RUTCOR
RESEARCH
REPORT

Solving Hard Mixed Integer Programming Problems with Xpress-MP: A MIPLIB 2003 Case Study

Richard Laundy [a] Michael Perregaard[a]
Gabriel Tavares [b c] Horia Tipi[c]
Alkis Vazacopoulos[c]

RRR 2–2007, January, 2007

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/~rrr

[a]Dash Optimization, Leamington Spa, UK.
E-Mail: {richard.laundy,michael.perregaard}@dashoptimization.com
[b]RUTCOR, Rutgers University, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA.
[c]Dash Optimization, Englewood Cliffs, USA.
E-Mail: {gabriel.tavares,horia.tipi,av}@dashoptimization.com

Rutcor Research Report
RRR 2–2007, January, 2007

Solving Hard Mixed Integer Programming Problems with Xpress-MP: A MIPLIB 2003 Case Study

Richard Laundy Michael Perregaard Gabriel Tavares
Horia Tipi Alkis Vazacopoulos

Abstract. Over the past decade there has been a huge improvement in the performance of state-of-the-art MIP codes. Combined with the increased performance of computers over this period the size and complexity of the problems that can be solved has increased enormously.
In this paper we take a closer look at some of the hardest problems in the MIPLIB 2003 library ([4]) to show how much progress has been made with Xpress-MP.

Contents

1 Introduction

Over the past decade there has been a huge increase in the performance of state-of-the-art Mixed Integer Programming (MIP) codes. This improvement in performance combined with the increased performance of computers, has dramatically increased the size and complexity of problems that can be solved. As a result, MIP is now commonly used in decision-oriented applications across a wide number of industries.

Xpress-MP is a suite of mathematical modeling and optimization tools used to solve linear, integer, quadratic, non-linear, and stochastic programming problems ([6, 17, 26]). The Xpress-MP suite is available on most computer platforms and in different capacities for solving problems of various sizes.

Xpress-MP was first released in 1983 as a tool for modeling and solving Linear Programming (LP) models on PCs, and it was extended in 1986 to solve MIP models as well. Since then a large number of improvements have been made to the code. Some changes fundamental to the improvement in MIP performance are:

- Robust high performance implementations of the underlying LP solver, which are up to 100 times faster than implementations of twenty years ago on the same hardware;
- Automatically generated cutting planes applied both at the root and in the branch-and-bound tree;
- Strong branching; and
- Heuristics.

Some of the innovations in MIP solution technology that have been implemented in Xpress, have resulted in orders of magnitude improvements in solution times, so that problems which were previously considered to be insoluble are now solved routinely within a short period of time. Some of these novel ideas are only applicable to certain problem classes, but their presence in a large toolbox of techniques allows Xpress to solve a wide range of problems derived from a large number of practical problems.

As a result of the vast improvements of the MIP technology within Xpress-MP, we decided that it was time to revisit some of the harder problems in MIPLIB 2003, to see what can be achieved. We show how the toolbox of techniques within Xpress can be harnessed to finally solve some of the problems which have seemed to be intractable in the past.

2 MIPLIB 2003

MIPLIB 2003 is a standard and widely used benchmark to compare the performance of various MIP algorithms (e.g., [2, 3, 7, 10, 11, 12, 15, 16, 19, 20, 21, 22, 23, 24, 25, 27, 35]). This library is publicly available at `http://miplib.zib.de` ([5]) and consists of 60 (minimization)

problems, each one having special characteristics, and described in more detail in Achterberg et al. [4].

The problems in the MIPLIB test set come from a wide range of applications and range from very easy problems to problems for which even solving the LP relaxation is a challenging task. According to the MIPLIB website (visited on October-2006), the MIPLIB 2003 instances were classified into three groups of difficulty as follows:

- 28 problems are solved within 1 hour with a commercial solver;
- 18 problems have their optimal solution known, but they do not satisfy the previous conditions;
- 14 problems are unsolved.

When we started our investigation there was no known solution to one of problems *stp3d* and to the best of our knowledge, only three of the unsolved problems had been solved prior to October 2006. Namely, problems *a1c1s1* and *timtab2* were reported to be solved optimally using GAMS Grid Computing in CONDOR [18]. The optimal solution of *swath* has also been reported in [18], which can be "quickly" solved by applying several rounds of subtour elimination cuts to this Travelling Salesman Problem.

