USER'S GUIDE FOR TOMLAB /Xpress R2004

Kenneth Holmström[2], Anders O. Göran[3] and Marcus M. Edvall[4]

February 26, 2007

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

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

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

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

Contents

1 Introduction

1.1 Overview

TOMLAB /Xpress includes an embedded license for Xpress^{MP} and the TOMLAB interfaces developed as part of the package.

The interface between Xpress^{MP} , Matlab and TOMLAB has been designed at two layers. The first layer gives direct access from Matlab to Xpress^{MP} , via calling one Matlab function that calls a pre-compiled DLL (or lib in Unix) that defines and runs the problem in Xpress^{MP} . The second layer is a Matlab function that takes the input in the TOMLAB format, and calls the first layer function. On return the function creates the output in the TOMLAB format.

Xpress^{MP} has a whole set of callback routines. There is one predefined Matlab routine for each callback. The user is in control of which ones to use, and should add his own code in Matlab for each callback. As default only the User Output Callback is called, that prints Error and Warning messages.

1.2 Contents of this Manual

Read carefully Section 2 on how to install the interface. Section 3 gives the basic information needed to run the Matlab interface. The more advanced feature, using callbacks, is described in Section 4. Some Matlab test routines are included, described in Section 5 (non-TOMLAB format) and Section 6 (TOMLAB format). All Matlab routines are described in Appendix A.

1.3 Prerequisite

In this manual we assume that the user is familiar with Xpress^{MP} TOMLAB and the Matlab language.

2 Installation of TOMLAB /Xpress

To check the TOMLAB /Xpress installation, try out the example files, e.g.

```
x=xptest1;
```

or

```
xpknaps
```

Note that xpknaps takes three input argument that one can play around with. If TOMLAB is installed, to run a first test using the test examples in the TOMLAB distribution, files in \tomlab\testprob, run

```
xptomtest1
```

To see how to quickly define a problem in the TOMLAB format using the TQ format (TOMLAB Format), run (and study the code)

```
xptomtest2
```

To see how to set parameters that influence the runs in TOMLAB /Xpress run

```
xpknapsTL
```

This file does the same as xpknap, but is using the TQ format, and setting the relevant parameters into the *Prob* structure before calling the TOMLAB driver routine *tomRun*.

3 Using the Matlab Interface

The two main routines in the two-layer design of the interface are shown in Table 1. Page and section references are given to detailed descriptions on how to use the routines. The user that is not using TOMLAB can skip reading about the routine *xpressTL*. A normal user, not using callbacks, only has to read about how to call the level 1 interface routine *xpress*.

Table 1: The interface routines.

Function	**Description**	**Section**	**Page**
xpress	The layer one Matlab interface routine, calls the MEX-file interface *xpressmp.dll*	A.1	38
xpressTL	The layer two TOMLAB interface routine that calls *xpress.m*. Converts the input *Prob* format before calling *xpress.m* and converts back to the output *Result* structure.	A.2	44

The Xpress^{MP} control variables, see Section 7 in the Xpress-Optimizer Reference Manual [1], are all possible to set from Matlab. They could be set as input to the interface routine *xpress*, but also in the callback routines. The user sets fields in a structure called *xpcontrol*, where the subfield names are the same as the names of the control variables. The following example shows how to set the values for one integer variable XPRS_LPITERLIMIT, one double variable XPRS_OPTIMALITYTOL, and one character variable valued variable XPRS_OBJNAME. TOMLAB /Xpress does not use the prefix XPRS_ in the Matlab structures.

```
xpcontrol.LPITERLIMIT   = 50;      % Setting maximal number of global iterations
xpcontrol.OPTIMALITYTOL = 1E-5;    % Changing reduced cost tolerance
xpcontrol.OBJNAME       = 'ObjF1'; % New name of the objective function
```

Character valued variables should have ≤ 64 characters.

Note that after the call to the Xpress^{MP} interface, two global variables are available, xpControlVariables and xpProblemAttrib. As subfields they hold the current values of all Xpress^{MP} control variables see Section 7 in [1], and all problem attributes, see Section 8 in [1]. To list all their fields, just define the variables as global and give their names

```
global xpControlVariables xpProblemAttrib
xpControlVariables
xpProblemAttrib
```

4 Callbacks in Matlab

There are eleven of the Xpress^{MP} callbacks defined in the interface. A logical vector defines the callbacks to be used in Xpress^{MP} . This vector is named *callback* and is one of the input variables to the level 1 interface routine *xpress* (Section A.1, page 38). If the i^{th} entry of the logical vector callback is set, the corresponding callback is defined. See Section 5.3 in [1]. The callback calls the *m*-file specified in Table 2. The user can edit this *m*-file directly, or make a new copy. It is important that a new copy is placed in a directory that is searched before the *xpress* directory when Matlab goes through the Matlab path.

Table 2: The *m*-file callback routines.

Index	m-file	Description	**Section**	**Page**
(1)	xpcb_usn	User Select Node Callback	D.11	71
(2)	xpcb_upn	User Preprocess Node Callback	D.10	70
(3)	xpcb_uon	User Optimal Node Callback	D.8	69
(4)	xpcb_uin	User Infeasible Node Callback	D.6	67
(5)	xpcb_uis	User Integer Solution Callback	D.7	68
(6)	xpcb_ucn	User Node Cut-off Callback	D.5	66
(7)	xpcb_ucb	User Choose Branching Variable Callback	D.4	64
(8)	xpcb_il	Simplex Log Callback	D.3	64
(9)	xpcb_gl	Global Log	D.2	61
(10)	xpcb_bl	Barrier Log Callback	D.1	61
(11)	xpcb_uop	User Output Callback	D.9	69

Before each call to the callback routine, the interface is defining the control variables and problem attributes as global variables in Matlab. By making the following global declarations in the callback *m*-file,

```
global xpControlVariables xpProblemAttrib
```

all control variables and problem attributes are accessible as subfields to the global variables.

The *Prob* structure is input to all callback *m*-file routines as the last parameter. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber$;, where ProblemNumber is some integer. If any callback is defined (see description of callback) then the arrays that define the current problem are set by the interface as fields in *Prob*. The defined fields are *Prob.c*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U* and. *Prob.QP.F*. The user then has full access to the original problem in the callback routine.

In one of the callback routines, *xpcb_gl*, a simple knapsack heuristic is implemented. This heuristic is also part of the standard Xpress^{MP} example files. Running the knapsack test program *xpknaps*, with the second input argument *Run* = 1 runs the knapsack heuristic in the callback. *xpknaps* sets *Prob.MIP.KNAPSACK = 1* to enable execution of the heuristic in *xpcb_gl*.

If there are no integer variables in the problem to be solved, i.e. a pure LP or QP problem, then the first seven callbacks as well as the 9^{th} callback (Global Log) are automatically disabled. If the problem is a mixed-integer problem, then the 8^{th} callback (Simplex Log Callback) is disabled. This change is made on around row 254 i *xpress.m* and the user may comment this row ($callback(8) = 0;$) to avoid it.

5 Test Routines in Non-Tomlab Format

A set of test routines have been defined illustrating the use of the *xpress* main routine. The test routines and utilities are shown in Table 3.

It is easy to call the test routines, e.g.

```
x = xptest1;
x = xptest2;
x = xptest3;
```

will call the three routines solving GAP problems. The *xpaircrew* test problem has no input parameters, just call:

```
xpaircrew;
```

The knapsack test routine runs three test examples. It is possible to change the cut strategy (third input parameter) and whether to use the knapsack heuristic in the callback routine *xpcb_gl* (second input parameter). To run the second test example, using the simple knapsack heuristic, and aggressive cuts, the call is

```
xpknaps(2,1,2);
```

The first parameter selects the test problem. Calling without any parameters

```
xpknaps
```

is the same as the call

```
xpknaps(1,0,0);
```

Table 3: The test routines and utilities.

Function	**Description**	**Section**	**Page**
abc2gap	Utility to convert a Generalized Assignment Problem (GAP) to standard form for Xpress^{MP} .	B.2	50
xpbiptest	Test of three large binary integer linear problems.	C.2	53
xpiptest	Test of three large integer linear problems.	C.3	54
xpknaps	Test of knapsack problems.	C.6	56
xptest1	Test of a Generalized Assignment Problem (GAP).	C.8	58
xptest2	Test of the same GAP problem as *xptest1*, but using sos1 variables.	C.9	58
xptest3	Test of a Generalized Assignment Problem (GAP).	C.10	59

6 Test Routines in Tomlab Format

A set of test routines have been defined illustrating the combined use of TOMLAB and Xpress^{MP} The test routines are shown in Table 4. The knapsack test routine *xpknapsTL* is similar to *xpknaps* discussed in the previous subsection. It runs three knapsack test examples. It is possible to change the cut strategy and whether to use the knapsack heuristic in the callback routine *xpcb_gl*. The problems are setup using the TOMLAB Format.

Table 4: The test routines and utilities.

Function	**Description**	**Section**	**Page**
xptomtest1	Tests of problems predefined in the TOMLAB IF format. LP, QP and MIP problems are solved calling the driver routine *tomRun*.	C.4	56
xptomtest2	Tests of a simple MIP problem defined in the TOMLAB (TQ) format. The problem is solved as an LP and MIP problem, with or without slacks defined. *tomRun*.	C.5	56
xpknapsTL	The same tests as in *xpknaps*, but the TOMLAB format and system is used.	C.7	57

The following example shows how to run a predefined problem, one of the problems in *xpknapsTL*, using the TOMLAB Init File (IF) format. Some fields that changes control variables are set to make Xpress^{MP} work slower. Then a simple predefined heuristic is run to try to improve the convergence. For this example it can get down the number of node visited from 135 to 35.

```
Prob = ProbInit('mip_prob',7); % Create Prob structure from predefined problem
Prob.MIP
ans =
    xpcontrol: []
      IntVars: [30x1 double]
    VarWeight: [30x1 double]
     KNAPSACK: 1
     callback: [14x1 double]
% Make Xpress-MP work slower by disabling presolve and cuts.
Prob.MIP.xpcontrol.CUTSTRATEGY = 0; % Use no cuts
Prob.MIP.xpcontrol.PRESOLVE = 0;    % Use no presolve
Prob.MIP.xpcontrol.MIPLOG = 3;      % Call the callback each node
Result = tomRun('xpress-mp',Prob,1); % Call Xpress-MP using Tomlab driver

===== * * * ================================================ * * *
TOMLAB SOL - Three weeks demonstration single user license
=================================================================
Problem: mip_prob -  7: Weingartner 1 - 2/28 0-1 knapsack  f_k -141278.000000000000000000
                                              User given f(x_*)-141278.000000000000000000

Solver: Xpress-MP.  EXIT=0.  INFORM=6.
Mex-interface to Xpress-MP LP/QP/MIP/MIQP solver
Global search complete - integer solution found

FuncEv  135 GradEv    0 Iter  135
CPU time: 0.250000 sec. Elapsed time: 0.250000 sec.
```

```
Prob.MIP.callback(9) = 1; % Enable Global Log callback
Result = tomRun('xpress-mp',Prob,1);
% Because Prob.MIP.KNAPSACK == 1, the predefined heuristic is tried

--- Global Log. Node      1. Node depth   1. Best Bound -142019.000000.
Node 1: LP-obj -142019;  Heuristic value -139508 *** Updated cutoff to -139507
--- Global Log. Node      2. Node depth   2. Best Bound -142019.000000.
Node 2: LP-obj -141897;  Heuristic value -140768 *** Updated cutoff to -140767
--- Global Log. Node      3. Node depth   3. Best Bound -142019.000000.
Node 3: LP-obj -141873;  Heuristic value -138493
--- Global Log. Node      4. Node depth   4. Best Bound -142019.000000.
Node 4: LP-obj -141726;  Heuristic value -139618
--- Global Log. Node      5. Node depth   3. Best Bound -142019.000000.
Node 5: Not applying heuristic (status is Cutoff in dual)
--- Global Log. Node      6. Node depth   5. Best Bound -142019.000000.
Node 6: LP-obj -141707;  Heuristic value -138168
--- Global Log. Node      7. Node depth   6. Best Bound -142019.000000.
Node 7: LP-obj -141655;  Heuristic value -138068
--- Global Log. Node      8. Node depth   5. Best Bound -142019.000000.
Node 8: Not applying heuristic (status is Cutoff in dual)
--- Global Log. Node      9. Node depth   6. Best Bound -142019.000000.
Node 9: Not applying heuristic (status is Infeasible)
--- Global Log. Node     10. Node depth   6. Best Bound -142019.000000.
Node 10: Not applying heuristic (status is Cutoff in dual)
--- Global Log. Node     11. Node depth   2. Best Bound -142019.000000.
Node 11: LP-obj -141465;  Heuristic value -139508
--- Global Log. Node     12. Node depth   3. Best Bound -142019.000000.
Node 12: LP-obj -141062;  Heuristic value -139933
--- Global Log. Node     13. Node depth   3. Best Bound -142019.000000.
Node 13: Not applying heuristic (status is Infeasible)
--- Global Log. Node     14. Node depth   3. Best Bound -142019.000000.
Node 14: Not applying heuristic (status is Cutoff in dual)
--- Global Log. Node     15. Node depth   3. Best Bound -141896.953125.
Node 15: LP-obj -141721;  Heuristic value -141278 *** Updated cutoff to -141277
--- Global Log. Node     16. Node depth   4. Best Bound -141896.953125.
Node 16: LP-obj -141694;  Heuristic value -141168
--- Global Log. Node     17. Node depth   5. Best Bound -141896.953125.
Node 17: LP-obj -141640;  Heuristic value -141168
--- Global Log. Node     18. Node depth   6. Best Bound -141896.953125.
Node 18: LP-obj -141572;  Heuristic value -141278 *** Updated cutoff to -141277
--- Global Log. Node     19. Node depth   7. Best Bound -141896.953125.
Node 19: LP-obj -141360;  Heuristic value -141258
--- Global Log. Node     20. Node depth   7. Best Bound -141896.953125.
Node 20: LP-obj -141565;  Heuristic value -141278 *** Updated cutoff to -141277
--- Global Log. Node     21. Node depth   8. Best Bound -141896.953125.
Node 21: LP-obj -141349;  Heuristic value -141247
--- Global Log. Node     22. Node depth   8. Best Bound -141896.953125.
Node 22: LP-obj -141498;  Heuristic value -141278 *** Updated cutoff to -141277
--- Global Log. Node     23. Node depth   8. Best Bound -141896.953125.
Node 23: Not applying heuristic (status is Infeasible)
--- Global Log. Node     24. Node depth   8. Best Bound -141896.953125. Best MIP f(x) -141278.000000
Node 24: B&B integer solution found; objval -141278
--- Global Log. Node     25. Node depth   8. Best Bound -141726.000000. Best MIP f(x) -141278.000000
Node 25: B&B integer solution found; objval -141056
```

```
--- Global Log. Node     26. Node depth   4. Best Bound -141720.515625. Best MIP f(x) -141278.000000
Node 26: B&B integer solution found; objval -140695
--- Global Log. Node     27. Node depth   3. Best Bound -141693.593750. Best MIP f(x) -141278.000000
Node 27: B&B integer solution found; objval -141524
--- Global Log. Node     28. Node depth   4. Best Bound -141640.000000. Best MIP f(x) -141278.000000
Node 28: B&B integer solution found; objval -140974
--- Global Log. Node     29. Node depth   3. Best Bound -141465.140625. Best MIP f(x) -141278.000000
Node 29: LP-obj -141341;  Heuristic value -139398
--- Global Log. Node     30. Node depth   3. Best Bound -141465.140625. Best MIP f(x) -141278.000000
Node 30: Not applying heuristic (status is Cutoff in dual)
--- Global Log. Node     31. Node depth   3. Best Bound -141465.140625. Best MIP f(x) -141278.000000
Node 31: Not applying heuristic (status is Infeasible)
--- Global Log. Node     32. Node depth   3. Best Bound -141360.000000. Best MIP f(x) -141278.000000
Node 32: Not applying heuristic (status is Cutoff in dual)
--- Global Log. Node     33. Node depth   7. Best Bound -141360.000000. Best MIP f(x) -141278.000000
Node 33: Not applying heuristic (status is Infeasible)
--- Global Log. Node     34. Node depth   7. Best Bound -141349.000000. Best MIP f(x) -141278.000000
Node 34: Not applying heuristic (status is Cutoff in dual)
--- Global Log. Node     35. Node depth   8. Best Bound -141349.000000. Best MIP f(x) -141278.000000
Node 35: Not applying heuristic (status is Infeasible)

===== * * * ================================================= * * *
TOMLAB SOL - Three weeks demonstration single user license
=================================================================
Problem: mip_prob -  7: Weingartner 1 - 2/28 0-1 knapsack  f_k -141278.000000000000000000
                                              User given f(x_*)-141278.000000000000000000

Solver: Xpress-MP.  EXIT=0.  INFORM=6.
Mex-interface to Xpress-MP LP/QP/MIP/MIQP solver
Global search complete - integer solution found

FuncEv   35 GradEv    0 Iter   35
CPU time: 3.156000 sec. Elapsed time: 3.156000 sec.
```

