

USER'S GUIDE FOR TOMLAB /XA V14¹

Kenneth Holmström², Anders O. Göran³ and Marcus M. Edvall⁴

February 26, 2007



¹More information available at the TOMLAB home page: <http://tomopt.com>. E-mail: tomlab@tomopt.com.

²Professor in Optimization, Mälardalen University, Department of Mathematics and Physics, P.O. Box 883, SE-721 23 Västerås, Sweden, kenneth.holmstrom@mdh.se.

³Tomlab Optimization AB, Västerås Technology Park, Trefasgatan 4, SE-721 30 Västerås, Sweden, anders@tomopt.com.

⁴Tomlab Optimization Inc., 855 Beech St #121, San Diego, CA, USA, medvall@tomopt.com.

Contents

Contents	2
1 Introduction	3
1.1 Overview	3
1.2 Contents of this Manual	3
1.3 More information	3
1.4 Prerequisites	3
2 Using the Matlab Interface	4
3 Setting XA Options	4
3.1 Setting options using the xaControl structure	4
3.2 IIS	4
4 TOMLAB /XA Solver Reference	5
4.1 xa	5
4.2 xaTL	11
4.3 xaControl	17
4.4 xaControl.STRATEGY	25
4.4.1 Integer Command Line Parameters	25
4.4.2 Branching Strategies	28
4.4.3 Branching Priority	29
4.4.4 LimitSearch Command Line Parameters	29
4.5 Model and Solver Status	30

1 Introduction

1.1 Overview

Welcome to the TOMLAB /XA User's Guide. TOMLAB /XA includes the XA solver suite from Sunset Software Technology and an interface to The MathWorks' MATLAB.

TOMLAB /XA is a solver package providing the user with functionality for solving linear, binary, integer and semi-continuous linear programming problems, as well as quadratic programming problems.

1.2 Contents of this Manual

- Section 1 provides a basic overview of the TOMLAB /XA solver package.
- Section 2 provides an overview of the Matlab interface to XA.
- Section 3 describes how to set XA solver options from Matlab.
- Section 4 gives detailed information about the interface routines *xa* and *xaTL*. The solver control options, branch and bound settings, as well as solver and model status codes are also explained in this section.

1.3 More information

Please visit the following links for more information:

- <http://tomopt.com/tomlab/products/xa/>
- <http://www.sunsetsoft.com>

1.4 Prerequisites

In this manual we assume that the user is familiar with linear, mixed-integer and quadratic programming, setting up problems in TOMLAB (in particular mixed-integer (**mip**) problems) and the Matlab language in general.

2 Using the Matlab Interface

The XA solver is accessed via the *tomRun* driver routine, which calls the *xaTL* interface routine. The solver itself is located in the library file *Xav14*. It is also possible to make a direct call the *xaTL* using the TOMLAB format, as well as a direct call to the solver interface by calling *xa*.

It is recommended that the TOMLAB format is used for maximum flexibility.

Table 1: The interface routines.

Function	Description	Section	Page
<i>xa</i>	The interface routine calls the XA library.	4.1	5
<i>xaTL</i>	The interface routine called by the TOMLAB driver routine <i>tomRun</i> or by a direct call. This routine then calls the interface routine <i>xa</i> , which calls the library file <i>Xav14</i>	4.2	11

3 Setting XA Options

All XA control parameters are possible to set from Matlab.

3.1 Setting options using the xaControl structure

The parameters can be set as subfields in the *Prob.MIP.xaControl* structure. The following example shows how to set a limit on the maximum number of iterations.

```
Prob = mipAssign(...)      % Setup problem, see help mipAssign for more information  
  
Prob.MIP.xaControl.ITERATION = 20000; % Setting maximum number of iterations
```

The maximum number of iterations can also be done through the TOMLAB parameter *MaxIter*:

```
Prob.optParam.MaxIter = 20000;
```

In the cases where a solver specific parameter has a corresponding TOMLAB general parameter, the latter is used only if the user has not given the solver specific parameter.

A complete description of the available XA parameters can be found in Section 4.3.

3.2 IIS

Irreducible Infeasible Sets (IIS) can be found with TOMLAB /XA. There are two options available to the user. The first one delivers the infeasible rows from the last simplex tableau. The second one will try to find a minimal number of constraints that need to be removed or corrected to make the model feasible. There may be more sets than the one delivered. The inputs and outputs are explained in Section 4.2.

4 TOMLAB /XA Solver Reference

A detailed description of the TOMLAB /XA solver interface is given below. Also see the M-file help for *xaTL.m*. It is strongly recommended that the user use the TOMLAB format when using XA. Please go to Section 4.2 for information about using XA with TOMLAB.

4.1 **xa**

Purpose

The XA linear, mixed-integer linear and quadratic programming (LP, MILP, QP) interface solves problems of the form

$$\begin{array}{ll} \min_x & f(x) = \frac{1}{2}x^T Fx + c^T x \\ s/t & x_L \leq x \leq x_U \\ & b_L \leq Ax \leq b_U \\ & x_i \text{ integer} \quad i \in I \end{array}$$

where $c, x, x_L, x_U \in \mathbb{R}^n$, $F \in \mathbb{R}^{n \times n}$, $A \in \mathbb{R}^{m \times n}$ and $b_L, b_U \in \mathbb{R}^m$. The variables $x \in I$, the index subset of $1, \dots, n$, are restricted to be integers. All variables are considered continuous if F is given.

If F is empty, an LP or MILP problem is solved.

Calling Syntax

[x,k, f_k, Inform, modsts, solsts, act, dact, status] = xa(c, A, x_L, x_U, b_L, b_U, xaControl, callback, PriLev, LogFile, SaveFile, Prob, IntVars, VarWeight, SC, SC2, F)

Description of Inputs

The following inputs are used:

- c* Linear objective function cost coefficients, vector $n \times 1$.
- A* Linear constraint matrix for linear constraints, dense or sparse matrix $m \times n$.
- x_L* Lower bounds on design parameters x . If empty assumed to be zero.
- x_U* Upper bounds on design parameters x .
- b_L* Lower bounds on the linear constraints.

The following parameters are optional:

- b_U* Upper bounds on the linear constraints. If empty, then $b_U = b_L$ is assumed, i.e. equality constraints.
- xaControl* Structure, where the fields are set to the XA control parameters that the user wants to specify values for. The control parameters are listed in Section 4.3
- callback* 0/1 vector of length 10. Defines the active callbacks during the solving process.

