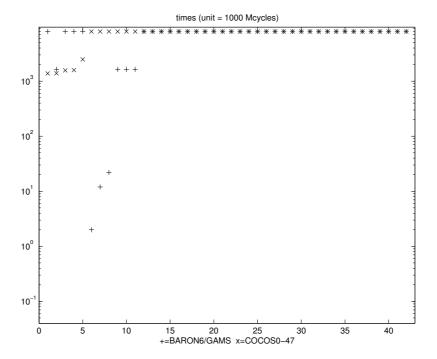


Figure 2: COCOS vs. BARON: Times for lib2s3, hard models with 100–999 variables. Marks at the upper border of the figure indicate inability to find a global minimizer within the alloted time

Section 4 contains comparisons with a basic combination strategy COCOS implemented in the COCONUT platform (SCHICHL [30, 31]). Our COCOS prototype strategy is already of comparable strength with the solvers GlobSol and Premium Solver. While not yet as good as BARON, it already solves a number of models with 100 and more variables that BARON could not solve within the allotted time (cf. Figure 2).

Since the COCONUT environment is not yet fully bug-free, COCOS was run in debug mode (printing lots of auxiliary information which slows everything down) and without any compiler optimization. Moreover, unresolved bugs in the implementation of some of the theoretically efficient techniques (exclusion boxes) caused further slowdown. We expect that the resolution of these bugs, full optimization, and better combinations of the strategies available will lead in the near future (by the end of 2004) to a package that is fully competitive with BARON, or even outperforms it. Indications for this can already be seen in Figure 1 (models are sorted by time taken by COCOS; the marks at the top of the figure indicates failure to find the global optimum): Among the models between 10 and 100 variables of library 2 that were hard in the

sense that the local solver MINOS did not find the global minimum, COCOS and BARON solved about the same number of models (but mostly different ones) within the time limit. A combination COCOS+BARON (which is in principle possible within the COCONUT environment) would almost double the efficiency.

Much more detailed results than can be given here are available online at [3].

Acknowledgments

We are happy to acknowledge cooperation with various people. Dan Fylstra (Frontline Solver), Baker Kearfott (GlobSol), Alex Meeraus (GAMS), Yahia Lebbah (ICOS), Janos Pinter (LGO), Nick Sahinidis (BARON), and Linus Schrage (LINGO) provided valuable support in obtaining and running their software. The first round of testing GlobSol was done by Boglárka Tóth (Szeged, Hungary). Mihály Markot tested and debugged preliminary versions of the COCOS strategy.

The results presented are part of work done in the context of the COCONUT project [30, 31] sponsored by the European Union, with the goal of integrating various existing complete approaches to global optimization into a uniform whole. Funding by the European Union under the IST Project Reference Number IST-2000-26063 within the FET Open Scheme is gratefully acknowledged.

2 Testing

Introduction. We present test results for the global optimization systems BARON, GlobSol, ICOS, LGO/GAMS, LINGO, OQNLP Premium Solver, for comparison the local solver MINOS, and (in the next section) a basic combination strategy COCOS implemented in the COCONUT environment. All tests were made on the COCONUT benchmarking suite described in Shcherbina et al. [32]. For generalities on benchmarking and the associated difficulties, in particular for global optimization, see Deliverable D10. Here we concentrate on the documentation of the testing conditions used and on the interpretation of the results obtained. For the interpretation of the main results, see the overview in Section 1.

The test set. A good benchmark must be one that can be interfaced with all existing solvers, in a way that a sufficient number of comparative results can

be obtained. There are various smaller-scale benchmark projects for partial domains, in particular the benchmarks for local optimization by MITTEL-MANN [21]. A very recent web site, the GAMS Global Library [12] started collecting real life global optimization problems with industrial relevance, but currently most problems on this site are without computational results. Our benchmark (described in more detail in [32]) includes most of the problems from these collections.

The test set consists of 1322 models varying in dimension (number of variables) between 1 and over 1000, coded in the modeling language AMPL [8]. They are sorted by size and source (library). Size k denotes models whose number of variables (after creating the corresponding DAG and simplifying it) has k decimal digits. Library 1 (from Global Library [12]) and Library 2 (from CUTE [5], in the version of VANDERBEI [33]) consist of global (and some local) optimization problems with nonempty feasible domain, while Library 3 (from EPFL [32]) consists of pure constraint satisfaction problems (constant objective function), almost all being feasible. The resulting 12 model classes are labeled as lib2s1 (= size 1 models from Library 2), etc..

Table 1: Number of test models

Number of variables	1 - 9	10 - 99	100 - 999	≥ 1000	
	size 1	size 2	size 3	size 4	total
Library 1	84	90	44	48	266
Library 2	347	100	93	187	727
Library 3	225	76	22	6	329
total	656	266	159	241	1322

We restricted testing to models with less than 1000 variables since the models of size 3 already pose so many difficulties that working on the (much more CPU time consuming) larger models is likely to give no additional insight for the current generation of solvers.

