# Quality Assurance and Global Optimization

Michael R. Bussieck[1], Arne Stolbjerg Drud[2], Alexander Meeraus[1], and Armin Pruessner[1]

[1] GAMS Development Corp., 1217 Potomac Street, NW
Washington, DC 20007, USA
{MBussieck, AMeeraus, APruessner}@gams.com
http://www.gams.com

[2] ARKI Consulting & Development A/S, Bagsvaerdvej 246A
2880 Bagsvaerd, Denmark ADrud@arki.dk
http://www.conopt.com

**Abstract.** GAMS Development and ARKI Consulting use Quality Assurance (QA) as an integral part of their software development and software publishing process. Research and development in the global optimization area has resulted in promising implementations. Initiated by customer demand, we have been adding three different global codes, BARON, LGO, and OQNLP, to our portfolio of nonlinear optimization solvers. As part of our QA effort towards the integration and testing of these new global solvers an open architecture for reliability and performance testing of mixed-integer nonlinear optimization codes has been released. This open testing framework has been placed in the public domain (www.gamsworld.org) to serve our customers, and researchers in general, by making reproducible tests a practical proposition. We give examples illustrating the quality assurance process for obtaining performance results for local and global nonlinear and mixed-integer nonlinear programming solvers, using the existing framework tools described in this article.

## 1 Introduction

The research and development efforts in the area of global (nonlinear) optimization codes have increased significantly in recent years, and a number of large-scale implementations have been successfully deployed in specialized research environments. To make these systems more widely available to users of optimization system, the global solvers need to be integrated into a modeling system that manages problem formulation and guides the solution process. In this paper we report on the quality assurance aspects of transforming a research product into a reliable commercial product. When introducing new solver technology into the marketplace, there are additional difficulties besides technical problems, due to the inherent risk to the customer using commercially unproven technology. Furthermore, the potential failure of a single global optimization code today can reduce the confidence level in future global optimization solvers.

This risk is comparable to the introduction of mixed-integer linear (MIP) solvers in the 1960s and 1970s.

To address some of the problems associated with introducing a new technology into a mature modeling environment, we have decided to share our internal quality assurance tools and make them available to everyone. Quality Assurance (QA), the process of assuring the quality of one organization's outcome, is in our case means to assure our customers of reliable, state-of-the-art technology. Although QA has become an essential component in most industrial and commercial undertakings, however, it has been more or less ignored by the Mathematical Programming (MP) community. This should not come as a surprise, since the main market for MP is the academic literature. Running the risk (again) of annoying some our colleagues, we would like to draw some historical analogies.

We, the optimization modelers, are in a transition from a slow and inefficient cottage industry, similar to manufacturing just prior to the Industrial Revolution. Lone experts during this period painstakingly manufactured hand-tooled, customized items. We are now transitioning to an industrial, customer-driven environment with standards, interchangeable parts, and low-cost, highly-reliable components. This transition is typical for engineering and problem solving, activities trying to maximize the use of existing parts and components and breaking larger problems into smaller, more tractable items. Standards are essential to increase the reliability and effectiveness of our products. Customers demand quality, which is itself a fuzzy concept (a bit like beauty being in the eye of the beholder). Quality Management and Standards have developed into an industry by itself, and ISO 9000, the International Standard Organization's Quality Management System Standard, has become the mainstay of many industries.

Another way to look at this positive development is the shift of the central theme in practical modeling. The first phase was dominated by *algorithmic issues.* Problem representations were defined by algorithms, and performance testing focused on detailed algorithmic issues. A good example is the seminal paper by Crowder, Dembo and Mulvey [3] on computational experiments. The second phase is dominated by *data and model representation* issues. This phase has resulted in a number of algebra-based modeling systems pioneered by LINDO [18], GAMS [1] and AMPL [7] (in that chronological order). We are now transitioning into a third phase dominated by *real-life problem solving.* There has been a permanent shift from a scientific, supply-driven regime into a market-oriented, user demand-driven business environment. In this environment, quality has become a central issue, as in any other industry.

In this paper we discuss some of the elements of an effective QA framework. In §2 we give a brief overview of the history of the GAMS modeling language and discuss the technical principles of modeling languages. We also discuss general global optimization and some issues associated with the quality assurance of global solvers. In §3 we describe the necessary tools for effective quality assurance testing, and in §4 we stress the importance of open testing architectures. In §5 we give examples implementing the framework we have described, and finally, in §6, we draw conclusions.

