

# XA

## Contents

---

<b>1</b>	<b>Introduction</b> . . . . .	<b>1</b>
<b>2</b>	<b>Usage</b> . . . . .	<b>1</b>
<b>3</b>	<b>Memory Usage</b> . . . . .	<b>1</b>
<b>4</b>	<b>Semi-Continuous &amp; Semi-Integer Variables</b> . . . . .	<b>2</b>
<b>5</b>	<b>Branch &amp; Bound Topics</b> . . . . .	<b>2</b>
5.1	Branching Priorities . . . . .	3
5.2	Branching Strategies . . . . .	3
5.3	Limitsearch Parameter . . . . .	4
<b>6</b>	<b>The XA Option File</b> . . . . .	<b>5</b>
<b>7</b>	<b>Iteration Log Formats</b> . . . . .	<b>8</b>

---

## 1 Introduction

This document describes the GAMS/XA linear and mixed-integer programming solver. The GAMS/XA solver (here also simply referred to as XA) is based on Sunset Software Technology's XA Callable Library, an implementation of high performance solvers for LP and MIP problems.

XA implements primal simplex, dual simplex, and barrier algorithms for solving linear problems. The primal/dual simplex method is very robust, and in most cases you should get good performance, especially from a warm start. The barrier method is particularly efficient on large models. Both algorithms benefit from XA's presolver, which reduces the size of the model by removing redundant constraints, substituting constraints, etc.

In most cases, GAMS/XA should perform satisfactorily without using any options. However, if fine-tuning is necessary or desired, XA provides many options and parameters designed for this purpose. These options are accessible via GAMS option statements or via an XA-specific option file.

## 2 Usage

If you have installed the GAMS system and configured XA as the default LP, RMIP and MIP solver, all LP, RMIP and MIP models without a specific solver option will use XA. If you installed another solver as the default, you can explicitly request that a particular model be solved by XA by inserting the statement

```
option LP = xa; { or MIP or RMIP }
```

somewhere before the `solve` statement.

## 3 Memory Usage

By default, the GAMS/XA link computes an estimate of the amount of memory that will be required by the solver, and passes this on to the solver. The solver makes an allocation of this amount and then uses this memory

during the course of program execution. Usually, this will be sufficient to solve the problem successfully. In some cases, though, the computed estimate will be too small, and GAMS/XA will indicate that a larger memory estimate is required. You will need to manually specify a larger memory estimate to solve the model.

A model-specified memory estimate can be made by adding the following line to your GAMS model before the solve statement:

```
<modelname>.workspace = xx;
```

where `xx` is the amount of memory in Mbytes. You can also define the environment variable `XAMEMORY` to be the amount of memory to use, in Mbytes. The computed memory estimate is the default, and is used only if no manual estimate is specified. The model-specified workspace limit overrides the computed estimate, and the `XAMEMORY` environment variable takes precedence over both of these.

In an attempt to insure that all models solve without running out of memory, XA makes one final memory check and if the user supplied memory amount is below what XA would consider reasonable for that size of problem, XA will then increase your amount to XA's minimal value.

On multi-processor machines, XA will automatically detect and use all available processors (CPU's) when solving MIP models. The memory estimate computed adds 50% more memory per processor to take full advantage of these processors, but this is sometimes not enough memory for XA to multi-process. In this case, a larger estimate must be specified manually.

## 4 Semi-Continuous & Semi-Integer Variables

XA supports semi-continuous and semi-integer variable types. Semi-continuous variables are variables that are either at zero or greater than or equal to their lower bound. E.g. a pump motor if operating must run between 2400 and 5400 r.p.m., but it may be switched off as well. Investment levels must exceed a specific threshold or no investment is made.

All semi-continuous variables must have a lower bound specification, e.g., `speed.lo(i) = 100`. Semi-integer variables must have an upper bound as well.

Prior to the introduction of these variable types, semi-continuous variables had to be emulated by adding one additional binary variable and one additional constraint for each semi-continuous variable. For models of any size, this approach very quickly increased the model's size beyond solvability. Now XA has implicitly defined these variables without requiring the addition of new variables and constraints to your model. This effectively increases the size of model that can be solved and does it in a very neat and clean way besides.

For example, to define variables 'a' and 'b' as semi-continuous enter:

```
SemiCont a , b ;
```

or to define semi-integer variables -

```
SemiInt y1 , y2 ;
```

Priority values (`.prior` suffix) can be associated with both semi-continuous and semi-integer variables. All the integer solving options are available for models with semi-continuous and semi-integer variables as well. For example, you can select solving strategies, `optcr` and `optca` values, etc.