The following example shows how to define and solve a problem using the TOMLAB Format (TQ). The matrices and vectors for the problem is defined in a *mat* file. Fields that changes the Xpress^{MP} control variables are set to show how to influence the work of the solver. In this case the changes slow down its performance.

```
clear all
load bilp1.mat
whos
  Name          Size         Bytes  Class

  A            27x1956      422496  double array
  b_L          27x1            216  double array
  b_U          27x1            216  double array
  c          1956x1          15648  double array
  noivars    1956x1                15648  double
  x_L        1956x1          15648  double array
  x_U        1956x1          15648  double array
```

```
Grand total is 58735 elements using 469880 bytes

Prob = mipAssign(c, A, b_L, b_U, x_L, x_U,[],'bilp1',[],[],noivars);
Result = tomRun('xpress-mp',Prob,1);

===== * * * =================================================== * * *
TOMLAB SOL - Three weeks demonstration single user license
=================================================================
Problem: No Init File    -  1: bilp1            f_k       0.000000000000000000
                                       sum(|constr|)      0.000000000000025960

Solver: Xpress-MP.  EXIT=0.  INFORM=6.
Mex-interface to Xpress-MP LP/QP/MIP/MIQP solver
Global search complete - integer solution found

FuncEv   67 GradEv    0 Iter   67
CPU time: 4.359000 sec. Elapsed time: 4.375000 sec.

=== * * * ===================================================== * * *

Prob.MIP
ans =
      IntVars: 1956
    VarWeight: []
     KNAPSACK: []
          fIP: []
          xIP: []
           PI: []
           SC: []
           SI: []
         sos1: []
         sos2: []

% Make Xpress-MP work slower by disabling presolve and cuts.
Prob.MIP.xpcontrol.CUTSTRATEGY = 0;
Prob.MIP.xpcontrol.MIPPRESOLVE = 0;
Prob.MIP.xpcontrol.PRESOLVE = 0;
Result = tomRun('xpress-mp',Prob,2);

===== * * * =================================================== * * *
TOMLAB SOL - Three weeks demonstration single user license
=================================================================
Problem: No Init File    -  1: bilp1            f_k       0.000000000000000000
                                       sum(|constr|)      0.000000000000020365

Solver: Xpress-MP.  EXIT=0.  INFORM=6.
Mex-interface to Xpress-MP LP/QP/MIP/MIQP solver
Global search complete - integer solution found
```

```
FuncEv  105 GradEv    0 Iter  105
CPU time: 0.578000 sec. Elapsed time: 0.578000 sec.

=== * * * ================================================== * * *

Prob.MIP.xpcontrol.MIPPRESOLVE = 7; % Try another MIPPRESOLVE value
Result = tomRun('xpress-mp', Prob, 1);

===== * * * ================================================ * * *
TOMLAB SOL - Three weeks demonstration single user license
=================================================================
Problem: No Init File    -  1: bilp1            f_k       0.000000000000000000
                                     sum(|constr|)      0.000000000000011373

Solver: Xpress-MP.  EXIT=0.  INFORM=6.
Mex-interface to Xpress-MP LP/QP/MIP/MIQP solver
Global search complete - integer solution found

FuncEv   82 GradEv    0 Iter   82
CPU time: 3.250000 sec. Elapsed time: 3.250000 sec.
```

7 xpControl - Control Parameters

Description

All parameters not specified by the user are automatically set to their default values. The following table lists all parameters that the user can specify before calling the solver.

After solver execution a global variable *xpControlVariables* will contain all the settings.

The following optional inputs can be used to control solver execution:

Symbol	Default	Description
AUTOPERTURB	1	Simplex: This indicates whether automatic perturbation is performed. If this is set to 1, the problem will be perturbed by the amount PERTURB whenever the simplex method encounters an excessive number of degenerate pivot steps, thus preventing the Optimizer being hindered by degeneracies. 0 = No perturbation performed. 1 = Automatic perturbation is performed.
BACKTRACK	3	Branch and Bound: This determines how the next node in the tree search is selected for processing. 1 = If MIPTARGET is not set, choose the node with the best estimate. If MIPTARGET is set (by the user or by the global search previously finding an integer solution), the choice is based on the Forrest-Hirst-Tomlin Criterion, which takes into account the best current integer solution and seeks a new node which represents a large potential improvement. 2 = Always choose the node with the best estimated solution. 3 = Always choose the node with the best bound on the solution.
BARDUALSTOP	1.0E-08	Newton barrier: This is a convergence parameter, representing the tolerance for dual infeasibilities. If the difference between the constraints and their bounds in the dual problem falls below this tolerance in absolute value, optimization will stop and the current solution will be returned.
BARGAPSTOP	1.0E-08	Newton barrier: This is a convergence parameter, representing the tolerance for the relative duality gap. When the difference between the primal and dual objective function values falls below this tolerance, the Optimizer determines that the optimal solution has been found.

Symbol	Default	Description
BARINDEFLIMIT	15	Newton Barrier. This limits the number of consecutive indefinite barrier iterations that will be performed. The optimizer will try to minimize (maximise) a QP problem even if the Q matrix is not positive (negative) semi-definite. However, the optimizer may detect that the Q matrix is indefinite and this can result in the optimizer not converging. If more that BARINDEFLIMIT indefinite iterations occur then the optimizer will stop.
BARITERLIMIT	200	Newton barrier: The maximum number of iterations. While the simplex method usually performs a number of iterations which is proportional to the number of constraints (rows) in a problem, the barrier method standardly finds the optimal solution to a given accuracy after a number of iterations which is independent of the problem size. The penalty is rather that the time for each iteration increases with the size of the problem. BARITERLIMIT specifies the maximum number of iterations which will be carried out by the barrier.
BARMEMORY	0	Newton barrier: This specifies the amount of memory in megabytes to be used by the barrier algorithm in its search for the optimal solution. If set to 0, this is determined automatically by the Optimizer.
BARORDER	0	Newton barrier: This specifies the ordering algorithm for the Cholesky factorization, used to preserve the sparsity of the factorized matrix. 0 = Choose automatically. 1 = Minimum degree method. This selects diagonal elements with the smallest number of nonzeros in their rows or columns. 2 = Minimum local fill method. This considers the adjacency graph of nonzeros in the matrix and seeks to eliminate nodes that minimize the creation of new edges. 3 = Nested dissection method. This considers the adjacency graph and recursively seeks to separate it into non-adjacent pieces.
BAROUTPUT	1	Newton barrier: This specifies the level of solution output provided. Output is provided either after each iteration of the algorithm, or else can be turned off completely by this parameter. 0 = No output. 1 = At each iteration.

Symbol	Default	Description
BARPRIMALSTOP	1.0E-08	Newton barrier: This is a convergence parameter, indicating the tolerance for primal infeasibilities. If the difference between the constraints and their bounds in the primal problem falls below this tolerance in absolute value, the Optimizer will terminate and return the current solution.
BARSTEPSTOP	1.0E-10	Newton barrier: A convergence parameter, representing the minimal step size. On each iteration of the barrier algorithm, a step is taken along a computed search direction. If that step size is smaller than BARSTEPSTOP, the Optimizer will terminate and return the current solution. **Note:** If the barrier method is making small improvements on BARGAPSTOP on later iterations, it may be better to set this value higher, to return a solution after a close approximation to the optimum has been found.
BARTHREADS	1	Newton barrier: The number of threads implemented to run the algorithm. This is usually set to the number of processors when running Parallel Xpress-MP on a single multi-processor machine. **Note:** The value of BARTHREADS depends on the user's authorization. If it is set to a value higher than that specified by the licence, then it will be reset by the Optimizer immediately prior to optimization. Obtaining its value after the optimization will give an indication of how many processors were actually used.
BIGM	N/A	The infeasibility penalty used if the "Big M" method is implemented. The default value is dependent on the matrix characteristics.
BIGMMETHOD	1	Simplex: This specifies whether to use the "Big M" method, or the standard phase I (achieving feasibility) and phase II (achieving optimality). In the "Big M" method, the objective coefficients of the variables are considered during the feasibility phase, possibly leading to an initial feasible basis which is closer to optimal. The side-effects involve possible round-off errors due to the presence of the "Big M" factor in the problem. 0 = For phase I / phase II. 1 = If "Big M" method to be used.
BRANCHCHOICE	0	Once a global entity has been selected for branching, this control determines whether the branch with the minimum of maximum estimate is followed first. 0 = Minimum estimate branch first.

Symbol	Default	Description
		1 = Maximum estimate branch first.
BREADTHFIRST	10	The number of nodes to include in the best-first search before switching to the local first search (NODESELECTION = 4).
CACHESIZE	N/A	Newton barrier: cache size in Kbytes on the user's computer. On Intel platforms -1 may be used to determine the cache size automatically. Default value is hardware/platform dependent. **Note:** If the size is unknown, it is better to choose a smaller size. If the size cannot be determined automatically under Windows, a default size of 512 kB is assumed.
CHOLESKYALG	1	Newton barrier: type of Cholesky factorization used. 0 = Pull Cholesky. 1 = Push Cholesky.
CHOLESKYTOL	1.0E-15	Newton barrier: The zero tolerance for pivot elements in the Cholesky decomposition of the normal equations coefficient matrix, computed at each iteration of the barrier algorithm. If the absolute value of the pivot element is less than or equal to CHOLESKYTOL, it merits special treatment in the Cholesky decomposition process.
COVERCUTS	N/A	Branch and Bound: The number of rounds of lifted cover inequalities at the top node. A lifted cover inequality is an additional constraint that can be particularly effective at reducing the size of the feasible region without removing potential integral solutions. The process of generating these can be carried out a number of times, further reducing the feasible region, albeit incurring a time penalty. There is usually a good payoff from generating these at the top node, since these inequalities then apply to every subsequent node in the tree search. Default is determined automatically.
CPKEEPALLCUTS	1	Cut pool: This indicates whether inactive user generated cuts should be deleted from the cut pool. Doing so will save memory, albeit at the expense of solution time if the cuts have to be generated again subsequently. 0 = Do not delete inactive cuts. 1 = Delete inactive cuts.
CPMAXCUTS	100	Cut pool: The initial maximum number of cuts that will be stored in the cut pool. During optimization, the cut pool is subsequently resized automatically.

