BENCH

Contents

1 Introduction

BENCH is a GAMS solver to help facilitate benchmarking of GAMS optimization solvers. BENCH calls all user-specified GAMS solvers for a particular modeltype and captures results in the standard GAMS listing file format. BENCH can call the GAMS/EXAMINER solver automatically to independently verify feasibility and optimality of the solution returned.

There are several advantages to using the BENCH solver instead of just creating a GAMS model or batch file to do multiple solves with the various solvers. The first is that the model does not need to be generated individually for each solve; BENCH spawns each solver using the matrix file generated during the initial call to BENCH. Furthermore, BENCH simplifies solution examination/verification by automatically utilizing EXAMINER. And finally, data can automatically be collected for use with the PAVER performance analysis server.

BENCH comes free with any GAMS system. Licensing is dependent on licensing of the subsolvers. Thus, BENCH runs all solvers for which the user has a valid license.

How to run a Model with BENCH:

BENCH is run like any other GAMS solver. From the command line this is:

```
>> gams modelname modeltype=bench
```

where `modelname` is the GAMS model name and `modeltype` the solver indicator for a particular model type (e.g. LP, MIP, RMIP, QCP, MIQCP, RMIQCP, NLP, DNLP, CNS, MINLP, or MCP). BENCH can also be specified via the option statement within the model itself before the solve statement:

```
option modeltype=bench;
```

The user must *specify the solvers* to be included by using the `solvers` option (specified in a solver option file called `bench.opt`). Otherwise, GAMS/BENCH returns with a warning message

```
Warning: no solvers selected. Nothing to be done.
```

For more information on using solver option files and the `solvers` option, see §2.

2 User-Specified Options

2.1 GAMS Options

BENCH works like other GAMS solvers, and many options can be set in the GAMS model. The most relevant GAMS options are `nodlim, optca, optcr, optfile, cheat, cutoff, prioropt`, and `tryint`. These options are global in the sense that they are passed on to all subsolvers called.

The options can be set either through an option statement

```
option optfile=1;
```

or through a model suffix, which sets them only for an individual model

```
modelname.optfile=1;
```

All of the options listed in the Chapter "Using Solver Specific Options" are implemented in BENCH and are passed on to the respective solvers. We remark that for a particular subsolver some of these options may not be valid. In this case, although they are passed on by BENCH to the respective subsolver, they may not be used.

The options listed below differ from the usual implementation and are based on *individual limits* for each solver called by BENCH.

Option	Description	Default
`iterlim`	Sets the **individual iteration limit**. The subsolver called by BENCH will terminate and pass on the current solution if the number of iterations for each solver exceeds this limit.	`10000`
`reslim`	Sets the **individual time limit** in seconds. The subsolver called by BENCH will terminate and pass on the current solution if the resource time for each solver exceeds this limit.	`1000`

2.2 The BENCH Options

BENCH solver options are passed on through solver option files. If you specify "<`modelname`>`.optfile = 1;`" before the SOLVE statement in your GAMS model. BENCH will then look for and read an option file with the name *bench.opt* (see "Using Solver Specific Options" for general use of solver option files). Unless explicitly specified in the BENCH option file, the solvers called by BENCH will not read option files. The syntax for the BENCH option file is

```
optname value
```

with one option on each line.

For example,

```
solvers conopt.1 minos snopt.2
```

This option determines the solvers to be called and is required. If the `solvers` option is omitted, then BENCH terminates with a warning message.

In this example, CONOPT will be called first with the option file *conopt.opt*. Then MINOS will be called with no option file and SNOPT will be called last with the option file *snopt.op2*. We note that the solvers are called in this order. This can

be of particular use since detailed solution information at the end of the GAMS listing file is for the final solver called. See the section describing the BENCH listing file for details.