## 2 Modeling Languages and Global Optimization Codes

The use of modeling languages has greatly simplified the solution of large-scale practical problems, both in academia and in industry. Because of its large and wide-ranging client base, GAMS has a deep impact on nonlinear programming (NLP) solver technology, and thus a responsibility for providing high quality and reliable commercial (local and global) NLP solvers.

In this section we describe some of the principles of modeling languages and some issues specific to global optimization codes which need to be addressed by any effective quality assurance framework.

### 2.1 Basic Technical Principles

Throughout the evolution as a company, GAMS has adhered to three basic technical principles:

**Separation of model and solution methods:** Separating the model from the various solution methods incorporated within that model ensures that the user is not locked into any particular method, and can switch rapidly between models at no additional cost. For example, users can seamlessly switch local and global solvers. This separation of model and solution method is now accepted as a standard approach to general modeling.

**Computing platform independence:** Platform independence ensures immediate application on any user's configuration, and eliminates conversion costs. Although most modeling systems in the commercial world are on the Windows platform, customer decisions are often influenced by availability across platforms.

**Multiple solvers, platforms, and model types:** To create the most flexible general model, GAMS' software incorporates solvers available from both the academic and the commercial worlds. Multiple solver, platform, and model type flexibility enables the user to tackle problems involving linear programs (LP), MIP, NLP, mixed-integer nonlinear programs (MINLP), mixed complementarity problems (MCP), mathematical programs with equilibrium constraints (MPEC), stochastic programming, and models written in MPSGE, a language for solving computable general equilibrium(CGE) models.

### 2.2 Global Optimization Principles

Most practical models involving nonlinearities are developed in a modeling language, and most NLP and MINLP solvers are linked to a modeling system. In our modeling system, GAMS, we try to improve the reliability of NLP modeling and thereby reduce the risk for our customers by offering a *portfolio* of NLP solvers (BARON, CONOPT, LGO, MINOS, MOSEK, OQNLP, PATHNLP, SNOPT - see [8] and the references therein) implementing a variety of different algorithms (interior point, GRG, SLP, SQP). This approach improves the probability of solving our customers' models.

Solving MINLP problems involves the sequential solution of a large number of NLP sub-problems (either in a branch-and-bound, extended cutting plane, or outer approximation algorithm), which increases the chance of failure of the overall algorithm. Our MINLP solvers DICOPT and SBB have been built around this *chance of failure*. By enabling these solvers to access any NLP sub-solver in our portfolio we effectively minimize the chance of failure. The use of a multi-solver architecture helps in cases of solver failure, and overcomes the weakness inherent in local optimization codes (*local solutions*, in particular *local infeasibility*). Nonlinear modelers have coped with this weakness by providing good starting points, and have implemented their own multi-start methods (usually with random points) at a modeling language level.

The incentive structure for developers of global optimization algorithms such as BARON [19], LGO [17], and OQNLP [20] in the academic environment is very different from that faced in a commercial setting. Algorithms in the academic world are operated in *expert mode* by the developers themselves, and are often highly specialized to meet the requirements of an abstract problem class. Commercial solvers, however, are deployed by users who have no interest in the algorithm itself but want to solve their business problem by using the algorithm in a *black box mode*. Instead of delivering extraordinary performance requiring substantial use of specific options or *code tweaking*, a commercial solver has to work reliably with decent performance in all cases using *default settings*. In case of algorithmic failure, the solver has to terminate gracefully and issue suggestions on how to overcome the failure.

Since its introduction, the GAMS system has provided nonlinear modeling to a wide audience of academic and commercial users. Global solvers can be almost seamlessly integrated into the portfolio of nonlinear solvers, offering a new service to our customers and opening a new market to global solver providers. The risk for our customers of investing in new solver technology can be reduced by providing access to quality assurance results for global solvers and a plug-compatible interface that allows painless transition from local solvers to global ones.

**Global Optimization Specific Issues** Global optimization requires special attention to several specific issues. These issues include:

**Termination criteria:** While the termination criteria for local solvers is concise (satisfying the Karush-Kuhn Tucker conditions), the stopping criteria for global codes is ambiguous. Mutli-start methods use different starting points to converge to different local optima, thereby maximizing the probability that one of the local solutions is indeed the global optima. Unfortunately, the algorithm cannot determine precisely if a current local optima is the global optima.

**Problem bounds:** Global solvers generally require modification of the original problem. In particular, global solvers can usually only guarantee global optimality if the problem is bounded for both variables and expressions (bounding box principle).

**Limited algebra:** Some global optimization solvers do not support all functions. In particular, many global solvers cannot handle black-box-type functions, where only function evaluations are returned. In particular, many require detailed knowledge of the algebra involved in the function itself.