The following inputs are used:, continued

The callback routines and their corresponding indices in the callback vector are:

- 1 - xacb_begin.m - before solving
- 2 - xacb_infeas.m - infeasible iteration
- 3 - xacb_feas.m - feasible iteration
- 4 - xacb_node.m - integer node generated
- 5 - xacb_intsol.m - integer solution found
- 6 - xacb_branch.m - user selects b & b variable
- 7 - xacb_barrier.m - barrier iteration
- 8 - xacb_resolve.m - problem solve – > fast modify resolve
- 9 - currently not used
- 10 - xacb_end.m - after solving

The user can either modify the existing xacb_*.m functions (use i.e. ”which xacb_feas”) to find them, OR copies can be made and placed before the original files in the MATLAB path.

The calling syntax for all XA callbacks is:

```
function xacb_*( xacbInfo, Prob )
```

and is described in more detail in xacb.m (help xacb.m)

- PriLev* Printing level in the *xa m*-file and the XA C-interface.
= 0 Silent
= 1 Warnings and Errors
= 2 Summary information
= 3 More detailed information

> 10 Pause statements, and maximal printing (debug mode)
- LogFile* Name of file to write XA log to. If empty, no log is written.
- SaveFile* Name of file to write MPS representation of the problem to. The name should be given without extension - .mps is added automatically. If empty, the problem is not saved.
- Prob* A structure. If TOMLAB calls XA, then Prob is the standard TOMLAB problem structure, otherwise the user optionally may set: Prob.P = ProblemNumber, where ProblemNumber is some integer.

If any callback is defined (see description of callback) then problem arrays are set as fields in Prob, and the Prob structure is always passed to the callback routines as the last parameter. The defined fields are Prob.QP.c, Prob.QP.F, Prob.x.L, Prob.x.U, Prob.A, Prob.b.L, Prob.b.U. (if input is [], then Prob.P=1 is set).

The following inputs are used:, continued

<i>IntVars</i>	Defines which variables are integers, of general type I or binary B Variable indices should be in the range [1,...,n]. IntVars is a logical vector $\implies x(\text{find}(\text{IntVars} > 0))$ are integers. IntVars is a vector of indices $\implies x(\text{IntVars})$ are integers (if [], then no integers of type I or B are defined). XA checks which variables has $x_L = 0$ and $x_U = 1$, i.e. binary.
<i>VarWeight</i>	Vector of branching priorities for MILP problems. Should be a n-vector, ideally of integers ≥ 1 . A lower value means higher priority in the variable selection phase.
<i>SC</i>	A vector with indices for the Type 1 Semi-Continuous variables, i.e. that takes either the value 0 or a value in the range $[x_L(i), x_U(i)]$.
<i>SC2</i>	A vector with indices for the Type 2 Semi-Continuous variables, i.e. that takes either the value of the corresponding upper bound, or a value in the range $[x_L(i), 0]$.
<i>F</i>	Square dense or sparse matrix. Empty if non-quadratic problem.
<i>iisRequest</i>	A flag telling whether to obtain an IIS when a problem is determined infeasible. The flag can be set to one of the following values: 0 - Do not search for an IIS (default). 1 - Implementation of irreducible inconsistent systems (IIS) of constraints. Algorithm: IIS. 2 - Method of locationing a minimal number of constraints such that if all are removed the model is feasible. Algorithm: Block. The IIS is returned through the output parameter: iis. Information about the IIS is automatically written to the LogFile if a LogFile is defined.

Description of Outputs

The following fields are used:

<i>x_k</i>	Solution vector with decision variable values (n x 1 vector).
<i>f_k</i>	Objective function value at optimum.
<i>Inform</i>	Result of XA run. 2 = Init Successful. 4 = Load Successful. 6 = Solve Successful. 8 = Undo Successful. 10 = Done Successful. 101 = Allocation amount must be greater than or equal to 0. 102 = Allocation amount must be greater than or equal to 0.

The following fields are used:, continued

- 103 = Memory allocation error, no memory available.
- 104 = Memory allocation error, requested memory not available.
- 110-118 = XA called out of sequence.
- 122 = BXA.DLL not in path.
- 124 = Wrong version of BXA.DLL or your version has been damaged.
- 126 = Wrong routine called after XADONE.
- 128 = Restart OS.
- 132, 134, 136 = Size is incorrect.
- 204 = Incorrect No or Yes variable.
- 205 = Incorrect command line parameter.
- 207 = Unreasonable command line parameter value.
- 208 = Incorrect value set.
- 209 = Incorrect value set.
- 210 = Incorrect value set.
- 211 = Unable to process output.
- 214 = Incorrect value set.
- 300 = Column number out of range.
- 301 = Too many or zero rows in problem.
- 302 = Too many or zero columns in problem.
- 303 = Free memory error, XA could not release it's memory. Probably XA data storage area has been clobbered.
- 304 = Column lower bound > upper bound.
- 305 = Row lower bound > upper bound.
- 306 = Problem too large for available memory.
- 307 = Too many nonzeros.
- 308 = Row number out of range.
- 309 = Duplicate column and/or row.
- 310 = Arrays overlap.
- 312 = Bad colptr values.
- 314 = Unreasonable rownos values.
- 316 = Duplicate rownos value for the same column.
- 318 = Unreasonable technology coefficient in a array.
- 320 = Unreasonable value in status array.
- 322 = Unreasonable value in priority array.
- 324 = Unreasonable value in increment array.
- 326 = Semi-continuous type 1 or type 2 column missing lower bound.
- 328 = Column can not have both semi-continuous type 1 and type 2 at the same time.
- 330 = Unrecognized line in MPS file.
- 400 = MPS formatted file is missing NAME line. The NAME must start in column 1 and be the first line in the file.
- 402 = Incomplete MPS file. File probably truncated.
- 404 = Row relationship error.
- 406 = Expecting a number.
- 408 = Bound relationship error.
- 410 = Split Column declaration. All rows a column intersects must be grouped together.