Option	Description	Default
allsolvers	Indicator if all valid solvers for given modeltype should be run.	0
cumulative	Indicator if resource time and iteration limits are interpreted as total limits for all solvers or for individual solvers. For example if enabled and reslim=1500 then all solvers have a total of 1500 seconds. If not enabled, then each solver has a limit of 1500 seconds. 0: limits enforced for each solver separately 1: limits enforced cumulatively	0
dualcstol	EXAMINER dualcstol option. Dual complementary slackness tolerance. By dual CS we refer to complementary slackness between dual variables and the primal constraints.	1e-7
dualfeastol	EXAMINER dualfeastol option. Dual feasibility tolerance. This tolerance is used for the checks on the dual variables and the dual constraints.	1e-6
examiner	Flag to call GAMS/EXAMINER to independently verify solution for feasibility and optimality. 0: skip solution verification. 1: verify solution using EXAMINER.	0
outlev	Log output level. 1: BENCH summary log output only. 2: BENCH summary log output and log output of each solver.	2
paver	Indicator if PAVER trace files should be written. Enabling causes a trace file solver.pvr to be written for each solver called. If the solver uses an option file, then the resulting file is solver-optnum.pvr, where optnum is the option file number. The files created can be submitted to the PAVER Server for automated performance analysis. See http://www.gamsworld.org/performance/paver 0: no PAVER trace files written 1: write trace file for each solver called	0
paverex	Indicator if PAVER trace files should be written for the Examiner run. Enabling causes a trace file solver-ex.pvr to be written for each solver called. If any Examiner check fails (independent or solver), then we return a model status 14 (no solution returned) and a solver status of 4 (terminated by solver). If no Examiner check is done, for example, because the return status is infeasible, then the status codes are returned as is. If the solver uses an option file, then the resulting file is solver-optnum-ex.pvr, where optnum is the option file number. The files created can be submitted to the PAVER Server for automated performance analysis. See http://www.gamsworld.org/performance/paver 0: no PAVER trace files for Examiner run written 1: write trace file for each solver called	0
primalcstol	EXAMINER primalcstol option. Primal complementary slackness tolerance. By primal CS we refer to complementary slackness between primal variables and the dual constraints.	1e-7
primalfeastol	EXAMINER primalfeastol option. Primal feasibility tolerance. This tolerance is used for the checks on the primal variables and the primal constraints.	1e-6
returnlastsol	Return the solution information of the last solver. 0: Do not return a solution. 1: Return the solution of last solver.	0

Option	Description	Default
solvers	solver[.n] Solver is the name of the GAMS solver that should be used, and n is the integer corresponding to optfile for the root node. If .n is missing, the optfile treated as zero (i.e., the solver) will not look for an options file. This is a required option.	none

2.3 Solvers Requiring Subsolvers

For GAMS solvers requiring subsolvers, for example the MINLP solvers DICOPT and SBB, BENCH requires some care. By default, BENCH assumes the default subsolvers for solvers requiring them. For example, if an MINLP model is solved via BENCH, then the MINLP solvers specified by the user in the `bench.opt` solver option file only call the default subsolvers available to them and *do not* run all valid subsolvers.

If users wish to do benchmarks with particular subsolvers, then subsolvers can be passed along via subsolver option files. To tell BENCH to use a particular solver, users can use the `solvers` option in the BENCH option file. For example, the BENCH option

```
solvers sbb dicopt
```

specifies that the solvers SBB and DICOPT are executed (and in that particular order). The subsolvers used by SBB and DICOPT will be the default solvers for your particular GAMS system.

To use a particular subsolver, users can append to the solver indicator. For example

```
solvers sbb sbb.1 sbb.2 dicopt dicopt.1
```

specifes that SBB without an option file is called first, then SBB with an option file called *sbb.opt* and then with an option file called *sbb.op2*. Then DICOPT is called without an option file and then with an option file called `dicopt.op2`. Within these option files, the particular subsolvers can be specified. The input of the solver option file is echoed in the listing file created by BENCH to help distinguish the different solver calls.

3 Benchmark Analysis Using the PAVER Server

Benchmark data obtained using GAMS/BENCH can be automatically analyzed using the PAVER Server. For more information on PAVER, see `http://www.gamsworld.org/performance/paver`.