**Solution quality metrics:** While for linear and local nonlinear solvers robustness and efficiency metrics are sufficient in characterizing solver performance and assure quality, for global optimization solvers quality of solution is another descriptive metric of performance. When analyzing global solver performance, higher solution quality is often obtained at a cost of efficiency. Thus, users should consider if feasibility of the (local) solution and efficiency or global optimality at a cost of efficiency is the primary criteria.

### 2.3 Quality Assurance

Although reproducibility of test results has been accepted as a critical step in most scientific fields, computational results in our field can rarely be reproduced. Limited access to test cases, non-reproducible methods for collecting results, and non-standard analysis tools seem to prevent the low-cost replication of such results by an independent auditor. Based on our experience, we rank the *replication of quality assurance results* as the most critical factor for establishing a new solver technology in the commercial world.

In the next section we discuss the necessary components for an effective quality assurance framework.

## 3 Effective Quality Assurance Testing

The key ingredients for effective testing are diverse *test cases*, efficient *data collection* tools, and automated *data analysis* tools.

The choice of test problems for benchmarks is difficult and inherently subjective. While there is no consensus on choosing appropriate models, the use of standard model libraries is important to ensure that testing is performed with a wide selection of models from diverse application areas. Consider that a particular solver may be fine-tuned for specific applications. The use of diverse standard model libraries *reduces the risk of bias* if a solver is fine-tuned for a specific family of models and allows more objective cross-comparisons of various solvers.

Most optimization engines or solvers output model statistics, objective function, resource time and general solve information. Many benchmarks and performance analyses involve either manually extracting the necessary information from the log output or writing solver and optimization engine-specific scripts to parse the output and extract the data to be analyzed. This can be cumbersome and is error prone and generally must be tailored to the specific engine or solver. In order to simplify the quality assurance process, data collection must be *automated*.

Finally, few standard performance metrics exist and the reproducibility of performance tests is often not a practical proposition. By introducing automated tools and standard performance measurements, we enable the quality assurance process to be *inexpensive, efficient, and reproducible.*

## 3.1 Test Cases

Testing global and local optimization solvers requires access to a collection of test models, including toy models, academic application models and, most importantly, for our purposes, commercial application models (which are in general difficult to collect due to the proprietary nature of most commercial models). While academic application models are useful in their own right, they do not always address the same problem types and model structure a commercial model may. In order to assure a new solver technology's quality in the commercial world, any practical model library should contain commercial application models as well.

Our publicly available collection of NLP and MINLP models is:

**GAMS Model library:** `http://www.gams.com/modlib/modlib.htm`, with more than 250 models from over 18 application areas.

**GLOBALLib:** `http://www.gamsworld.org/global/globallib.htm`, with more than 390 scalar NLP models.

**MINLPLib:** `http://www.gamsworld.org/minlp/minlplib.htm`, with about 180 scalar MINLP models. Also see [2].

**MPECLib:** `http://www.gamsworld.org/mpec/mpeclib.htm`, which produces over $10,000$ NLP models .

New models are added continuously to these model libraries. In order to add commercial models with proprietary data, we make use of the CONVERT utility.

**Translating Models Using CONVERT** For commercial application models, where proprietary data should be hidden, GAMS provides the CONVERT tool, which translates a model into *scalar* format. This hides all proprietary parts of a model and makes public access to customer models possible.

Modeling languages such as GAMS or AMPL have a rich syntax that is usually based on sets and indexed variables, equations, and parameters. This syntax, and the corresponding structure in the model and the data, is very useful for the model developer. However, usually such structures are not used by the solvers. Most solvers see the world as a list of variables, $X_1$ to $X_n$, a list of equations or constraints, $E_1$ to $E_m$, and the relationship between these variables and equations, as represented in some form. As long as the model seen by the solver remains unchanged, it is therefore acceptable for a translator to remove the structure. The GAMS translator CONVERT transforms models into a very simple, internal scalar format. This internal format can then be written out in many different formats. With GAMS as output format, the scalar model consists of the following:

– Declarations of the variables, with extra declarations for the subsets of positive, integer, or binary variables,
– Declarations of the equations,
– The symbolic form of these equations, and
– Assignment statements for non-default bounds and initial values.

All operations involving sets are unrolled, and all expressions involving parameters are evaluated and replaced by their numerical values. Since there are no sets or indexed parameters in the scalar models, most of the differences between modeling systems have disappeared. Therefore, the GAMS format can be easily transformed into another language's format. For example, in AMPL the keyword "var" is used instead of "Variable," bounds and initial values are written using a different format, the equation declarations are missing, the equation definitions start with "subject to $E_i$" instead of "$E_i$..", and a few operators are named differently.

