OSL

Contents

1 Introduction

This document describes the GAMS interface to OSL.

OSL is the IBM Optimization Subroutine Library, containing high performance solvers for LP (linear programming), MIP (mixed integer programming) and QP (quadratic programming) problems. GAMS does not support the QP capabilities of OSL, you have to use a general non-linear solver like MINOS or CONOPT for that.

OSL offers quite a few algorithms and tuning parameters. Most of these are accessible by the GAMS user through an option file. In most cases GAMS/OSL should perform satisfactory without using any options.

2 How to Run a Model with OSL

OSL is capable of solve models of the following types: LP, RMIP and MIP. If you did not specify OSL as a default LP, RMIP or MIP solver, then you can use the following statement in your GAMS model:

```
option lp = osl; { or RMIP or MIP }
```

It should appear before the SOLVE statement.

3 Overview of OSL

OSL offers several algorithms for solving LP problems: a primal simplex method (the default), a dual simplex method, and three interior point methods: primal, primal-dual and primal-dual predictor-corrector. For network problems there is a network solver.

Normally the primal simplex method is a good method to start with. The simplex method is a very robust method, and in most cases you should get good performance with this solver. For large models that have to be solved many times it may be worthwhile to see if one of the other methods gives better results. Also changing the tuning parameters may influence the performance. The `method` option can be used to use another algorithm than the default one.

3.1 The Simplex Method

The most used method is the primal simplex method. It is very fast, and allows for restarts from an advanced basis. In case the GAMS model has multiple solves, and there are relatively minor changes between those LP models, then the solves after the first one will use basis information from the previous solve to do a 'jump start'. This is completely automated by GAMS and normally you should not have to worry about this.

In case of a 'cold start' (the first solve) you will see on the screen the message `'Crash...'`. This will try to create a better basis than the scratch ('all-slack') basis the Simplex method would normally get. The crash routine does not use much time, so it is often beneficial to crash. Crashing is usually not used for subsequent solves because that would destroy the advanced basis. The default rule is to crash when GAMS does not pass on a basis, and not to crash otherwise. Notice that in a GAMS model you can use the `bratio` option to influence GAMS whether or not to provide the solver with a basis. The default behavior can be changed by the `crash` option in the option file.

By default the model is also scaled. Scaling is most of the time beneficial. It can prevent the algorithm from breaking down when the matrix elements have a wide range: i.e. elements with a value of 1.0e-6 and also of 1.0e+6. It can also reduce the solution time. The presolver is called to try to reduce the size of the model. There are some simple reductions that can be applied before the model is solved, like replacing singleton constraints (i.e. `x =l= 5 ;`) by bounds (if there was already a tighter bound on X we just can remove this equation). Although most modelers will already use the GAMS facilities to specify bounds (`x.lo` and `x.up`), in many cases there are still possibilities to do these reductions. In addition to these reductions OSL can also remove some redundant rows, and substitute out certain equations. The presolver has several options which can be set through the `presolve` option.

The presolve may destroy an advanced basis. Sometimes this will result in very expensive restarts. As a default, the presolve is not used if an advanced basis is available. If using the presolve procedure is more useful than the use of an advanced basis, one can still force a presolve by using an option file.

GAMS/OSL uses the order: scale, presolve, crash. There is no possibility to change this order unless you have access to the source of the GAMS/OSL program. After the model is solved we have to call the postsolver in case the presolver was used. The postsolver will reintroduce the variables and equations the presolve substituted out, and will calculate their values. This solution is an optimal solution, but not a basic solution. By default we call simplex again to find an optimal basis. This allows us to restart from this solution. It is possible to turn off this last step by the option `postsolve 0`.

Occasionally you may want to use the dual simplex method (for instance when the model is highly primal degenerate, and not dual degenerate, or when you have many more rows than columns). You can use the `method dsimplex` option to achieve this. In general the primal simplex (the default) is more appropriate: most models do not have the characteristics above, and the OSL primal simplex algorithm is numerically more stable.

The other options for the Simplex method like the refactorization frequency, the Devex option, the primal and the dual weight and the change weight option are only to be changed in exceptional cases.

3.2 The Interior Point Methods

OSL also provides you with three interior point solvers. You should try them out if your model is large and if it is not a restart. The primal-dual barrier method with predictor-corrector is in general the best algorithm. This can be set by `method interior3`. Note that the memory estimate of OSL is in many cases not sufficient to solve a model with this method. You can override OSL's estimate by adding to using the workspace model suffix as shown in Section 4.2. We have seen models

where we had to ask for more than twice as much as the estimate. It is worthwhile to check how the interior points are doing on your model especially when your model is very large.