The solve time complexity for semi-continuous variables is comparable with the solve times for binary models, while the semi-integer case compares to integer.

## 5 Branch & Bound Topics

XA is designed to solve a vast majority of LP problems using the default settings. In the integer case, however, the default settings may not result in optimal speed and reliability. By experimenting with the control parameters performance can be improved (or worsened!) dramatically.

## 5.1 Branching Priorities

Using priorities can significantly reduce the amount of time required to obtain a good integer solution. If your model has a natural order in time, or in space, or in any other dimension then you should consider using priority branching. For example, multi-period production problem with inventory would use the period value as the priority setting for all variable active in that period, or a layered chip manufacturing process where the priority assigned to binary variables is top down or bottom up in that layer.

If priorities are given to binary, integer, or semi-continuous variables, then these are used to provide a user-specified order in which variables are branched. XA selects the variable with the highest priority (lowest numerical value) for branching and the Strategy determines the direction, up or down.

Priorities are assigned to variables using the `.prior` suffix. For example:

```
NAVY.PRIOROPT = 1 ;
...
Z.PRIOR(J,"SMALL") = 10 ;
Z.PRIOR(J,"MEDIUM") = 5 ;
Z.PRIOR(J,"LARGE" ) = 1 ;
```

The value 1 indicates the highest priority (branch first), and the value 10 the lowest priority (branch last). Valid priority values should range between -32000 and 32000. The default priority value is 16000.

## 5.2 Branching Strategies

Ten branch & bound strategies are provided to meet the demands of many different types of problems. Each strategy has five variations (six if you include the basic strategy, or “no variation”) that affect the solve time, speed to first solution, and the search for the best integer solution. The order in which integer variables are processed during the search is important. This order is called the *branching order* of integer variables. Solution times can vary significantly with the method selected.

In general, XA will solve your MIP problems much faster when all model variables are given some kind of objective function value. This biases the basis in a specific direction and usually leads to satisfactory first integer solutions.

The strategy used can be changed by setting the “strategy” option in an XA options file.

Branch & Bound Strategy	Description of Selection Criteria
1	Proprietary method. Default value. Excellent strategy, also add priority to integer variable and try 1P for additional performance gains.
2	Minimum change in the objective function. This strategy has not been very successful at solving MIP problems.
3	Priority based upon column order. This strategy probably does not have much meaning because you typically do not set the column order in GAMS .
4	Column closest to its integer bound. This strategy tends to send a variable to its lower bounds.
6	Column always branches up (high). Second choice after 1. Excellent choice when your model is a multi-period problem; additional performance gains when priority value are equated with period number; also try 6P if using priorities.
7	Column always branches down (low). Useful if variable branched down doesn't limit capacity or resources. One suggestion is to use priorities in the reverse order from that described in Strategy 6.
8	Column farthest from its integer bound. Next to Strategies 1 and 6 this is probably the next choice to try. Using priorities and variation P is also suggested.
9	Column randomly selected, useful when solving very large problems. Priority values helpful in multi-period models.
10	Apparent smoothest sides on the polytope. Priorities helpful.

Each XA B&B strategy has many variations. Sometimes these variations reduce the solution time but may not

yield the optimal integer solution. If you are interested in obtaining a fast and 'good' integer solution (which may not be the optimal integer solution), try these variations. You should be aware, though, that using these variations will invalidate the best bound and optimality gap statistics printed by the link at the end of the solve. To choose a variation, either append its letter to the strategy number or add its offset to the strategy number. For example, to choose variations B and P of strategy 6, you could either set “`strategy 6BP`” or “`strategy 1806`”.

Variation	Effect
A (+100)	This variation reduces the amount of time XA spends estimating the value of a potential integer solution. The values calculated are rough estimates and may eliminate nodes that would lead to better integer solutions. Variation A may not appear with variation B.
B	This variation spends very little time calculating estimated integer solutions at each node and is the most radical in performance and integer solution value and may eliminate nodes that would lead to better integer solutions. Variation B may not appear with variation A.
C	Each time an improving integer solution is found XA splits the remaining node list in half based upon the length of the current list. This technique allows XA to search nodes that might not normally be explored. The reported integer solution value may not be the optimal integer solution because nodes may be eliminated that would lead to this solutions. Variation C may not appear with variation D.
D	Each time an improving integer solution is found XA splits the remaining node list based upon the difference in current projected objective and the best possible objective value divided by two. This technique allows XA to search nodes that might not normally be explored. The reported integer solution value may not be the optimal integer solution because nodes may be eliminated that would lead to this solutions. Variation D may not appear with variation C.
P	Each time a node is generated XA calculates the effects of each non-integer on future objective function values, which is calculation intensive. By assigning branching priorities to your integer variables XA will only perform this calculation on the non-integer variables with the lowest branching priority. This frequently reduces the number of calculations. Variation P may appear with any variation, but to be effective you must assign integer branching priorities.