Symbol	Default	Description
CPMAXELEMS	200	Cut pool: The initial maximum number of nonzero coefficients which will be held in the cut pool. During optimization, the cut pool is subsequently resized automatically.
CPUTIME	1	Which time to be used in reporting solution times. 0 = If elapsed time is to be used. 1 = If CPU time is to be used.
CRASH	2	Simplex: This determines the type of crash used when the algorithm begins. During the crash procedure, an initial basis is determined which is as close to feasibility and triangularity as possible. A good choice at this stage will significantly reduce the number of iterations required to find an optimal solution. The possible values increase proportionally to their time-consumption. 0 = Turns off all crash procedures. 1 = For singletons only (one pass). 2 = For singletons only (multi pass). 3 = Multiple passes through the matrix considering slacks. 4 = Multiple (<= 10) passes through the matrix but only doing slacks at the very end. n ¿ 10. As for value 4 but performing at most n - 10 passes.
CROSSOVER	1	Newton barrier: This control determines whether the barrier method will cross over to the simplex method when at optimal solution has been found, to provide an end basis and advanced sensitivity analysis information. 0 = No crossover. 1 = Crossover to a basic solution. **Note:** The full primal and dual solution is available whether or not crossover is used.
CSTYLE	1	Convention used for numbering arrays. 0 = Indicates that the FORTRAN convention should be used for arrays (i.e. starting from 1). 1 = Indicates that the C convention should be used for arrays (i.e. starting from 0).

Symbol	Default	Description
CUTDEPTH	N/A	Branch and Bound: Sets the maximum depth in the tree search at which cuts will be generated. Generating cuts can take a lot of time, and is often less important at deeper levels of the tree since tighter bounds on the variables have already reduced the feasible region. A value of 0 signifies that no cuts will be generated. Default value is determined automatically.
CUTFREQ	N/A	Branch and Bound: This specifies the frequency at which cuts are generated in the tree search. If the depth of the node modulo CUTFREQ is zero, then cuts will be generated. Default value is determined automatically.
CUTSTRATEGY	-1	Branch and Bound: This specifies the cut strategy. A more aggressive cut strategy, generating a greater number of cuts, will result in fewer nodes to be explored, but with an associated time cost in generating the cuts. The fewer cuts generated, the less time taken, but the greater subsequent number of nodes to be explored. -1 = Automatic selection of the cut strategy. 0 = No cuts. 1 = Conservative cut strategy. 2 = Moderate cut strategy. 3 = Aggressive cut strategy.
DEFAULTALG	1	This selects the algorithm that will be used to solve the LP if no algorithm flag is passed to the optimization routines. 1 = Automatically determined. 2 = Dual simplex. 3 = Primal simplex. 4 = Newton barrier.
DEGRADEFACTOR	1.0	Branch and Bound: Factor to multiply estimated degradations associated with an unexplored node in the tree. The estimated degradation is the amount by which the objective function is expected to worsen in an integer solution that may be obtained through exploring a given node.
DENSECOLLIMIT	N/A	Newton barrier: Columns with more than DENSECOLLIMIT elements are considered to be dense. Such columns will be handled specially in the Cholesky factorization of this matrix. Default value is determined automatically.
DUALGRADIENT	-1	This specifies the pricing method for the dual algorithm. -1 = Determine automatically.

Symbol	Default	Description
		0 = Devex. 1 = Steepest edge.
ELIMTOL	1.0E-03	The Markowitz tolerance for the elimination phase of the presolve.
ETATOL	1.0E-12	Zero tolerance on eta elements. During each iteration, the basis inverse is premultiplied by an elementary matrix, which is the identity except for one column - the eta vector. Elements of eta vectors whose absolute value is smaller than ETATOL are taken to be zero in this step.
EXTRACOLS	0	The initial number of extra columns to allow for in the matrix. If columns are to be added to the matrix, then, for maximum efficiency, space should be reserved for the columns before the matrix is input by setting the EXTRACOLS control. If this is not done, resizing will occur automatically, but more space may be allocated than the user actually requires.
EXTRAELEMS	N/A	The initial number of extra matrix elements to allow for in the matrix, including coefficients for cuts. If rows or columns are to be added to the matrix, then, for maximum efficiency, space should be reserved for the extra matrix elements before the matrix is input by setting the EXTRAELEMS control. If this is not done, resizing will occur automatically, but more space may be allocated than the user actually requires. The space allowed for cut coefficients is equal to the number of extra matrix elements remaining after rows and columns have been added but before the global optimisation starts. EXTRAELEMS is set automatically by the optimiser when the matrix is first input to allow space for cuts, but if you add rows or columns, this automatic setting will not be updated. So if you wish cuts, either automatic cuts or user cuts, to be added to the matrix and you are adding rows or columns, EXTRAELEMS must be set before the matrix is first input, to allow space both for the cuts and any extra rows or columns that you wish to add. Default is hardware/platform dependent.
EXTRAMIPENTS	0	The initial number of extra global entities to allow for.
EXTRAPRESOLVE	N/A	The initial number of extra elements to allow for in the presolve. Default is hardware/platform dependent. **Note:** The space required to store extra presolve elements is allocated dynamically, so it is not necessary to set this control.

Symbol	Default	Description
EXTRAROWS	N/A	The initial number of extra rows to allow for in the matrix, including cuts. If rows are to be added to the matrix, then, for maximum efficiency, space should be reserved for the rows before the matrix is input by setting the EXTRAROWS control. If this is not done, resizing will occur automatically, but more space may be allocated than the user actually requires. The space allowed for cuts is equal to the number of extra rows remaining after rows have been added but before the global optimisation starts. EXTRAROWS is set automatically by the optimiser when the matrix is first input to allow space for cuts, but if you add rows, this automatic setting will not be updated. So if you wish cuts, either automatic cuts or user cuts, to be added to the matrix and you are adding rows, EXTRAROWS must be set before the matrix is first input, to allow space both for the cuts and any extra rows that you wish to add. Default value depends on the matrix characteristics.
FEASTOL	1.0E-06	This is the zero tolerance on right hand side values, bounds and range values, i.e. the bounds of basic variables. If one of these is less than or equal to FEASTOL in absolute value, it is treated as zero.
GOMCUTS	N/A	Branch and Bound: The number of rounds of Gomory cuts at the top node. These can always be generated if the current node does not yield an integral solution. However, Gomory cuts are not usually as effective as lifted cover inequalities in reducing the size of the feasible region. Default determined automatically.
HEURDEPTH	0	Branch and Bound: Sets the maximum depth in the tree search at which heuristics will be used to find MIP solutions. It may be worth stopping the heuristic search for solutions after a certain depth in the tree search. A value of 0 signifies that heuristics will not be used.
HEURFREQ	5	Branch and Bound: This specifies the frequency at which heuristics are used in the tree search. Heuristics will only be used at a node if the depth of the node is a multiple of HEURFREQ.
HEURMAXNODES	1000	Branch and Bound: This specifies the maximum number of nodes at which heuristics are used in the tree search.
HEURMAXSOL	10	Branch and Bound: This specifies the maximum number of heuristic solutions that will be found in the tree search.
HEURSTRATEGY	-1	Branch and Bound: This specifies the heuristic strategy. -1 = Automatic selection of heuristic strategy. 0 = No heuristics.

Symbol	Default	Description
		1 = Rounding heuristics.
INVERTFREQ	N/A	Simplex: The frequency with which the basis will be inverted. The basis is maintained in a factorized form and on most simplex iterations it is incrementally updated to reflect the step just taken. This is considerably faster than computing the full inverted matrix at each iteration, although after a number of iterations the basis becomes less well-conditioned and it becomes necessary to compute the full inverted matrix. The value of INVERTFREQ specifies the maximum number of iterations between full inversions. The default frequency is determined automatically.
INVERTMIN	3	Simplex: The minimum number of iterations between full inversions of the basis matrix. See the description of INVERTFREQ for details.
KEEPBASIS	1	Simplex: This determines which basis to use for the next iteration. The choice is between using that determined by the crash procedure at the first iteration, or using the basis from the last iteration. 0 = Problem optimization starts from the first iteration, i.e. the previous basis is ignored. 1 = The previously loaded basis (last in memory) should be used.
KEEPMIPSOL	1	Branch and Bound: The number of integer solutions to keep. During a global search, typically any number of integer solutions may be found, which may or may not represent optimal solutions. The value of KEEPMIPSOL represents the number of integer solutions which will be stored. Goal Programming: The number of goal programming solutions to keep in the pre-emptive case. Pre-emptive goal programming solves a sequence of problems giving a sequence of partial solutions. The value of KEEPMIPSOL represents the number of partial solutions to keep. By default only the best solution is kept.
KEEPNROWS	1	Status for nonbinding rows. -1 = Delete N type rows and make space available as spare rows. 0 = Delete N type rows. 1 = Keep N type rows.
LNPBEST	50	Number of infeasible global entities to create lift-and-project cuts for during each round of Gomory cuts at the top node (see GOMCUTS).

Symbol	Default	Description
LNPITERLIMIT	10	Number of iterations to perform in improving each lift-and-project cut. **Note:** By setting the number to zero a Gomory cut will be created instead.
LPITERLIMIT	2147483645	Simplex: The maximum number of iterations that will be performed before the optimization process terminates. For MIP problems, this is the maximum number of iterations at each node explored by the Branch and Bound method.
LPLOG	100	Simplex: The frequency at which the simplex log is printed. $n < 0$ = Detailed output every -n iterations. 0 = Log displayed at the end of the optimization only. $n > 0$ = Summary output every n iterations.
MARKOWITZTOL	0.01	The Markowitz tolerance used for the factorization of the basis matrix.
MATRIXTOL	1.0E-09	The zero tolerance on matrix elements. If the value of a matrix element is less than or equal to MATRIXTOL in absolute value, it is treated as zero.
MAXCUTTIME	0	The maximum amount of time allowed for generation of cutting planes and re-optimization. The limit is checked during generation and no further cuts are added once this limit has been exceeded. 0 = No time limit. $n > 0$ = Stop cut generation after n seconds.
MAXIIS	1	This controls the number of Irreducible Infeasible Sets to be found. -1 = Search for each of the IIS. 0 = Search for none. $n > 0$ = Search for the first n IIS.
MAXMIPSOL	0	Branch and Bound: This specifies a limit on the number of integer solutions to be found by the Optimizer before it pauses and asks whether or not to continue. It is possible that during optimization the Optimizer will find the same objective solution from different nodes. However, MAXMIPSOL refers to the total number of integer solutions found, and not necessarily the number of distinct solutions.

Symbol	Default	Description
MAXNODE	100000000	Branch and Bound: The maximum number of nodes that will be explored before the Optimizer pauses and asks whether or not to continue.
MAXPAGELINES	23	Number of lines between page breaks in printable output.
MAXSLAVE	0	Number of worker processes to use in the parallel MIP optimization. **Note:** Set this to the number of processors available to solve the MIP problem.
MAXTIME	0	The maximum time in seconds that the Optimizer will run before it terminates, including the problem setup time and solution time. For MIP problems, this is the total time taken to solve all the nodes. 0 = No time limit. $n > 0$ = If an integer solution has been found, stop MIP search after n seconds, otherwise continue until an integer solution is finally found. $n < 0$ = Stop in LP or MIP search after -n seconds.
MIPABSCUTOFF	1.0E+40	Branch and Bound: If the user knows that they are interested only in values of the objective function which are better than some value, this can be assigned to MIPABSCUTOFF. This allows the Optimizer to ignore solving any nodes which may yield worse objective values, saving solution time. It is set automatically after an LP Optimizer command, unless it was previously set by the user. The cutoff may be updated automatically whenever a MIP solution is found using the MIPRELCUTOFF and MIPADDCUTOFF controls. The default is for minimization problems. Default has negative sign if maximization.
MIPABSSTOP	0.0	Branch and Bound: The absolute tolerance determining whether the global search will continue or not. It will terminate if abs(MIPOBJVAL - BESTBOUND) <= MIPABSSTOP, where MIPOBJVAL is the value of the best solution's objective function, and BESTBOUND is the current best solution bound. For example, to stop the global search when a MIP solution has been found and the Optimizer can guarantee it is within 100 of the optimial solution, set MIPABSSTOP to 100.

Symbol	Default	Description
MIPADDCUTOFF	-1.0E-05	Branch and Bound: The amount to add to the objective function of the best integer solution found to give the new cutoff. Once an integer solution has been found whose objective function is equal to or better than MIPABSCUTOFF, improvements on this value may not be interesting unless they are better by at least a certain amount. If MIPADDCUTOFF is nonzero, it will be added to MIPABSCUTOFF each time an integer solution is found which is better than this new value. This cuts off sections of the tree whose solutions would not represent substantial improvements in the objective function, saving processor time. The control MIPABSSTOP provides a similar function but works in a different way.
MIPLOG	-100	Global print control. -n = Print out summary log at each n'th node. 0 = No printout in global. 1 = Only print out summary statement at the end. 2 = Print out detailed log at all solutions found. 3 = Print out detailed log at each node.
MIPPRESOLVE	N/A	Branch and Bound: Type of integer processing to be performed. If set to 0, no processing will be performed. Default value depends on the matrix characteristics. 0 = Reduced cost fixing will be performed at each node. This can simplify the node before it is solved, by deducing that certain variables' values can be fixed based on additional bounds imposed on other variables at this node. 1 = Logical preprocessing will be performed at each node. This is performed on binary variables, often resulting in fixing their values based on the constraints. This greatly simplifies the problem and may even determine optimality or infeasibility of the node before the simplex method commences. 2 = Probing of binary variables is performed at the top node. This sets certain binary variables and then deduces effects on other binary variables occurring in the same constraints.
MIPRELCUTOFF	1.0E-04	Branch and Bound: Percentage of the LP solution value to be added to the value of the objective function when an integer solution is found, to give the new value of MIPABSCUTOFF. The effect is to cut off the search in parts of the tree whose best possible objective function would not be substantially better than the current solution. The control MIPRELSTOP provides a similar functionality but works in a different way.