In order to enable creation of the necessary data files for submission to PAVER, users must enable the `paver` option as described in §2.

For example, suppose a user has a set of models and wishes to compare three solvers, say CONOPT3, MINOS, and SNOPT. The user would then create a `bench.opt` solver option file with the entries

```
solvers conopt3 minos snopt
paver 1
```

Solving the models using `bench` as the solver will create PAVER data files, namely one for each solver: `conopt.pvr`, `minos.pvr`, and `snopt.pvr`, which can be submitted to the PAVER server at

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

for automated analysis. Note that all PAVER trace files are appended to if they exist and if subsequent solves are made.

4 Solution Verification Using Examiner

4.1 Examiner Checks

BENCH can automatically call the GAMS/EXAMINER[1] solver to check the solution for feasibility and complementarity. In particular, EXAMINER checks for

- primal feasibility: feasibility of both primal variables and primal constraints.
- dual feasibility: feasibility of both dual variables and dual constraints.

[1] See `http://www.gams.com/solvers/examiner.pdf`

- primal complementary slackness: complementary slackness of primal variables to dual constraints.
- dual complementary slackness: complementary slackness of dual variables to primal constraints.

where EXAMINER does two types of checks:

- **Solvepoint:** the point returned by the subsolver. The subsolver returns both level and marginal values for the rows and columns: Examiner uses these exactly as given.
- **Solupoint:** EXAMINER uses the variable levels (primal variables) and equation marginals (dual variabless) to compute the equation levels and variable marginals. The variable levels and equation marginals used are those returned by the subsolver.

By default, BENCH does not call EXAMINER to verify the solution. To enable solution verification, specify

```
examiner 1
```

in the `bench.opt` solver option file. Of interest are also the EXAMINER tolerances `dualcstol, dualfeastol, primalcstol`, and `primalfeastol` which can also be set in the BENCH solver option file. See the BENCH solver options for details. For more information, see the EXAMINER documentation.

4.2 Examiner Output in BENCH

Examiner output, if solution verification is enabled, is given in the log output during the actual solve and summary information is given in the final BENCH summary under the `Examiner` column. Models either pass (`P`) or fail (`F`) based on the default Examiner or user-specified tolerances given. If EXAMINER does not do a check, for example, because the solver returns a model status of infeasible, then the `Examiner` column is given as (`N`).

If Examiner is not enabled, then `n/a` is listed under the `Examiner` column.

The first entry under `Examiner` is the Examiner status for using solver provided variable constraint level values (`solvpoint`). The second entry is the `solupoint`, where GAMS computes the constraint levels from the variable levels returned by the solver.

An example is given below, where we specified to use the solvers BDMLP, MINOS, XPRESS, and CPLEX on the GAMS Model Library model `trnsport.gms`:

```
Solver     Modstat  Solstat    Objective       ResUsd    Examiner
-------------------------------------------------------------------
BDMLP          1       1        153.6750        0.000       P/P
MINOS          1       1        153.6750        0.000       P/P
XPRESS         1       1        153.6750        0.040       P/P
CPLEX          1       1        153.6750        0.000       P/P
```

In the example below, EXAMINER is enabled, but does not perform any checks because the return status of the solver lists the model as infeasible (see the `Examiner` column (`N/N`).

```
Solver     Modstat  Solstat    Objective       ResUsd    Examiner
-------------------------------------------------------------------
BDMLP          5       1          0.0000        0.000       N/N
```

For models having discrete variables, for example MIP, MIQCP, or MINLP, we also show the best bound. A sample output using the model `magic.gms` is shown below.

```
Solver    Modstat Solstat      Objective      BestBound      ResUsd  Examiner
-------------------------------------------------------------------------------
CPLEX           8       1    991970.0000    985514.2857       0.000      n/a
XPRESS          1       1    988540.0000    988540.0000       0.060      n/a
MOSEK           8       1    988540.0000    988540.0000       0.170      n/a
```

5 Output

5.1 The BENCH Log File

The BENCH log output contains complete log information for each solver called. The individual solver calls are indicated by the entry

```
--- Spawning solver : (Solver Name)
```

followed by the log output of the individual solver.