In about a one-year period of time prior to this report, the number of open problems from MIPLIB decreased from 17 to 8 unsolved cases:

Dec-05 A solution of *arki001* was proven to be optimal by Balas and Saxena [10] using the split closure of the problem;

Jun-06 According to the MIPLIB website, problems *glass4* and *roll3000* could be solved by the state-of-the-art MIP solvers;

Sep-06 The optimal values of *a1c1s1*, *timtab2* and *swath* were found by GAMS grid Condor computing [18].

Nov-06 The optimal values of *atlanta-ip*, *msc98-ip* and *rd-rplusc-21* were found using Xpress 2006B by Laundy et al. [30].

In this paper, we address several optimization strategies using Xpress-MP[1] that lead to finding the optimal solutions presented by Laundy et al. [30], and the optimum of the problem *protfold*, previously unsolved (see Section 5). We also present improved solve times (to optimality) for five of the problems that have been solved in the past year (see Section 4), and present improved solutions for all the remaining seven open problems (see Section 7).

Before considering the harder MIPLIB problems we first look at some of the easier ones. The Xpress-MP performance on the group of MIPLIB problems which can be solved within 1 hour is investigated in Section 3 taking into account the following aspects:

[1]Version 17.10.04 of release 2006B has been considered in all the computational experiments.

- The integrality gap closed at the top node;
- The computing times of the problems, using older and more recent versions of the Xpress-MP software;
- The solution times obtained using Xpress MIP parallel computing;
- The solution times obtained by fine-tuning the Xpress-MP software.

3 Problems that can be solved within 1 hour

Table 1 includes the 29 instances from MIPLIB 2003 for which Xpress 2006B can find the optimal solution within 1 hour using *default* settings.

It should be remarked that the 1 hour criterion has been applied in every case to a computer system based on a dual Xeon 3.0 GHz, 64 bit, 4 GB of RAM, running Windows XP. Although the computer has 2 CPUs it should be noted that the results correspond to a Xpress run in serial mode.

If special settings and/or if a faster computer system had been used instead, the number of problems that Xpress could solve in 1 hour would be larger.

The control parameter *miprelstop* sets the relative gap stopping criterion. The cutoff parameters *miprelcutoff* and *mipaddcutoff*, when having a nonzero value, instruct the optimizer to cutoff subproblems which can only provide a "little" better than the current best objective value. Since by default Xpress assigns nonzero values to the previous controls, then *miprelstop*, *miprelcutoff* and *mipaddcutoff* must be assigned with zero values in order to assure that the solutions obtained by Xpress are optimal.

The most important facts about MIPLIB 2003, which can be derived from the results of Table 1, are the following:

- Six problems can be solved at the root node;
- Nine problems are solved by Xpress 2006B in less than 1 second (using a standard computer);
- The Xpress 2006B average computing time on the group of easier problems is 164.6 seconds. If ones disregards those problems solved within 1 second, then the average solve time of the remaining 20 problems is 265 seconds;
- The three more *challenging* instances on the group of easier problems are *mas74*, *mzzv11* and *pk1*, respectively having Xpress 2006B solve times of 3 332, 359 and 228 seconds.

Table 1: B&B nodes and solution times of the MIPLIB 2003 instances, which are solved to optimality by Xpress-MP 2006B with *default* settings in 1 hour.

Problem	*Xpress B&B Nodes to*		*Xpress Time*† *to*		*Optimal*
	Solution	*Optimality*	*Solution*	*Optimality*	*Value* (OPT)
10teams	26	51	2 s	2 s	924
aflow30a	3 416	7 023	23 s	43 s	1 158
air04	166	185	36 s	36 s	56 137
air05	209	261	31 s	32 s	26 374
cap6000	1 389	1 937	7 s	9 s	-2 451 377
disctom	1	1	4 s	4 s	-5 000
fiber	56	69	<1 s	<1 s	405 935.18
fixnet6	9	11	<1 s	<1 s	3 983
gesa2	1	1	<1 s	<1 s	25 779 856.4
gesa2-o	1	1	<1 s	<1 s	25 779 856.4
harp2	30 942	67 239	90 s	149 s	-73 899 798.84
manna81	1	1	<1 s	<1 s	-13 164
mas74	46 709	2 380 685	35 s	3 332 s	11 801.1857
mas76	1	219 797	<1 s	159 s	40 005.0541
misc07	2 220	21 359	8 s	65 s	2 810
mod011	1 001	1 277	64 s	72 s	-54 558 535
modglob	1	11	<1 s	<1 s	20 740 508.1
mzzv11	1 047	1 145	345 s	359 s	-21 718
mzzv42z	121	125	48 s	48 s	-20 540
nw04	28	161	15 s	17 s	16 862
opt1217	1	1	<1 s	<1 s	-16
p2756	8	15	<1 s	<1 s	3 124
pk1	157 892	232 391	149 s	228 s	11
pp08a	124	213	<1 s	1 s	7 350
pp08aCUTS	88	181	<1 s	1 s	7 350
qiu	6 245	10 383	120 s	147 s	-132.873137
rout	4 165	10 323	34 s	67 s	1077.56
set1ch	1	1	<1 s	<1 s	54 537.75
vpm2	19	1 341	<1 s	2 s	13.75

†Obtained on a dual Xeon 3.0 GHz, 64 bit, 4 GB of RAM, running Windows XP.