Symbol	Default	Description
MIPRELSTOP	0.0	Branch and Bound: This determines whether or not the global search will terminate. Essentially it will stop if: abs(MIPOBJVAL - BESTBOUND) <= MIPRELSTOP * BESTBOUND, where MIPOBJVAL is the value of the best solution's objective function and BESTBOUND is the current best solution bound. For example, to stop the global search when a MIP solution has been found and the Optimizer can guarantee it is within 5
MIPTARGET	1.0E+40	Branch and Bound: The target object function for the global search (only used by certain node selection criteria). This is set automatically after an LP optimization routine, unless it was previously set by the user.
MIPTOL	5.0E-06	Branch and Bound: This is the tolerance within which a decision variable's value is considered to be integral.
MPSBOUNDNAME	64 blanks	The bound name sought in the MPS file.
MPSECHO	1	Determines whether comments in MPS matrix files are to be printed out during matrix input. 0 = MPS comments are not to be echoed. 1 = MPS comments are not to be echoed.
MPSERRIGNORE	0	Number of errors to ignore whilst reading an MPS file.
MPSFORMAT	-1	Specifies the format of MPS files. -1 = To determine the file type automatically. 0 = For fixed format. 1 = If MPS files are assumed to be in free format by input.
MPSNAMELENGTH	8 (MAX 64)	Maximum length of MPS names in characters. If reset, this must be before any problem is input. Internally it is rounded up to the smallest multiple of 8. MPS names are right padded with blanks.
MPSOBJNAME	64 blanks	The objective function name sought in the MPS file.
MPSRANGENAME	64 blanks	The range name sought in the MPS file.
MPSRHSNAME	64 blanks	The right hand side name sought in the MPS file.
NODESELECTION	N/A	Minimum number of iterations. Default value depends on the matrix characteristics. 1 = Local first: Choose between descendant and sibling nodes if available; choose from all outstanding nodes otherwise. 2 = Best first: Choose from all outstanding nodes.

Symbol	Default	Description
		3 = Local depth first: Choose between descendant and sibling nodes if available; choose from the deepest nodes otherwise. 4 = Best first, then local first: Best first is used for the first BREADTHFIRST nodes, after which local first is used. 5 = Pure depth first: Choose from the deepest outstanding nodes.
OMNIDATANAME	64 blanks	Data for OMNI data name field.
OPTIMALITYTOL	1.0E-06	Simplex: This is the zero tolerance for reduced costs. On each iteration, the simplex method searches for a variable to enter the basis which has a negative reduced cost. The candidates are only those variables which have reduced costs less than the negative value of OPTIMALITYTOL.
OUTPUTLOG	1	This controls the level of output produced by the Optimizer during optimization. The possible options are to print all messages or to disable printing altogether. 0 = Turn all output off. 1 = Print messages.
OUTPUTMASK	64 '?'s	Mask to restrict the row and column names written to file.
OUTPUTTOL	1.0E-05	Zero tolerance on print values.
PENALTY	N/A	Minimum absolute penalty variable coefficient. Default depends on the matrix characteristics.
PERTURB	0.0	The factor by which the problem will be perturbed prior to optimization if the control AUTOPERTURB has been set to 1. A value of 0.0 results in an automatically determined perturbation value.
PIVOTTOL	1.0E-09	Simplex: The zero tolerance for matrix elements. On each iteration, the simplex method seeks a nonzero matrix element to pivot on. Any element with absolute value less than PIVOTTOL is treated as zero for this purpose.
PPFACTOR	1.0	The partial pricing candidate list sizing parameter.
PRESOLVE	1	This control determines whether presolving should be performed prior to starting the main algorithm. Presolve attempts to simplify the problem by detecting and removing redundant constraints, tightening variable bounds, etc. In some cases, infeasibility may even be determined at this stage, or the optimal solution found.

Symbol	Default	Description
		-1 = Presolve applied, but a problem will not be declared infeasible if primal infeasibilities are detected. The problem will be solved by the LP optimization algorithm, returning an infeasible solution, which can sometimes be helpful. 0 = Presolve not applied. 1 = Presolve applied. 2 = Presolve applied, but redundant bounds are not removed. This can sometimes increase the efficiency of the barrier algorithm. **Note:** Memory for presolve is dynamically resized. If the Optimizer runs out of memory for presolve, an error message (245) is produced.
PRESOLVEOPS	511 (0-8 are set)	This specifies the operations which are performed during the presolve. 0 = singleton column removal. 1 = singleton row removal. 2 = forcing row removal. 3 = dual reductions. 4 = redundant row removal. 5 = duplicate column removal. 6 = duplicate row removal. 7 = strong dual reductions. 8 = variable eliminations. 9 = no IP reductions. 13 = linearly dependant row removal.
PRICINGALG	0	Simplex: This determines the pricing method to use on each iteration, selecting which variable enters the basis. In general Devex pricing requires more time on each iteration, but may reduce the total number of iterations, whereas partial pricing saves time on each iteration, although possibly results in more iterations. -1 = If partial pricing is to be used. 0 = If the pricing is to be decided automatically. 1 = If Devex pricing is to be used.
PSEUDOCOST	0.01	Branch and Bound: The default pseudo cost used in estimation of the degradation associated with an unexplored node in the tree search. A pseudo cost is associated with each integer decision variable and is an estimate of the amount by which the objective function will be worse if that variable is forced to an integral value.
REFACTOR	0 (1)	Indicates whether the optimization should restart using the current representation of the factorization in memory. Default is 1 for reoptimizing.

Symbol	Default	Description
		0 = Do not refactor on reoptimizing. 1 = Refactor on reoptimizing. **Note:** In the tree search, the optimal bases at the nodes are not refactorized by default, but the optimal basis for an LP problem will be refactorized. If you are repeatedly solving LPs with few changes then it is more efficient to set REFACTOR to 0.
REL10STYLE	0	Determines whether the old style convention should be used for dual values, slacks and reduced costs. 0 = Use standard convention for solution values. 1 = Use convention in Release 10 and earlier for solution values.
RELPIVOTTOL	1.0E-06	Simplex: At each iteration a pivot element is chosen within a given column of the matrix. The relative pivot tolerance, RELPIVOTTOL, is the size of the element chosen relative to the largest possible pivot element in the same column.
SBBEST	N/A	Number of infeasible global entities on which to perform strong branching. Default determined automatically.
SBITERLIMIT	N/A	Number of dual iterations to perform the strong branching. Default determined automatically.
SBSELECT	N/A	The size of the candidate list of global entities for strong branching. Default determined automatically. **Note:** Before strong branching is applied on a node of the branch and bound tree, a list of candidates is selected among the infeasible global entities. These entities are then evaluated based on the local LP solution and prioritised. Strong branching will then be applied to the SBBEST candidates. The evaluation is potentially expensive and for some problems it might improve performance if the size of the candidate list is reduced.
SCALING	35	This determines how the Optimizer will rescale a model internally before optimization. If set to 0, no scaling will take place.
	Bit	Meaning
	0	Row scaling.
	1	Column scaling.
	2	Row scaling again.
	3	Maximum.
	4	Curtis-Reid.
	5	0 = scale by geometric mean., 1 = scale by maximum element.

Symbol	Default	Description
SOLUTIONFILE	1	Determines whether the binary solution file (problem_name.sol) is used to store optimal solutions. The Optimizer always stores the final LP solution in memory. Depending on the value of SOLUTIONFILE, the Optimizer may also store the final LP solution, or, in the case of a MIP, the best known MIP solution to the binary solution file. Sometimes it is advantageous to disable use of the solution file, where file access is inconvenient or incurs a performance overhead. However, certain functions that use the solution obtain it from the binary solution file, and their behaviour is affected by this control. 0 = The binary solution file is not used. If solving a MIP, MIP solutions are not stored anywhere by the Optimizer. If required, they must be stored by the user in the user's own memory structures. This can be achieved by setting an integer solution callback function using XPRSsetcbintsol, which will be called whenever a MIP solution is found. Functions which require the binary solution file will not work and will report an error. All other functions will access the LP solution stored in memory rather than the binary solution file. If a MIP problem is being solved, and a function is called that accesses the solution, this means the LP solution to the last branch and bound node (linear relaxation) solved will be used, and not the best known MIP solution. 1 = The binary solution file is used to store the final LP solution, or, if a MIP solution has been found, the best known MIP solution.
SOSREFTOL	1.0E-03	The minimum gap between the ordering values of elements in a special ordered set.
TRACE	0	Control of the infeasibility diagnosis during presolve - if nonzero, infeasibility will be explained.
TREECOVERCUTS	1	Branch and Bound: The number of rounds of lifted cover inequalities generated at nodes other than the top node in the tree. Compare with the description for COVERCUTS.
TREEGOMCUTS	1	Branch and Bound: The number of rounds of Gomory cuts generated at nodes other than the first node in the tree. Compare with the description for GOMCUTS.

Symbol	Default	Description
VARSELECTION	-1	Branch and Bound: This determines the formula used to calculate the estimate of each integer variable, and thus which integer variable is selected to be branched on at a given node. The variable selected to be branched on is the one with the minimum estimate. The variable estimates are also combined to calculate the overall estimate of the node, which, depending on the BACKTRACK setting, may be used to choose between outstanding nodes. -1 = Determined automatically. 1 = The minimum of the 'up' and 'down' pseudo costs. 2 = The 'up' pseudo cost plus the 'down' pseudo cost. 3 = The maximum the 'up' and 'down' pseudo costs, plus twice the minimum of the 'up' and 'down' pseudo costs. 4 = The maximum of the 'up' and 'down' pseudo costs. 5 = The 'down' pseudo cost. 6 = The 'up' pseudo cost.

8 xpProblemAttrib - Problem Attributes

Description

During the optimization process, various properties of the problem being solved are stored and made available to users of TOMLAB /Xpress in the form of problem attributes. These can be accessed in the global structure xpProblemAttrib. A full list of the attributes available and their types may be found in this Section.

The following problem attributes are available after optimization:

Symbol	Description
ACTIVENODES	Number of outstanding nodes.
BARAASIZE	Number of nonzeros in AA^T.
BARCROSSOVER	Indicates whether or not the basis crossover phase has been entered. 0 = The crossover phase has not been entered. 1 = The crossover phase has been entered.
BARDENSECOL	Number of dense columns found in the matrix.
BARDUALINF	Sum of the dual infeasibilities for the Newton barrier algorithm.
BARDUALOBJ	Dual objective value calculated by the Newton barrier algorithm.
BARITER	Number of Newton barrier iterations.
BARLSIZE	Number of nonzeros in L resulting from the Cholesky factorization.
BARPRIMALINF	Sum of the primal infeasibilities for the Newton barrier algorithm.
BARPRIMALOBJ	Primal objective value calculated by the Newton barrier algorithm.
BARSTOP	Convergence criterion for the Newton barrier algorithm.
BESTBOUND	Value of the best bound determined so far by the global search.
BOUNDNAME	Active bound name.
BRANCHVALUE	The value of the branching variable at a node of the Branch and Bound tree.
COLS	Number of columns (i.e. variables) in the matrix. **Note:** If the matrix is in a presolved state, this attribute returns the number of columns in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.

Symbol	Description
CUTS	Number of cuts being added to the matrix.
DUALINFEAS	Number of dual infeasibilities. **Note:** If the matrix is in a presolved state, this attribute returns the number of dual infeasibilities in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
ELEMS	Number of matrix nonzeros (elements). If the matrix is in a presolved state, this attribute returns the number of matrix nonzeros in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
ERRORCODE	The most recent Optimizer error number that occurred. This is useful to determine the precise error or warning that has occurred, after an Optimizer function has signalled an error by returning a non-zero value. The return value itself is not the error number. Refer to Xpress manual available from http://tomopt.com is an error should occur.
IIS	Number of IIS found.
LPOBJVAL	Value of the objective function of the last LP solved.
LPSTATUS	LP solution status. 1 = Optimal. 2 = Infeasible. 3 = Objective worse than cutoff. 4 = Unfinished. 5 = Unbounded. 6 = Cutoff in dual.
MATRIXNAME	The matrix name. **Note:** This is the name read from the MATRIX field in an MPS matrix, and is not related to the problem name used in the Optimizer.
MIPENTS	Number of global entities (i.e. binary, integer, semi-continuous, partial integer, and semi-continuous integer variables) but excluding the number of special ordered sets. **Note:** If the matrix is in a presolved state, this attribute returns the number of global entities in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
MIPINFEAS	Number of integer infeasibilities at the current node.

Symbol	Description
MIPOBJVAL	Objective function value of the best integer solution found.
MIPSOLNODE	Node at which the last integer feasible solution was found.
MIPSOLS	Number of integer solutions that have been found.
MIPSTATUS	Global (MIP) solution status. 1 = Problem has not been loaded. 2 = LP has not been optimized. 3 = LP has been optimized. Once the MIP optimization proper has begun, only the following four status codes will be returned. 4 = Global search incomplete - no integer solution found. 5 = Global search incomplete - an integer solution has been found. 6 = Global search complete - no integer solution found. 7 = Global search complete - integer solution found. **Note:** If a 3 status code is returned, it implies that the optimization halted during or directly after the LP optimization - for instance, if the LP relaxation is infeasible or unbounded. In this case please check the value of LP solution status using LPSTATUS.
NAMELENGTH	The length (in 8 character units) of row and column names in the matrix. To allocate a character array to store names, you must allow 8*NAMELENGTH+1 characters per name (the +1 allows for the string terminator character).
NODEDEPTH	Depth of the current node.
NODES	Number of nodes solved so far in the global search. The node numbers start at 1 for the first (top) node in the Branch and Bound tree. Nodes are numbered consecutively.
OBJFIXED	Contribution to the objective function from artificial (fixed) variables. **Note:** If the matrix is in a presolved state, this attribute returns the contribution to the objective from the artificial variables in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
OBJNAME	Active objective function row name.
OBJRHS	Fixed part of the objective function. **Note:** If the matrix is in a presolved state, this attribute returns the fixed part of the objective in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
OBJSENSE	Sense of the optimization being performed.