3.3 The Network Method

A linear model can be reformulated to a network model if

- The objective variable is a free variable.
- The objective variable only appears in the objective function, and not somewhere else in the model. This in fact defines the objective function.
- The objective function is of the form `=e=`.
- Each variable appears twice in the matrix (that is excluding the objective function) once with a coefficient of +1 and once with a coefficient of -1. In case there is a column with two entries that are the same, GAMS/OSL will try a row scaling. If there are no matrix entries left for a column (only in the objective function) or there is only one entry, GAMS/OSL will try to deal with this by adding extra rows to the model.

4 GAMS Options

The following GAMS options are used by GAMS/OSL.

4.1 Options Specified Through the Option Statement

The following options are specified through the option statement. For example,

```
set iterlim = 100 ;
```

sets the iterations limit to 100.

Option	Description
`iterlim`	Sets the iteration limit. The algorithm will terminate and pass on the current solution to GAMS. In case a pre-solve is done, the post-solve routine will be invoked before reporting the solution.
`reslim`	Sets the time limit in seconds. The algorithm will terminate and pass on the current solution to GAMS. In case a pre-solve is done, the post-solve routine will be invoked before reporting the solution.
`optca`	Absolute optimality criterion for a MIP problem.
`optcr`	Relative optimality criterion for a MIP problem.
`bratio`	Determines whether or not to use an advanced basis.
`sysout`	Will echo the OSL messages to the GAMS listing file. This option is useful to find out exactly what OSL is doing.

4.2 Options Specified Through Model Suffixes

The following options are specified through the use of model suffix. For example,

```
mymodel.workspace = 10;
```

sets the amount of memory used to 10 MB. `Mymodel` is the name of the model as specified by the `model` statement. In order to be effective, the assignment of the model suffix should be made between the `model` and `solve` statements.

Option	Description
workspace	Gives OSL x MB of workspace. Overrides the memory estimation.
optfile	Instructs OSL to read the option file *osl.opt*.
cheat	Cheat value: each new integer solution must be at least x better than the previous one. Can speed up the search, but you may miss the optimal solution. The cheat parameter is specified in absolute terms (like the OPTCA option). The OSL option improve overrides the GAMS cheat parameter.
cutoff	Cutoff value. When the Branch and Bound search starts, the parts of the tree with an objective worse than x are deleted. Can speed up the initial phase in the Branch and Bound.
prioropt	Instructs OSL to use priority branching information passed by GAMS through the variable.prior parameters.

5 Summary of OSL Options

All these options should be entered in the option file *osl.opt* after setting the mymodel.optfile parameter to 1. The option file is not case sensitive and the keywords must be given in full. Examples for using the option file can be found at the end of this section.

5.1 LP Algorithmic Options

Option	Description
crash	crash an initial basis
iterlim	iteration limit
method	solution method
pdgaptol	barrier method primal-dual gap tolerance
presolve	perform presolve. Reduce size of model.
reslim	resource limit
simopt	simplex option
scale	perform scaling
toldinf	dual feasibility tolerance
tolpinf	primal feasibility tolerance
workspace	memory allocation

5.2 MIP Algorithmic Options

Option	Description
bbpreproc	branch and bound preprocessing
cutoff	cutoff or incumbent value
cuts	allocate space for extra cuts
degscale	scale factor for degradation
improve	required level of improvement over current best integer solution
incore	keep tree in core
iweight	weight for each integer infeasibility
maxnodes	maximum number of nodes allowed
maxsols	maximum number of integer solutions allowed
optca	absolute optimality criterion
optcr	relative optimality criterion

Option	Description
strategy	MIP strategy
target	target value for objective
threshold	supernode processing threshold
tolint	integer tolerance

5.3 Screen and Output File Options

Option	Description
bastype	format of basis file
creatbas	create basis file
creatmps	create MPS file
iterlog	iteration log
mpstype	type of MPS file
networklog	network log
nodelog	MIP log