3.1 Integrality gap closed at the top node

Some of the recent literature on MIP (e.g., [11, 14]) has focused on how much of the integrality gap could be closed at the top node by generating cuts.

The *integrality gap* is the difference between the LP relaxation bound and the optimal value. The integrality gap *closed* at the root node is the difference between the original LP value and the more restricted LP formulation value obtained from the original LP with additional valid cuts.

The percentage of integrality gap closed by Xpress at the top node of the search tree (using default settings) is listed in Table 2 for the easy group of MIPLIB 2003 problems. The relevant Xpress 2006B cutting generation results on these instances are the following:

- On average the integrality gap closed at the top node is 62.3%;
- In twelve cases (41% of the instances) the integrality gap closed at the top node is larger than 90%;
- The integrality gap at the top node is closed completely for four problems (*10teams*, *disctom*, *manna81* and *mzzv42z*). Note that the LP relaxation of instance *disctom* coincides with its optimal objective.

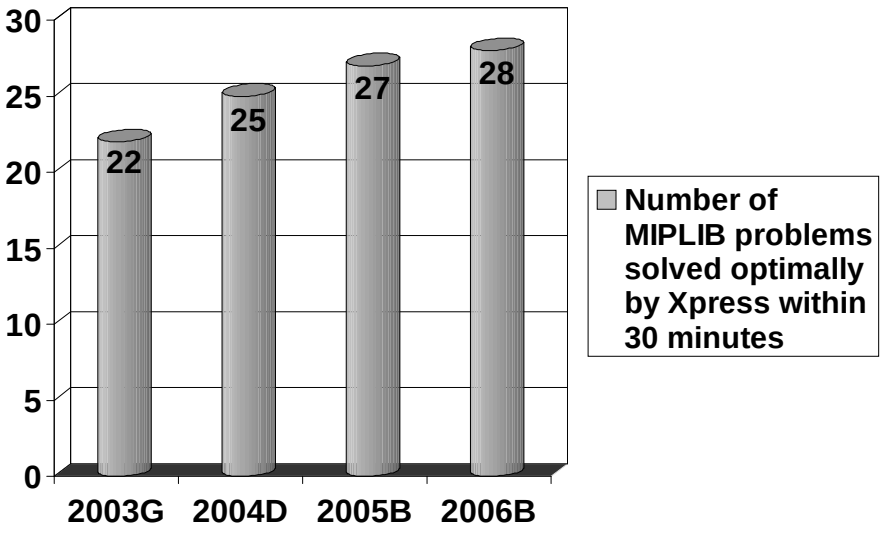

Figure 1: Number of MIPLIB 2003 problems solved optimally by Xpress within 30 minutes using defaults settings (on a Pentium 4, 3.6GHz, running Windows XP), across different releases, in the past 4 years.

Table 2: Integrality gap closed at the top node of the B&B tree of the MIPLIB 2003 instances solved to optimality, which are solved by Xpress-MP 2006B with *default* settings in 1 hour.

Problem	*LP Relaxation Value (LP)*	*Relaxation+Cutting Value (CUT)*	*% Gap Closed* $\left(\frac{CUT-LP}{OPT-LP}\right)$
pk1	0	0	0.0%
misc07	1 415	1 425	0.7%
rout	981.864286	983.41708	1.6%
mas76	38 893.9036	38 956.33192	5.6%
mas74	10 482.7953	10 563.0769	6.1%
qiu	-931.638845	-880.9154	6.4%
nw04	16 310.66667	16 436.85707	22.9%
air05	25 877.6093	25 999.48936	24.6%
air04	55 535.4364	55 710.9086	29.2%
harp2	-74 353 341.5	-74 160 911.74	42.4%
cap6000	-2 451 537.33	-2 451 448.988	55.1%
gesa2-o	25 476 489.7	25 662 530.85	61.3%
aflow30a	983.167425	1 094.550188	63.7%
mod011	-62 121 982.6	-56 639 764.37	72.5%
opt1217	-20.0213904	-17	75.1%
mzzv11	-22 945.2407	-21 940.91258	81.8%
vpm2	9.88926460	13.225837	86.4%
fixnet6	1 200.884	3 706.818534	90.1%
fiber	156 082.518	389 732.2951	93.5%
pp08aCUTS	5 480.60616	7 260.112137	95.2%
gesa2	25 476 489.7	25 771 846.66	97.4%
pp08a	2 748.34524	7 258.410726	98.0%
modglob	20 430 947.6	20 737 214.57	98.9%
p2756	2 688.75	3 121.349003	99.4%
set1ch	32 007.7299	54 536.11115	99.9%
10teams	917	924	100.0%
manna81	-13 297	-13 164	100.0%
disctom	-5 000	-5 000	100.0%
mzzv42z	-21 622.9985	-20 540	100.0%