The following fields are used:, continued

412 = In Column section, a column has duplicate row entries.
420 = Column has a Power sequence with a negative lower bound.
600 = Misspelled or missing DBF table filename.
602 = Error reading DBF table.
604 = Special row and column name appear together.
606 = Row increment setting; increment settings only apply to integer columns.
608 = Row priority setting, priority settings only apply to integer columns.
610 = Row field name not found in DBF table.
612 = Column field name not found in DBF table.
614 = Coefficient field name not found in DBF table.
616 = No data available in DBF table.
700 = No data previously loaded for solving.
702 = Row name size not equal to previously loaded name size.
704 = Column name size not equal to previously loaded size.
706 = Specified name size does not match row or column name size.
900 = Unknown error.
901 = Incorrect call.
930 = XA is solving an integer programming problem. A node is generated splitting the feasible region of each integer variables. The default maximum number of nodes is $\sqrt{\text{number of Integers variables}} + \text{number of 0/1 and semi-continuous columns} + 4000$ This default settings is too small. Set MAXNODES # to increase.
998 = Internal system administration error.
999 = Code has XA's internal memory.
20001 = Ranging a free/null row with MPS file.
20030 = Row name referenced in COLUMNS section is undefined.
20040 = Column has no technological coefficients, XA is fixing with zero primal activity.
20045 = Column name referenced in BOUNDS section is undefined.
20050 = Unable to save "advance basis" solution.
20060 = A free column only appears in a FREE row. SET MPSXCOMPATIBLE YES to FIX this column to zero.
20080 = Duplicate row names in ROWS section.
20090 = Unrecognized line in MPS formatted file.
902-919 = Argument xx has a unreasonable value. This can also occur when you leave off an argument.

modsts Model status. See Table [11](#)

solsts Solver status. See Table [12](#)

act Primal activities for all columns+rows.

dact Dual activities for all columns+rows.

status Status of all rows+cols at optimum.

The following fields are used:, continued

iis Structure containing IIS information. Fields:

iisStatus Status flag. Possible values:
1 - IIS was obtained.
0 - IIS was not requested.
-1 - Problem is infeasible but no IIS found.
-2 - Problem is not infeasible.

iisMessage Status message.

rowind The row indices of the IIS set.

Description

The interface routine *xa* calls XA to solve LP, QP, and MILP problems. The matrices *A* and *F* are transformed in *xa.m* to the XA sparse matrix format.

Error checking is made on the lengths of the vectors and matrices.

4.2 xaTL

Purpose

The XA LP, MILP, and QP Interface solves linear programming (LP), quadratic programming (QP) and mixed integer linear programming (MILP). *xaTL* solves problems of the form

$$\begin{array}{ll} \min_x & f(x) = \frac{1}{2}x^T Fx + c^T x \\ s/t & x_L \leq x \leq x_U \\ & b_L \leq Ax \leq b_U \\ & x_i \text{ integer} \quad i \in I \end{array}$$

where $c, x, x_L, x_U \in \mathbb{R}^n$, $F \in \mathbb{R}^{n \times n}$, $A \in \mathbb{R}^{m \times n}$ and $b_L, b_U \in \mathbb{R}^m$. The variables $x \in I$, the index subset of $1, \dots, n$, are restricted to be integers. All variables are considered continuous if F is given.

Calling Syntax

Prob = lpAssign(...); or

Prob = mipAssign(...); or

Prob = qpAssign(...);

Result = xaTL(Prob); or

Result = tomRun('xa', Prob, 1);

Description of Inputs

Problem description structure. The following fields are used:

Prob Problem structure in TOMLAB format. Use lpAssign, mipAssign or qpAssign to define the Prob structure.
Fields used in input structure Prob:

x_L, x_U Lower and upper bounds on variables, size n x 1.

b_L, b_U Lower and upper bounds on linear constraints, size m x 1.

A Linear constraint matrix, dense or sparse m x n matrix.

NOTE - all bounds vectors - if [], +/- Inf is assumed.

QP.c Linear objective function coefficients, size n x 1.

QP.F Quadratic matrix of size n x n.

PriLevOpt Print level in XATL, the XA m-file and XAmex C-interface.
= 0 Silent
= 1 Warnings and Errors
= 2 Summary information
= 3 More detailed information

> 10 Pause statements, and maximal printing (debug mode)

Problem description structure. The following fields are used:, continued

optParam Structure with optimization parameters. The following fields are used:
MaxIter Limit of iterations. If a value is given here, it is set as *xaControl.ITERATION*. Note that a value given directly in *Prob.MIP.xaControl.ITERATION* takes precedence.

Prob.XA Structure with XA specific parameters. The following fields are used:

LogFile Name of file to write XA log to. If empty, no log is written.

SaveFile Name of file to write MPS representation of the problem to. The name should be given without extension - .mps is added automatically. If empty, the problem is not saved.

callback 0/1 vector of length 10. Defines the active callbacks during the solving process.

The callback routines and their corresponding indices in the callback vector are:

- 1 - *xacb.begin.m* - before solving
- 2 - *xacb.infeas.m* - infeasible iteration
- 3 - *xacb.feas.m* - feasible iteration
- 4 - *xacb.node.m* - integer node generated
- 5 - *xacb.intsol.m* - integer solution found
- 6 - *xacb.branch.m* - user selects b & b variable
- 7 - *xacb.barrier.m* - barrier iteration
- 8 - *xacb.resolve.m* - problem solve – > fast modify resolve
- 9 - currently not used
- 10 - *xacb.end.m* - after solving

The user can either modify the existing *xacb_*.m* functions (use i.e. "which *xacb_feas*") to find them, OR copies can be made and placed before the original files in the MATLAB path.

The calling syntax for all XA callbacks is:

```
function xacb_*(xacbInfo, Prob )
```

and is described in more detail in *xacb.m* (help *xacb.m*)

iis A flag telling whether to obtain an IIS when a problem is determined infeasible. The flag can be set to one of the following values:

- 0 - Do not search for an IIS (default).
- 1 - Implementation of irreducible inconsistent systems (IIS) of constraints. Algorithm: IIS.
- 2 - Method of locating a minimal number of constraints such that if all are removed the model is feasible. Algorithm: Block.

The IIS is returned to the field: *Result.XA.iis* Information about the IIS is automatically written to the *LogFile* if a *LogFile* is defined.