Symbol	Description
	1.0 = For minimization problems. -1.0 = For maximization problems.
PARENTNODE	The parent node of the current node in the tree search.
PRESOLVESTATE	Problem status as a bit map. 0 = Problem has been loaded. 1 = Problem has been LP presolved. 2 = Problem has been MIP presolved. 7 = Solution in memory is valid.
PRIMALINFEAS	Number of primal infeasibilities. **Note:** If the matrix is in a presolved state, this attribute returns the number of primal infeasibilities in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
QELEMS	Number of quadratic elements in the matrix. If the matrix is in a presolved state, this attribute returns the number of quadratic elements in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
RANGENAME	Active range name.
RHSNAME	Active right hand side name.
ROWS	Number of rows (i.e. constraints) in the matrix. **Note:** If the matrix is in a presolved state, this attribute returns the number of rows in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
SIMPLEXITER	Number of simplex iterations performed.
SETMEMBERS	Number of variables within special ordered sets (set members) in the matrix. **Note:** If the matrix is in a presolved state, this attribute returns the number of variables within special ordered sets in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
SETS	Number of special ordered sets in the matrix.

Symbol	Description
	Note: If the matrix is in a presolved state, this attribute returns the number of special ordered sets in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.
SPARECOLS	Number of spare columns in the matrix.
SPAREELEMS	Number of spare matrix elements in the matrix.
SPAREMIPENTS	Number of spare global entities in the matrix.
SPAREROWS	Number of spare rows in the matrix.
SUMPRIMALINF	Scaled sum of primal infeasibilities. If the matrix is in a presolved state, this attribute returns the scaled sum of primal infeasibilities in the **presolved** matrix. If you require the value for the original matrix, make sure you obtain the value when the matrix is not presolved. The PRESOLVESTATE attribute can be used to test if the matrix is presolved or not. See also Working with Presolve.

9 Return Codes

Description

The following table shows the possible return codes from TOMLAB /Xpress.

Return Code	Description
0	Subroutine completed successfully.
1	Bad input encountered.
2	Bad or corrupt file - unrecoverable.
4	Memory error.
8	Corrupt use.
16	Program error.
32	Invalid call or invalid argument.
128	Sum of the primal infeasibilities for the Newton barrier algorithm.

A The Matlab Interface Routines - Main Routines

A.1 xpress

Purpose

Xpress^{MP} mixed-integer linear and quadratic programming (MILP, MIQP) and linear and quadratic programming (LP, QP) interface. Xpress^{MP} solves problems of the form

$$
\begin{array}{cccccc}
\min\limits_{x} & f(x) = 0.5 * x^T * F * x + c^T * x & & & & \\
s/t & x_L & \leq & x & \leq & x_U \\
 & b_L & \leq & Ax & \leq & b_U \\
 & & & x_i \text{ integer} & & i \in I
\end{array}
$$

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 $1, ..., n$, are restricted to be integers.

Calling Syntax

[x, slack, v, rc, f_k, ninf, sinf, Inform, basis, lpiter, glnodes] = xpress(c, A, x_L, x_U, b_L, b_U, xpcontrol, callback, PriLev, Prob, IntVars, PI, SC, SI, sos1, sos2, F, LogFile, SaveFile, SaveMode, iisRequest, iisFile, saRequest);

Description of Inputs

Problem description structure. The following fields are used:

c Linear objective function cost coefficients, vector $n \times 1$.

F Square dense or sparse matrix. Empty if non-quadratic problem.

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 as 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 $b_U = b_L$ assumed.

xpcontrol Structure, where the fields are set to the Xpress^{MP} control parameters that the user wants to specify values for. The control parameters are listed in Section 7 in the Xpress-Optimizer Reference Manual [1]. The prefix XPRS_ is not used.

callback Logical vector defining which callbacks to use in Xpress^{MP} . If the i^{th} entry of the logical vector *callback* is set, the corresponding callback is defined. See Section 5.3 in [1]. The callback calls the m-file specified in Table 9 below. The user may edit this file, or make a new copy, which is put in a directory that is searched before the *xpress* directory in the Matlab path.

PriLev Printing level in the *xpress* m-file and the Xpress^{MP} C-interface.

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

	= 0 Silent = 1 Summary information = 2 More detailed information
Prob	A structure. If TOMLAB calls *xpress*, 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 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.c*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U* and *Prob.QP.F*. If the input structure is empty ([ ]), then $Prob.P = 1$ is set. If *Prob.MIP.KNAPSACK = 1* and *callback*(9) == 1, then the simple heuristic in *xpcb_GL* is used. If callback(9) is set, and *Prob.MIP.KNAPSACK*, or *Prob.MIP* is undefined, *xpress* is setting *Prob.MIP.KNAPSACK = 0*, to avoid the call to the heuristic.
IntVars	Defines which variables are integers, of the general type I or binary B. Variable indices should be in the range [1,...,n]. If $IntVars$ is a logical vector then all variables i where $IntVars(i) > 0$ are defined to be integers. If $IntVars$ is determined to be a vector of indices then $x(IntVars)$ are defined as integers. If the input is empty ([ ]), then no integers of type I or B are defined. The interface routine *xpress* checks which of the integer variables have lower bound $x_L = 0$ and upper bound $x_U = 1$, i.e. are binary 0/1 variables.
PI	Integer variables of type *Partially Integer* (PI), i.e. takes an integer value up to a specified limit, and any real value above that limit. PI must be a structure array where: $PI.var$ is a vector of variable indices in the range $[1, ..., n]$. $PI.lim$ is a vector of limit values for each of the variables specified in PI.var, i.e. for variable i, that is the PI variable with index j in $PI.var$, then $x(i)$ takes integer values in $[x_L(i), PI.lim(j)]$ and continuous values in $[PI.lim(j), x_U(i)]$.
SC	A vector with indices for the integer variables of type *Semi-continuous* (SC), i.e. that takes either the value 0 or a real value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SC(j)$, where i is an variable number in the range $[1, ..., n]$.
SI	A vector with indices for the integer variables of type *Semi-integer* (SI), i.e. that takes either the value 0 or an integer value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SI(j)$, where i is an variable number in the range $[1, ..., n]$.
sos1	A structure defining the *Special Ordered Sets of Type One* (sos1). Assume there are k sets of type sos1, then $sos1(k).var$ is a vector of indices for variables of type sos1 in set k. $sos1(k).row$ is the row number for the reference row identifying the ordering information for the sos1 set, i.e. A(sos1(k).row,sos1(k).var) identifies this information. As ordering information, also the objective function coefficients c could be used. Then as row number, 0 is instead given in $sos1(k).row$.
sos2	A structure defining the *Special Ordered Sets of Type Two* (sos2). Specified exactly as sos1 sets, see $sos1$ input variable description.

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

LogFile File to write Xpress-MP log output to. Default is empty '' in which case nothing is written. Please note that Xpress-MP appends it's output to the log file.

SaveFile Filename for writing the problem prior to calling the Xpress-MP solver. If empty, no file is written. The type of output is determined by the SaveMode parameter. Xpress-MP will always add an extension to the filename given here. The extension depends on the SaveMode chosen, see below.

SaveMode Character string with any combination of the following character flags:
p - full precision of numerical values.
o - one element per line.
n - scaled.
s - scrambled vector names.
l - output in LP format.
The extension added to the SaveFile name is .mat, unless the 'l' flag is used in which case the extension is .lp.

iisRequest Flag indicating whether to compute an IIS and return it to MATLAB. This option can only be set for an LP problem. If an IIS is found, XPRESS automatically changes the problem to make it feasible and reoptimizes it.
= 0, Don't return IIS to MATLAB (default).
= 1, Compute IIS and return it to MATLAB if an LP problem has been proven infeasible. The IIS is returned through the output parameter 'iis'.

iisFile Flag indicating whether to write a file describing the IIS set or not. If is set to 1, a file: LPprob.iis will be written. Otherwise, no file is written..

saRequest Structure telling whether and how you want XPRESS to perform a sensitivity analysis (SA). You can complete an SA on the objective function and right hand side vector. The saRequest structure contains two sub structures:

.obj and .rhs

They have one field each:

.index

In case of .obj.index, .index contains the indices of the columns whose objective function coefficients sensitivity ranges are required.

In case of .rhs.index, .index contains the indices of the rows whose RHS coefficients sensitivity ranges are required.

In both cases, the .index array has to be sorted, ascending.

To get an SA of objective function on the four variables 120 to 123 (included) and variable 6 the saRequest structure would look like this:

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

saRequest.obj.index = [6 120 121 122 123];

The result is returned through the output parameter 'sa'.

Table 9: Callback functions.

Index	m-file	Description
(1)	xpcb_USN	User Select Node Callback
(2)	xpcb_UPN	User Preprocess Node Callback
(3)	xpcb_UON	User Optimal Node Callback
(4)	xpcb_UIN	User Infeasible Node Callback
(5)	xpcb_UIS	User Integer Solution Callback
(6)	xpcb_UCN	User Node Cut-off Callback
(7)	xpcb_UCB	User Choose Branching Variable Callback
(8)	xpcb_IL	Simplex Log Callback
(9)	xpcb_GL	Global Log Callback
(10)	xpcb_BL	Barrier Log Callback
(11)	xpcb_UOP	User Output Callback
(12)	xpcb_CMI	User Defined Cut Manager Init Routine
(13)	xpcb_CMS	User Defined Cut Manager Termination Routine
(14)	xpcb_CM	User Defined Cut Manager Routine
(15)	xpcb_TCM	User Defined Top Cut Manager Routine

Description of Outputs

The following fields are used:

x	Solution vector x with decision variable values ($n \times 1$ vector).
slack	Slack variables ($m \times 1$ vector).
v	Lagrangian multipliers (dual solution vector) ($m \times 1$ vector).
rc	Reduced costs. Lagrangian multipliers for simple bounds on x.
f_k	Objective function value $f(x) = c^T * x$ at optimum.
ninf	Number of infeasibilities.
sinf	Sum of infeasibilities.
Inform	Result of Xpress^{MP} run:
0	Optimal solution found.
2	Unbounded solution.
4	Infeasible problem.
5	Some error occurred.

The following fields are used:, continued

See the Xpress^{MP} problem attributes XPRS_LPSTATUS (for LP and QP) and XPRS_MIPSTATUS (for MILP and MIQP) for more exact information. They are available in the global variable *xpProblemAttrib*.

basis Basis status of constraints and variables, ($m + n \times 1$ vector).

lpiter Number of simplex iterations.
glnodes Number of nodes visited.
iis Structure containing IIS information (niis x 1). niis is the number of IISs found (see MAXIIS parameter). The fields:

iisStatus Status flag. (Only set in the first element of the iis array.) Possible values:

2 = IIS was written to file LPprob.iis.
1 = IIS was obtained.
-1 = Problem was infeasible but no IIS found.
-2 = Problem was not infeasible.

iisMessage Error message on error. (Only set in the first element of the iis array.)

colind The column indices of the IIS set.

rowind The row indices of the IIS set.

sa Structure with information about the requested SA, if requested. The fields:

obj Ranges for the variables in the objective function.

rhs Ranges for the right hand side values.

These fields are structures themselves. All four structures have identical field names:

status Status of the SA operation. Possible values:

1 = Successful.
0 = SA not requested.
-1 = Error: MIP problem was presolved.

lower The lower range.

upper The upper range.

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables listed in Section 7 in the Xpress-Optimizer Reference Manual [1]. Available with fresh variables in each callback, and after the optimization.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes listed in Section 8 in the Xpress-Optimizer Reference Manual [1]. Available with fresh values in each callback, and after the optimization.

Description

The interface routine *xpress* calls Xpress^{MP} to solve LP, QP, MILP and MIQP problems. The matrix A is transformed in xpress.m to the Xpress^{MP} sparse matrix format.

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

A.2 xpressTL

Purpose

The TOMLAB /Xpress MILP, MIQP, LP and QP Interface. It solves linear programming (LP), quadratic programming (QP) , mixed integer linear programming (MILP) and mixed integer quadratic programming problems (MIQP). *xpressTL* solves problems of the form

$$\begin{array}{llcccc} \min\limits_{x} & f(x) = 0.5 * x^T * F * x + c^T * x & & & & \\ s/t & x_L & \leq & x & \leq & x_U \\ & b_L & \leq & Ax & \leq & b_U \\ & & x_i & \text{integer} & & i \in I \end{array}$$

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 $1, ..., n$, are restricted to be integers.

Calling Syntax

Prob = ProbCheck(Prob, 'xpress');
Result = xpressTL(Prob);

Description of Inputs

Prob, the problem structure. The following fields are used:

QP.c	Linear objective function cost coefficients, vector $n \times 1$.
QP.F	Square $n \times n$ dense or sparse matrix. Empty if non-quadratic problem.
A	Linear constraint matrix for linear constraints, dense or sparse $m \times n$ matrix.
x_L	Lower bounds on design parameters x. If empty assumed to be $-Inf$.
x_U	Upper bounds on design parameters x. If empty assumed to be Inf.
b_L	Lower bounds on the linear constraints.
b_U	Upper bounds on the linear constraints.
PriLev	Printing level in the *xpress* m-file and the Xpress^{MP} C-interface. = 0 Silent = 1 Summary information = 2 More detailed information
MIP.IntVars	Defines which variables are integers, of the general type I or binary B. Variable indices should be in the range [1,...,n]. If $IntVars$ is a logical vector then all variables i where $IntVars(i) > 0$ are defined to be integers. If $IntVars$ is determined to be a vector of indices then $x(IntVars)$ are defined as integers. If the input is empty ([ ]), then no integers of type I or B are defined. The interface routine *xpress* checks which of the integer variables have lower bound $x_L = 0$ and upper bound $x_U = 1$, i.e. are binary 0/1 variables.
MIP.PI	Integer variables of type *Partially Integer* (PI), i.e. takes an integer value up to a specified limit, and any real value above that limit. PI must be a structure array where: $PI.var$ is a vector of variable indices in the range $[1, ..., n]$.