If you wish to improve your solution times, you should experiment with different Strategies to determine which is best for your problems. We have found that Strategies 1 and 6 work quite well. Also try strategies 1A, 1B, 6A, 6B, and 9. As you gain experience with these Strategies you will be able to make an informed choice.

### 5.3 Limitsearch Parameter

LIMITSEARCH is used to limit the number of nodes to search by implicitly or explicitly stating a bound on the value of an integer solution. The integer solution obtained, if any, will have a functional value no worse than LIMITSEARCH. The next integer solution will have a monotonically improving objective function value until an optimal integer solution is found and if verified.

If you can estimate the objective function value of a good integer solution, you can avoid nodes that lead to worse solutions and, consequently, speed up the search. However, too restrictive a value may lead to no integer solution at all, if an integer solution with an objective value better that the LIMITSEARCH value does not exist. If the search terminates with 'NO INTEGER SOLUTION', you must begin the search again with a less restrictive LIMITSEARCH value. The LIMITSEARCH command line parameter has three methods of specifying a lower limit on the objective function.

LIMITSEARCH Value	Meaning
##	Only search for integer solutions between this value and the 'optimal continuous' solution.
##% (#%)	Only search for integer solutions with #% of the 'optimal continuous' solution. Solve for the integer solution that is within #% of the 'optimal integer solution'. This can reduce the search time significantly, but the reported integer solution may not be the optimal integer solution: it will only be within #% of it. This is similar to the GAMS <code>optcr</code> option, but setting <code>optcr</code> reports the actual gap: this is the recommended way to run GAMS/XA.

## 6 The XA Option File

The option file is called `xa.opt`. The GAMS model should contain the following line to signal GAMS/XA to use the option file:

```
<modelname>.optfile = 1 ;
```

where `<modelname>` is the name of the model specified in the model statement. For instance:

```
model m /all/ ;
m.optfile = 1 ;
option LP = XA ;
solve m using LP minimize z ;
```

The XA option file allows you to solver-specific options that are not anticipated by GAMS. Where an XA option and a GAMS option both set the same thing, the setting in the XA option file takes precedence. Option file lines beginning with an asterisk `*` are comment lines. For example:

```
* Integer solving strategy.
  Strategy 6P
* Write log information to the screen every 5 seconds.
  Set FreqLog 00:05
* Do NOT scale the problem.
  Set Scale No
```

The contents of the option file are echoed to the screen. If no options file is found where one is expected, a warning is sent to the log file and the solve continues.

Here is a list of available XA options.

Option	Description	Default
BASIS	After XA has solved your problem, the solution is saved for the next time the problem is solved. This can greatly reduce the number of iterations and execution time required. The Dual Simplex algorithm is used when XA detects advance basis restarts. You can instruct XA to not use the Dual Simplex algorithm for restarts as follows, Set DualSimplex No. <code>file.ext</code> : The filename containing an 'advance basis' for restarting. Default file extension is SAV.	none
none:	No 'advance basis' is specified, but the 'final basis' is saved in the problem filename with an extension of SAV.	
never:	No 'advance basis' is specified, and the final 'basis' is not saved.	