An example of the log output using the transportation model (`trnsport.gms`) from the GAMS model library. We specify the solvers `BDMLP`, `XPRESS`, `MINOS` and `CPLEX` via the option file `bench.opt`:

```
GAMS Rev 138  Copyright (C) 1987-2004 GAMS Development. All rights reserved
Licensee: GAMS Development Corp.                          G040421:1523CR-LNX
          GAMS Development Corp.                                      DC3665
--- Starting compilation
--- trnsport.gms(69) 3 Mb
--- Starting execution
--- trnsport.gms(45) 4 Mb
--- Generating model transport
--- trnsport.gms(66) 4 Mb
---    6 rows, 7 columns, and 19 non-zeroes.
--- Executing BENCH

GAMS/BENCH    Jan 19, 2004 LNX.00.NA 21.3 004.027.041.LXI

  GAMS Benchmark Solver

  Reading user supplied options file /home/gams/support/bench.opt
  Processing...
  > solvers bdmlp minos xpress cplex


--- Spawning solver : BDMLP


 BDMLP 1.3     Jan 19, 2004 LNX.00.01 21.3 058.050.041.LXI

 Reading data...
 Work space allocated           --    0.03 Mb

    Iter    Sinf/Objective  Status    Num  Freq
       1    2.25000000E+02  infeas      1     1
       4    1.53675000E+02  nopt        0
 SOLVER STATUS: 1 NORMAL COMPLETION
 MODEL STATUS : 1 OPTIMAL
 OBJECTIVE VALUE     153.67500

--- Spawning solver : MINOS


MINOS-Link    Jan 19, 2004 LNX.M5.M5 21.3 029.050.041.LXI GAMS/MINOS 5.51

    GAMS/MINOS 5.51, Large Scale Nonlinear Solver
```

```
    B. A. Murtagh, University of New South Wales
    P. E. Gill, University of California at San Diego,
    W. Murray, M. A. Saunders, and M. H. Wright,
    Systems Optimization Laboratory, Stanford University

 Work space allocated           --    1.01 Mb

  Reading Rows...
  Reading Columns...

 EXIT - Optimal Solution found, objective:        153.6750


--- Spawning solver : XPRESS


Xpress-MP     Jan 19, 2004 LNX.XP.XP 21.3 024.027.041.LXI Xpress lib 14.24
Xpress-MP licensed by Dash to GAMS Development Corp. for GAMS

Reading data . . . done.

Reading Problem gmsxp_xx
Problem Statistics
           6 (      0 spare) rows
           7 (      0 spare) structural columns
          19 (      0 spare) non-zero elements
Global Statistics
           0 entities        0 sets        0 set members
Presolved problem has:      5 rows       6 cols      12 non-zeros

   Its         Obj Value      S   Ninf  Nneg        Sum Inf  Time
     0           .000000      D      3     0     900.000000     0
     3        153.675000      D      0     0        .000000     0
Uncrunching matrix
     3        153.675000      D      0     0        .000000     0
Optimal solution found
optimal LP solution found: objective value 153.675

--- Spawning solver : CPLEX


GAMS/Cplex    Jan 19, 2004 LNX.CP.CP 21.3 025.027.041.LXI For Cplex 9.0
Cplex 9.0.0, GAMS Link 25

Reading data...
Starting Cplex...
Tried aggregator 1 time.
LP Presolve eliminated 1 rows and 1 columns.
Reduced LP has 5 rows, 6 columns, and 12 nonzeros.
Presolve time =    0.00 sec.

Iteration      Dual Objective            In Variable           Out Variable
     1              73.125000    x(seattle.new-york) demand(new-york) slack
     2             119.025000     x(seattle.chicago)  demand(chicago) slack
     3             153.675000    x(san-diego.topeka)   demand(topeka) slack
     4             153.675000  x(san-diego.new-york)  supply(seattle) slack
```

```
Optimal solution found.
Objective :         153.675000



--- BENCH SUMMARY:

  Solver     Modstat  Solstat    Objective       ResUsd    Examiner
  ---------------------------------------------------------------
  BDMLP          1       1        153.6750        0.000       n/a
  MINOS          1       1        153.6750        0.000       n/a
  XPRESS         1       1        153.6750        0.040       n/a
  CPLEX          1       1        153.6750        0.000       n/a

--- Restarting execution
--- trnsport.gms(66) 0 Mb
--- Reading solution for model transport
--- trnsport.gms(68) 3 Mb
*** Status: Normal completion
```