5.4 Advanced LP Options

Option	Description
adjac	save storage on AA^T
chabstol	absolute pivot tolerance for Cholesky factorization
chweight	rate of change of multiplier in composite objective function
chtinytol	cut-off tolerance in Cholesky factorization
crossover	when to switch to simplex
densecol	dense column threshold
densethr	density threshold in Cholesky
devex	devex pricing method
droprowct	constraint dropping threshold
dweight	proportion of feasible objective used in dual infeasible solution
factor	refactorization frequency
fastits	switching frequency to devex
fixvar1	tolerance for fixing variables in barrier method when infeasible
fixvar2	tolerance for fixing variables in barrier method when feasible
formntype	formation of normal matrix
maxprojns	maximum number of null space projections
muinit	initial value for μ
mulimit	lower limit for μ
mulinfac	multiple of μ to add to the linear objective
mufactor	reduction factor for μ in primal barrier method
nullcheck	null space checking switch
objweight	weight given to true objective function in phase I of primal barrier
pdstepmult	step-length multiplier for primal-dual barrier
pertdiag	diagonal perturbation in Cholesky factorization
possbasis	potential basis flag
postsolve	basis solution required
projtol	projection error tolerance starting value for multiplier in composite objective function
pweight	
rgfactor	reduced gradient target reduction
rglimit	reduced gradient limit
simopt	simplex option
stepmult	step-length multiplier for primal barrier algorithm

5.5 Examples of GAMS/OSL Option File

The following option file *osl.opt* may be used to force OSL to perform branch and bound preprocessing, a maximum of 4 integer solutions and to provide a log of the branch and bound search at every node.

```
bbpreproc 1
maxsols 4
nodelog 1
```

6 Detailed Description of OSL Options

Option	Description	Default
adjac	Generation of AA^T 0: Save storage on forming AA^T in the interior point methods. 1: Use a fast way of computing AA^T.	1
bastype	Format of basis file 0: do not write level values to the basis file 1: write level values to the basis file	0
bbpreproc	Preprocess the Branch and Bound tree. 0: Do not preprocess the 0-1 structure. 1: Use super-nodes, copy matrix 2: Regular branch-and-bound, copy matrix 3: Use super-nodes, overwrite matrix 4: Regular branch-and-bound, overwrite matrix The pre-processor examines only the 0-1 structure. On models with only general integer1 variables (i.e. integer variables with other bounds than 0 and 1) the preprocessor will not do any good. A message is written if this happens. The preprocessor needs space for extra cuts. If no space available, the branch-and-bound search may fail. Use the cuts option to specify how much extra room has to be allocated for additional cuts. Notice that the presolve already may reduce the size of the model, and so create extra space for additional cuts.	0
chabstol	Absolute pivot tolerance for the Cholesky factorization. Range - [1.0e-30, 1.0e-6]	1.0e-15
chtinytol	Cut-off tolerance in the Cholesky factorization. Range - [1.0e-30, 1.0e-6]	1.0e-18
chweight	Rate of change for multiplier in composite objective function. Range - [1e-12,1] This value determines the rate of change for pweight or dweight. It is a nonlinear factor based on case-dependent heuristics. The default of 0.5 gives a reasonable change if progress towards feasibility is slow. A value of 1.0 would give a greater change, while 0.1 would give a smaller change, and 0.01 would give very slow change.	0.5
Crash	Crash an initial basis. This option should not be used with a network or interior point solver. The option is ignored if GAMS provides a basis. To tell GAMS never to provide the solver with a basis use 'option bratio = 1;' in the GAMS program. 0: No crash is performed 1: Dual feasibility may not be maintained 2: Dual feasibility is maintained 3: The sum of infeasibilities is not allowed to increase 4: Dual feasibility is maintained and the sum of infeasibilities is not allowed to increase The options to maintain dual feasibility do not have much impact due to the way GAMS sets up the model. Normally, only options 0 or 1 need be used.	1, if no basis provided by GAMS