We also excluded a small number of models from these test sets because of difficulties unrelated to the solvers. In particular, the functions if, log10, tan, atan, asin, acos and acosh are currently not supported by the ampl2dag converter. Since they are used in the models ColumnDesign-original, FatigueDesign-original, djtl, hubfit (if), bearing (log10), yfit, yfitu (tan), artif, helix, s332, TrussDesign-full, TrussDesign01 (atan), dallasl, dallasm, dallass (asin), chebyqad, cresc4, cresc50, cresc100, cresc132 (acos), coshfun (acosh), these models were excluded.

A few of the problems in Library 3 (pure constraint satisfaction problems) in fact contained objective functions, and hence were excluded, too. This was the case for the models h78, h80, h81, logcheb, median_exp, median_nonconvex, robotarm, steenbre. A few other models, namely ex8_3_12, ex8_3_14, concon, mconcon, osborneb, showed strange behavior, making us suspect that it is due unspotted bugs in our converters.

Finally, several problems (elec100, elec200, polygon100, arglina, chandheq, dtoc1nd, eigenc2, flosp2hh, flosp2hl, flosp2hm, integreq, s368, logcheb, arglina, nnls, all of size 3) caused memory overflow for our solution checker, and were therefore excluded.

The models where none of the solvers found a feasible point and some other attempts to get one failed, are regarded in the following as being infeasible (though some of these might possess undiscovered feasible points).

The computers. Because of the large number of models to be solved, we performed our tests on a number of different computers, called Lisa, Hektor, Zenon, Theseus and Bagend. Their brand and their general performance characteristic are displayed below. The standard unit time STU is defined according to Shcherbina et al. [32]; for Bogomips and Linpack, see

http://www.tldp.org/HOWTO/mini/BogoMips-2.html http://www.netlib.org/benchmark/linpackjava

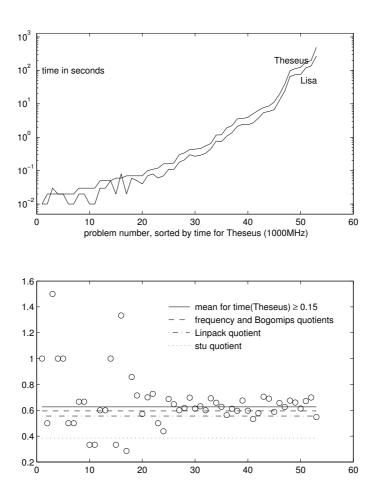
Computer	CPU type	OS	CPU/MHz	Bogomips	STU/sec	Linpack
Lisa	AMD Athlon	Linux	1678.86	3348.88	50	7.42
	XP2000+					
Hektor	AMD Athlon	Linux	1544.51	3080.19	53	6.66
	XP1800+					
Zenon	AMD Family 6	Windows	1001	_	74	46.78
	Model 4	NT 4.0				
Theseus	Pentium III	Linux	1000.07	1992.29	130	4.12
Bagend	AMD Athlon	Linux	1666.72	3329.22	36	5.68
	MP2000+					

To decide on the best way to compare across computers, we ran the models from lib1s1 with BARON on both Lisa and Theseus, and compared the resulting ratios of CPU times with the ratios of performance indices, given by the following table.

	Lisa	Theseus	Ratios	Inverse ratios
Frequency	1678.86	1000.07	1.68	0.60
Bogomips	3348.88	1992.29	1.68	0.59
STU	50.00	130.00	0.38	2.60
Linpack	7.42	4.12	1.80	0.56

As Figure 3 with the results shows, the appropriate index to use is the frequency. We therefore measure times in multiples of 1000 Mcycles, obtained by multiplying the CPU time by the nominal frequency of the CPU in MHz, and dividing the result by 1000. Figure 3 also shows that small times are not well comparable; we therefore decided to round the resulting numbers t to 1 digit after the decimal point if t < 10, and to the nearest integer if $t \ge 10$.

Figure 3: Times and timing ratios for lib1s1 with BARON



The solvers. The following tables summarize some of the main properties of these solvers, as far as known to us. Missing information is indicated by

a question mark, and partial applicability by a + or - in parentheses; the dominant technique (if any) exploited by the solver is denoted by ++.

The first two rows give the name of the solvers and the access language used to pass the problem description. The next two rows indicate whether it is possible to specify integer constraints (although we don't test this feature), and whether it is necessary to specify a finite search box within which all functions can be evaluated without floating point exceptions.

Solver	Minos	LGO	BARON	ICOS	GlobSol
access language	GAMS	С	GAMS	AMPL	Fortran90
optimization?	+	+	+	-	+
integer constraints	_	+	+	_	_
search bounds	_	required	recommended	_	required
black box eval.	+	+	_	_	_
complete	_	(-)	+	+	+
rigorous	_	_	_	+	+
local	++	+	+	+	(+)
CP	_	_	+	++	+
other interval	_	_	-+	++	
convex/LP	_	_	++	+	_
dual	+	_	+	_	_
available	+	+	+	+	+
free	_	_	_	+	+