Problem description structure. The following fields are used:, continued

<i>Prob.MIP</i>	Structure holding information about mixed integer optimization. Also found here is the <i>xaControl</i> structure in which XA parameter settings can be made. The fields used are:
<i>IntVars</i>	Defines which variables are integers, of general type I or binary B Variable indices should be in the range [1,...,n]. IntVars is a logical vector ==> x(find(IntVars > 0)) are integers. IntVars is a vector of indices ==> x(IntVars) are integers (if [], then no integers of type I or B are defined). XA checks which variables has x_L=0 and x_U=1, i.e. binary.
<i>VarWeight</i>	Vector of branching priorities for MILP problems. Should be a n-vector, ideally of integers >= 1. A lower value means higher priority in the variable selection phase.
<i>SC</i>	A vector with indices for the Type 1 Semi-Continuous variables, i.e. that takes either the value 0 or a value in the range [x_L(i) , x_U(i)].
<i>SC2</i>	A vector with indices for the Type 2 Semi-Continuous variables, i.e. that takes either the value of the corresponding upper bound, or a value in the range [x_L(i) , 0].
<i>F</i>	Square dense or sparse matrix. Empty if non-quadratic problem.
<i>xaControl</i>	Structure, where fields are set to the XA control parameters that the user wants to specify values for. Please refer to Section 4.3 for more information on how to set the fields.

Description of Outputs

Result structure. The following fields are used:

<i>f_k</i>	Function value at optimum.
<i>x_k</i>	Solution vector.
<i>x_0</i>	Initial solution vector not known, set as empty.
<i>g_k</i>	Exact gradient computed at optimum, computed as c or c + Fx.
<i>xState</i>	State of variables. Free==0; On lower == 1; On upper == 2; Fixed == 3;.
<i>bState</i>	State of constraints. Free==0; On lower == 1; On upper == 2; Equality == 3;.
<i>v_k</i>	Lagrangian multipliers (for bounds + dual solution vector). v_k = [rc;v]. rc n-vector of reduced costs. v holds m dual variables.
<i>rc</i>	Reduced costs. If ninf=0, last m == -v_k.
<i>ExitFlag</i>	Exit status, TOMLAB standard.

Result. The following fields are used:, continued

Inform

Result of XA run.

- 2 = Init Successful.
- 4 = Load Successful.
- 6 = Solve Successful.
- 8 = Undo Successful.
- 10 = Done Successful.
- 101 = Allocation amount must be greater than or equal to 0.
- 102 = Allocation amount must be greater than or equal to 0.
- 103 = Memory allocation error, no memory available.
- 104 = Memory allocation error, requested memory not available.
- 110-118 = XA called out of sequence.
- 122 = BXA.DLL not in path.
- 124 = Wrong version of BXA.DLL or your version has been damaged.
- 126 = Wrong routine called after XADONE.
- 128 = Restart OS.
- 132, 134, 136 = Size is incorrect.
- 204 = Incorrect No or Yes variable.
- 205 = Incorrect command line parameter.
- 207 = Unreasonable command line parameter value.
- 208 = Incorrect value set.
- 209 = Incorrect value set.
- 210 = Incorrect value set.
- 211 = Unable to process output.
- 214 = Incorrect value set.
- 300 = Column number out of range.
- 301 = Too many or zero rows in problem.
- 302 = Too many or zero columns in problem.
- 303 = Free memory error, XA could not release it's memory. Probably XA data storage area has been clobbered.
- 304 = Column lower bound > upper bound.
- 305 = Row lower bound > upper bound.
- 306 = Problem too large for available memory.
- 307 = Too many nonzeros.
- 308 = Row number out of range.
- 309 = Duplicate column and/or row.
- 310 = Arrays overlap.
- 312 = Bad colptr values.
- 314 = Unreasonable rownos values.
- 316 = Duplicate rownos value for the same column.
- 318 = Unreasonable technology coefficient in a array.
- 320 = Unreasonable value in status array.
- 322 = Unreasonable value in priority array.
- 324 = Unreasonable value in increment array.
- 326 = Semi-continuous type 1 or type 2 column missing lower bound.

Result. The following fields are used:, continued

- 328 = Column can not have both semi-continuous type 1 and type 2 at the same time.
- 330 = Unrecognized line in MPS file.
- 400 = MPS formatted file is missing NAME line. The NAME must start in column 1 and be the first line in the file.
- 402 = Incomplete MPS file. File probably truncated.
- 404 = Row relationship error.
- 406 = Expecting a number.
- 408 = Bound relationship error.
- 410 = Split Column declaration. All rows a column intersects must be grouped together.
- 412 = In Column section, a column has duplicate row entries.
- 420 = Column has a Power sequence with a negative lower bound.
- 600 = Misspelled or missing DBF table filename.
- 602 = Error reading DBF table.
- 604 = Special row and column name appear together.
- 606 = Row increment setting; increment settings only apply to integer columns.
- 608 = Row priority setting, priority settings only apply to integer columns.
- 610 = Row field name not found in DBF table.
- 612 = Column field name not found in DBF table.
- 614 = Coefficient field name not found in DBF table.
- 616 = No data available in DBF table.
- 700 = No data previously loaded for solving.
- 702 = Row name size not equal to previously loaded name size.
- 704 = Column name size not equal to previously loaded size.
- 706 = Specified name size does not match row or column name size.
- 900 = Unknown error.
- 901 = Incorrect call.
- 930 = XA is solving an integer programming problem. A node is generated splitting the feasible region of each integer variables. The default maximum number of nodes is $\sqrt{\text{number of Integers variables}} + \text{number of 0/1 and semi-continuous columns} + 4000$ This default settings is too small. Set MAXNODES # to increase.
- 998 = Internal system administration error.
- 999 = Code has XA's internal memory.
- 20001 = Ranging a free/null row with MPS file.
- 20030 = Row name referenced in COLUMNS section is undefined.
- 20040 = Column has no technological coefficients, XA is fixing with zero primal activity.
- 20045 = Column name referenced in BOUNDS section is undefined.
- 20050 = Unable to save "advance basis" solution.
- 20060 = A free column only appears in a FREE row. SET MPSXCOMPATIBLE YES to FIX this column to zero.
- 20080 = Duplicate row names in ROWS section.
- 20090 = Unrecognized line in MPS formatted file.
- 902-919 = Argument xx has a unreasonable value. This can also occur when you leave off an argument.

Iter

Number of iterations / nodes visited.