Option	Description	Default
creatbas	Create a basis file in MPS style. String is the file name. Can only be used if GAMS passes on a basis (i.e. not the first solve, and if BRATIO test is passed). The basis is written just after reading in the problem and after the basis is setup. The option bastype determines whether values are written to the basis file. **On UNIX precede the file name by ../ due to a bug in OSL (see Section 7.3).**	None
creatmps	This option can be used to create an MPS file of the GAMS model. This can be useful to test out other solvers or to pass on problems to the developers. String is file name. Its should not be longer than 30 characters. Example: creatmps trnsport.mps. The option mpstype determines which type (1 or 2 nonzeroes per line) is being used. The MPS file is written just after reading in the problem, before scaling and presolve. On **UNIX precede the file name by ../ due to a bug in OSL (see Section 7.3)** . Examples of the MPS files generated by OSL are in the Appendix.	None
crossover	Switching to simplex from the Interior Point method. Use of this option can be expensive. It should be used if there is a need to restart from the optimal solution in the next solve statement. 0: Interior point method does not switch to Simplex 1: Interior point method switches to Simplex when numerical problems arise. 2: Interior point method switches to Simplex at completion or when in trouble. 3: As in 2, but also if after analyzing the matrix it seems the model is more appropriate for the Simplex method. 4: Interior point immediately creates a basis and switches over to Simplex.	1
cutoff	Cutoff or incumbent value. Overrides the CUTOFF in GAMS.	None
cuts	Allocate space for extra cuts generated by Branch and Bound preprocessor.	10 + m/10, where m is the number of rows
degscale	The scale factor for all degradation. Range - [0.0, maxreal]	1.0
densecol	Dense column threshold. Columns with more non- zeroes than DENSECOL are treated differently. Range - [10, maxint]	99,999
densethr	Density threshold in Cholesky. If DENSETHR $\leq$ 0 then everything is considered dense, if DENSETHR $\geq$ 1 then everything is considered sparse. Range - [-maxreal, maxreal]	0.1
devex	Devex pricing 0: Switch off Devex pricing (not recommended for normal use). 1: Approximate Devex method. 2: Primal: Exact Devex method, Dual: Steepest Edge using inaccurate initial norms. 3: Exact Steepest Edge method. -1,-2,-3: As 1,2,3 for the dual. For the primal the default is used until feasible, and then mode 1,2 or 3. Devex pricing is only used by the simplex method. Devex pricing is used after the first "cheap" iterations which use a random pricing strategy. In the primal case a negative value tells OSL to use the non-default devex pricing strategy only in phase II (in phase I the default devex strategy is used then). For the primal 1 (the default) or -2 are usually good settings, for the dual 3 is often a good choice.	1
Droprowct	The constraint dropping threshold. Range - [1,30]	1
dweight	Proportion of the feasible objective that is used when the solution is dual infeasible. Range - [0.0,1.0]	0.1
factor	Refactorization frequency. A factorization of the basis matrix will occur at least each factor iterations. OSL may decide to factorize earlier based on some heuristics (loss of precision, space considerations) Range - [0,999]	100+m/100, where m is the number of rows

Option	Description	Default
fastits	When positive, OSL switches to Devex after fastits random iterations, (to be precise at the first refactorization after that) but before it will price out a random subset, with correct reduced costs. When negative, OSL takes a subset of the columns at each refactorization, and uses this as a working set. Range - [-maxint,+maxint]. By default OSL primal simplex initially does cheap iterations using a random pricing strategy, then switches to Devex either because the success rate of the random pricing becomes low, or because the storage requirements are becoming high. This option allows you to change this pricing change. This option is only used when simplex was used with simopt 2 which is the default for models solved from scratch. For more information see the OSL manual.	0
fixvar1	Tolerance for fixing variables in the barrier method if the problem is still infeasible. Range - [0.0, 1.0e-3]	1.0e-7
fixvar2	Tolerance for fixing variables when the problem is feasible. Range - [0.0, 1.0e-3]	1.0e-8
formntype	Formation of normal matrix 0: Do formation of normal matrix in the interior point methods in a way that exploits vectorization. Uses lots of memory, but will switch to option 2 if needed. 1: Save memory in formation of normal matrix.	0
Improve	The next integer solution should be at least improve better than the current one. Overrides the cheat parameter in GAMS.	None
Incore	Keep tree in core 0: The Branch & Bound routine will save tree information on the disk. This is needed for larger and more difficult models. 1: The tree is kept in core. This is faster, but you may run out of memory. For larger models use incore 0, or use the workspace model suffix to request lots of memory. GAMS/OSL can not recover if you are running out of memory, i.e. we cannot save the tree to disk, and go on with incore 0.	0
Iterlim	Iteration limit. Overrides the GAMS iterlim option. Notice that the interior point codes require much less iterations than the simplex method. Most LP models can be solved with the interior point method with 50 iterations. MIP models may often need much more than 1000 LP iterations.	1000
iterlog	LP iteration log frequency. How many iterations are performed before a new line to the log file (normally the screen) is written. The log shows the iteration number, the primal infeasibility, and the objective function. The same log frequency is passed on to OSL, so in case you have option sysout=on in the GAMS source, you will see an OSL log in the listing file with the same granularity. The resource usage is checked only when an iteration log is written. In case you set this option parameter to a very large value, a resource limit overflow will not be detected.	20
iweight	Weight for each integer infeasibility. Range [0.0, maxreal]	1.0
maxnodes	Maximum number of nodes allowed	9,999,999
maxprojns	Maximum number of null space projections. Range - [1,10]	3
maxsols	Maximum number of integer solution allowed.	9,999,999