Option	Description	Default
DUALSIMPLEX	By default, XA uses the dual simplex algorithm to restart from an advanced basis (e.g. on nodes in the B&B tree). If DualSimplex is set to No, it will use primal simplex instead.	Yes
FORCE	If your LP model is infeasible, XA makes adjustments in column and row bounds to make the problem feasible. No adjustments are made to binary columns or RHS values of SOS sets. Depending upon how tightly constrained the problem is, XA may be prevented from making additional adjustment that would lead to an integer solution. No adjustments are made to make a column's lower bound less than zero.	No
LIMITSEARCH	See Section 5.3.	
MATLIST	Problem is displayed in equation format. This is probably the most useful command when debugging the model. The GAMS equation and variable listings perform a similar function. Var : List columns in row. Con : List rows in columns. Both: Var and Con None: no listing	None
SET BARRIER	Activates XA's primal-dual interior point algorithm. Useful when solving very large-scale LP models. Yes: Uses primal-dual interior point algorithm, MIP models automatically crossover the simplex algorithm for branching & bounding. No : Use the simplex algorithm. X : Crossover to the simplex algorithm after solving. (automatic when solving MIP models.	
SET BELL	Turns XA's termination bell on or off. No : do not ring the bell Yes: ring the bell	No
SET BVPRIORITY	Sets the default priority of all binary variables to #. By default, all variables have priority 1600, so a value < 1600 causes binary variables to be branched on before general integer variables. A value > 1600 has the opposite affect.	1600
SET CRASH	Method of generating initial basis. 0: Minimize primal infeasibility 1: Minimize dual infeasibility. 2: Both 0 & 1.	0
SET DEGENERITER	Degenerate anticycling aide. Number of consecutive degenerate pivots before anti-cycling code is activated.	square root of the number of rows.
SET FREQLLOG	Frequency in time to print the iteration log line. A negative number (e.g. -00:02) overwrites the same line. This command reduces the overhead of printing too many iteration lines.	00:01 (one log line per second).
SET INTGAP	Minimum objective function improvement between each new integer solutions. Reported integer solution may not be the optimal integer solution because of premature termination.	0.00
SET INTLIMIT	After finding # improving integer solutions, XA terminates with the best solution found thus far. Reported integer solution may not be the optimal integer solution because of premature termination.	no limit on the number of integer solutions.
SET INTPCT	Percentage of available integer columns to consider fixing at each integer node. Useful on very large binary problems. If 100 is entered then all integer columns that are integer at the end of solving the relaxed LP problem are fixed at the current integer bounds.	0.0 - meaning no fixing.

Option	Description	Default
SET IROUND	XA reports either rounded or unrounded integer column primal activity. Yes: causes XA to report rounded integer column activity. XA rounds integer activity values to the closest bound based upon the LTOLERANCE and UTOLERANCE values. No : causes XA to report unrounded integer variable activity, but these activities will always be within the requested integer tolerance.	Yes
SET ITERATION	Maximum number of iteration. XA terminates if limit is exceeded, and if solving an integer problem the best integer solution found thus far is returned. Reported integer solution may not be the optimal integer solution because of premature termination.	200000000
SET LTOLERANCE SET UTOLERANCE	The tolerances XA uses to decide that an integer column's value is integral. For instance, you might consider using a UTOLERANCE of 0.02 (a boat 98% full is for all practical purposes really 100% full). But beware, these integer activities within the specified tolerances are used in calculating constraint relationships and the objective function value. For example, if LTOLERANCE = 0.001, UTOLERANCE = 0.05, and Y has a reported (rounded) activity of 4.0, then $3 * Y$ is in the range $[3 * 3.95, 3 * 4.001]$ .	5.0e-6
SET MARKOWITZ	Numeric Stability vs. sparsity in basis updating and inverse. A larger number favors sparsity at the expense of numeric stability.	10
SET MAXCPU	Number of processors to use when solving MIP models. In general, MIP models should solve # times faster than on a single processor machine. Consider requesting 50% more memory per processor. Defaults to the number of processors on the machine. This number can be greater than the number of physical processors.	
SET MAXNODES	Memory estimate for number of branch and bound nodes. Default value: 4,000 plus the number of binary variables plus square root of the number of integer columns.	
SET NODELIMIT	Maximum number of branch and bound nodes. Default value: unlimited.	
SET PERTURBATE	Indicates the amount of perturbation for highly degenerate problems. A positive value allows XA to generate a uniformly distributive random variable between 0 and #. A negative value uses a constant perturbation of the absolute value of #. Note: This option should not be used except when all else fails. XA has build-in routines to handle degeneracy.	0, indicating no perturbation value
SET PRICING	Variable pricing strategies, useful if XA appears to make a substantial number (rows/2) of pivots that do not move the objective function or reduce the sum of infeasibilities. This feature requires more memory because an additional set of matrix coefficient are loaded into memory. 0: Standard reduced cost pricing 1: Automatic DEVEX pricing switch over. 2: Infeasible DEVEX pricing. 3: Feasible DEVEX pricing (our next choice). 4: Both infeasible and feasible DEVEX pricing.	0
SET REDUCDCOST	Dual activity tolerance to zero.	1e-7
SET REINVERTFREQ	The basis update factors are thrown away and the basis reinverted with this frequency.	40
SET RELAXED	Integer problem is solved as a standard LP and with no integer columns in the formulation. No : integer problems are solved with branch and bound method Yes: solve problems as if all columns were continuous columns	No