Prob, the problem structure. The following fields are used:, continued

	$PI.lim$ is a vector of limit values for each of the variables specified in PI.var, i.e. for variable i, that is the PI variable with index j in $PI.var$, then $x(i)$ takes integer values in $[x_L(i), PI.lim(j)]$ and continuous values in $[PI.lim(j), x_U(i)]$.
MIP.SC	A vector with indices for the integer variables of type *Semi-continuous* (SC), i.e. that takes either the value 0 or a real value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SC(j)$, where i is an variable number in the range $[1, ..., n]$.
MIP.SI	A vector with indices for the integer variables of type *Semi-integer* (SI), i.e. that takes either the value 0 or an integer value in the range $[x_L(i), x_U(i)]$, assuming for some j, that $i = SI(j)$, where i is an variable number in the range $[1, ..., n]$.
MIP.sos1	A structure defining the *Special Ordered Sets of Type One* (sos1). Assume there are k sets of type sos1, then $sos1(k).var$ is a vector of indices for variables of type sos1 in set k. $sos1(k).row$ is the row number for the reference row identifying the ordering information for the sos1 set, i.e. A(sos1(k).row,sos1(k).var) identifies this information. As ordering information, also the objective function coefficients c could be used. Then as row number, 0 is instead given in $sos1(k).row$.
MIP.sos2	A structure defining the *Special Ordered Sets of Type Two* (sos2). Specified exactly as sos1 sets, see $MIP.sos1$ input variable description.
MIP.KNAPSACK	True if a knapsack problem is to be solved and a knapsack heuristic is to be used. Also $MIP.callback(9) = 1$ must be set if the heuristic is to be executed.
MIP.xpcontrol	Structure, where the fields are set to the Xpress^{MP} control parameters that the user wants to specify values for. The control parameters are listed in Section 7 in the Xpress-Optimizer Reference Manual [1]. The prefix XPRS_ is not used.
MIP.callback	Logical vector defining which callbacks to use in Xpress^{MP} . If the i^{th} entry of the logical vector *callback* is set, the corresponding callback is defined. See Section 5.3 in [1]. The callback calls the m-file specified in the Table 12 below. The user may edit this file, or make a new copy, which is put in a directory that is searched before the *xpress* directory in the Matlab path.
optParam	Structure with special fields for optimization parameters Fields used are: *MaxIter* - Maximal number of iterations or node visits. .
XPRESS.LogFile	File to write Xpress-MP log output to. Default is empty '' in which case nothing is written. Please note that Xpress-MP appends it's output to the log file.
XPRESS.SaveFile	Filename for writing the problem prior to calling the Xpress-MP solver. If empty, no file is written. The type of output is determined by the SaveMode parameter. Xpress-MP will always add an extension to the filename given here. The extension depends on the SaveMode chosen, see below.
XPRESS.SaveMode	Character string with any combination of the following character flags:

Prob, the problem structure. The following fields are used:, continued

p - full precision of numerical values.
o - one element per line.
n - scaled.
s - scrambled vector names.
l - output in LP format.
The extension added to the SaveFile name is .mat, unless the 'l' flag is used in which case the extension is .lp.

iis Flag indicating whether to compute an IIS and return it to MATLAB. This option can only be set for an LP problem. If an IIS is found, XPRESS automatically changes the problem to make it feasible and reoptimizes it.
= 0, Don't return IIS to MATLAB (default).
= 1, Compute IIS and return it to MATLAB if an LP problem has been proven infeasible. The IIS is returned through the output parameter 'iis'.

iisFile Flag indicating whether to write a file describing the IIS set or not. If is set to 1, a file: LPprob.iis will be written. Otherwise, no file is written..

sa Structure telling whether and how you want XPRESS to perform a sensitivity analysis (SA). You can complete an SA on the objective function and right hand side vector. The saRequest structure contains two sub structures:

.obj and .rhs

They have one field each:

.index

In case of .obj.index, .index contains the indices of the columns whose objective function coefficients sensitivity ranges are required.

In case of .rhs.index, .index contains the indices of the rows whose RHS coefficients sensitivity ranges are required.

In both cases, the .index array has to be sorted, ascending.

To get an SA of objective function on the four variables 120 to 123 (included) and variable 6 the saRequest structure would look like this:

saRequest.obj.index = [6 120 121 122 123];

The result is returned through the output parameter 'sa'.

Table 12: Callback functions.

Index	m-file	Description
(1)	xpcb_USN	User Select Node Callback
(2)	xpcb_UPN	User Preprocess Node Callback
(3)	xpcb_UON	User Optimal Node Callback
(4)	xpcb_UIN	User Infeasible Node Callback
(5)	xpcb_UIS	User Integer Solution Callback
(6)	xpcb_UCN	User Node Cut-off Callback
(7)	xpcb_UCB	User Choose Branching Variable Callback
(8)	xpcb_IL	Simplex Log Callback
(9)	xpcb_GL	Global Log Callback
(10)	xpcb_BL	Barrier Log Callback
(11)	xpcb_UOP	User Output Callback
(12)	xpcb_CMI	User Defined Cut Manager Init Routine
(13)	xpcb_CMS	User Defined Cut Manager Termination Routine
(14)	xpcb_CM	User Defined Cut Manager Routine
(15)	xpcb_TCM	User Defined Top Cut Manager Routine

Description of Outputs

Result structure. The following fields are used:

Iter	Number of iterations, or nodes visited.
ExitFlag	0: OK. 1: Maximal number of iterations reached. 2: Unbounded feasible region. 4: No feasible point found. 5: Error of some kind.
Inform	If a MIP problem the control variable XPRS_MIPSTATUS (xpControlVariables.MIPSTATUS) else XPRS_LPSTATUS (xpControlVariables.LPSTATUS).
x_0	Initial starting point not known, set as empty.
QP.B	Optimal active set, basis vector, in TOMLAB QP standard. $B(i) = 1$: Include variable $x(i)$ is in basic set. $B(i) = 0$: Variable $x(i)$ is set on its lower bound. $B(i) = -1$: Variable $x(i)$ is set on its upper bound.
f_k	Function value at optimum, $f(x_k)$.
g_k	Gradient value at optimum, c or $c + F * x$.
x_k	Optimal solution vector x_k.
v_k	Lagrangian multipliers (for bounds and dual solution vector). Set as $v_k = [rc; v]$, where rc is the n-vector of reduced costs and v holds the m dual variables.
xState	State of each variable. 0 = nonbasic (on x_L), 1 = nonbasic (on x_U), 2 = superbasic (between bounds), 3 = basic (between bounds)

Result structure. The following fields are used:, continued

bState	State of each constraint. 0 = nonbasic (on b_L), 1 = nonbasic (on b_U), 2 = superbasic (between bounds), 3 = basic (between bounds)
Solver	Solver used - Xpress^{MP} .
SolverAlgorithm	Solver algorithm used.
FuncEv	Number of function evaluations. Set to *Iter*.
GradEv	Number of gradient evaluations. Set to *Iter*.
ConstrEv	Number of constraint evaluations. Set to *Iter*.
Prob	Problem structure used.
MIP.ninf	Number of infeasibilities.
MIP.sinf	Sum of infeasibilities.
MIP.slack	Slack variables ($m \times 1$ vector).
MIP.lpiter	Number of LP iterations.
MIP.glnodes	Number of nodes visited.
MIP.basis	Basis status of constraints and variables ($m + n \times 1$ vector) in the Xpress^{MP} format, fields xState and bState has the same information in the Tomlab format.
MIP.xpControlVariables	Structure with all Xpress^{MP} control variables listed in Section 7 in the Xpress-Optimizer Reference Manual [1].
MIP.xpProblemAttrib	Structure with all Xpress^{MP} problem attributes listed in Section 8 in the Xpress-Optimizer Reference Manual [1].
XPRESS.iis	Structure containing IIS information (niis x 1). niis is the number of IISs found (see MAXIIS parameter). The fields:
iisStatus	Status flag. (Only set in the first element of the iis array.) Possible values: 2 = IIS was written to file LPprob.iis. 1 = IIS was obtained. -1 = Problem was infeasible but no IIS found. -2 = Problem was not infeasible.
iisMessage	Error message on error. (Only set in the first element of the iis array.)
colind	The column indices of the IIS set.
rowind	The row indices of the IIS set.
XPRESS.sa	Structure with information about the requested SA, if requested. The fields:
obj	Ranges for the variables in the objective function.
rhs	Ranges for the right hand side values.

Result structure. The following fields are used:, continued

These fields are structures themselves. All four structures have identical field names:

status	Status of the SA operation. Possible values: 1 = Successful. 0 = SA not requested. -1 = Error: MIP problem was presolved.
lower	The lower range.
upper	The upper range.

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables listed in Section 7 in the Xpress-Optimizer Reference Manual [1]. Available with fresh variables in each callback, and after the optimization.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes listed in Section 8 in the Xpress-Optimizer Reference Manual [1]. Available with fresh variables in each callback, and after the optimization.

Description
The TOMLAB Xpress^{MP} MILP, MIQP, QP and LP interface calls the interface routine *xpress.m*. Values $> 10^{20}$ and Inf values are set to 10^{20}, and the opposite for negative numbers. An empty objective coefficient c-vector is set to the zero-vector.

Examples
See *mip_prob*

M-files Used
xpress.m, *mipRun.m*

See Also
mipSolve

B The Matlab Interface Routines - Utility Routines

B.1 xpr2mat

Purpose

xpr2mat reads an (X)MPS file and more. The file is converted to matrices and vectors made available in MATLAB. MPS and extended MPS for LP, MILP, QP and MIQP are the supported file types, however it is possible to supply a wide range of file types.

Calling Syntax

$[F, c, A, b\_L, b\_U, x\_L, x\_U, IntVars]$ = xpr2mat(Name, PriLev, FreeRows);

Description of Input

Name	Name of the MPS file without extension. xpr2mat can recognize many different file extensions, e.g.: .mps, .lp, .mat, .qps.
PriLev	Print level of cpx2mat. Set to 0 to have it silent, 1 to print warnings, and 2 to print debug information.
FreeRows	Set to 1 to delete free rows. 0 leaves the free rows. Default: 1.

Description of Output

F	The quadratic term matrix. Empty for non-QP problems.
c	The linear term vector.
A	The constraint matrix.
b_L	The lower bounds of the constraints.
b_U	The upper bounds of the constraints.
x_L	The lower box bounds of x.
x_U	The upper box bounds of x.
IntVars	Logical vector describing what variables that are integer or binary variables. Empty if the problem is not a mixed integer problem.

B.2 abc2gap

Purpose

Converting a general assignment problem (GAP) to a standard form suitable for a MIP solver.

The GAP problem is formulated as

$$\min_{x_{ij}} \quad f(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} * x_{ij}$$

$$\begin{array}{ll} s/t & \sum_{j=1}^{n} x_{ij} = 1 \quad , i = 1, ..., m \\ & \sum_{i=1}^{m} a_{ij} * x_{ij} \le b_j \quad , j = 1, .., n \\ & x \in B^{m \times n}, B = \{0, 1\}. \end{array}$$

Calling Syntax

[c, x_L, x_U, b_L, b_U, a, sos1] = abc2gap(A, b, C, SOS1);

Description of Input

A	A $m \times n$ constraint matrix for GAP constraints.
b	A $m \times 1$ right hand side vector.
C	A $m \times n$ cost matrix for GAP constraints.
SOS1	Logical variable, default false. If true, generate output for sos1 handling with Xpress^{MP} Otherwise generate output giving an equivalent formulation with standard integer variables.

Description of Output

c	Linear objective function cost coefficients, vector $m * n \times 1$.
x_L	Lower bounds on design parameters x.
x_U	Upper bounds on design parameters x.
b_L	Lower bounds on the $m + n$ linear constraints.
b_U	Upper bounds on the linear constraints.
a	Sparse $m + n \times m * n$ matrix for linear constraints.
sos1	If input variable $SOS1$ is true, structure with sos1 variable information in the form suitable for the Matlab Xpress^{MP} interface routine *xpress*, otherwise empty.

Description

Converting a general assignment problem (GAP) to standard form suitable for a mixed-integer programming solver.

Either binary or sos1 variables are used.

B.3 xp2control

Purpose

Xpress^{MP} Matlab MEX-interface internal callback routine

Calling Syntax

xp2control(xpicv,xpdcv,xpccv1,xpccv2,xpccv3,xpccv4,xpccv5,xpccv6)

Description of Input

xpicv	Vector of doubles with Xpress^{MP} Integer Control Variables.
xpdcv	Vector of doubles with Xpress^{MP} Double Control Variables.
xpccv1	String with 1st Xpress^{MP} String Control Variable.
xpccv2	String with 2nd Xpress^{MP} String Control Variable.
xpccv3	String with 3rd Xpress^{MP} String Control Variable.
xpccv4	String with 4th Xpress^{MP} String Control Variable.
xpccv5	String with 5th Xpress^{MP} String Control Variable.
xpccv6	String with 6th Xpress^{MP} String Control Variable.

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.

Description

Xpress^{MP} Matlab MEX-interface internal callback routine. Creates a global Matlab structure variable xpControlVariables, where the fields corresponds to the Xpress^{MP} control variable names as given in Section 7 in the Xpress-Optimizer Reference Manual [1].

B.4 xp2problem

Purpose

Xpress^{MP} Matlab MEX-interface internal callback routine

Calling Syntax

xp2problem(xpipv,xpdpv,xpcpv1,xpcpv2,xpcpv3,xpcpv4,xpcpv5)

Description of Input

xpipv	Vector of doubles with Xpress^{MP} Integer Problem Variables.
xpdpv	Vector of doubles with Xpress^{MP} Double Problem Variables.
xpcpv1	String with 1st Xpress^{MP} String Problem Variable.
xpcpv2	String with 2nd Xpress^{MP} String Problem Variable.
xpcpv3	String with 3rd Xpress^{MP} String Problem Variable.
xpcpv4	String with 4th Xpress^{MP} String Problem Variable.
xpcpv5	String with 5th Xpress^{MP} String Problem Variable.