3.2 Historical overview of the Xpress performance

The recent development of new algorithms and the improvement of existing methods allows Xpress to solve (optimally) problems (within minutes) that could not be solved just two or three years ago (in weeks/months). This claim is illustrated in Figure 1, which gives the number of MIPLIB 2003 problems that could be solved by the last four annual releases of Xpress-MP within 30 minutes. Using the same computer (a dual Xeon 3.0 GHz, 64 bit, running XP), Xpress 2003G could only solve 22 instances, whereas Xpress 2006B now solves six more instances within a 30 minutes time window.

Table 3 provides the speedup factor of the later Xpress releases (2004D, 2005B and 2006B) with respect to Xpress 2003G for the easier group of MIPLIB 2003 problems. Instances (*fixnet6*, *gesa2*, *gesa2-o*, *modglob*, *modglob*, *p2756* and *pp08a*) that could be solved by all the four solvers within 1 second were disregarded.

Table 3: Comparative performance of the Xpress software releases between 2003 and 2006 in the MIPLIB 2003 easier instances.

Problem	*Solve Time*† *of 2003G* (14.27)	*Optimality Speedup Factor to 2003G by*		
		2004D (15.30.12)	*2005B* (16.10.09)	*2006B* (17.10.04)
10teams	451 s	30.1×	32.2×	**225.5**×
aflow30a	141 s	0.8×	0.8×	**3.3**×
air04	75 s	3.1×	**3.8**×	2.1×
air05	54 s	1.5×	1.2×	**1.7**×
cap6000	46 s	2.0×	2.2×	**4.6**×
disctom	⩾20 000 s	⩾**6 670**×	⩾**6 670**×	⩾5 000×
fiber	3 s	**3.0**×	**3.0**×	**3.0**×
harp2	437 s	1.4×	1.5×	**2.7**×
manna81	⩾6 000 s	⩾1 500×	⩾**6 000**×	⩾**6 000**×
mas74	3 781 s	1.2×	**1.3**×	1.1×
mas76	112 s	**1.1**×	0.9×	0.7×
misc07	98 s	1.1×	0.9×	**1.3**×
mod011	129 s	1.4×	1.4×	**1.6**×
mzzv11	97 889 s	<1.5×	**119.7**×	113.2×
mzzv42z	⩾20 000 s	n/a	⩾**415**×	⩾385×
nw04	52 s	**4.7**×	4.0×	2.7×
opt1217	⩾35 000 s	n/a	n/a	⩾**35 000**×
pk1	**139 s**	0.6×	0.7×	0.6×
pp08aCUTS	2 s	**2.0**×	**2.0**×	**2.0**×
qiu	196 s	**1.3**×	**1.3**×	1.2×
rout	86 s	0.3×	0.9×	**1.2**×
set1ch	2 965 s	0.8×	**2 965**×	**2 965**×
vpm2	6 s	2.0×	2.0×	**3.0**×

†Obtained on a dual Xeon 3.0 GHz, 64 bit, 4 GB of RAM, running Windows XP.

Some facts derived from the results of Table 3 are remarked:

- 2003G is the fastest solver in a single case (*pk1*);
- 2004D is the fastest solver in six cases;
- 2005B is the fastest solver in eleven cases;
- The latest release of Xpress (2006B) is the fastest solver in fourteen cases (i.e., in about $\frac{2}{3}$ of the cases analyzed);

From the results of Table 3, it is also clear that the development of new algorithms was the key element in solving some problems in a given release of Xpress, which are unsolved by previous releases. For instance, problem *opt1217* can be solved in less than 1 second using 2006B, but it can not be solved (in hours) by any of the previous releases of Xpress.