Result. The following fields are used:, continued

<i>FuncEv</i>	Number of function evaluations. Set to Iter.
<i>GradEv</i>	Number of gradient evaluations. Set to Iter if QP/MIQP, otherwise 0. FuncEv and ConstrEv set to Iter. GradEv=0.
<i>ConstrEv</i>	Number of constraint evaluations. Set to 0.
<i>QP.B</i>	Basis vector in TOMLAB QP standard.
<i>Solver</i>	Name of the solver (XA).
<i>SolverAlgorithm</i>	Description of the solver.
<i>MIP.slack</i>	Slack variables (m x 1 vector).
<i>MIP.ninf</i>	Number of infeasibilities.
<i>MIP.sinf</i>	Sum of infeasibilities.
<i>MIP.lpiter</i>	Number of LP iterations.
<i>MIP.glnodes</i>	Number of nodes visited.
<i>MIP.basis</i>	basis status of constraints + variables, (m + n x 1 vector) in the XA format, fields xState and bState has the same information in the Tomlab format.
<i>XA.iis</i>	Structure containing IIS information. Fields:
<i>iisStatus</i>	Status flag. Possible values: 1 - IIS was obtained. 0 - IIS was not requested. -1 - Problem is infeasible but no IIS found. -2 - Problem is not infeasible.
<i>iisMessage</i>	Status message.
<i>rowind</i>	The row indices of the IIS set.

Description

The TOMLAB XA LP, MILP and QP interface calls the interface routine *xa.m*. Values $> 10^{10}$ and *Inf* values are set to 10^{10} , and the opposite for negative numbers. An empty objective coefficient *c*-vector is set to the zero-vector.

4.3 xaControl

The following table describes the XA parameters that the user can set using the xaControl structure.

Table 6: XA control parameters in xaControl, The following fields are used:

<i>LPMETHOD</i>	<p>LP optimizer algorithm that will be used. For QP problems Barrier will always be used.</p> <p>0 - Default. Dual Simplex. 1 - Primal Simplex. 2 - Dual Simplex. 3 - Barrier. 4 - Network Primal Simplex algorithm (not active). 5 - Network Dual Simplex algorithm (not active). 6 - Network Generalize Primal Simplex algorithm (not active). 7 - Network Generalize Dual Simplex algorithm(not active).</p>
<i>BASIS</i>	<p>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. Don't worry about new or deleted variables, constraints, etc. XA will take care of all these adjustments. The Dual Simplex algorithm is used when XA detects advance basis restarts. You can instruct XA to use the Primal Simplex algorithm for restarts as follows, Set DualSimplex No.</p>
'filename.ext'	<p>The filename containing an 'advance basis' for restarting. Default file extension is SAV.</p>
'none'	<p>No 'advance basis' is specified, but the 'final basis' is saved in the problem filename with an extension of SAV.</p>
'never'	<p>No 'advance basis' is specified, and the final 'basis' is not saved.</p>
	<p>Filename containing basis information (if present), after calling XA this file is updated with new basis information. Overrides status setting. '*' means to use the FILENAME setting value. Defaults to no basis file.</p>
<i>FILENAME</i>	<p>File name to write information. Please specify drive and path information so you can find your file(s). BASIS '*' write to this name . sav, .b01 and .r01. ToRCC Yes writes to this name .rcc. Default value: RUNTIME in current directory. Example: 'path\filename'</p>

Table 6: XA control parameters in xaControl, The following fields are used:, continued

LIMITSEARCH LimitSearch is used to limit the number of nodes to search by implicitly or explicitly stating a bound on the value of the integer solution. See Section 4.4.4 for detailed explanation.
 Default value is no limiting value, or, -1.0e23 for maximizing and 1.0e23 for minimizing.
 Possible values: # (1), #% (2), (##) (3).

MAXIMIZE Sense of optimization.
 No (0) - minimize the objective function (default value).
 Yes (1) - maximize the objective function.

PRESOLVE Prior to solving a model, the XA Optimizer attempts to simplify and reduce the size (number of rows, columns and non-zero coefficients). The amount and type of model reduction is represented by the following 'bit' pattern:

Type Reduction	Bit Pattern
	12345 67890 123
Singleton Column	1xxxx xxxxx xxx
Singleton Row	x1xxx xxxxx xxx
Min/Max Row Adjust	xx1xx xxxxx xxx
Dual Checking	xxx1x xxxxx xxx
Min/Max Column Adjust	xxxx1 xxxxx xxx
Quick Dual Check	xxxxx 1xxxx xxx
Identical Column	xxxxx x1xxx xxx
Dependent Rows	xxxxx xx1xx xxx
Row Doubleton Reduction	xxxxx xxx1x xxx
Free Column Elimination	xxxxx xxxx1 xxx
Matrix Reduction	xxxxx xxxxx 1xx
Restore Original Bounds	xxxxx xxxxx x1x
Expensive Dual Test	xxxxx xxxxx xx1

A one (1) activates and a zero (0) prohibits the a specific reduction feature. Default Pattern : '1111111111100' (string)

Consideration: When you know your model does not have a lot of unnecessary rows or columns, or if you plan to solve variations of the same model 1,000s of times you should consider reducing the amount of pre-solve checking because these routines can be CPU intensive.

Example, xaControl.PRESOLVE = '1110111111100';

CRASH Method of generating initial basis.

Table 6: XA control parameters in xaControl, The following fields are used:, continued

	<p>0 - Minimize primal infeasibility (default value). 1 - Minimize dual infeasibility. 2 - Both 0 and 1. 3 - All slack basis.</p>
<i>DEGENITER</i>	<p>Degenerate anti-cycling aide. Number of consecutive degenerate pivots before anti-cycling code is activated. Default value: square root of the number of rows.</p>
<i>ELEMSIZE</i>	<p>The smallest element that can be stored in the LU factors arrays. If the absolute value is a number less than this it is considered zero (0.0). Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility. Default value: 1.0e-12</p>
<i>ELIMINATE</i>	<p>See Command Line Parameter PRESOLVE. 'No' (0), 'Yes' (1), 'Off' (2).</p>
<i>FREQLOG</i>	<p>Frequency in seconds to print the iteration log line. A negative number (e.g. -2) overwrites the same line. This command reduces the overhead of printing too many iteration lines. Default value: 1 (one log line per second).</p>
<i>IBOUNDS</i>	<p>Upper bound for integer column. Default value: 1.0</p>
<i>INTGAP</i>	<p>Minimum objective function improvement between each new integer solutions. Reported integer solution may not be the optimal integer solution because of premature termination. Default value 0.00.</p>
<i>INTLIMIT</i>	<p>After finding this number of improving integer solution XA terminates with the best solution thus far. Defaults - no limit the number of integer solutions. Reported integer solution may not be the optimal integer solution because of premature termination.</p>
<i>INTPCT</i>	<p>Percent of available integer columns to consider fixing at each integer node. Useful on very large binary problems. Default is zero (0.0) meaning no fixing. 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.</p>

