Basic Solver Usage

Contents

1 Introduction

For the novice GAMS user, solver usage can be very simple: you run the model and inspect the listing file to see what the solution is. No knowledge of solver options or solver return codes is required. While this is enough for some users, most will quickly find they need some basic knowledge of how to control the solver and interpret the results. This section describes the GAMS options that are used to control a solver, how the GAMS solvers interpret these options, and how to interpret the model and solver status codes the solvers return.

While most solvers allow the user to set additional, solver-specific options, we will not be concerned with those here. In most cases, it is not necessary to use any solver-specific options: use of the generic GAMS options is sufficient. This carries an important benefit: since the solvers interpret the GAMS options in a consistent way, a GAMS option setting applies to all solvers, not just to a specific one.

2 GAMS Options

Options exist in two forms: global or model-specific. The option statement sets a global GAMS option, e.g.

```
option iterlim = 100;
```

while the model suffix sets a GAMS option for an individual model:

```
mymodel.iterlim = 10;
```

In addition, the default value of a global GAMS option can be set on the GAMS command line:

```
gams trnsport iterlim = 100
```

If a model-specific option is set, this takes precedence over the global setting. You can unset any model-specific option by assigning it the default value of NA:

```
mymodel.iterlim = NA;
```

The GAMS options for controlling solvers follow. Included with each option is a description of how this option is interpreted by a GAMS solver.

Option	Description
iterlim	Sets a limit on the simplex iterations (i.e. pivots) performed by the solver. If this limit is hit, the solver will terminate and return solver status 2 ITERATION INTERRUPT. Note that this option does not apply to other types of iterations (e.g. barrier iterations, major iterations in a nonlinear solver). These limits must be set by solver-specific options. In case many subproblems are solved via pivotal methods (e.g. in Branch and Bound or in an NLP solver), iterlim may be used as either a per-subproblem or cumulative pivot limit: this is solver dependent.
reslim	Sets the time limit in seconds. If this limit is hit, the solver will terminate and return solver status 3 RESOURCE INTERRUPT. The solver should start the clock fairly early, so that time required to read in the problem and do any reformulation, preprocessing, or presolving is included in the time limit.
optfile	If nonzero, the solver should read an option file. If optfile=1 the name of the option file is *solvername*.opt. If optfile is between 2 and 999, the value determines the extension used. For example, optfile=2 implies *solvername*.op2, optfile=67 implies *solvername*.o67, optfile=525 implies *solvername*.525, etc.
nodlim	Sets the branch and bound node limit. This is a limit on the total number of nodes in the tree, not on the number of active nodes. If this limit is hit, the solver will terminate and return solver status 4 TERMINATED BY SOLVER.
optca	MIP absolute optimality criterion. The absolute gap is defined to be $\|BP - BF\|$, where the best found value *BF* is the objective function value of the best integer solution found thus far and the best possible value *BP* is the current bound on the problem's solution. If the absolute gap is no greater than optca, the solver will terminate and return solver status 1 NORMAL COMPLETION and model status 8 INTEGER SOLUTION. Note that this is a termination test only; setting this option should not change the global search.
optcr	MIP relative optimality criterion. The solver will stop as soon as it has found a feasible solution proven to be within optcr of optimal. The precise definition of optcr depends on the solver. $\|BP-BF\|/\|BP\|$ and $\|BP-BF\|/\|BF\|$ are both in common use and specific solvers make different adjustments when the denominator approaches zero. See the specific solver chapters for details.
prioropt	Instructs the solver to use the priority branching information passed by GAMS through variable suffix values *variable*.prior. If and how priorities are used is solver-dependent.
cheat	MIP cheat value: Each new integer solution must be at least cheat better than the previous one. This can speed up the search, but the search may miss the optimal solution. The cheat option is specified in absolute terms (like the optca option), so that non-negative values are appropriate for both minimizattion and maximization models. Using the cheat option invalidates any reporting of the best bound or optimality gaps.
cutoff	Cutoff value: When the branch and bound search starts, the parts of the tree with an objective worse than cutoff are deleted. This can sometimes speed up the initial phase of the branch and bound algorithm, at the cost of ignoring integer solutions whose value is worse than cutoff.
tryint	Signals the solver to make use of a partial or near-integer-feasible solution stored in current variable values to get a quick integer-feasible point. If or how tryint is used is solver-dependent.
bratio	GAMS uses the bratio value to determine if an advanced basis exists (see the GAMS User's Guide). The result of this test is passed as a logical flag to the solver. All the pivotal algorithms in GAMS solvers will make use of this advanced basis to speed up problem solution.
domlim	Sets the domain violation limit. Domain errors are evaluation errors in the nonlinear functions (e.g. $\sqrt{x}$ for $x < 0$). When a domain violation occurs the domain error count is increased by one; a solver will terminate if this count exceeds domlim and return solver status 5 EVALUATION ERROR LIMIT. Note that some solvers operate in a mode where trial function evaluations are performed; these solvers will not move to points at which evaluation errors occur, so the evaluation errors at trial points are not counted against the limit.