3.3 Using Xpress parallel MIP to speedup solve times

In this and in the following sections we will provide two alternative ways to improve the solution times for MIP using Xpress 2006B.

The first approach considered is based on using parallel algorithms. The Xpress-MP MIP and barrier algorithms can be run using more than one thread by setting the following controls:

- **barrier** – Parameter *barthreads* is the number of threads implemented to run the parallel barrier algorithm;
- **branch-and-bound** – Parameter *mipthreads* is the number of threads implemented to run the parallel MIP code during branch-and-bound.

The parallel MIP performs separate tree search dives in the different threads in a way similar to the operation of the earlier distributed parallel code described in [29]. The path taken by the parallel MIP is non-deterministic and so the solution vector and the time taken may differ between different runs.

Table 4 provides the average (the minimum and the maximum) parallel speedup of several MIPLIB 2003 problems using Xpress 2006B. These results were collected over 10 runs of Xpress (using *mipthreads*=2) for each problem.

The single processor option turned out to be superior to the parallel option in 3 cases. The two CPUs option however was considerably better in 10 cases, having an average speedup of 1.7 times.

It is interesting to note that two matrices (*air04* and *pk1*) have a parallel speedup factor larger than 2 times, obtained in a computer having only two CPUs.

Table 4: 2–CPU parallel Xpress 2006B speedup of the MIPLIB 2003 instances. Results correspond to 10 runs of each problem.

Problem	*1–CPU Solve Time* $\left(t^{\text{serial}}\right)$	*2–CPU Solve Time* $\left(t^{\text{parallel}} \pm \text{st.dev.}\right)$	*Speedup* [min, max]	*Speedup Expected* $\left(\frac{t^{serial}}{t^{parallel}}\right)$
10teams	2 s	6.2±2.5 s	[0.2×, 0.5×]	0.3×
mzzv11	359 s	445.1±53.1 s	[0.7×, 0.9×]	0.8×
mzzv42z	48 s	61.4±0.8 s	[0.8×, 0.8×]	0.8×
rout	67 s	76.9±13.7 s	[0.7×, 1.1×]	0.9×
vpm2	2 s	2.2±0.4 s	[0.8×, 1.1×]	0.9×
nw04	17 s	17.6±0.5 s	[0.9×, 1.0×]	1.0×
mas74	3 332 s	3 083.2±805.8 s	[0.9×, 1.5×]	1.1×
harp2	149 s	120.2±24.2 s	[1.0×, 1.6×]	1.2×
cap6000	9 s	5.9±0.6 s	[1.4×, 1.7×]	1.5×
mod011	72 s	45.3±3.0 s	[1.5×, 1.7×]	1.6×
qiu	147 s	90.2±2.3 s	[1.6×, 1.7×]	1.6×
air05	32 s	18.5±1.0 s	[1.6×, 1.8×]	1.7×
misc07	65 s	37.9±2.4 s	[1.6×, 1.8×]	1.7×
aflow30a	43 s	23.7±1.3 s	[1.7×, 1.9×]	1.8×
mas76	159 s	83.1±5.9 s	[1.8×, 2.1×]	1.9×
air04	36 s	16.8±0.6 s	[2.1×, 2.2×]	2.1×
pk1	228 s	104.4±13.7 s	[1.9×, 2.5×]	2.2×

3.4 Fine-tuning Xpress to speedup solve times

Xpress-MP has a large toolbox of techniques which are effective on numerous problem classes. Solving a MIP effectively often requires a collection of these techniques applied at different levels and frequency. Whilst the default strategies for applying these techniques are usually very good they cannot be optimal for all problems and it is often possible to fine tune Xpress to improve performance by varying the optimization strategies. One of the techniques which can make a huge difference to the solution time is the use of cutting planes (see [13], [33], [34], [9], [37]). Getting the balance between cutting and branching is crucial, so the first stage for fine-tuning is to establish how much emphasis should be put on *cutting* and *branching*.

The frequency at which heuristics are run by the solver should also be considered. If the solutions found by branch-and-bound are of "good" quality, then the use of heuristics is unnecessary. However if no solutions are easily found or if they are of "poor" quality then heuristics should be applied extensively.

Table 5 lists a subset of the MIPLIB 2003 problems, their solve times using automatic choice of values for the control parameters, and the Xpress-tuned control parameter settings and corresponding solution times.

The average speedup between the computing times of Xpress and Xpress-tuned is 1.9 times. The *aflow30a* instance gets the largest speedup, with the Xpress-tuned version solv-

Table 5: Control parameters to improve Xpress 2006B performance on some MIPLIB 2003 problems.