Solver	Premium	LINGO	αBB	GloptiPoly	OptQuest
	Solver	Global			
access language	Visual Basic	LINGO	MINOPT	Matlab	Visual Basic
optimization?	+	+	+	(+)	+
integer constraints	+	+	+	_	+
search bounds	+	-	?	_	+
black box eval.	_	_	_	_	+
complete	+	+	+	+	_
rigorous	(+)	_	_	_	_
local	+	+	+	_	+
CP	+	+	_	_	_
interval	++	+	+	_	_
convex	+	++	++	+	_
dual	_	+	_	++	_
available	+	+	_	+	+
free	_			+	_

The next three rows indicate whether black box function evaluation is supported, whether the search is complete (i.e., is claimed to cover the whole search region if the arithmetic is exact and sufficiently fast) or even rigorous (i.e., the results are claimed to be valid with mathematical certainty even in the presence of rounding errors). Note that general theorems forbid a complete finite search if black box functions are part of the problem formulation, and that a rigorous search is necessarily complete. In view of the goals of the COCONUT project we were mainly interested in complete solvers. However, we were curious how (some) incomplete solvers perform. Five further rows indicate the mathematical techniques used to do the global search. We report whether local optimization techniques, constraint propagation, other interval techniques, convex analysis and linear programming (LP), or dual (multiplier) techniques are part of the toolkit of the solver.

The final two rows indicate whether the code is available (we include in this list of properties the solver αBB because of its good reported properties, although we failed to obtain a copy of the code), and whether it is free (in the public domain).

In this report, we study the behavior of the solvers BARON/GAMS version 6 [28, 29], GlobSol (version released 11 September 2003) [18], ICOS (beta-test version from February 16, 2004) [19] LGO/GAMS [26], LINGO version 8.0 [20], OQNLP/GAMS [10], Premium Solver (Interval Global Solver from the Premium Solver Platform of Frontline Systems, Version 5) [9]. (Gloptipoly is limited to polynomial systems of dimension < 20, and was not yet tested.)

To enable us to assess how difficult it is (i) to find a global minimum, and (ii) to verify it as global – in many instances, part (ii) is significantly harder than part (i) –, results (without timings) from the local solver MINOS [23] are also included in the tables.

ICOS only handles pure constraint satisfaction problems, and hence was tested only on Library 3. Two of the solvers (BARON and ICOS) also allow the generation of multiple solutions, but due to the lack of a reliable basis for comparison, this feature has not yet been tested. Two of the solvers (BARON and LINGO) allow one to pose integer constraints, and one (Premium Solver) allows nonsmooth expressions. Neither of these features has been tested in this study.

Passing the models. The access to all test models is through an AMPL interface, which translates the AMPL model definition into the internal form of a directed acyclic graph (DAG) which is labelled in such a way as to provide a unique description of the model to be solved. This internal description could

be simplified by a program dagsimplify which performs simple presolve and DAG simplification tasks. Moreover, all maximization problems are converted to minimization problems, with objective multiplied by -1. This preprocessing ensures that all solvers start with a uniform level of description of the model. The DAG is then translated into the input format required by each solver (without simplification).

A testing environment was created to make as much as possible of the testing work automatic. We had to rerun many calculations for many models whenever bugs were fixed, new versions of a solver became available,, new solvers were added, improvements in the testing statistics were made, etc.; this would have been impossible without the support of such a testing environment.

Performance criteria. All solvers are tested with the default options suggested by the providers of the codes. (Most solvers may be made to work significantly better by tuning the options to particular model classes; hence the view given by our comparisons may look more pessimistic than the view users get who spend time and effort on tuning a solver to their models. However, in a large-scale, automated comparison, it is impossible to do such tuning.)

The timeout limit used was (scaled to a 1000 MHz machine) around 170 seconds CPU time for models of size 1, around 850 seconds for models of size 2, and around 1700 seconds for models of size 3. The precise value changed between different runs since we experimented with different units for measuring time on different machines. But changing (not too much) the value for the timeout limit hardly affects the cumulative results, since the overwhelming majority of the models was either completed very quickly, or extremely slow. (Once we shall have settled all aspects of the benchmarking procedure, we shall rerun all tests with uniform timeout limits, to have publishable data.)

The solver WinLGO and GlobSol required a bounded search region, and we bounded each variable between -1000 and 1000, except in a few cases where this lead to a loss of the global optimum.

The reliability of claimed results is the most poorly documented aspect of current global optimization software. Indeed, as was shown by Neumaier & Shcherbina [25] as part of the current project, even famous state-of-theart solvers like CPLEX8.0 (and many other commercial MILP codes) may lose an integral global solution of an innocent-looking mixed integer linear program. We use the following five categories to describe the quality claimed:

Sign	Description
X	model not accepted by the solver
Ι	model claimed infeasible by the solver
G	result claimed to be a global optimizer
L	result claimed to be a local (possibly global) optimizer
U	unresolved (no solution found or error message)
Т	timeout reached (qualifies L and U)

Note that the unresolved case may contain cases where a feasible but nonoptimal point was found, but the system stopped before claiming a local or global optimum.

Checking for best function value. The program solcheck from the COCONUT Environment checks the feasibility of putative solutions of solver results. This was necessary since we found lots of inconsistencies where different solvers produced different results, and we needed a way of checking whether the problem was in the solver or in our interface to it. A point was considered to be (nearly) feasible if each constraint $c(x) \in [\underline{c}, \overline{c}]$ was satisfied within an absolute error of tol for bounds with absolute value < 1, and a relative error of tol for all other bounds. Equality constraints were handled by the same recipe with $\underline{c} = \overline{c}$.