Table 6: XA control parameters in xaControl, The following fields are used:, continued

<i>IROUND</i>	<p>XA reports either rounded or unrounded integer column primal activity. 1 is YES (Default is 1), 0 is NO 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.</p>
<i>ITERATION</i>	<p>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. Default value: 2000000000 Reported integer solution may not be the optimal integer solution because of premature termination.</p>
<i>LIMITNODES</i>	<p>Maximum number of branch and bound nodes to generate. XA terminates if limit is exceeded. Default value: 2000000000 Reported integer solution may not be the optimal integer solution because of premature termination, or an integer solution may not have been found.</p>
<i>L-, UTOLERANCE</i>	<p>The tolerance (closeness) XA uses to solve integer column to its numeric sequence. Values between LTOLERANCE and UTOLERANCE are candidates for branch and bound. For instance, you might consider using an UTOLERANCE of 0.02 (a boat 98% full for all practical purposes to 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 $3 * Y$ ranges of $3 * 3.95$ to $3 * 4.001$. Default values: 5.0e-6.</p>

Table 6: XA control parameters in xaControl, The following fields are used:, continued

<i>MARKOWITZ</i>	<p>Numeric Stability vs. sparsity in basis updating and inverse. Markowitz number is used during matrix inversion and updating the LU factors. During inversion, row and column factors are scanned for the largest and smallest numbers (in an absolute sense). The pivot element is chosen such that the ratio of the pivot element to these numbers is less than the MARKOWITZ number if possible. The larger the MARKOWITZ number is, the larger will this ratio be, which leads to sparsity but can cause numeric instability. The best number of numeric stability is 1.0 which tends to leads to build-up (density) in the LU factors, which effect speed. So simply put - 1) A value of 1.0 is the best for numerical stability and the worst for density. 2) A value of 1000.0 is (can be) bad for numerical stability but good to maintain sparsity of LU factors. Sparsity favors a larger number at the expensive of numeric stability. Default 10</p>
<i>MAXNODES</i>	<p>Maximum 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.</p>
<i>MPRICING</i>	<p>Number of variables to price per simplex iteration. Zero (0) defaults to 2 x square root of the number columns. Default 0.</p>
<i>PERTUBATE</i>	<p>After making 'degeniter' degenerate iterations (pivots), zero pivot values are adjusted by this amount in an attempt to move away from this region. After a non-degenerate pivot is made the 'degeniter' count is reset to zero. Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility. Default value: 0.0e0</p>
<i>PRICING</i>	<p>Variable pricing strategies. 0 - Standard reduced cost pricing. 1 - Automatic DEVEX pricing switch over. 2 - Infeasible DEVEX pricing. 3 - Feasible DEVEX pricing(default value). 4 - Both infeasible and feasible DEVEX pricing.</p>
<i>TOLERANCE_DUAL</i>	<p>Dual activity tolerance is considered to be zero. Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility. Default value: 1e-7</p>

Table 6: XA control parameters in xaControl, The following fields are used:, continued

<i>REINVERTFREQ</i>	<p>After making this number of iterations (pivots), the basis is refactor based upon the current columns in the basis and all LU pivoting factor are removed. Reducing the number of pivots before 're-inversion' usually reduces the number of iterations to solve a problem but at the expense of performing more basis inversion which use more CPU time.</p> <p>Default value: 40</p>
<i>REJPIVOT</i>	<p>After a column is selected to enter the basis, a column is selected to leave the basis to maintain model feasibility. All candidate columns to leave the basis which have a 'marginal' value greater than the 'ypivot' value are initially consider as outgoing columns. If absolute value of the selected outgoing column's 'marginal' value is less than the 'rejpivot' value then this entering column is 'initially' rejected and the search for another column to enter the basis is performed and the outgoing column selection is performed again until the 'rejpivot' value is exceeded. If all enter columns are rejected then the entering column with the largest 'marginal' value is selected without regards to the 'rejpivot' value.</p> <p>Increasing this value, for example to 1.0e-4, improves numerical stability on numerically unstable models but increases solve times.</p> <p>Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility.</p> <p>Default value: 1.0e-6</p>
<i>RELAXED</i>	<p>Integer problem is solved as a standard LP and with no integer columns in the formulation.</p> <p>0 - integer problem are solved with branch and bound method (default value).</p> <p>1 - solve problems as if all columns where continuous columns.</p>
<i>SCALE</i>	<p>Problem Scaling technique.</p> <p>'No' (0) - Data not scaled.</p> <p>'Yes' (1) - Column and row scaling (default value).</p> <p>'2' (2) - Row scaling only.</p>
<i>SPROUTS, RUNNER</i>	<p>When solving mixed integer models the Optimizer must decide when to generate a 'marker' on the branch and bound node tree. This 'marker' is used to very quickly return to this location and begin exploring other branch directions. The 'marker' table is scanned at regular intervals to try to improve integer solutions.</p> <p>Default value is 0 means no 'markers' are generated.</p> <p>A positive SPROUTS value indicates the frequency of 'marker' generation.</p> <p>A negative SPROUTS value indicates the amount of reduction in the number of non-integer columns before a 'marker' is generated.</p>

Table 6: XA control parameters in xaControl, The following fields are used:, continued