Option	Description	Default
method	Solution Method psimplex: Uses primal simplex method as LP solver dsimplex: Uses dual simplex simplex: Uses either the primal or dual simplex based on model characteristics network: Uses the network solver interior0: Uses an appropriate barrier method interior1: Uses the primal barrier method interior2: Uses the primal-dual barrier method interior3: Uses the primal-dual predictor-corrector barrier method. In case any of the simplex algorithms is used the listing file will list the algorithm chosen by OSL (this was not possible for interior0). the best interior point algorithm overall seems to be the primal-dual with predictor-corrector. Note that if an interior point method has been selected for a MIP problem, only the relaxed LP is solved by the interior point method.	psimplex
Mpstype	Format of MPS file 1: one nonzero per line in the COLUMN section 2: two nonzeroes per line in the COLUMN section.	2
Mufactor	The reduction factor for μ in the primal barrier method.Range - [1.0e-6,0.99999]	0.1
muinit	Initial value for μ in primal barrier method. Range - [1.0e-20, 1.0e6]	0.1
mulimit	Lower limit for μ. Range - [1.0e-6,1.0]	1.0e-8
mulinfac	Multiple of μ to add to the linear objective. Range - [0.0, 1.0]	0.0
networklog	Iteration log frequency for the network solver. See iterlog.	100
nodelog	MIP node log frequency. Node log is written to the screen when a new integer is found or after nodelog nodes. OSL writes a log line each node. This is captured in the status file, and displayed in the listing file when you have option sysout=on; in the GAMS source. The status file can become huge when many nodes are being examined.	20
Nullcheck	Null space checking switch. See the OSL manual.	None
Objweight	Weight given to true objective function in phase 1 of the primal barrier method. Range - [0.0, 1.0e8]	0.1
optca	Absolute optimality criterion. The definition of this criterion is the same as described in the GAMS manual.	0.0
optcr	Relative optimality criterion. The definition of this criterion is the same as described in the GAMS manual.	0.10
pdgaptol	Barrier method primal-dual gap tolerance. Range - [1.0e-12, 1.0e-1]	1.0e-7
pdstepmult	Step-length multiplier for primal-dual barrier. Range - [0.01, 0.99999]	0.99995
pertdiag	Diagonal perturbation in the Cholesky factorization. Range - [0.0, 1.0e-6]	1.0e-12
possbasis	Potential basis flag. If greater than zero, variables that are remote from their bounds are marked potentially basic. Range - [0,maxint]	1
postsolve	Basis solution required. If the presolver was not used this option is ignored. 0: No basis solution is required. The reported solution will be optimal but it will not be a basis solution. This will require less work than being forced to find a basic optimal solution. 1: Basis solution is required. After the postsolve, the Simplex method is called again to make sure that the final optimal solution is also a basic solution	1