There are a few cases where a model cannot be translated into a particular language because special functions (e.g. `errorf`) or variable types (e.g. `SemiCont`) are not available in that language. GAMS models have an objective variable rather than an objective function. If there is one defining equation for a given variable, CONVERT will eliminate the objective variable and will introduce a real objective function for formats that support objective functions (e.g. AMPL). For more details on CONVERT see [2].

Other issues which impact the translation include scaling of the model and presolve capabilities by the particular solver or optimization engine. In general, long term strategies should include a mathematical programming *standard scalar format*, which allows automorphic translation to and from this format. This has been accomplished for linear model with the long-accepted standard MPS format [9] and recently work focused on a broader set of models has been done by introducing a format based on XML [13].

## 3.2 Data Collection Tools

Reproducible data production and collection requires an automated system that provides information about the testing environment (version of software, hardware, etc.), and status, performance, and exception information for the individual test case. Such a system is automatically available in GAMS through the *trace facility* for all solvers connected. The trace facility provides access to model statistics, non-default input options, and solver and solution statistics and writes information conveniently to an ASCII interface. Table 1 shows the possible trace file column headers and the associated data.

**Table 1.** GAMS Trace Facility ASCII Interface (current trace file data)

| Heading | Description |
|---|---|
| Filename | GAMS model filename |
| Modeltype | LP, MIP, NLP, etc. |
| Solvername | |
| NLP def. | Default NLP solver |
| MIP def. | Default MIP solver |
| Juliantoday | start day/time of job |
| Direction | 0=min, 1=max |
| Equnum | Total number of equations |
| Varnum | Total number of variables |
| Dvarnum | Total number of discrete variables |
| Nz | number of nonzeros |
| Nlnz | number of nonlinear nonzeros |
| Optfile | 1=optfile included, 0=none |
| Modelstatus | GAMS model status |
| Solverstatus | GAMS solver status |
| Obj | Value of objective function |
| Objest | Estimate of objective function |
| Res used | Solver resource time used (sec) |
| Iter used | Number of solver iterations |
| Dom used | Number of domain violations |
| Nodes used | Number of nodes used |

The resulting data can easily be analyzed for example through user automation scripts, Excel spreadsheets or some of the online tools available as part of Performance World: `http://www.gamsworld.org/performance`.

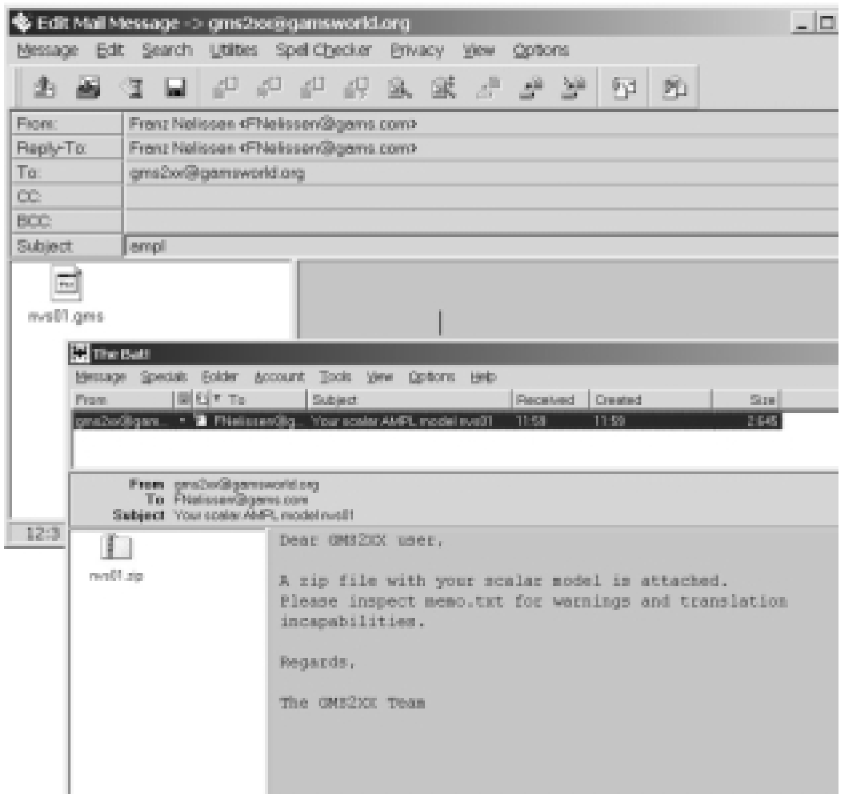
## 3.3 Data Analysis Tools

In addition to custom programs for processing testing results, we have implemented a variety of performance measurement tools. Together with experts in the field of benchmarking at Performance World, we have also extended and developed new ways to present testing results in an easy and consistent way.