Problem	*Xpress Control Parameters*	*Xpress Default Time* $\left(t^{default}\right)$	*Xpress Tuned Time* $\left(t^{special}\right)$	*Speedup* $\left(\frac{t^{default}}{t^{special}}\right)$
aflow30a	*cutstrategy*=3 *heurstrategy*=0 *varselection*=3	43 s	15 s	2.9×
air04	*lnpiterlimit*=75, *cutfreq*=1 *heurstrategy*=0 *sbbest*=0	36 s	17 s	2.1×
air05	*cutstrategy*=0 *sbeffort*=0 *nodeselection*=5, *varselection*=4	32 s	16 s	2.0×
cap6000	*cutstrategy*=1 *cutdepth*=0 *sbeffort*=0	9 s	7 s	1.3×
harp2	*cutstrategy*=3, *covercuts*=20 *lnpbest*=200, *lnpiterlimit*=75 *nodeselection*=5	146 s	77 s	1.9×
mas74	*cutstrategy*=0 *heurstrategy*=0 *nodeselection*=3, *varselection*=3	3 332 s	2 248 s	1.5×
mas76	*cutstrategy*=3, *covercuts*=5, *gomcuts*=5 *lnpiterlimit*=50, *cutfreq*=3, *treecovercuts*=2 *treegomcuts*=3, *heurstrategy*=3, *sbeffort*=0.5 *nodeselection*=3, *varselection*=3	159 s	85 s	1.9×
misc07	*cutstrategy*=0 *heurstrategy*=0, *sbeffort*=0 *nodeselection*=2, *varselection*=1	65 s	37 s	1.8×
mod011	*covercuts*=10 *nodeselection*=2	72 s	48 s	1.5×
mzzv11	*covercuts*=20 *cutfreq*=3 *heurstrategy*=3	359 s	329 s	1.1×
mzzv42z	*cutstrategy*=0 *heurstrategy*=3, *sbbest*=0 *nodeselection*=3, *varselection*=5	48 s	28 s	1.7×
nw04	*covercuts*=3 *gomcuts*=3	17 s	7 s	2.4×
pk1	*cutfreq*=5 *varselection*=3	228 s	136 s	1.7×
qiu	*lnpiterlimit*=50 *heurstrategy*=0 *sbeffort*=0.5	147 s	112 s	1.3×
rout	*gomcuts*=20 *sbeffort*=1.5	67 s	24 s	2.8×

ing the problem in 15 seconds vs. the 43 seconds solve time obtained using Xpress default parameter values. The largest savings in time occurred with *mas74* where the corresponding Xpress-tuned solver spends 18 minutes less computing time than Xpress using default settings.

It is interesting to analyze what control parameters are predominant in Table 5.

Three control parameters appear frequently with low or high values (see Table 6). On the one hand, in 33% of the cases it is better to turn off – or use it in a very limited way – cutting, heuristics or strong branching. On the other hand the extensive use of cuts, heuristics or strong branching is sometimes the best way to speedup the optimization time.

Table 6: Number of cases with small or large values for the Xpress-tuned settings (*cutstrategy*, *heurstrategy* and *sbeffort*) of Table 5.

Control Parameter	*Impact on*	*Cases Where Value is*	
		Small	*Large*
cutstrategy	cutting	5 (33%)	3 (20%)
heurstrategy	heuristics	5 (33%)	2 (13%)
sbeffort	branching	5 (33%)	1 (7%)

The branching related controls for node and variable selection (respectively, *nodeselection* and *varselection*) appear seven times (about 50% of the cases) in Table 5. This high frequency number indicates that the branching choice is very important in determining the efficiency of the solver.

The following points are also remarked:

- Increasing generation of lift-and-project ([8, 9, 14, 36]) cuts (using *lnpbest* and *lnpiterlimit*) at the top node speeds up the computing times of four instances (26% of the cases);
- Specifying how many rounds of cuts should be applied at the top node (using *covercuts* and *gomcuts*) speeds up the computing times of six instances (40% of the cases);
- Specifying how frequently (using *cutfreq* and *cutdepth*) the generation of cuts should be applied during B&B speeds up the computing times of five instances (33% of the cases).

4 Problems that have been solved in the past year

In the following subsections we report how Xpress-MP 2006B can be used to solve to optimality five problems – *a1c1s1*, *arki001*, *glass4*, *roll3000* and *swath* – which were the last five confirmed problems solved prior to Laundy et al. [30].

We remark that two MIPs (*a1c1s1* and *swath*) from MIPLIB 2003 are solved for the first time using a standard computer, and that these optimal solutions are obtained without the use of logical cuts derived from the knowledge of the model formulation.