	<p>'Marker' generate is memory intensify and the amount of memory requested in the call to XAINIT should increase to the maximum amount of memory on your machine.</p> <p>When solving mixed integer models the Optimizer must decide how to handle node generation, as either depth first or breadth first. The 'depth first' approach continues to generate nodes deeper into the branch tree and only backtracks nodes when an infeasibility or a integer solution is found. The 'breadth first' approach generates and solves both sides of a node tree before deciding which node to continue with to the next level. This process continues until an infeasibility or integer solution is found. One can control which approach is used with the RUNNER command line parameter. Default value is 'No' (0) meaning 'breadth first' node generation. A value of 'Yes' (1) meaning to use 'depth first' node generation.</p>
<i>STICKWITHIT</i>	<p>Once an integer problem is solved, a bit in the 'status' array for each column is set to indicate the branching direction for the next XA iteration. This branching advice is following until this number of infeasible branches are made. Default value 10. After this number of infeasible branches, the standard branching direction for the particular branch and bound strategy is used.</p>
<i>TOLERANCE</i> <i>_TCOEFFICIENTS</i>	<p>The smallest technological coefficient allowed in your 'a' 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. Default value: 1.0e-7</p>
<i>TIMELIMIT</i>	<p>Maximum time in seconds allowed to solving the problem. XA terminates if limit is exceeded, and if solving an integer problem the best integer solution found thus far is returned. Wall clock time. Default value: 2000000000 seconds. If set too low, reported integer solution may not be the optimal integer solution because of premature termination.</p>
<i>TRANSENTRY</i>	<p>After the entering and outgoing basis columns are selected the LU factors are updated. The absolute value of the size of the updated LU factors updated is limited to the pivoting 'marginal' value times 'transentry' value. Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility. Default value: 1.0e-13</p>

Table 6: XA control parameters in xaControl, The following fields are used:, continued

<i>TOLERANCE_PRIMAL</i>	Primal activity values are considered to be zero. Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility. Default value: 1e-8.
<i>YPIVOT</i>	When selecting a column to leave the basis the column's marginal values that are less than 'ypivot' in an absolute sense are rejected. Making pivots with very small values can lead to numeric stability and should be avoided when possible while making pivots with too large a 'ypivot' value can lead to infeasible pivoting. Extreme caution should be exercised when changing this value because of the overall effect on problem feasibility. Default value: 1.0e-10.
<i>STOPAFTER</i>	Amount of time in seconds to continue solving after finding the first integer solution. Default value: 0, indicating no termination. Reported integer solution may not be the optimal integer solution because of premature termination.
<i>STOPUNCHANGED</i>	Amount of time in seconds to continue solving without finding a better/improving integer solution. Default value: 0, indicating no termination. Reported integer solution may not be the optimal integer solution because of premature termination.
<i>STRATEGY</i>	MIP solving strategy. See Table 8 for possible values. Default value: 1
<i>TORCC</i>	Write an RCC file of problem. The filename is determined by the FILENAME command line parameter. 'No' (0) - Do not write an RCC formatted file (default value). 'Yes' (1) - Write problem in RCC format.

4.4 xaControl.STRATEGY

Integer models are ones where one or more decision variable must not have fractional values. This is a very much harder problem than an ordinary LP. In the worst case, the amount of time to solve a family of related problems goes up exponentially as the size of the problem grows, for all algorithms that solve such problems to a proven an optimal integer answer.

We use the word integer to describe any binary, generalized integer, semi-continuous, prime, fibonacci, power and special sequence variables. For example, you can define a variable as semi-continuous, generalized integer with increments of 0.25. The only restriction is that you can not declare a variable to be both type I and type II semi-continuous.

Mixed integer linear programming problems are solved by first: optimizing all variables as being continuous variables; and secondly: the problem is searched for integer solutions. That is, feasible solutions are ones satisfying the constraints and which give integer values to the integer variables. The search is started from the optimal continuous solution. The integer decision variables are forced to take integer values using a 'brand and bound' technique with heuristic branching rules. You should be prepared to solve much smaller MIP models than the corresponding LP model. There exist models that are considered challenging, with a few dozen variables. Conversely, some models with tens of thousands of variables solve readily. But a MIP model with hundreds of variables should always be approached, initially at least, with a certain amount of caution.

There are numerous parameters and options available to control the solution strategy. XA has the capability of stopping before an optimum is proved, processing/printing the best integer answer obtained so far. For many MIP models, stopping early is a practical necessity. Fortunately, a solution that has been proved by the algorithm to be within, say, 1% of optimality often turns out to be the true optimum, and the bulk of the computation time is spent proving the optimality. For many modeling situations, a near-optimal solution is acceptable.

Whatever the solution method you choose, when trying to solve a difficult MIP model, it is usually crucial to understand the workings of the physical system you are modeling, and try to find some insight that will assist your chosen algorithm to work better.

The numeric bound on integer variable determines if the variable is binary (0 or 1) or a generalized (0,1,2,3, ...) integer variable. If you do not specify an upper bound for an integer variable then default value is 1.

4.4.1 Integer Command Line Parameters

XA is designed to solve a vast majority of LP problems using the default settings. When solving integer problems the default setting may not provide the best for speed and reliability. By experimenting with these parameters performance can be improved dramatically. The following table shows some of the options.

MIP performance options	
Topic	Integer Specific Command Line Parameter
<i>Branch and Bound Choice</i>	Strategy a (0).
<hr/>	
<i>Stopping Criteria</i>	
Limit the range to search for an integer solution.	LIMITSEARCH
Time limit of 1 hour and 25 minutes	Set TIMELIMIT 5100.

Table 7: MIP performance options, continued

Topic	Integer Specific Command Line Parameter
Iteration limit of 100,000.	Set ITERATION 100000.
Iteration limit of 100,000.	Set ITERATION 100000.
Stop after find 3 improving integer solutions.	Set INTLIMIT 3
Stop after running for 10 minutes after finding the first integer solution.	Set STOPAFTER 600
<hr/>	
<i>Priority branching</i>	
Set integer variable branching order to the order the columns are defined in problem.	Priority o, or in reverse order of definition, Priority O.
Set branching order by ascending number of nonzeros per column.	Priority s, or descending number, Priority S.
Branching order determined by ascending objective coefficient values.	Priority c, or descending number, Priority C.
Set the default priority for binary variables to branch before generalized integers.	Set BVPriority 1000
Set the default priority for binary variables to branch after all generalized integers.	Set BVPriority 20000
<hr/>	
<i>Near Integer Optimal Solutions</i>	
Solve for a solution with in 1000 units of the optimal integer solution.	Set INTGAP 1000.
Solve within 10 percent of the optimal integer solution.	LIMITSEARCH (10%).
<hr/>	
<i>Integer Variable Tolerances</i>	
Tolerance to variable's upper bound sequence value.	Set UTOLERANCE number.