To evaluate the test results, the best function value is needed for each model. We checked in all cases the near feasibility of the best points used to verify the claim of global optimality or feasibility. In a later stage of testing we intend to prove rigorously the existence of a nearby feasible point. More specifically:

The results of all solvers tested were taken into account, and the best function value was chosen from the minimum of the (nearly) feasible solutions by any solver. For models where this did not give a (nearly) feasible point, we tried to find feasible points by ad hoc means, which were sometimes successful. If there was still no feasible solution for a given model, the (local) solution with the minimal residual was chosen (but the result marked as infeasible).

To test which accuracy requirements on the constraint residuals were adequate, we counted the number of solutions of BARON and LINGO on lib1s1 which were accepted as feasible with various solcheck tolerances. Based on the results given in the following table, it was decided that a tolerance of 10^{-5} was adequate. (The default tolerances used for running BARON and LINGO were 10^{-6} .)

solver	tolerance	all	accepted	+G	G!	G?	I?
BARON	1e-4	91	88	75	36	1	0
	1e-5	91	88	75	36	1	0
	1e-6	91	88	56	24	13	0
	1e-7	91	88	49	19	18	0
LINGO	1e-4	91	91	82	66	3	0
	1e-5	91	91	81	65	4	0
	1e-6	91	91	77	63	6	0
	1e-7	91	91	52	41	28	0

Test results. In online files [3], we give a complete list of results we currently have on the nine model classes (i.e., excluding the models with 1000 or more variables, and the few models described before.) For the purposes of comparison in view of roundoff, we rounded function values to 3 significant decimal digits, and regarded function values of $< 10^{-4}$ as zero (but print them in the tables as nonzeros) when the best known function value was zero. Otherwise, we regard the global minimum as achieved if the printed (rounded) values agree.

Notation in the tables. In the summary statistic tables the following notation is used:

Column	Description
library	describes the library
all	library/size
accepted	the number of models accepted by the solver
+G	number of models for which the global solution was found
G!	number of models for which the global solution was
	correctly claimed to be found
G?	number of models for which a global solution was claimed
	but the true global solution was in fact significantly better
	or the global solution is reported but in fact that is an
	infeasible point
I?	number of models for which the model was claimed
	infeasible although a feasible point exists

For models where a local optimizer finds the global optimum (without knowing it), the purpose of a global code is to check whether there is indeed no better point; this may well be the most time-consuming part of a complete search. For the remaining models the search for the global optimum is already hard. We therefore split the evaluation into

- 'easy location models' where the local optimizer MINOS found the global solution,
- and 'hard location models' where a local optimizer did not find the global solution.

For the easy and hard models (according to this classification), the claimed status is given in the columns; the rows contain a comparison with the true status:

Column	Description
wrong	number of wrong claims, i.e. the sum of G? and I? from
	the summary statistic table
+G	how often the solution found was in fact global
-G	how often it was in fact not global
Ι	how many models are in fact infeasible

For each library detailed tables for all models and all solvers tested, and many more figures (of the same type as the few presented here) are also available, for reasons of space they are presented online on the web [3].

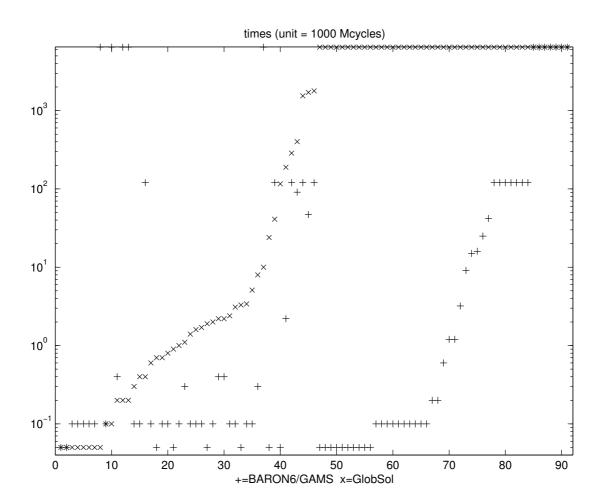
GlobSol. To test GlobSol, we used an evaluation version of LAHEY Fortran 95 compiler. Note that we had difficulties with the Intel Fortran compiler.

In the first round we tested GlobSol on Library 1 size 1 problems (contains 91 problems) with the same time limit as used for the other solvers. GlobSol failed to solve most of the problems (only 29 problems were solved) within the strict time limit. For this reason we decided to use a very permissive time limit.

Figure 4 compares the two global solvers BARON and GlobSol on the size 1 problems from Library 1. The figure contains timing results for the models described in the figure caption, sorted by the time used by GlobSol. Conversion times for putting the models into the format required by the solvers are not included. Times (given in units of 1000 Mcycles) below 0.05 are places on the bottom border of each figure, models for which the global minimum was not found by the solver get a dummy time above the timeout value, and are placed at the top border of each figure. In this way one can assess the successful completion of the global optimization task.