In particular, the tools as part of the *PAVER server* (Performance Analysis and Visualization for Efficient Reproducibility) [15], accessible online at

`http://www.gamsworld.org/performance/paver`

can be used for performance analysis of performance data collected in trace files in an automated fashion. The server provides automation and visualization tools for automatically generating HTML-based reports of submitted trace files and gives information on robustness, efficiency and solver quality of solution. Users can submit up to 8 trace files (each containing performance data for a


Edit Mail Message -> gms2xx@gamsworld.org
Message Edit Search Utilities Spell Checker Privacy View Options
nvs01.gms
The Bat!
Message Specials Folder Account Tools View Options Help
nvs01.zip
12:3

From: Franz Nelissen <FNelissen@gams.com>
Reply-To: Franz Nelissen <FNelissen@gams.com>
To: gms2xx@gamsworld.org
CC:
BCC:
Subject: ampl

| From | To | Subject | Received | Created | Size |
|---|---|---|---|---|---|
| gms2xx@gams... | FNelissen@g... | Your scalar AMPL model nvs01 | 11:59 | 11:59 | 2,645 |

From gms2xx@gamsworld.org
To FNelissen@gams.com
Subject Your scalar AMPL model nvs01

```
Dear GMS2XX user,

A zip file with your scalar model is attached.
Please inspect memo.txt for warnings and translation
incapabilities.

Regards,

The GMS2XX Team
```

**Fig. 1.** GMS2XX/CONVERT E-mail submission and results

particular solver) online, and the server will automatically generate quality and performance reports and do cross-comparisons of all solvers.

**Test Cases:** The CONVERT tool is also available as an e-mail based translation service (GMS2XX) at GAMS World

`http://www.gamsworld.org/translate.htm`

to facilitate the translation of GAMS models into other modeling languages, such as AMPL, BARON, CplexLP, CplexMPS, LGO, LINGO, and MINOPT. Users submit their model as an e-mail attachment to `gms2xx@gamsworld.org` with the translation language listed in the subject line. The translated model is then returned via e-mail attachment. Figure 1 shows the process of converting the MINLPLib model `nvs01.gms` from GAMS to AMPL via the GMS2XX translation service. Table 2 shows the original model `nvs01.gms` and the converted AMPL model `ampl.mod`.

**Data Collection Tools:** The trace facility has an open API and allows ASCII trace file importing and exporting. Non-GAMS users can therefore produce a

**Table 2.** GMS2XX Example: Part of original GAMS Model `nvs01.gms` and translated model `ampl.mod` using the GMS2XX translation service

`nvs01.gms`

```
*  MINLP written by GAMS Convert at 02/21/03 13:01:13
*  Equation counts
*     Total       E       G       L       N       X       C
*         4       2       2       0       0       0       0
*  Variable counts
*                 x       b       i     s1s     s2s      sc      si
*     Total    cont  binary integer    sos1    sos2   scont    sint
*         4       2       0       2       0       0       0       0
*  FX     0       0       0       0       0       0       0       0

*  Nonzero counts
*     Total   const      NL     DLL
*        10       3       7       0
*
*  Solve m using MINLP minimizing objvar;

Variables  i1,i2,x3,objvar;
Positive Variables x3;
Integer Variables i1,i2;
Equations  e1,e2,e3,e4;

e1.. 420.169404664517*sqrt(900 + sqr(i1)) - x3*i1*i2 =E= 0;
e2..  - x3 =G= -100;
e3.. 296087.631843*(0.01+0.0625*sqr(i2))/(7200 + sqr(i1))-x3 =G= 0;
```

`ampl.mod`

```
#  MINLP written by GAMS Convert at 02/21/03 14:03:38
#
#  Reformulation has removed 1 variable and 1 equation

var i1 integer := 100, >= 0, <= 200;
var i2 integer := 100, >= 0, <= 200;
var x3 := 100, >= 0, <= 100;

minimize obj: 0.04712385*i2*(900 + i1^2)^0.5;

subject to

e1: 420.169404664517*sqrt(900 + i1^2) - x3*i1*i2 = 0;
e2:  - x3 >= -100;
e3: (2960.87631843 + 18505.4769901875*i2^2)/(7200 + i1^2) - x3 >= 0;
```