Option	Description
sysout	If option sysout=on GAMS will echo *all* the solver messages to the GAMS listing file. This is useful for debugging or to get additional information about a solver run. Normally, only those messages flagged by solver as destined for the listing file get listed. sysout exists only as a global option, and can be set from the command line using an integer (e.g. sysout=1)
workfactor	Specifies a factor to be applied to the solver-computed memory estimate. E.g. setting workfactor=2 doubles the memory estimate. In case a solver allocates memory dynamically as it is needed, this option will have no affect. In case workfactor and workspace are both specified, the workspace setting takes precedence.
workspace	Specifies the amount (in MB) of memory the solver should allocate. This is used to override the solver-computed memory estimate. In case a solver allocates memory dynamically as it is needed, this option will have no affect. workspace exists only as a model-specific option.

3 The Solver Option File

To specify solver-specific options, it is necessary to use a solver option file. Two things are required to do this: you must create an option file having a proper name, and you must tell the solver to read and use this option file.

To tell a solver to use an option file, you can set the optfile model suffix to a positive value. For example,

```
model mymodel /all/;
mymodel.optfile = 1;
solve mymodel using nlp maximizing dollars;
```

The option file takes its name from the solver being used: *solvername.XXX*, where '*solvername*' is the name of the solver that is specified, and the suffix *XXX* depends on the value to which the model suffix optfile has been set. If its value is 1, the suffix is *opt*. For example, the option file for CONOPT is called *conopt.opt*; for DICOPT, it is *dicopt.opt*.

If you do not set the .optfile suffix to a nonzero value, no option file will be used even if one exists.

To allow different option file names for the same solver, the .optfile model suffix can take on values between 2 and 999. In this case, the option file extension is computed from the .optfile value by replacing the characters in opt with the digits in the characters in the .optfile value, starting from the right. For example,

optfile model suffix value	Name of option file
0	No option file used
1	*solvername*.opt
2	*solvername*.op2
3	*solvername*.op3
10	*solvername*.o10
91	*solvername*.o91
100	*solvername*.100
999	*solvername*.999

For example, setting mymodel.optfile to 23 will result in the option file *conopt.o23* being used for CONOPT, and *dicopt.o23* being used for DICOPT.

The format of the options file is not completely standard and changes marginally from solver to solver. This section illustrates some of the common features of the option file format. Please check the solver-specific documentation before using an option file.

Blank lines in an option file are ignored. Each nonblank line falls into one of two categories

- a comment line
- an option specification line

A comment line begins with an asterisk (*) in the first column, is not interpreted by either GAMS or the solver, and is used purely for documentation. Each option specification line can contain only one option. The format for specifying options is as follows:

```
keyword(s)    [modifier]    [value]
```

The keyword may consist of one or more words and is not case sensitive. The value might be an integer, a real, or a string. All solvers will accept real numbers expressed in scientific (i.e. `E`) format. Note that not all options require modifiers or values.

Any errors in the spelling of keyword(s) or modifiers will lead to that option being misunderstood and therefore ignored. Errors in the value of an option can result in unpredictable behavior. When detected, errors are either ignored or pushed to a default or limiting value, but not all can or will be detected. Option values should be chosen thoughtfully and with some care.

Consider the following CPLEX options file,

```
* CPLEX options file
barrier
crossover 2
```

The first line begins with an asterisk and therefore contains comments. The first option specifies the use of the barrier algorithm to solver the linear programming problem, while the second option specifies that the crossover option 2 is to be used. Details of these options can be found in the CPLEX section of this manual.

Consider the following MINOS options file,

```
*MINOS options file
scale option 2
completion partial
```

The first option sets the scale option to a value of 2. In this case, the key word `'scale option'` consists of two words. In the second line, the `completion` option is set to `partial`. Details of these options can be found in the MINOS section of this manual.

4 GAMS Dot Options

Dot options in a solver option file allow users to associate values to variables and equations using the GAMS name of the variables and equations. The general syntax of a dot option in the option file is as follows:

(variable/equation name)`.optionname` *(value)*

Dot options can be specified for *all*, a *block*, a *slice*, and a *single* variable and equation. Please note that a specific dot option might only applied to variables or equations (e.g. the GAMS/Gurobi dot option `prior` applies to variables only). The following example makes the use of the dot option clear.

For example, suppose we have a GAMS declaration:

```
Set i /i1*i5/;
Set j /j2*j4/;
variable v(i,j); equation e(i,j);
```

Consider the following lines in an option file with the imaginary option name `dotopt`:

Line in option file	**Explanation**
`variables.dotopt 1`	Sets the value of all variables to 1
`equations.dotopt 2`	Sets the value of all equations to 2
`v.dotopt 3`	Sets the value of the variables in block `v` to 3
`e.dotopt(*,*) 4`	Sets the value of the equations in block `e` to 4
`v.dotopt(*,'j2') 5`	Sets the value of the variables `v` that have `j2` in the second index position (slice) to 5
`e.dotopt('i3',*) 6`	Sets the value of the equations `e` that have `i3` in the first index position (slice) to 6
`w.dotopt('i2') 7`	Sets the value of the single variables `v('i2')` to 7
`e.dotopt('i3','j3') 8`	Sets the value of the single equations `e('i3','i3')` to 8

The values of the dot option are applied in correspondence to the sequence they appear in the option file. In the current example, the values of `dotopt` for the equation `e` would be as follows:

e.dotopt	i1	i2	i3
j2	4	4	6
j3	4	4	8
j4	4	4	6