Clearly, GlobSol is much less efficient in both time and solving capacity than BARON. To a large extent this may be due to the fact that GlobSol strives to achieve mathematical rigor, resulting in significant slowdown due to the need

Figure 4: Times for lib1s1, all models, GlobSol vs. BARON



of rigorously validated techniques. (There may be also systematic reasons in the comparison, since GlobSol does not output a best approximation to a global solution but boxes, from which we had to extract a test point.) In view of these results, we refrained from further testing GlobSol.

The summary statistics on library 11s1 can be found in the following table.

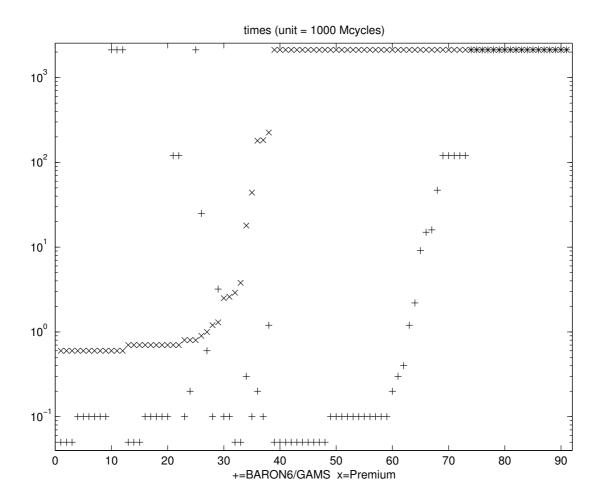
GlobSol summary statistics								
library	all	accepted	+G	G!	G?	I?		
lib1s1	91	77	29	29	29	0		

A more detailed table gives more information. Included is an evaluation of

the status claimed about model feasibility and the global optimum, and the true status (based on the best point known to us).

GlobSol on lib1s1										
status	all	wrong	easy	locati	on	hard	locati	on		
			+G	-G	Ι	+G	-G	I		
all	91	29	15	23	0	14	39	0		
G	58	29	15	12	0	14	17	0		
U	17	0	0	5	0	0	12	0		
UT	2	0	0	1	0	0	1	0		
X	14	0	0	5	0	0	9	0		

Figure 5: Times for lib1s1, all models, Premium Solver vs. BARON



Premium Solver. The same tests as for GlobSol were performed for Premium Solver, with slightly worse performance. In view of these results, we refrained from further testing Premium Solver.

Premium summary statistics									
library	all	all accepted $+G$ $G!$ $G?$ $I?$							
lib1s1	91	75	35	27	14	1			

	Premium Solver on lib1s1							
status	all	wrong	easy	locati	on	hard	locati	on.
			+G	-G	Ι	+G	-G	Ι
all	91	15	29	25	2	6	29	0
G	41	14	22	11	0	5	3	0
L	12	0	4	2	0	1	5	0
LT	9	0	2	3	1	0	3	0
U	12	0	1	1	0	0	10	0
X	16	0	0	7	1	0	8	0
I	1	1	0	1	0	0	0	0

BARON, LINGO, OQNLP, LGO and MINOS. The following tables contain the summary statistics for the performance of the other solvers tested, apart from COCOS. ICOS is a pure constraint solver, which cannot handle models with an objective function, and hence was tested only on library 3.

BAI	RON/	GAMS sum	mary	stati	istics	
library	all	accepted	+G	G!	G?	I?
lib1s1	91	88	66	35	1	0
lib1s2	80	79	64	34	2	0
lib1s3	38	34	15	0	0	0
lib2s1	324	300	232	91	7	0
lib2s2	99	89	62	19	2	2
lib2s3	86	81	31	10	7	3
lib3s1	217	196	165	1	0	3
lib3s2	69	65	48	0	0	2
lib3s3	20	18	9	0	0	1

	LINC	GO summai	ry stat	tistics		
library	all	accepted	+G	G!	G?	Ι?
lib1s1	91	91	70	59	8	0
lib1s2	80	80	42	33	12	0
lib1s3	38	38	9	3	1	0
lib2s1	324	324	195	170	37	0
lib2s2	99	99	55	33	11	0
lib2s3	86	86	30	21	3	0
lib3s1	217	217	167	17	2	0
lib3s2	69	69	42	3	0	0
lib3s3	20	20	6	2	0	0

OQNLP/GAMS summary statistics						
library	all	accepted	+G	G!	G?	I?
lib1s1	91	91	70	0	0	1
lib1s2	80	80	62	0	0	2
lib1s3	38	27	11	0	0	4
lib2s1	324	316	246	0	0	3
lib2s2	99	95	75	0	0	2
lib2s3	86	74	61	0	0	3
lib3s1	217	213	189	0	0	6
lib3s2	69	67	53	0	0	4
lib3s3	20	17	7	0	0	6

LGO/GAMS summary statistics						
library	all	accepted	+G	G!	G?	Ι?
lib1s1	91	85	51	0	0	0
lib1s2	80	78	37	0	0	8
lib1s3	38	22	3	0	0	10
lib2s1	324	311	208	0	0	6
lib2s2	99	91	48	0	0	17
lib2s3	86	48	14	0	0	10
lib3s1	217	212	142	0	0	48
lib3s2	69	67	28	0	0	22
lib3s3	20	10	2	0	0	6