Option	Description	Default
presolve	Perform model reduction before starting optimization procedure. This should not be with network solver. If by accident, all the constraints can be removed from the model, OSL will not be able to solve it and will abort. presolve can sometimes detect infeasibilities which can cause it to fail, in which case the normal solver algorithm is called for the full problem. -1: Do not use presolve 0: Remove redundant rows, the variables summing to zero are fixed. If just one variable in a row is not fixed, the row is removed and appropriate bounds are imposed on that variable. 1: As 0, and doubleton rows are eliminated (rows of the form $px_j + qx_k = b$). 2: As 0, and rows of the form $x_1 = x_2 + x_3 ..., x > 0$ are eliminated. 3: All of the above are performed. The listing file will report how successful the presolve was. The presolver writes information needed to restore the original to a disk file, which is located in the GAMS scratch directory. The postsolve routine will read this file after the smaller problem is solved, and then simplex is called again to calculate an optimal basis solution of the original model. If no basis solution is required use the option postsolve 0.	0: for cold starts, -1: for restarts
Projtol	Projection error tolerance. Range - [0.0, 1.0]	1.0e-6
pweight	Starting value of the multiplier in composite objective function. Range: [1e-12,1e10]. OSL uses in phase I when the model is still infeasible a composite objective function of the form *phase_I_objective + pweight * phase_I_objective* . The phase I objective has a +1 or -1 where the basic variables exceed their lower or upper bounds. This gives already a little bit weight to the phase II objective.pweight starts with 0.1 (or whatever you specify as starting value) and is decreased continuously. It is decreased by a large amount if the infeasibilities increase and by small amounts if progress to feasibility is slow.	0.1
Reslim	Resource limit. Overrides the GAMS reslim option.	1000s
rgfactor	Reduced gradient target reduction factor. Range - [1.0e-6,0.999999]	0.1
rglimit	Reduced gradient limit. Range - [0.0, 1.0]	0.0
scale	Scaling done on the model. Scaling cannot be used for network models. It is advised to use scaling when using the primal barrier method, the other barrier methods (primal-dual and primal- dual predictor-corrector) in general should not be used with scaling. 0: Do not scale the model 1: Scale the model	1, except for methods mentioned above
simopt	Simplex option 0: Use an existing basis 1: Devex pricing is used 2: Random pricing is used for the first "fast" iterations, then a switch is made to Devex pricing 3: Use previous values to reconstruct a basis. The dual simplex method can only have options 0 or 1.	2: if no basis provided by GAMS and crashing off, 0: otherwise
stepmult	Step-length multiplier for primal barrier algorithm. Range - [0.01, 0.99999]	0.99

Option	Description	Default
strategy	MIP strategy for deciding branching. 1: Perform probing only on satisfied 0-1 variables. This is the default setting in OSL. When a 0-1 variable is satisfied, OSL will do probing to determine what other variables can be fixed as a result. If this bit is not set, OSL will perform probing on all 0-1 variables. If they are still fractional, OSL will try setting them both ways and use probing to build an implication list for each direction. 2: Use solution strategies that assume a valid integer solution has been found. OSL uses different strategies when looking for the first integer solution than when looking for a better one. If you already have a solution from a previous run and have set a cutoff value, this option will cause OSL to operate as though it already has an integer solution. This is beneficial for restarting and should reduce the time necessary to reach the optimal integer solution. 4: Take the branch opposite the maximum pseudo-cost. Normally OSL will branch on the node whose smaller pseudo-cost is highest. This has the effect of choosing a node where both branches cause significant degradation in the objective function, probably allowing the tree to be pruned earlier. With this option, OSL will branch on the node whose larger pseudo-cost is highest. The branch taken will be in the opposite direction of this cost. This has the effect of forcing the most nearly integer values to integers earlier and may be useful when any integer solution is desired, even if not optimal. Here the tree could potentially grow much larger, but if the search is successful and any integer solution is adequate, then most of it will never have to be explored. 8: Compute new pseudo-costs as variables are branched on. Pseudo-costs are normally left as is during the solution process. Setting this option will cause OSL to make new estimates, using heuristics, as each branch is selected. 16: Compute new pseudo-costs for unsatisfied variables. Pseudo-costs are normally left as is during the solution process. Setting this option will cause OSL to make new estimates, using heuristics, for any unsatisfied variables' pseudo-costs in both directions. This is done only the first time the variable is found to be unsatisfied. In some cases, variables will be fixed to a bound by this process, leading to better performance in the branch and bound routine. This work is equivalent to making two branches for every variable investigated. 32: Compute pseudo-costs for satisfied variables as well as unsatisfied variables. Here, if 16 is also on, OSL will compute new estimated pseudo-costs for the unsatisfied as well as the unsatisfied ones. Again, this is very expensive, but can improve performance on some problems. strategy can be a combination of the above options by adding up the values for the individual options. 48 for instance will select 16 and 32.	None
Target	Target value for the objective.	5% worse than the relaxed solution
threshold	Supernode processing threshold. Range [0.0, maxreal]	0
tolpinf	Primal infeasibility tolerance. Row and column levels less than this values outside their bounds are still considered feasible. Range: [1e-12,1e-1].	1e-8
toldinf	Dual infeasibility tolerance. Functions as optimality tolerance for primal simplex. Range: [1e-12,1e-1].	1e-7
tolint	Integer tolerance Range - [1.0e-12, 1.0e-1]	1.0e-6
workspace	Memory allocation. Overrides the memory estimation and the workspace model suffix. Workspace is defined in Megabytes.	None

7 Special Notes

This section covers some special topics of interest to users of OSL.