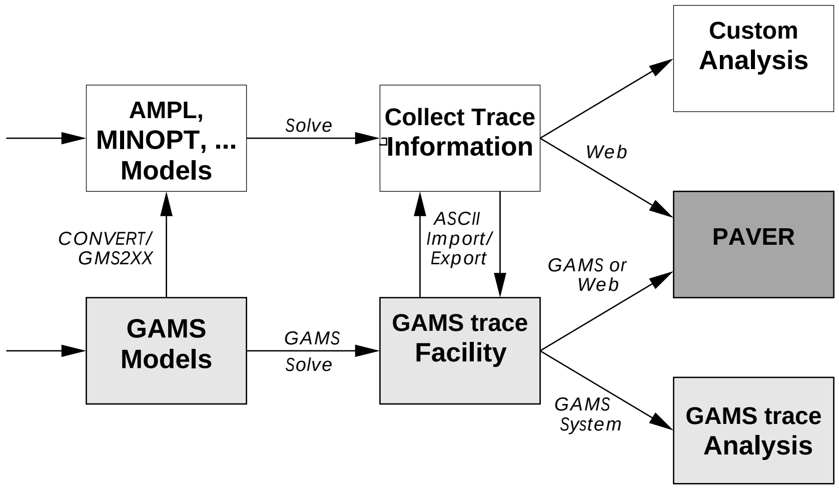


**Fig. 2.** Quality Assurance Process. The process is system and software independent at each phase. Models can be translated to other languages, benchmark runs performed with any optimization engine, and subsequent analysis done using web-based, GAMS, or customized tools

trace file by any preferred method, for example by writing scripts to parse the log output from an optimization engine or manually writing a trace record.

**Data Analysis Tools:** All performance tools are implemented in open source GAMS programs. Free access to these programs for users without a GAMS system is guaranteed through the Web interface PAVER described previously. The server is not restricted to trace files obtained through GAMS for generating performance analysis reports but can accept data from any optimization engine providing trace-like data files.

The open architecture and steps in the quality assurance process are illustrated in Figure 2. A user with a GAMS system can solve the model and automatically capture solve and model statistics via the trace file utility. The data can be analyzed then via GAMS or the PAVER web interface. Users of other modeling systems or optimization engines can use the GMS2XX tool to convert the model to another language. Once solved with this other engine, trace files can be created either manually or through some other automated fashion. Analysis can then proceed either customized or through the PAVER web interface. Note also that the ASCII trace interface allows simple import and export of data between GAMS and other formats.

## 4 Examples

In this section we illustrate how to create reproducible quality assurance results and performance results for local and global NLP and MINLP solvers using the framework described previously.

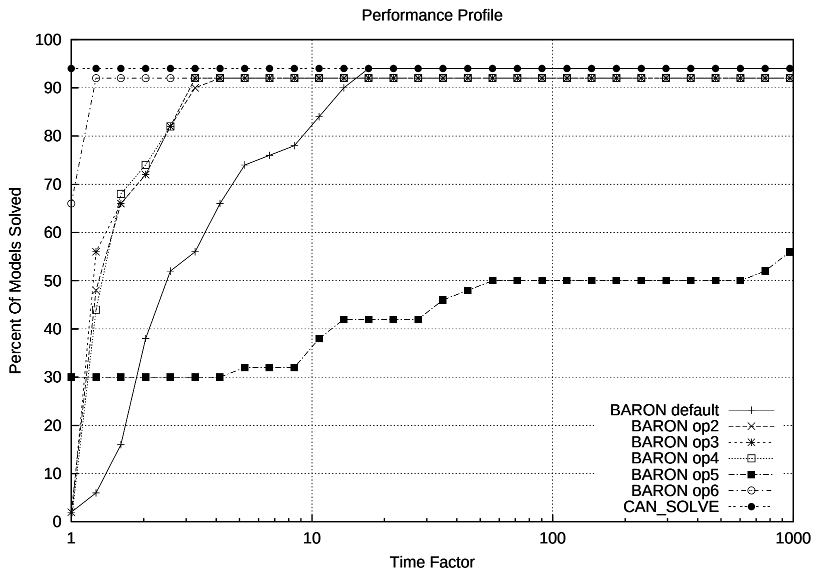


**Fig. 3.** PAVER Performance Profiles for LMP1 models. Graphs show measure of competitiveness of each "solver" in terms of efficiency. BARON with option 6 is the most efficient (see the profile for `Time Factor = 1`, solving 65% of the models the fastest. In terms of probability of success, BARON default has the highest rating. Roughly 95% of the models can be solved by BARON default given any amount of time

## 4.1 NLP Example: Linear multiplicative models

The first example involves 50 different instances of the NLP model `lmp1.gms` (linear multiplicative models [12]) found in the GAMS Model Library. We used the global optimization code BARON and ran it in default mode and with five different option files, specifying varying solver options to fine tune for this particular class of problems. We specified a time limit of 300 seconds. Performance data was collected via trace files from within GAMS and we used the PAVER web submission utility to compare performance.