MINOS/GAMS summary statistics						
library	all	accepted	+G	G!	G?	I?
lib1s1	91	91	53	0	0	0
lib1s2	80	80	43	4	0	4
lib1s3	38	38	16	0	1	4
lib2s1	324	324	213	10	6	13
lib2s2	99	97	65	4	2	4
lib2s3	86	85	34	1	0	11
lib3s1	217	213	151	2	0	27
lib3s2	69	69	36	1	0	12
lib3s3	20	20	8	1	0	6

ICOS summary statistics						
library	all	accepted	+G	G!	G?	I?
lib3s1	217	198	128	122	2	0
lib3s2	69	0	0	0	0	0
lib3s3	20	18	3	3	0	0

The corresponding reliability analysis tables are as follows.

	Reliability analysis for BARON
	global minimum found/accepted
size 1	$463/584 \approx 79\%$
size 2	$174/233 \approx 75\%$
size 3	$55/133 \approx 41\%$
all	$692/950 \approx 73\%$
	correctly claimed global/accepted
size 1	$127/584 \approx 22\%$
size 2	$53/233 \approx 23\%$
size 3	$10/133 \approx 8\%$
all	$190/950 \approx 20\%$
	wrongly claimed global/claimed global
size 1	$8/135 \approx 6\%$
size 2	$4/57 \approx 7\%$
size 3	$7/17 \approx 41\%$
all	$19/209 \approx 9\%$
	claimed infeasible/accepted and feasible
size 1	$3/537 \approx 1\%$
size 2	$4/202 \approx 2\%$
size 3	$4/102 \approx 4\%$
all	$11/841 \approx 1\%$

	Reliability analysis for LINGO
	global minimum found/accepted
size 1	$432/632 \approx 68\%$
size 2	$139/248 \approx 56\%$
size 3	$45/144 \approx 31\%$
all	$616/1024 \approx 60\%$
	correctly claimed global/accepted
size 1	$246/632 \approx 39\%$
size 2	$69/248 \approx 28\%$
size 3	$26/144 \approx 18\%$
all	$341/1024 \approx 33\%$
	wrongly claimed global/claimed global
size 1	$47/293 \approx 16\%$
size 2	$23/92 \approx 25\%$
size 3	$4/30 \approx 13\%$
all	$74/415 \approx 18\%$
	claimed infeasible/accepted and feasible
size 1	$0/513 \approx 0\%$
size 2	$0/164 \approx 0\%$
size 3	$0/63 \approx 0\%$
all	$0/740 \approx 0\%$

	Reliability analysis for OQNLP			
	global minimum found/accepted			
size 1	$505/620 \approx 81\%$			
size 2	$190/242 \approx 79\%$			
size 3	$79/118 \approx 67\%$			
all	$774/980 \approx 79\%$			
	claimed infeasible/accepted and feasible			
size 1	$10/582 \approx 2\%$			
size 2	$8/220 \approx 4\%$			
size 3	$13/95 \approx 14\%$			
all	$31/897 \approx 3\%$			

	Reliability analysis for LGO		
	global minimum found/accepted		
size 1	$401/608 \approx 66\%$		
size 2	$113/236 \approx 48\%$		
size 3	$19/80 \approx 24\%$		
all	$533/924 \approx 58\%$		
	claimed infeasible/accepted and feasible		
size 1	$54/519 \approx 10\%$		
size 2	$47/170 \approx 28\%$		
size 3	$26/45 \approx 58\%$		
all	$127/734 \approx 17\%$		

	Reliability analysis for MINOS
	· · · · · · · · · · · · · · · · · · ·
	global minimum found/accepted
size 1	$417/628 \approx 66\%$
size 2	$144/246 \approx 59\%$
size 3	$58/143 \approx 41\%$
all	$619/1017 \approx 61\%$
	correctly claimed global/accepted
size 1	$12/628 \approx 2\%$
size 2	$9/246 \approx 4\%$
size 3	$2/143 \approx 1\%$
all	$23/1017 \approx 2\%$
	wrongly claimed global/claimed global
size 1	$6/18 \approx 33\%$
size 2	$2/11 \approx 18\%$
size 3	$1/3 \approx 33\%$
all	$9/32 \approx 28\%$
	claimed infeasible/accepted and feasible
size 1	$40/526 \approx 8\%$
size 2	$20/189 \approx 11\%$
size 3	$21/96 \approx 22\%$
all	$81/811 \approx 10\%$

The results speak for themselves, here only a few observations.

The GAMS solvers LGO and OQNLP are very cautious, never claiming a global minimum. This reflects the observed unreliability of the internal claims (as seen by studying the logfile) of the LGO version used by GAMS, and their decision rather to err on the conservative side.

For pure CSPs when the objective function is constant, any feasible point is a global solution. But as the results show, this is not recognized by any of the solvers except ICOS.

It is remarkable that under GAMS, the local solver MINOS sometimes claims to have found a global result. This is the case, e.g., because some models are recognized as linear programs for which every local solution is global.