Global Parameters Used

xpProblemAttrib	Structure with all Xpress^{MP} problem attributes listed in Section 8 in the Xpress-Optimizer Reference Manual [1]. Set before the callback.

Description

Xpress^{MP} Matlab MEX-interface internal callback routine. Creates a global Matlab structure variable xpProblemAttrib, where the fields corresponds to the Xpress^{MP} problem attribute names.

C The Matlab Interface Routines - Test Routines

C.1 xpaircrew

Purpose

Test of an air-crew schedule generation problem.

Calling Syntax

xpaircrew

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.
MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description

Test of an air-crew schedule generation problem. Based on D.M.Ryan, Airline Industry, Encyclopedia of Operations Research and Management Science. Two subfunctions are used (defined at the end of the xpaircrew.m file): The function *generateToDs* create ToDs, i.e. Tours of Duty. The function *sectordata* generates some test data.

M-files Used

abc2gap, *xpress*

C.2 xpbiptest

Purpose

Test of TOMLAB /Xpress level 1 interface solving three larger binary integer linear optimization problems calling the Xpress^{MP} solver.

Calling Syntax

function xpbiptest(Cut, PreSolve, MipPre, NodeSel, BackTrack, xpcontrol)

Description of Input

Cut	Value of the CUTSTRATEGY control parameter, default $Cut = -1$. $Cut = -1$, auto select of $Cut = 1$ or $Cut = 2$. $Cut = 0$, no cuts. $Cut = 1$, conservative cut strategy. $Cut = 2$, aggressive cut strategy
PreSolve	Value of the PRESOLVE control parameter, default $PreSolve = 1$. $PreSolve = 0$, no presolve. $PreSolve = 1$, do presolve.
MipPre	Value of the MIPPRESOLVE control parameter, where the default value is dependent on the matrix characteristics. It determines the type of integer processing to be performed in the Branch and Bound. $MipPre = 0$, no processing will be performed. If bit 0 is set, do reduced cost fixing at each node. If bit 1 is set, do logical preprocessing on binary variables at each node. If bit 2 is set, do probing of binary variables is performed at the top node. A value $MipPre = 7$ will set all three bits as 1.
NodeSel	Value of the NODESELECT control parameter. The default value is dependent on the matrix characteristics. It determines which nodes will be considered for solution once the current node has been solved. $NodeSel = 1$, choose among the two descendant nodes, if none among all active nodes. $NodeSel = 2$, all nodes are always considered. $NodeSel = 3$, depth-first search exploring both descendants first. $NodeSel = 4$, all nodes are considered for the first BREADTHFIRST nodes, after which the usual default behavior is resumed. Setting xpcontrol.BREADTHFIRST influences the last choice.

Description of Input

BackTrack	Value of the BACKTRACK control parameter, default value is 3. Determines how the next node in the tree search is selected for processing. $BackTrack = 1$, if MIPTARGET is not set, choose the node with the best estimate. Otherwise the node choice is based on the Forrest-Hirst-Tomlin Criterion, which takes into account the best current integer solution and seeks a new node which represents a large potential improvement. $BackTrack = 2$, always choose the node with the best estimated solution. $BackTrack = 3$, always choose the node with the best bound on the solution.
xpcontrol	The initial xpcontrol structure. Here the user may set additional control parameters, e.g. xpcontrol.BREADTHFIRST. Default empty.

Global Parameters Used

MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description

Test of three larger binary integer linear optimization problems calling the Xpress^{MP} solver. The test problem 1 and 2 have 1956 variables, 23 equalities and four inequalities with both lower and upper bounds set.

Test problem 1, in *bilp1.mat*, is randomly generated. It has several minima with optimal zero value. Xpress^{MP} runs faster if avoiding the use of a cut strategy, and skipping presolve. Test problem 2, in *bilp2.mat*, has a unique minimum. Runs faster if avoiding the use of presolve.

Test problem 3, in *bilp1211.mat*, has 1656 variables, 23 equalities and four inequalities with lower and upper bounds set. Runs very slow without the use of cuts. A call *xpbiptest*$(0, 0)$ gives the fastest execution for the first two problems, but will be extremly slow for the third problem.

It might be interesting the follow the progress towards the solution by setting the global log callback. This could be done by removing the comment from the line

```
% callback(9) = 1;
```

in the code.

Timings are made with the Matlab functions *tic* and *toc*.

M-files Used

xpress, *xpprint*

C.3 xpiptest

Purpose

Test of the TOMLAB /Xpress level 1 interface solving three larger integer linear optimization problems calling the Xpress^{MP} solver.

Calling Syntax

function xpiptest(Cut, PreSolve, MipPre, NodeSel, BackTrack, xpcontrol)

Description of Input

Cut	Value of the CUTSTRATEGY control parameter, default $Cut = -1$. $Cut = -1$, auto select of $Cut = 1$ or $Cut = 2$. $Cut = 0$, no cuts. $Cut = 1$, conservative cut strategy. $Cut = 2$, aggressive cut strategy
PreSolve	Value of the PRESOLVE control parameter, default $PreSolve = 1$. $PreSolve = 0$, no presolve. $PreSolve = 1$, do presolve.

Description of Input

MipPre	Value of the MIPPRESOLVE control parameter, where the default value is dependent on the matrix characteristics. It determines the type of integer processing to be performed in the Branch and Bound. $MipPre = 0$, no processing will be performed. If bit 0 is set, do reduced cost fixing at each node. If bit 1 is set, do logical preprocessing on binary variables at each node. If bit 2 is set, do probing of binary variables is performed at the top node. A value $MipPre = 7$ will set all three bits as 1.
NodeSel	Value of the NODESELECT control parameter. The default value is dependent on the matrix characteristics. It determines which nodes will be considered for solution once the current node has been solved. $NodeSel = 1$, choose among the two descendant nodes, if none among all active nodes. $NodeSel = 2$, all nodes are always considered. $NodeSel = 3$, depth-first search exploring both descendants first. $NodeSel = 4$, all nodes are considered for the first BREADTHFIRST nodes, after which the usual default behavior is resumed. Setting xpcontrol.BREADTHFIRST influences the last choice.

Description of Input

BackTrack	Value of the BACKTRACK control parameter, default value is 3. Determines how the next node in the tree search is selected for processing. $BackTrack = 1$, if MIPTARGET is not set, choose the node with the best estimate. Otherwise the node choice is based on the Forrest-Hirst-Tomlin Criterion, which takes into account the best current integer solution and seeks a new node which represents a large potential improvement. $BackTrack = 2$, always choose the node with the best estimated solution. $BackTrack = 3$, always choose the node with the best bound on the solution.
xpcontrol	The initial xpcontrol structure. Here the user may set additional control parameters, e.g. xpcontrol.BREADTHFIRST. Default empty.

Global Parameters Used

MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description

Test of three larger integer linear optimization problems calling the Xpress^{MP} solver. The test problems have 61 variables and 138 linear inequalities. 32 of the 138 inequalities are just zero rows in the matrix A. The three problems are stored in $ilp061.mat$, $ilp062.mat$ and $ilp063.mat$.

Code is included to remove the 32 zero rows, and compute better upper bounds using the positivity of the matrix elements, right hand side and the variables. But this does not influence the timing much, the Xpress^{MP} presolve will do all these problem changes.

It might be interesting the follow the progress towards the solution by setting the global log callback. This could be done by removing the comment from the line

```
% callback(9) = 1;
```

in the code.

A call $xpiptest(2, 1, 3, 3, 3)$ probably gives the fastest execution. Timings are made with the Matlab functions *tic* and *toc*.

M-files Used

xpress, *xpprint*

C.4 xptomtest1

Purpose
Test of using TOMLAB to call Xpress^{MP} for problems defined in the TOMLAB IF format.

Calling Syntax
xptomtest1

Description
Test of using TOMLAB to call Xpress^{MP} for problems defined in the TOMLAB IF format. The examples show the solution of LP, QP and MILP problems.

M-files Used
tomRun.

See Also
xpressTL.

C.5 xptomtest2

Purpose
Test of using TOMLAB to call Xpress^{MP} for problems defined in the TOMLAB TQ format.

Calling Syntax
xptomtest2

Description
Test of using TOMLAB to call Xpress^{MP} for problems defined in the TOMLAB TQ format. The routine *mipAssign* is used to define the problem. A simple problem is solved with Xpress^{MP} both as an LP problem and as a MILP problem. The problem is solved both with and without explicitly defining the slack variables.

M-files Used
mipAssign, *tomRun* and *PrintResult*.

See Also
xpressTL and *xpress*.

C.6 xpknaps

Purpose
Xpress^{MP} Matlab level 1 interface Knapsack test routine

Calling Syntax
xpknaps(P, Run, Cut)

Description of Input

P	Problem number 1-3. Default 1.
Run	If empty or Run ≤ 0, run normal Xpress^{MP} global solve. If Run > 0 run simple knapsack heuristic in callback xpcb_GL.m Default 0.
Cut	Cut strategy. 0 = no cuts, 1 = cuts, 2 = aggressive cuts. Default 0.

Global Parameters Used

MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.
xpControlVariables	Structure with all Xpress^{MP} control variables.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes.

Description

The Xpress^{MP} Matlab level 1 interface knapsack test routine runs three different test problems. It is possible to change cut strategy and use heuristics defined in callbacks.

Currently defined knapsack problems:

Problem	Name	Knapsacks	Variables
1	Weingartner 1	2	28
2	Hansen, Plateau 1	4	28
3	PB 4	2	29

M-files Used

xpress

C.7 xpknapsTL

Purpose

Xpress^{MP} Matlab level 1 interface Knapsack test routine

Calling Syntax

xpknapsTL(P, Run, Cut)

Description of Input

P	Problem number 1-3. Default 1.
Run	If empty or Run ≤ 0, run normal Xpress^{MP} global solve. If Run > 0 run simple knapsack heuristic in callback xpcb_GL.m Default 0.
Cut	Cut strategy. 0 = no cuts, 1 = cuts, 2 = aggressive cuts. Default 0.

Global Parameters Used

MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.
xpControlVariables	Structure with all Xpress^{MP} control variables.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes.

Description

The Xpress^{MP} Matlab level 2 interface knapsack test routine runs three different test problems. It is possible to change cut strategy and use heuristics defined in callbacks.

Currently defined knapsack problems:

Problem	Name	Knapsacks	Variables
1	Weingartner 1	2	28
2	Hansen, Plateau 1	4	28
3	PB 4	2	29

M-files Used

xpress

C.8 xptest1

Purpose
Test routine 1, calls Xpress^{MP} Matlab level 1 interface to solve a GAP problem.

Calling Syntax
x = xptest1

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes.
MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description
Running a generalized assignment problem (GAP) from Wolsey [2, 9.8.16, pp165]. In this test the linear sos1 constraints are defined explicitly.

Given the matrices A (constraints) and C (costs), *xptest1* is using the utility *abc2gap* to reformulate the problem into the standard form suitable for Xpress^{MP}

The number of iterations are increased, no presolve is used, and an aggressive cut strategy.

M-files Used
abc2gap, *xpress*

C.9 xptest2

Purpose
Test routine 2, calls Xpress^{MP} Matlab level 1 interface to solve a GAP problem.

Calling Syntax
x = xptest2

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes.
MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description
Running a generalized assignment problem (GAP) from Wolsey [2, 9.8.16, pp165]. In this test sos1 variables are used.

Given the matrices A (constraints) and C (costs), *xptest2* is using the utility *abc2gap* to reformulate the problem into the standard form suitable for Xpress^{MP}

The number of iterations are increased, no presolve is used, and an aggressive cut strategy is applied.

M-files Used
abc2gap, *xpress*

See Also
xptest3

C.10 xptest3

Purpose
Test routine 3, calls Xpress^{MP} Matlab level 1 interface to solve a GAP problem.

Calling Syntax
x = xptest3

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes.
MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description
Running a generalized assignment problem (GAP) from Wolsey [2, 9.6, pp159]. In this test the linear sos1 constraints are defined explicitly.

Given the matrices A (constraints) and C (costs), *xptest1* is using the utility *abc2gap* to reformulate the problem into the standard form suitable for Xpress^{MP}

The number of iterations are increased, no presolve is used, and no cut strategy is used.

M-files Used
abc2gap, *xpress*

See Also
xptest2

C.11 xptestqp1

Purpose
Simple test of calling Xpress^{MP} Matlab level 1 interface to solve a QP problem.

Calling Syntax
x = xptestqp1(MIP)

Description of Input

MIP	If $MIP = 1$, run as a MIQP problem, trying to make the first variable integer valued, otherwise run as a pure QP problem. Default $MIP = 0$.

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes.
MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description
Simple test of calling Xpress^{MP} Matlab level 1 interface to solve a QP or MIQP problem. The problem is the first test problem in the TOMLAB $qp_prob.m$ file.

M-files Used
xpress

C.12 xptestqp2

Purpose

Simple test of MIQP problem running Xpress^{MP} . Simple test of calling Xpress^{MP} Matlab level 1 interface to solve a QP problem.

Calling Syntax

x = xptestqp2(MIP)

Description of Input

MIP	If $MIP = 1$ (default), run as a MIQP problem, trying to make the first variable integer valued, otherwise run as a pure QP problem.

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes.
MAX_x	Maximal number of x elements printed in output statements. Default 20.
MAX_c	Maximal number of constraint elements printed in output statements. Default 20.