4.1 *a1c1s1*

Problem *a1c1s1* is a lot sizing problem first looked at by Van Vyve and Pochet [18, 30]. For this problem a large part of the initial integrality gap can be reduced by cutting, but the final 10% of the gap is very hard to close either by adding extra cuts or by branching. For this reason alternative strategies were required.

The procedure that we have considered to find this optimum consists of the following steps. First a partial branch-and-bound is run with full strong branching to create an initial set of 128 subproblems. For each subproblem that could not be solved within half an hour, we would split it again into 5 subproblems, and keep repeating this action until all subproblems were solved.

The test computer used is an Intel Core Duo 2 running at 3GHz with 4GB RAM. In this experiment, we have used both cores with Xpress *mipthreads=2*. The full time needed to prove optimality was about 32 hours.

The total number of subproblems created during the application of our method was 230, of which 212 represent "final" subproblems, i.e. problems which do not satisfy the splitting rule. The time needed to create these subproblems was 3 800 sec, and the time needed to solve them was approximately 110 000 seconds of which 72 000 seconds correspond to the "final" subproblems. The total number of nodes solved by our procedure was approximately 2.1 million nodes.

To solve each subproblem we have used several Xpress controls to get better optimization performance. Namely, we have considered a cutoff value of 11 534, we have used a node selection rule that would choose the best node from *all* outstanding nodes (i.e. *nodeselection*= 2), and put more effort on Xpress cutting and on strong branching than that used by the default settings.

Ferris [18] has also recently (September,2006) reported solving *a1c1s1*. For this achievement [18] they used the GAMS grid CONDOR system spending 3 452 hours of CPU time. Since our CPU time was about 58 hours (considering the 2 CPUs used), our approach using Xpress 2006B needs about 60 times less CPU consumption than the approach of [18].

Problem *a1c1s1* has an optimal objective value of 11 503.444125 .

4.2 *arki001*

The problem arki001 is a relaxation of a non-linear problem arising in the metallurgy industry. Although it is possible to find good solutions to this problem it was not proven that the best solution found to this problem was indeed optimal until 2006 when Balas and Saxena [10, 11] solved the problem with the use of cuts from the split closure. The solve time reported in [10] was about 65 hours, of which 54 hours were spent generating rank-1 split cuts and 11 hours were spent solving the strengthened MIP.

Achterberg [5] solved this instance using SCIP in under 9 hours, computing nearly 3 million nodes, using an AMD Opteron 2.2 GHz. Heavy use of strong branching has been pointed out to be the key technique to solve problem *arki001*.

Achterberg [1] using SCIP with aggressive strong branching and conflict analysis resolution solved this problem in 5.2 hours, computing about 1.8 million nodes.

Laundy et al. [30] report solving *arki001* with Xpress 2006A in 4 hours.

Xpress 2006B solves this problem in 3.9 hours on a dual Xeon 3.0 GHz, 4GB, XP, computing (using 1 thread) 2.85 million nodes. To achieve this goal, it is necessary to set the control parameters: *cutstrategy*=3, *covercuts*=5, *gomcuts*=10, *lnpbest*=150, *cutfreq*=1, *treegomcuts*=4, *heurstrategy*=0, *varselection*=3 and *sbeffort*=0.1. In conclusion, one has to use aggressive cut generation, both at the root node (including lift-and-project cuts) and during branch-and-bound, use no heuristics, and put limited effort in strong branching.

Problem *arki001* has an optimal objective value of 7 580 813.046.

4.3 *glass4*

The problem *glass4* is a nesting problem first studied by Luzzi ([32]). This particular instance is much easier for Xpress to solve if a cutoff value is used as this allows reduced cost fixing and allows all nodes with a bound greater than the cut-off to be fathomed.

Using the *ideal* cutoff value (we have used 1 200 012 601) Xpress 2006B proves optimality to problem *glass4* in 19 minutes using a dual Xeon 3.0 GHz, 4GB of RAM and XP, computing almost 500 000 nodes.

Without the use of a cutoff value, it is very hard to solve this problem. So in this case a 2-stage procedure can be used. First, the solver should run to find a solution (applying internal heuristics if necessary) which is close to the ideal cutoff value. In the second stage, the solver uses the cutoff value found in the previous stage, and if the cutoff value is close enough to the optimum then Xpress 2006B can solve it very efficiently, as was described above.

Setting the Xpress control parameter *varselection* to 2 is very helpful for this instance. The above 19 minutes run would become 12 minutes if this choice was adopted instead.