The PAVER results returned via e-mail give a variety of information including robustness and efficiency information, as well as quality of solution. We show the *performance profiles* results [4] which provide a measure of competitiveness of a solver.

Figure 3 shows competitiveness of each BARON run with various options if we are interested only in solver efficiency. At a Time Factor of 1 (x-axis), the graph shows the percentage of models solved the fastest by a particular solver. In particular, BARON with option 6 (BARON op6) solves 65% of the models the fastest. As Time Factor goes to $\infty$, the graph gives solver robustness information, indicating the percentage of models a solver can solve at all given any amount of time. In this case BARON default has the highest probability of success at roughly 95%. Also note the graph associated with CAN_SOLVE, which shows the probability of success that any one of the six "solvers" can solve the models.

### 4.2 MINLP Example: Models from Gupta and Ravindran

In this example we choose 24 MINLP models from [10], available as part of the MINLPLib. We ran all available GAMS MINLP solvers (BARON, DICOPT, OQNLP, SBB). We also ran MINLP, a branch and bound code based on FILTER by Fletcher and Leyffer [14]. The solver MINLP was run remotely through the Network-Enabled Optimization Server (NEOS) [5], [11], [16], an online server for solving optimization problems. All solvers were run using a time limit of 20 seconds. All performance information for GAMS solvers was collected in trace files. For the MINLP solver run remotely through the NEOS server, we wrote a script to parse the output and create a trace file.

In order to evaluate solver performance we wrote a GAMS script to compare objective value information. The script also computes absolute and relative gaps and compares CPU time. The script first reads the trace file with the performance data, computes statistical information and then writes the output to a standard text file. The program is listed below.

*MINLP Example: GAMS Analysis Script*

```
$set col    julian,dir,eqnum,varnum,dvarnum,nz,nlnz,opt,modelstat
$set col    %col%,solvestat,obj,objest,res,iter,dom,nodes
Set  col    /%col%/,
     c(col) /modelstat,solvestat,obj,objest,res,nodes/

Alias(u1,u2,*);
$eolcom ,#

*=== Read in trace file
Table tracedata(*,*,col)
$ondelim
modelname,solvername,%col%
$offlisting
$call cat trace.trc minlpbb.csv | cut -d, -f1,3,6- > trace.tmp
$include trace.tmp
$onlisting
$offdelim
;

*=== Extract driving sets
Set modelname, solvername;
loop((u1,u2,)$tracedata(u1,u2,'julian'),
  modelname(u1)  = yes;
  solvername(u2) = yes;
);
```

```
parameter srep(u1,u2,col)
          mstat(u1,u2);

*=== Load reference solution
$gdxin minlpstat
$load mstat

*=== Select columns
srep(modelname,solvername,c) =
       tracedata(modelname,solvername,c);
srep(modelname,'Reference','obj') =
       mstat(modelname,'BestInt');

Parameter gap;
gap(modelname,solvername,'agap') =
       round(srep(modelname,solvername,'obj')
     - srep(modelname,'Reference','obj'),3);
gap(modelname,solvername,'agap')$(
       srep(modelname,solvername,'modelstat')<>1
   and srep(modelname,solvername,'modelstat')<>2
   and srep(modelname,solvername,'modelstat')<>8
                              ) = inf;
gap(modelname,solvername,'rgap') =
       gap(modelname,solvername,'agap')
     / abs(srep(modelname,'Reference','obj'));
gap(modelname,solvername,'obj')  =
       srep(modelname,solvername,'obj');
gap(modelname,solvername,'cpu')  =
       srep(modelname,solvername,'res');

display srep, gap;
```

The output created by the script shows the computed parameters `srep` and `gap`. It shows the objective function value, the absolute and relative gaps, as well as the CPU time used for a subset of models for each of the solvers in the comparison.