Table 7: MIP performance options, continued

Topic	Integer Specific Command Line Parameter
Tolerance to variable's lower bound sequence.	Set LTOLERANCE number.
Report the integer variables with unrounded results.	Set IROUND No (0).
Change the default upper bound on implicitly bounded integer variables from 1 to 1000.	Set IBOUND 1000.
<hr/>	
<i>Restarting Integer Models</i>	
Use the previous integer solution stored in filename.r01 and re-start problem right were it was interrupted.	Set RESTART Yes (1).
Use the previous integer solution as a guide to solving this problem.	Basis filename.
Stop after solving the lp part of an integer problem. Do not attempt to solve the integer part of problem.	Set Relaxed Yes.
<hr/>	
<i>Performance Adjustments</i>	
Perform integer variable elimination's at each node and presolve between each node generated.	Set INTPCT 30
<hr/>	
<i>Tree Depth Strategy</i>	
Limit the run time and tree depth for the branching node at each subdivided branching node.	TREETIME 10 (units are seconds). TREEDEPTH 8 ($\neq 8$ nodes).
<hr/>	
<i>Primal/Dual Solver</i>	
Use the dual simplex method to solve subproblems.	Set LPMETHOD 2.

4.4.2 Branching Strategies

Nine 'branch and bound' strategies are provided to meet the demands of many different types of problems. Each strategy has four variations which effects the solve time, speed to first solution, and finding best integer solution. The order in which integer variables are processed during the search for an integer solutions is important. This order is called the 'branching order' of integer variables. Solution times can vary significantly with the method selected.

A command line parameter, STRATEGY, is available to select the strategy of your choice. The STRATEGY parameter may be specified with one of following values:

Table 8: Branch and Bound options

B & B Strategy	Description of Selection Criteria
1	Proprietary method. Default value.
2	Minimum change in the objective function.
3	Priority based upon column order.
4	Column closest to it's integer bound.
5	Column always branches up (high).
6	Column always branches down (low).
7	Column farthest from it's integer bound.
8	Column randomly selected, useful when solving very large problems.
9	Apparent smoothest sides on the polytope.

Each XA Branch and Bound strategy has 2 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.

Table 9: Branch and Bound variations

Variation	Effect
A (1)	This variation reduces the amount of time XA spends estimating the value of a would be integer solution. The values calculated are rough estimates. STRATEGY 1A Variation A may not appear with variation B.
B (2)	This variation spends very little time calculating estimated integer solutions at each node and is the most radical in performance and integer solution. STRATEGY 8B Variation B may not appear with variation A.
C (3)	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.

Table 9: Branch and Bound variations, continued

Variation	Effect
	<p>STRATEGY 6C Variation C may not appear with variation D.</p>
<i>D (4)</i>	<p>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.</p> <p>STRATEGY 1D Variation D may not appear with variation C.</p>
<i>P (5)</i>	<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 non-integer variable(s) with the lowest branching priority, which frequently reduces the number of calculations.</p> <p>STRATEGY 1P or STRATEGY 6BP Variation P may appear any variation, but to be effective you must assign integer branching priorities.</p>

4.4.3 Branching Priority

Branching priorities are completely independent of branching strategies.

1. XA develops a lists of integer/binary/semi- variables which are NOT integral (not sequence appropriate).
2. This list is trimmed to variables having the same lowest priority.
3. If strategy value is suffixed by P then 2)'s list is used to compute changes in the objective value, else 1)'s list is used. A large saving of compute time can be found depending upon the size of 1) compared to 2). Plus 2) list can also help the solver sequence the b and b candidates, e.g., period 1 then period 2, ordering. This have been found to help out a lot.
4. Now the branching strategy is used to select the variable and direction to branch, up or down.

The priority values used be XA in increasing order of preference. 1) Priority, 2) BV PRIORITY for BV variables with default priority value of 16000 are change to this value. This is XA internal default priority value which is used to initialize all variables.

4.4.4 LimitSearch Command Line Parameters

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 a functional value better than the LIMITSEARCH value does 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 (3) methods of specifying a lower limit on the objective function.

Table 10: LIMITSEARCH Methods

Value	Meaning
## (1)	Only search for integer solutions between this value and the 'optimal continuous' solution.
##% (2)	Only search for integer solutions with #% of the 'optimal continuous' solution.
(#%) (3)	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 but will be within #% of it.

You must experiment with different Strategies and LimitSearch values to determine which is best for your problems. We have found that one of the following settings generally works quite well:

- STRATEGY 1, LIMITSEARCH (5%), STOPAFTER 900
- STRATEGY 6, LIMITSEARCH (5%), STOPAFTER 900

Also try strategies 1A, 1B, 6A, 6B, and 9. The LimitSearch value should be as large as possible and still meets your business objectives (distance from the optimal integer solution). As you gain experience with these 'strategies', you will be able to make an 'informed' choice.

4.5 Model and Solver Status

Table 11: Model Status Codes

Code	Model Status (modsts)
1	Optimal (Integer) Solution.
2	Integer Solution (not proven the optimal integer solution).
3	Unbounded solution.
4	Infeasible solution.
5	Callback function indicates Infeasible solution.

Table 11: Model Status Codes, continued

Code	Model Status (modsts)
<i>6</i>	Intermediate infeasible solution.
<i>7</i>	Intermediate nonoptimal solution.
<i>9</i>	Intermediate Non-integer solution.
<i>10</i>	Integer Infeasible.
<i>13</i>	More memory required to load/solve model.
<i>32</i>	Integer branch and bound process currently active, model has not completed solving.
<i>99</i>	Currently solving model, model has not completed solving.

Note: Integer problems return a Model Status code of 1 if the optimal integer solution is found. If XA has not proven that its integer solution is optimal, then a Model Status code of 2 is returned.

Table 12: Solver Status Codes

Code	Solver Status (solsts)
<i>1</i>	Normal Completion.
<i>2</i>	Iteration Interrupt.
<i>3</i>	Resource Interrupt, like time limit exceeded.
<i>4</i>	Terminated by User, probably a Cntrl Z.
<i>8</i>	Node Table Overflow.
<i>10</i>	Solver Failure.