5.2 The BENCH Listing File

The BENCH listing file is similar to the standard GAMS format. It contains a benchmark summary of all solvers called. The regular solve summary for BENCH does not return a solution (although the solution of the final solve can be returned using the `returnlastsol` option). For the example below we use the `batchdes.gms` model from the GAMS model library using the solvers SBB and DICOPT. We specify the two solvers using a *bench.opt* option file with the entry:

```
solvers sbb.1 dicopt
```

Note that SBB will use a solver option file called *sbb.opt*.

```
               S O L V E      S U M M A R Y

    MODEL   batch               OBJECTIVE  cost
    TYPE    MINLP               DIRECTION  MINIMIZE
    SOLVER  BENCH               FROM LINE  183

 **** SOLVER STATUS     1 NORMAL COMPLETION
 **** MODEL STATUS      14 NO SOLUTION RETURNED
 **** OBJECTIVE VALUE                0.0000

 RESOURCE USAGE, LIMIT          0.000      1000.000
 ITERATION COUNT, LIMIT         0         10000
   Reading user supplied options file /home/models/bench.opt
   Processing...
   > solvers sbb.1 dicopt
```

Note that the model status return code for BENCH itself is always SOLVER STATUS 1 and MODEL STATUS 14, since BENCH itself does not return a solution by default. To obtain the status codes and solution information of the last solver, the `returnlastsol` option can be enabled. See the BENCH solver option section.

In addition the listing file contains complete solve summary information for each solver called. Also, note that the option file used for SBB and its contents are echoed to the SBB summary.

```
                B E N C H M A R K      S U M M A R Y
```

```
   SOLVER             SBB
   SOLVER STATUS    1 NORMAL COMPLETION
   MODEL STATUS     8 INTEGER SOLUTION
   OBJECTIVE VALUE           167427.6571
   RESOURCE USAGE, LIMIT          0.080       1000.000
   ITERATION COUNT, LIMIT       139         100000
   EVALUATION ERRORS, LIMIT       0              0
   OPTION FILE        sbb.opt

Reading user supplied options file sbb.opt
Processing...
> rootsolver conopt2
> subsolver snopt



   SOLVER             DICOPT
   SOLVER STATUS    1 NORMAL COMPLETION
   MODEL STATUS     8 INTEGER SOLUTION
   OBJECTIVE VALUE           167427.6571
   RESOURCE USAGE, LIMIT          0.100        999.920
   ITERATION COUNT, LIMIT       117          99861
   EVALUATION ERRORS, LIMIT       0              0
```

Note that the listing file does not contain detailed solution information since BENCH does not return any values.

6 Interrupting BENCH with Ctl-C

BENCH passes all *Control-C* (Ctl-C) signals to the respective subsolvers. If a terminate signal via Ctl-C is sent in the middle of a solver run (i.e. not initially when the solver begins execution), the individual subsolver is terminated.

To terminate not only the subsolver but also BENCH, a Ctl-C signal should be sent at the beginning of a solver's execution. Thus, several Ctl-C in rapid succession will terminate BENCH.

Benchmark summary information will be written to the listing file for each solver that has successfully completed without any signal interrupt.