*MINLP Example: GAMS Analysis Script*

```
----     167  PARAMETER gap


                     obj      agap       rgap        cpu
NVS01.OQNLP       15.806     3.336      0.268      2.218
NVS01.BARON       12.470                           0.110
NVS01.SBB         12.470                           0.410
NVS01.DICOPT      12.470                           0.035
NVS01.MINLPBB     12.470                          20.000
NVS02.OQNLP        6.575     0.611      0.102     20.046
NVS02.BARON        5.964                           0.080
```

```
NVS02.SBB          5.964                          0.437
NVS02.DICOPT       5.964                          0.346
NVS02.MINLPBB      5.964                         20.000
NVS03.OQNLP       16.000                         20.015
NVS03.BARON       16.000                          0.060
NVS03.SBB         16.000                          0.305
NVS03.DICOPT      16.000                          0.069
NVS03.MINLPBB     16.000                         20.000
NVS04.OQNLP        0.720                          0.265
NVS04.BARON        0.720                          0.050
NVS04.SBB          0.720                          0.195
NVS04.DICOPT       2.120     1.400      1.944     0.227
NVS04.MINLPBB      0.720                         20.000
```

## 5 Conclusions

We have addressed the necessary steps for moving global optimization codes from an academic research lab into the commercial production environment. Reproducible quality assurance testing is the key to success. We have proposed a testing framework, which includes model collections, data collection tools and data analysis tools, and allow seamless testing inside the GAMS system, yet remains open for outside use by non-GAMS customers and researchers in general. We offer these models and tools along with our open invitation to use them as a means to strengthen collaboration among all groups interested in quality assurance for global optimization.

## References

1. Brooke, A., Kendrick, D., Meeraus, A.: GAMS: A User's Guide, The Scientific Press, Redwood City, California (1988)
2. Bussieck, M. R., Drud, A. S., Meeraus, A.: MINLPLib - A Collection of Test Models for Mixed-Integer Nonlinear Programming, Informs J. Comput., **15** (1) (2003) 114–119
3. Crowder, H., Dembo, R. S., Mulevy, J. M.: On Reporting Computational Experiments with Mathematical Software, ACM Transactions of Mathematical Software, **5** (2) (1979) 193–203
4. Dolan, E. D., Moré, J. J.: Benchmarking Optimization Software with Performance Profiles, Math. Programming, **91** (2002) 201–213
5. Dolan, E. D.: The NEOS Server 4.0 Administrative Guide, Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory (2001)
6. Dolan, E. D., Moré, J. J.: Benchmarking Optimization Software with COPS, Technical Memorandum ANL/MCS-TM-246, Argonne National Laboratory, Argonne, Illinois (2000)
7. Fourer, R., Gay, D. M.: AMPL: A Modeling Language for Mathematical Programming, The Scientific Press, Redwood City, California (1993)

8. GAMS Development Corp.: GAMS - The Solver Manuals, GAMS Development Corp., Washington, D.C. (2003)
9. Gay, D. M.: Electronic Mail Distribution of Linear Programming Test Problems, Mathematical Programming Society COAL Newsletter (1985)
10. Gupta, O. K. Ravindran, A.: Branch and Bound Experiments in Convex Nonlinear Integer Programming, Management Science, **13** (1985) 1544–1546
11. Gropp, W., Moré, J. J.: Optimization Environments and the NEOS Server, In: M. D. Buhmann and A. Iserles (Eds.), Approximation Theory and Optimization, Cambridge University Press, Cambridge (1997) 167–182
12. Konno, H., Kuno, T.: Linear Multiplicative Programming, Math. Prog., **56** (1992) 51–64
13. Kristjansson, B.: Optimization Modeling in Distributed Applications: How New Technologies such as XML and SOAP allow OR to provide web-based Services, INFORMS Roundtable, Savannah, Georgia (2001)
14. Leyffer, S.: User Manual for MINLP_BB, University of Dundee Numerical Analysis Report NA/XXX (1999)
15. Mittelmann, H., Pruessner, A.: A Server for Automated Performance Analysis and Benchmarking of Optimization Software, submitted (2003)
16. NEOS: http://www-neos.mcs.anl.gov/ (1997)
17. Pintér, J. D: LGO - A Model Development System for Continuous Global Optimization. User's Guide. (Current revised edition.), Pintér Consulting Services, Halifax, Nova Scotia (2002)
18. Schrage, L. S.: LINDO - An Optimization Modeling System, Scientific Press series; Fourth ed. Boyd and Fraser, Danvers, Massachussets (1991)
19. Tawarmalani, M., Sahinidis, N. V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications, Kluwer Academic Publishers, Dordrecht (2002)
20. Ugray, Z., Lasdon, L., Plummer, J., Glover, F., Kelly, J., Marti, R.: A Multistart Scatter Search Heuristic for Smooth NLP and MINLP Problems, INFORMS J. Comp., to appear (2002)