Option	Description	Default
SET RESTART	When solving integer, binary, or semi-continuous problems XA may terminate before exploring all your integer nodes. If so you have the option of restarting XA and picking up right where you left off. Just before XA terminates, it writes out basis information in the basis file (extension sav). If an integer, binary or semi-continuous solution has been found an integer basis file is created (extension b01). And if all your integer nodes are not explored then a third file is created for restarting the unchanged problem (extension r01). The BASIS command line parameter determines the filename used with these extensions. <b>Yes:</b> if an r01 file exists with my identical problem then restart integer problem where I left off and if XA terminates before all nodes are explored again then write restart information to this file <b>No :</b> do not use or create the r01 file to restart my problem	No
SET SCALE	Problem Scaling technique. <b>No :</b> Data not scaled. <b>Yes:</b> Column and row scaling. <b>2 :</b> Row scaling only.	Yes
SET STICKWITHIT	If an integer basis (.b01) file is reloaded to indicate branching direction for the current XA solve, this branching advice is following until # infeasible branches are made. After # infeasible branches, the standard branching direction for the particular branch & bound strategy is used.	10
SET TCTOLERANCE	The smallest technological coefficient allowed in your matrix array. This tolerance is useful when extensive calculations are performed on these coefficients, where the results should be zero (but because of rounding errors) ends up being something like 1.0e-15.	1.0e-7
SET TIMELIMIT	Maximum time allowed to solving the problem. XA terminates if this limit is exceeded, and if solving an integer problem the best integer solution found thus far is returned. Units are wall clock time. If set too low, reported integer solutions may not be the optimal integer solution because of premature termination.	200000000 seconds
SET YPIVOT	When selecting a column to leave the basis, columns with absolute marginal values less than ypivot are rejected. Pivoting in columns with very small values can lead to numeric instability and should be avoided when possible. Setting ypivot too large can lead to infeasible pivoting. Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility.	1e-9
SET XTOZERO	Primal activity tolerance to zero	1e-7
STOPAFTER	Amount of time (hh:mm:ss) to continue solving after finding the first integer solution. Reported integer solution may not be the optimal integer solution because of premature termination.	0 (indicates no termina- tion)
STRATEGY	MIP search strategy.	1
TOMPS	Write an MPS file of problem. <i>gams.mps</i> file is created or rewritten. <b>No:</b> Do not write an MPS formatted file (default value). <b>Yes:</b> Write problem in MPS format. <b>Secure:</b> Write problem in MPS format and change column and row names to: C0, C1,... and R0, R1, ...	No

## 7 Iteration Log Formats

The iteration log is something many users watch closely, especially when solving MIP models. Setting MUTE YES or Set FreqLog 0 suppresses the display of the iteration log. During LP iterations, the log format varies depending on the algorithm chosen. Its format is self-explanatory. The default MIP log looks like:

Node	IInf	ToGo.Map	Best.Obj	Cur.Obj	Int.Obj	#	Column	+/-	Iter
####	###	#####	#####	#####	#####	#	#####	#	####
<b>Description</b>									
Node	Active node, the smaller the better, value increases and decreases as the branch-and-bound proceeds.								
IInf	Number of discrete columns having fractional values. This number converges to 0 as XA approaches an integer solution.								
ToGo.Map	A numeric picture of open nodes. The i'th digit (from the right) represents the number of open nodes in the i'th group of ten nodes. For example, 435 means: <ul style="list-style-type: none"> <li>• 4 unexplored nodes between nodes 20 and 29.</li> <li>• 3 unexplored nodes between nodes 10 and 19.</li> <li>• 5 unexplored nodes between nodes 0 and 9.</li> </ul>								
Best.Obj	Best possible integer objective function. As the branch-and-bound algorithm proceeds this number bounds the Optimal Integer Solution. This number does not change very fast.								
Cur.Obj	Objective function for the current node. If an integer solution is found in this node cannot be any better than this value.								
Int.Obj	Objective function of the best integer solution found so far. This value improves as additional integer solutions are found.								
#	Number of improving integer solutions found thus far.								
Column	Column selected by the branch-and-bound process.								
+/-	Branching direction: up(+) or down(-).								
Iter	Cumulative total of simplex iterations used (including the relaxed LP).								

Display of the iteration log line may be toggled on and off by entering a CTRL/U during the iteration process. Use the Set FreqLog command line parameter to minimize the number of lines displayed. Logging each iteration can significantly slow down the solution process.