7.1 Cuts Generated During Branch and Bound

The OSL documentation does not give an estimate for the number of cuts that can be generated when the `bbpreproc` is used. By default we now allow #rows/10 extra rows to be added. Use the `cuts` option to change this. Also we were not able to find out how many cuts OSL really adds, so we can not report this. You will see a message on the screen and in the listing file when OSL runs out of space to add cuts. When this happens, I have seen the branch & bound fail later on with the message: OSL *data was overwritten*, so make sure your `cuts` option is large enough. For this reason we have as a default: no preprocessing.

Note that when `cuts` is 0 and `presolve` is turned on, there may still be enough room for the preprocessor to add cuts, because the presolve reduced the number of rows.

7.2 Presolve Procedure Removing All Constraints from the Model

The PRESOLVE can occasionally remove all constraints from a model. This is the case for instance for the first solve in [WESTMIP]. An artificial model with the same behavior is shown below. OSL will not recover from this. Turn off the presolve procedure by using the option `presolve -1` to prevent this from happening. An appropriate message is written when this happens.

```
Variable x ;
Equation e ;
e..  x =e= 1 ;
model m /all/ ;
option lp = osl ;
solve m using lp minimizing x ;
```

7.3 Deleting the Tree Files

The MIP solver EKKMSLV uses two files to store the tree. Due to a bug in OSL we are unable to store these files in the GAMS scratch directory (the open statement we issue is ignored by OSL). Therefore after you solved a MIP with OSL with INCORE 0 (the default), two files *fort.82* and *fort.83* would be in the current directory. The shell script *gamsosl.run* will try to overcome this by doing a **cd** to the scratch directory. If this fails, you will see an error message on the screen.

An undesirable side effect of this is that all options that relate to user specified files have to be preceded by ../ to have the file going to the directory where the user started GAMS from. If you do not do this, the file will go to the current directory, which is the scratch directory, and the file will be removed when GAMS terminates. The affected commands are `creatmps` and `creatmbas`. So if you want to write an MPS file with the name *myfile.mps*, use the option `creatmps ../myfile.mps`.

8 The GAMS/OSL Log File

The iteration log file that normally appears on the screen has the following appearance on the screen for LP problems:

```
Reading data...
 Starting OSL...
 Scale...
 Presolve...
 Crashing...
 Primal Simplex...
    Iter             Objective   Sum Infeasibilities
```

```
      20        2155310.000000          48470.762716
      40        1845110.000000          37910.387877
      60        1553010.000000          26711.895409
      80         191410.000000                   0.0
     100         280780.000000                   0.0
     120         294070.000000                   0.0
 Postsolve...
 Primal Simplex...
     121         294070.000000     Normal Completion
                                   Optimal
```

For MIP problems, similar information is provided for the relaxed problem, and in addition the branch and bound information is provided at regular intervals. The screen log has the following appearance:

```
Reading data...
 Starting OSL...
 Scale...
 Presolve...
 Crashing...
 Primal Simplex...
    Iter             Objective   Sum Infeasibilities
      20             19.000000              8.500000
      40              9.500000              6.250000
 **** Not much progress: perturbing the problem
      60              3.588470              3.802830
      80              0.500000              2.000000
     100              2.662888              0.166163
     116              0.000000     Relaxed Objective
Branch\& Bound...
    Iter  Nodes  Rel Gap       Abs Gap    Best Found
     276     19   n/a           6.0000        6.0000 New
     477     39   n/a           6.0000        6.0000
     700     59   n/a           6.0000        6.0000
     901     79   n/a           6.0000        6.0000
    1119     99  65.0000        5.9091        6.0000
    1309    119  65.0000        5.9091        6.0000
    1538    139  17.0000        5.6667        6.0000
    1701    159  17.0000        5.6667        6.0000
    1866    179  17.0000        5.6667        6.0000
    2034    199  17.0000        5.6667        6.0000
    2158    219  11.0000        5.5000        6.0000
    2362    239  11.0000        5.5000        6.0000
    2530    259   8.0000        5.3333        6.0000
    2661    275   5.0000        3.3333        4.0000 Done
Postsolve...
 Fixing integer variables...
 Primal Simplex...
    2661              4.000000     Normal Completion
                                   Integer Solution
The solution satisfies the termination tolerances
```

The branch and bound information consists of the number of iterations, the number of nodes, the current relative gap, the current absolute gap and the current best integer solution.