Another remarkable observation is that the models from Library 1, which were collected specifically as test problems for global optimization, do not behave much differently from those of Library 2, which were collected as test problems for local optimization routines.

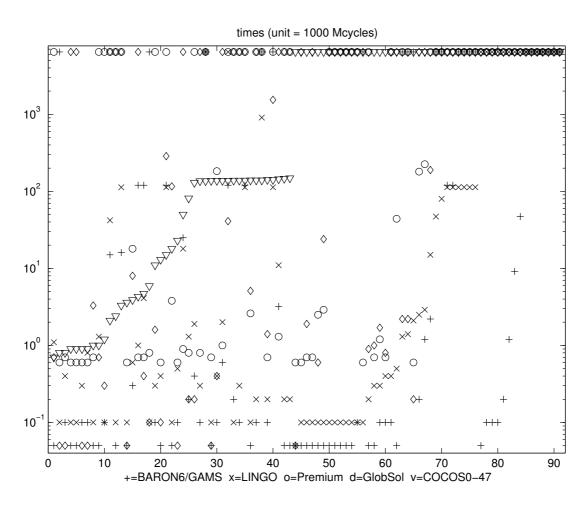
3 Comparing the COCOS combination strategy

In this section we compare COCOS, a basic combination strategy implemented in the COCONUT environment (version 0.99), with BARON, and on lib1s1 (problems from GlobalLib with less than 10 variables) also with GlobSol, LINGO, and Premium Solver, the other complete solvers used in our tests.

lib1s1 summary statistics							
solver	all	accepted	+G	G!	G?	Ι?	
LINGO	91	91	70	59	8	0	
BARON	91	88	66	35	1	0	
OQNLP	91	91	53	0	0	0	
MINOS	91	91	53	0	0	0	
LGO	91	85	51	0	0	0	
COCOS	91	91	41	26	12	0	
Premium	91	75	35	27	14	1	
GlobSol	91	77	29	29	29	0	

Clearly, COCOS cannot yet compete with the best solvers, but it performs already at the same level as GlobSol and Premium Solver. On the other sublibraries, the performance is as follows.

Figure 6: Times for lib1s1, all models, several solvers



COCOS summary statistics							
library	all	accepted	+G	G!	G?	I?	
lib1s1	91	91	41	26	12	0	
lib1s2	80	80	35	9	7	0	
lib1s3	38	38	6	0	6	0	
lib2s1	324	323	132	76	32	0	
lib2s2	99	99	37	9	5	0	
lib2s3	86	86	16	4	3	0	
lib3s1	217	217	129	40	3	0	
lib3s2	69	69	38	16	2	0	
lib3s3	20	20	6	3	0	0	

Reliability analysis for COCOS		
	global minimum found/accepted	
size 1	$302/631 \approx 48\%$	
size 2	$110/248 \approx 44\%$	
size 3	$28/144 \approx 19\%$	
all	$440/1023 \approx 43\%$	
	correctly claimed global/accepted	
size 1	$142/631 \approx 23\%$	
size 2	$34/248 \approx 14\%$	
size 3	$7/144 \approx 5\%$	
all	$183/1023 \approx 18\%$	
	wrongly claimed global/claimed global	
all	/pprox%	
size 1	$47/189 \approx 25\%$	
size 2	$14/48 \approx 29\%$	
size 3	$9/16 \approx 56\%$	
all	$70/253 \approx 28\%$	
	claimed infeasible/accepted and feasible	
size 1	$0/398 \approx 0\%$	
size 2	$0/136 \approx 0\%$	
size 3	$0/38 \approx 0\%$	
all	$0/572 \approx 0\%$	

For the hard models (where the local solver MINOS did not find the global optimum), the performance is already close to that of the currently best solvers BARON and LINGO; see the figures in Section 1.

Performance on problems with few variables, or easy problems (where the local solver MINOS already found the global optimum) was less good. The current version of COCOS is dominated for small or easier problems by overhead due to printing debugging information and the presence of choices that are really useful only for larger, hard problems. We did not yet have time to tune the parameters in the COCOS strategy to get a uniformly good performance; this will be done in the near future.

References

[1] R.S. Barr, B.L. Golden, J.P. Kelly, M.G.C. Resende, and W.R. Stewart, Designing and reporting on computational experiments with heuristic