Description

Simple test of MIQP problem running Xpress^{MP} . The MIQP problem is from the Xpress-Optimizer Reference Manual [1], page 166. The problem is defined as

$$\begin{array}{cccccc} \min\limits_{x} & f(x) & = & -6x_1 + 2x_1^2 - 2x_1x_2 + 2x_2^2 & & \\ s/t & 0 & \leq & x_1, x_2 & \leq & \infty \\ & & & x_1 + x_2 & \leq & 1.9 \\ & & & x_1 \ \text{integer.} & & \end{array}$$

M-files Used

xpress

D The Matlab Interface Routines - Callback Routines

D.1 xpcb_bl

Purpose

Xpress^{MP} Barrier Log Callback routine

Calling Syntax

[quit,xpcontrol] = xpcb_bl(x, slack, pi, rc, Prob)

Description of Input

x	Solution vector x with decision variable values ($n \times 1$ vector)
slack	Vector of slack variables.
pi	Lagrange multipliers for the linear constraints, i.e. the dual variables.
rc	Lagrange multipliers for the inequality variable constraints, i.e. the reduced costs.
Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: *Prob.P = ProblemNumber;*, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*. Also *Prob.MIP.KNAPSACK* is set and variables defining the set of integer variables.

Description of Output

quit	Return flag. If non-zero, Xpress^{MP} will exit.
xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} ontrol variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} roblem attributes. Set before the callback.

Description

At each iteration running the barrier algorithm, this routine is called.

Examples

Default some printing is done, and the user should instead write the Matlab statements wanted. The definition of a few control variables are shown as comments.

See Also

See the documentation for the Xpress^{MP} routine *XPRSsetcbbarlog*.

D.2 xpcb_gl

Purpose

Xpress^{MP} Global Log Callback routine

Calling Syntax

[quit,xpcontrol] = xpcb_gl(x, xBIS, Prob)

Description of Input

x Latest solution vector x with decision variable values ($n \times 1$ vector). If control variable $MIPINFEAS = 0$, then x is a new integer solution. If $MIPINFEAS > 0$, then x is the latest simplex solution.

xBIS Solution vector $xBIS$ with best integer solution found ($n \times 1$ vector), otherwise empty. If control variable $MIPINFEAS = 0$, then $xBIS$ is the best integer solution found before this step. The new integer solution might or might not be an improvement. If $MIPINFEAS > 0$, then $xBIS$ is either empty, or the best integer solution found so far.

Description of Input

Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber;$, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

quit	Return flag. If non-zero, Xpress^{MP} will exit.
xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description

This is the global log callback routine. How often it is called is dependent on the control variable *MIPLOG*:

- $MIPLOG = 0$. No printout in global.
- $MIPLOG = 1$. Print out summary statement at the end.
- $MIPLOG = 2$. Print out all solutions found, i.e. all integer valued solutions.
- $MIPLOG = 3$. Print out each node.
- $MIPLOG < 0$. Print out each $-MIPLOG^{th}$ node.

The default value is $MIPLOG = -100$. If to apply the simple KNAPSACK heuristic that is programmed as an example in this callback routine, then $xpcontrol.MIPLOG = 3$ should be set.

The following logic describes what x and xBIS are set as, and the relations to the problem attributes that contains function values.

```
if XPRS_MIPINFEAS > 0
   x    is LP      solution, f(x) = XPRS_LPOBJVAL
   xBIS is empty or the best integer solution, f(x) = XPRS_MIPOBJVAL
   XPRS_MIPOBJVAL == 1E20 before the first integer solution is found
end
if XPRS_MIPINFEAS == 0
   if XPRS_LPOBJVAL == XPRS_MIPOBJVAL
      x    is the best integer solution found, f(x) = XPRS_LPOBJVAL
      xBIS is the old best integer solution found, unknown f(xBIS).
           f(xBIS) could be computed as Prob.QP.c' * xBIS;
   else
      x    is the a new integer solution, but not the best, f(x) = XPRS_LPOBJVAL
      xBIS is the best integer solution found, f(xBIS) = XPRS_MIPOBJVAL
   end
end
```

Examples

The routine writes out the node number, the node depth, the best bound and the best integer solution so far found. The Matlab code shows an implementation of a simple heuristic, an Xpress^{MP} standard example similar to the example file:

`\xpressmp\examples\optimizer\knapsack\knapsack.c.`

This implementation assumes that a minimum problem is solved. The heuristic is used if $Prob.MIP.KNAPSACK$ is $true$. Also $xpcontrol.MIPLOG = 3$ must be set.

See Also
See the documentation for the Xpress^{MP} routine *XPRSsetcbgloballog*.

D.3 xpcb_il

Purpose
Xpress^{MP} Simplex Log Callback routine

Calling Syntax
[quit,xpcontrol] = xpcb_il(x, slack, pi, rc, Prob)

Description of Input

x	Solution vector x with decision variable values ($n \times 1$ vector)
slack	Vector of slack variables.
pi	Lagrange multipliers for the linear constraints, i.e. the dual variables.
rc	Lagrange multipliers for the inequality variable constraints, i.e. the reduced costs.
Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber$;, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

quit	Return flag. If non-zero, Xpress^{MP} will exit.
xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description
Called at the 0 and last simplex iteration, as well as each LPLOG iteration, where LPLOG is the XPRS_LPLOG control variable.

Examples
Default the routine prints the problem number, the iteration number and the current value of the objective function. The user could instead write the Matlab statements wanted. The definition of a few control variables are shown as comments.

See Also
See the documentation for the Xpress^{MP} routine *XPRSsetcblplog*.

Bugs
The variables x, *slack*, *pi* and *rc* are just set as empty. It is not possible to retrieve these variables during the simplex iterations. They will probably be deleted later on.

D.4 xpcb_ucb

Purpose

Xpress^{MP} User Choose Branching Variable Callback routine

Calling Syntax

[iPtr, iDir, estdeg, xpcontrol] = xpcb_ucb(iPtr, iDir, estdeg, Prob)

Description of Input

iPtr	Pointer to the variable or the set to branch.
iDir	1 or 3 (for sets) means upward branch. 0 or 2 (for sets) means downward branch.
estdeg	Estimated degradation using the selected variable or set.
Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber;$, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are $Prob.QP.c$, $Prob.QP.F$, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

iPtr	Pointer to the variable or the set to branch.
iDir	1 or 3 (for sets) means upward branch. 0 or 2 (for sets) means downward branch.
estdeg	Estimated degradation using the selected variable or set.
xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description

At each global iteration, the User Choose Branching Variable Callback routine is called. It gives the user the possibility to set the wanted branching variable. New values for the control variables are also possible to return.

Examples

Default the node number, the branch pointer, the direction and the estimated degradation is printed. The user should instead write the Matlab statements to set the branch pointer, the direction and the estimated degradation.

See Also

See the documentation for the Xpress^{MP} routine *XPRSsetcbchgbranch*. It is demonstrated how to choose branching on the most fractional integer.

D.5 xpcb_ucn

Purpose

Xpress^{MP} User Node Cut-Off Callback routine

Calling Syntax

[xpcontrol] = xpcb_ucn(node, Prob)

Description of Input

node	Node selected by Xpress^{MP}
Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber;$, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description

Declares a user node cutoff callback function, called every time a node is cut off as a result of an improved integer solution being found during the Branch and Bound search. New values for the control variables may be returned as sub fields in the *xpcontrol* variable.

Examples

Default the node number is printed.

See Also

See the documentation for the Xpress^{MP} routine *XPRSsetcbnodecutoff*.

D.6 xpcb_uin

Purpose

Xpress^{MP} User Infeasible Node Callback routine

Calling Syntax
[xpcontrol] = xpcb_uin(Prob)

Description of Input

Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber$;, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description
At each global iteration, when an infeasible node is found the User Infeasible Node Callback routine is called. The infeasible node is picked up using the global structure *xpProblemAttrib.NODES*. New values for the control variables is returned as sub fields in the *xpcontrol* variable.

Examples
Default the infeasible node number is printed.

See Also
See the documentation for the Xpress^{MP} routine *XPRSsetcbinfnode*.

D.7 xpcb_uis

Purpose
Xpress^{MP} User Integer Solution Callback routine

Calling Syntax
[xpcontrol] = xpcb_uis(Prob)

Description of Input

Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber$;, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description
At each global iteration, when an integer solution is found the User Integer Solution Callback routine is called. The integer valued node is picked up using the global structure *xpProblemAttrib.NODES*. New values for the control variables is returned as sub fields in the *xpcontrol* variable.

Examples

Default the node number with an integer solution is printed, together with the objective function value.

See Also
See the documentation for the Xpress^{MP} routine *XPRSsetcbintsol*.

D.8 xpcb_uon

Purpose
Xpress^{MP} User Optimal Node Callback routine

Calling Syntax
[Feasible] = xpcb_uon(Prob)

Description of Input

Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber;$, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

Feasible	If 0, the node is accepted as optimal.

Global Parameters Used

xpControlAttrib	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemVariables	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description
When an optimal solution for the current node is found the User Optimal Node Callback routine is called.

Examples
Default the node number and the function value is printed.

See Also
See the documentation for the Xpress^{MP} routine *XPRSsetcboptnode*.

D.9 xpcb_uop

Purpose
Xpress^{MP} User Output Callback routine

Calling Syntax
xpcb_uop(msg, msgLevel, Prob)

Description of Input

msg	Error message string.
msgLevel	Error Level 4 = Error 3 = Warning 2 = Dialogue 1 = Information 0 = Flush buffers -1 = No message
Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: *Prob.P = ProblemNumber*;, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

xpcontrol	Structure with the control fields that the user wishes to be set in Xpress^{MP}

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description

Every time the *Xpress*^{MP} solver wants to output a message, this routine is called. An error or warning message is always printed in the Matlab command window, if this callback is enabled. If the control variable *OUTPUTLOG* is true, then also normal messages are printed in the Matlab command window.

See Also

See the documentation for the Xpress^{MP} routine *XPRSsetcbmessage*.

D.10 xpcb_upn

Purpose

Xpress^{MP} User Preprocess Node Callback routine

Calling Syntax

[Feasible] = xpcb_upn(Prob)

Description of Input

Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: *Prob.P = ProblemNumber*;, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are *Prob.QP.c*, *Prob.QP.F*, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

Feasible	A 1 shows the node is LP infeasible, a 0 that it is feasible.

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description

The User Preprocess Node Callback routine is called before the analysis of the node. The user might have an efficient method in determining if the node is LP feasible or not. If the return flag is changed to one, the node is not further considered.

Examples
Default the node number is printed.

See Also
See the documentation for the Xpress^{MP} routine *XPRSsetcbprenode*.

D.11 xpcb_usn

Purpose
Xpress^{MP} User Select Node Callback routine

Calling Syntax
[xpcontrol] = xpcb_usn(node, Prob)

Description of Input

node	Node selected by Xpress^{MP}
Prob	A structure. If TOMLAB calls *xpress*, then *Prob* is the standard TOMLAB problem structure, otherwise the user optionally can set: $Prob.P = ProblemNumber$;, where ProblemNumber is some integer. When the callback routine is called then the arrays that define the current problem are added as fields in *Prob*. The additional fields are $Prob.QP.c$, $Prob.QP.F$, *Prob.x_L*, *Prob.x_U*, *Prob.A*, *Prob.b_L*, *Prob.b_U*.

Description of Output

node	User selected node.

Global Parameters Used

xpControlVariables	Structure with all Xpress^{MP} control variables. Set before the callback.
xpProblemAttrib	Structure with all Xpress^{MP} problem attributes. Set before the callback.

Description
Every time the code backtracks to select a new node during the MIP search, the User Select Node Callback routine is called. It is then possible to return another node number, if the user has a better strategy for selecting the new node.

Examples
Default the node number is printed.

See Also
See the documentation for the Xpress^{MP} routine *XPRSsetcbchgnode*.

E IIS and SA

It is possible to perform infeasibility and sensitivity analysis with TOMLAB /Xpress. The inputs and outputs are described in detail in Section A.1 and A.2.

E.1 IIS

If TOMLAB /Xpress reports that your problem is infeasible, then you can invoke the TOMLAB /Xpress infeasibility finder to help you analyze the source of the infeasibility. This diagnostic tool computes a set of infeasible constraints and column bounds that would be feasible if one of them (a constraint or variable) were removed. Such a set is known as an irreducibly inconsistent set (IIS).

To work, the infeasibility finder must have a problem that satisfies two conditions:

- the problem has been optimized by the primal or dual simplex optimizer or by the barrier optimizer with crossover, and
- the optimizer has terminated with a declaration of infeasibility.

Correcting Multiple Infeasibilities

The infeasibility finder can find several irreducibly inconsistent sets (IIS), however the TOMLAB default is one (controlled by MAXIIS). Consequently, even after you detect and correct one such IIS in your problem, it may still remain infeasible. In such a case, you need to run the infeasibility finder more than once or increase MAXIIS to detect those multiple causes of infeasibility in your problem.

Interpreting IIS Output

The size of the IIS reported by TOMLAB /Xpress depends on many factors in the model. If an IIS contains hundreds of rows and columns, you may find it hard to determine the cause of the infeasibility. Fortunately, there are tactics to help you interpret IIS output:

- If the problem contains equality constraints, examine the cumulative constraint consisting of the sum of the equality rows.
- Try preprocessing with the TOMLAB /Xpress presolver. The presolver may even detect infeasibility by itself. If not, running the infeasibility finder on the presolved problem may help by reducing the problem size and removing extraneous constraints that do not directly cause the infeasibility but still appear in the IIS.
- Other simplifications of the constraints in the IIS, such as combining variables, multiplying constraints by constants, and rearranging sums, may make it easier to interpret the IIS.

E.2 SA

The availability of a basis for an LP allows you to perform sensitivity analysis for your model, if it is an LP. Such analysis tells you by how much you can modify your model without affecting the solution you found. The modifications supported by the sensitivity analysis function include changes of the right hand side vector and changes of the objective function.

References

[1] Dash Optimization Ltd. *Xpress-Optimizer Reference Manual, Release 13*, October 31, 2001.

[2] Laurence A. Wolsey. *Integer Programming*. John Wiley and Sons, New York, 1998.