9 Examples of MPS Files Written by GAMS/OSL

This appendix shows the different output formats for MPS and basis files. We will not explain the MPS format or the format of the basis file: we will merely illustrate the function of the options `mpstype` and `bastype`. Running [TRNSPORT] with the following option file

```
mpsfile trnsport.mps
```

will result in the following MPS file:

```
NAME          GAMS/OSL
ROWS
 N  OBJECTRW
 E  R0000001
 L  R0000002
 L  R0000003
 G  R0000004
 G  R0000005
 G  R0000006
COLUMNS
    C0000001  R0000001     -0.225000   R0000002      1.000000
    C0000001  R0000004      1.000000
    C0000002  R0000001     -0.153000   R0000002      1.000000
    C0000002  R0000005      1.000000
    C0000003  R0000001     -0.162000   R0000002      1.000000
    C0000003  R0000006      1.000000
    C0000004  R0000001     -0.225000   R0000003      1.000000
    C0000004  R0000004      1.000000
    C0000005  R0000001     -0.162000   R0000003      1.000000
    C0000005  R0000005      1.000000
    C0000006  R0000001     -0.126000   R0000003      1.000000
    C0000006  R0000006      1.000000
    C0000007  OBJECTRW      1.000000   R0000001      1.000000
RHS
    RHS1      R0000002    350.000000   R0000003    600.000000
    RHS1      R0000004    325.000000   R0000005    300.000000
    RHS1      R0000006    275.000000
BOUNDS
 FR BOUND1    C0000007      0.000000
ENDATA
```

MPS names have to be 8 characters or less. GAMS names can be much longer, for instance: X(”Seattle”,”New-York”). We don’t try to make the names recognizable, but just give them labels like R0000001 etc. Setting the option `mpstype 1` gives:

```
NAME          GAMS/OSL
ROWS
 N  OBJECTRW
 E  R0000001
 L  R0000002
 L  R0000003
 G  R0000004
 G  R0000005
 G  R0000006
COLUMNS
    C0000001  R0000001     -0.225000
    C0000001  R0000002      1.000000
    C0000001  R0000004      1.000000
```

```
    C0000002  R0000001     -0.153000
    C0000002  R0000002      1.000000
    C0000002  R0000005      1.000000
    C0000003  R0000001     -0.162000
    C0000003  R0000002      1.000000
    C0000003  R0000006      1.000000
    C0000004  R0000001     -0.225000
    C0000004  R0000003      1.000000
    C0000004  R0000004      1.000000
    C0000005  R0000001     -0.162000
    C0000005  R0000003      1.000000
    C0000005  R0000005      1.000000
    C0000006  R0000001     -0.126000
    C0000006  R0000003      1.000000
    C0000006  R0000006      1.000000
    C0000007  OBJECTRW      1.000000
    C0000007  R0000001      1.000000
RHS
    RHS1      R0000002    350.000000
    RHS1      R0000003    600.000000
    RHS1      R0000004    325.000000
    RHS1      R0000005    300.000000
    RHS1      R0000006    275.000000
BOUNDS
 FR BOUND1    C0000007      0.000000
ENDATA
```

To illustrate the creation of a basis file, we first solve the transport model as usual, but we save work files so we can restart the job:

```
gams trnsport save=t
```

Then we create a new file called *t2.gms* with the following content:

```
transport.optfile=1;
solve trnsport using lp minimizing z;
```

and we run *gams t2 restart=t* after creating an option file containing the line `creatbas trnsport.bas`. This results in the following basis file being generated.

```
NAME
 XL C0000002  R0000001
 XU C0000004  R0000004
 XU C0000006  R0000005
 XU C0000007  R0000006
ENDATA
```

When we change the option to bastype 1 we get:

```
NAME
 BS R0000002               0.000000
 BS R0000003               0.000000
 LL C0000001               0.000000
 XL C0000002  R0000001     0.000000                 0.000000
 LL C0000003               0.000000
 XU C0000004  R0000004     0.000000               325.000000
```

```
 LL C0000005                0.000000
 XU C0000006  R0000005      0.000000              300.000000
 XU C0000007  R0000006      0.000000              275.000000
ENDATA
```