7 Benchmark Example

In this section we will give a small example showing how to use the BENCH solver and automate the subsequent analysis using the PAVER Server. In particular, we will run the three versions of CONOPT (CONOPT1, CONOPT2, and CONOPT3, as well as CONOPT3 with no scaling) on the default instance of the COPS models for nonlinear programming. We will use the 17 models available from the GAMS Model Library.

Firs we need to extract all of the models from the GAMS Model Library. We can create a file which will extract these automatically. Create a file called `getcops.gms` with the entries below:

```
$call gamslib camshape
$call gamslib catmix
$call gamslib chain
$call gamslib elec
$call gamslib flowchan
```

```
$call gamslib gasoil
$call gamslib glider
$call gamslib jbearing
$call gamslib lnts
$call gamslib methanol
$call gamslib minsurf
$call gamslib pinene
$call gamslib polygon
$call gamslib popdynm
$call gamslib robot
$call gamslib rocket
$call gamslib torsion
```

Running the file using `gams getcops.gms` extracts the models. Then create a BENCH solver option file called `bench.opt` with the entries

```
solvers conopt1 conopt2 conopt3 conopt3.1
paver 1
```

The first entry tells BENCH to run the solvers CONOPT1, CONOPT2, and CONOPT3 and then CONOPT3 with the option file `conopt3.opt`. The second entry tells BENCH to create PAVER trace files. These can be submitted to the PAVER for automated performance analysis. Now create an option file `conopt3.opt` with the entry

```
lsscal f
```

which tells CONOPT3 not to use scaling.

We can now run the models in batch mode, for example by creating a GAMS batch file `runcops.gms` with the following entries:

```
$call gams camshape.gms nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams catmix.gms   nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams chain.gms    nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams elec.gms     nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams flowchan.gms nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams gasoil.gms   nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams glider.gms   nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams jbearing.gms nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams lnts.gms     nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams methanol.gms nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams minsurf.gms  nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams pinene.gms   nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams polygon.gms  nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams popdynm.gms  nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams robot.gms    nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams rocket.gms   nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
$call gams torsion.gms  nlp=bench optfile=1 reslim=10 iterlim=999999 domlim=99999
```

Running the file using the command `gams runcops.gms` runs all models with all three solvers through the GAMS/BENCH solver. Furthermore, three PAVER trace files are created: `conopt1.pvr`, `conopt2.pvr`, `conopt3.pvr` and `conopt3-1.pvr`, where the latter is for CONOPT3 with no scaling. Users can then submit the three trace files to the PAVER Server for automated analysis.

The resulting performance plot in Figure 25.1 shows the efficiency of each solver/solver option.

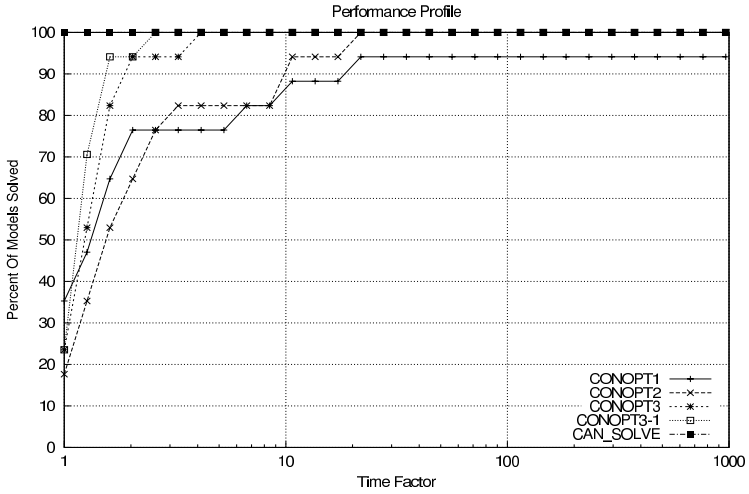

Figure 25.1: PAVER: Process Overview