- methods, Journal of Heuristics 1 (1995), 9-32. http://www.research.att.com/~mgcr/abstracts/guidelines.html
- [2] F. Benhamou and F. Goualard, Universally Quantified Interval Constraints. In *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP'2000)*, pages 67–82, 2000.
- [3] COCONUT test results, WWW-directory, 2004. http://www.mat.univie.ac.at/ neum/glopt/coconut/tests/figures/
- [4] H.P. Crowder, R.S. Dembo and J.M. Mulvey, On reporting Computational Experiments with Mathematical Software, ACM Transactions on Mathematical Software, 5 (1979), 193–203.
- [5] N.I.M. Gould, D. Orban and Ph.L. Toint, CUTEr, a constrained and unconstrained testing environment, revisited, WWW-document, 2001. http://cuter.rl.ac.uk/cuter-www/problems.html
- [6] E.D. Dolan and J.J. Moré, Benchmarking Optimization Software with COPS, Tech. Report ANL/MCS-246, Argonne Nat. Lab., November 2000. http://www-unix.mcs.anl.gov/~more/cops
- [7] E.D. Dolan and J.J. Moré, Benchmarking Optimization Software with Performance Profiles, Tech. Report ANL/MCS-P861-1200, Argonne Nat. Lab., January 2001 http://www-unix.mcs.anl.gov/more/cops
- [8] R. Fourer, D.M. Gay and B.W. Kernighan, AMPL: A Modeling Language for Mathematical Programming, Duxbury Press, Brooks/Cole Publishing Company, 1993. http://www.ampl.com/cm/cs/what/ampl/
- [9] Frontline Systems, Inc., Solver Technology Global Optimization, WWW-document (2003). http://www.solver.com/technology5.htm
- [10] GAMS Solver descriptions, GAMS/OQNLP, WWW-document (2003). http://www.gams.com/solvers/solvers.htm#OQNLP
- [11] GAMS World, WWW-document, 2002. http://www.gamsworld.org

- [12] GLOBAL Library, WWW-document, 2002. http://www.gamsworld.org/global/globallib.htm
- [13] H.J. Greenberg, Computational testing: Why, how, and how much, ORSA Journal on Computing 2 (1990), 94–97.
- [14] W. Huyer, A comparison of some algorithms for bound constrained global optimization, WWW-document (2004). http://www.mat.univie.ac.at/ neum/glopt/contrib/compbound.pdf
- [15] ILOG. ILOG Solver. Reference Manual. 2002.
- [16] R.H.F. Jackson, P.T. Boggs, S.G. Nash and S. Powell, Guidelines for reporting results of computational experiments. Report of the ad hoc committee, Mathematical Programming 49 (1990/91), 413–426.
- [17] E. Janka, Vergleich Stochastischer Verfahren zur Globalen Optimierung, Diplomarbeit, Mathematisches Inst., Universität Wien, 1999.

 A shorter online version in English language is at http://www.mat.univie.ac.at/~neum/glopt/janka/gopt_eng.html.
- [18] R.B. Kearfott, Rigorous Global Search: Continuous Problems, Kluwer, Dordrecht 1996.
 www.mscs.mu.edu/ globsol
- [19] Y. Lebbah, ICOS (Interval COnstraints Solver), WWW-document (2003). http://www-sop.inria.fr/coprin/ylebbah/icos/
- [20] Lindo Systems, Inc., New LINGO 8.0, WWW-document (2003). http://www.lindo.com/table/lgofeatures8t.html
- [21] H. Mittelmann, Benchmarks. WWW-document, 2002. http://plato.la.asu.edu/topics/benchm.html
- [22] M. Mongeau, H. Karsenty, V. Rouzé, J.-B. Hiriart-Urruty, Comparison of public-domain software for black box global optimization, Optimization Methods and Software 13 (2000), 203–226. http://mip.ups-tlse.fr/publi/rapp99/99.50.html
- [23] B.A. Murtagh and M.A. Saunders, MINOS 5.4 User's Guide, Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, December 1983 (revised February 1995). http://www.sbsi-sol-optimize.com/Minos.htm

- [24] A. Neumaier, Complete Search in Continuous Global Optimization and Constraint Satisfaction, to appear in: Acta Numerica 2004 (A. Iserles, ed.), Cambridge University Press 2004.

 http://www.mat.univie.ac.at/ neum/papers.html#glopt03
- [25] A. Neumaier and O. Shcherbina, Safe bounds in linear and mixed-integer programming, Math. Programming A 99 (2004), 283-296. http://www.mat.univie.ac.at/ neum/papers.html#mip
- [26] J.D. Pinter, Global Optimization in Action, Kluwer, Dordrecht 1996. http://www.dal.ca/~jdpinter/l_s_d.html
- [27] H. D. Ratliff and W. Pierskalla, Reporting Computational Experience in Operations Research, Operations Research 29 (2) (1981), xi–xiv.
- [28] H.S. Ryoo and N.V. Sahinidis, A branch-and-reduce approach to global optimization, J. Global Optim. 8 (1996), 107–139. http://archimedes.scs.uiuc.edu/baron/baron.html
- [29] N. Sahinidis and M. Tawarmalani, Global Optimization with GAMS/BARON http://www.gams.com/presentations/present_baron2.pdf
- [30] H. Schichl, Global optimization in the COCONUT project. in: Proceedings of the Dagstuhl Seminar "Numerical Software with Result Verification", Springer Lecture Notes in Computer Science 2991, Springer, Berlin, 2004.
- [31] H. Schichl, Mathematical Modeling and Global Optimization, Habilitation Thesis (2003), Cambridge Univ. Press, to appear.
- [32] O. Shcherbina, A. Neumaier, Djamila Sam-Haroud, Xuan-Ha Vu and Tuan-Viet Nguyen, Benchmarking global optimization and constraint satisfaction codes, Proc. COCOS'02, Springer, to appear.
- [33] B. Vanderbei, Nonlinear Optimization Models, WWW-document. http://www.orfe.princeton.edu/~rvdb/ampl/nlmodels/