Problem *glass4* has an optimal objective value of 1 200 012 600.

4.4 *roll3000*

The problem *roll3000* is a rolling stock problem first studied by Kroon [28].

Xpress 2006A belongs to the first group of solvers that could solve this instance to optimality (see [5, 31]). The special settings of Xpress used to solve this problem included a node selection strategy that would choose the best node from all outstanding nodes (i.e. *nodeselection*= 2), and put more emphasis on cutting (in particular in generating "extra" Gomory cuts). The computer platform used is a dual Xeon 3.0 GHz, 64 bit and 4 GB RAM. Although this computer has two CPUs, *roll3000* has been solved in serial mode. Xpress 2006A needed about 2.8 million nodes and 15.5 hours to provide this certificate of optimality.

Using special settings (*covercuts*=20, *gomcuts*=20, *lnpbest*=100, *lnpiterlimit*=50, *cutfreq*=1, *treecovercuts*=2, *treegomcuts*=2, *heurstrategy*=3, *sbeffort*=4), Xpress 2006B can

solve this problem on the same computer in 13 minutes. The key control parameters include the heavy use of strong branching, especially at the top nodes of the search tree, and heavy generation of Gomory and lift-and-project cuts ([8, 9, 14, 36]) at the top node. Xpress 2006B needed approximately 100 000 nodes to prove optimality for this problem, which represents about 28 times fewer nodes than those created by Xpress 2006A (with other settings).

Ferris [18] states that *roll3000* was solved in about 50 hours of CPU time using GAMS grid computing in CONDOR. Achterberg [1] using SCIP with aggressive strong branching and moderate conflict analysis solved this problem in 10.3 hours by computing 4.1 million nodes using a computer based on a 3.2GHz P4. The previous solve times and hardware options are an indication that Xpress 2006B should be able to solve problems of the same nature as *roll3000*, which in this particular case deals with rolling stock and line planning.

Problem *roll3000* has an optimal objective value of 12 890.

4.5 *swath*

The problem *swath* is a mission planning model for synthetic aperture radar surveillance. The model optimizes the tours for aircrafts and is similar to the travelling salesman problem (TSP).

It is much easier to solve TSP problems if problem specific cuts are added. Ferris [18] recently (September, 2006) solved *swath* after adding 5 rounds of subtour elimination cuts, resulting in 32 extra constraints. The "enlarged" problem was solved in less than 20 minutes using a single machine with a standard MIP solver.

However this problem can also be solved without problem specific cuts by using the same optimization procedure that was used to solve problem *a1c1s1* (see Section 4.1).

Using both cores of an Intel Core Duo 2 computer the elapsed time needed to solve the problem *swath* to optimality was about 66 hours.

The total number of subproblems created during the application of our method was 516, of which 422 represent final subproblems. The time needed to create these subproblems was 900 sec, and the time needed to solve them was approximately 234 000 seconds of which 79 000 seconds correspond to the "final" subproblems. The total number of nodes solved by our procedure was about 60 million nodes.

In this case, we have again used several Xpress controls to allow us to get better optimization performance. Namely, we have considered a cutoff value of 468, we have used a node selection rule that would choose the best node from *all* outstanding nodes, and we put more effort on Xpress cutting (namely on lift-and-project cuts) and on strong branching.

We note that after more than 36 000 hours of CPU time, the GAMS grid CONDOR system (see [18]) was unable to prove optimality for the original (i.e., unchanged) *swath* problem. This fact combined together with our result clearly indicates that better optimization strategies and decisions (i.e., what levels of strong branching, cutting, etc.) can lead to the solution of very hard MIPs which at first sight seem impossible to solve.

Problem *swath* has an optimal objective value of 467.407491 ([18]).

5 Problems that have been solved for the first time

We now turn our attention to some of the harder problems in MIPLIB 2003 which were unsolved prior to September 2006.

Three MIP problems from MIPLIB 2003 – *atlanta-ip*, *msc98-ip* and *rd-rplusc-21* – were solved for the first time using Xpress 2006B by Laundy et al. [30]. In this section we shall describe the various algorithmic approaches used in [30] to get certificates of optimality for the previous problems. In addition, the optimal objective of problem *protfold* is presented for the first time, which was found by using Xpress 2006B.

A summary of the results is displayed in Table 7, which includes the computing times and the corresponding optimal objectives.

Table 7: Objective value and computing time needed to find the optimal solutions of four (previously unsolved) MIPLIB 2003 problems using Xpress-MP 2006B.

Problem	*Computer System*	*Solve Time*	*Optimal Objective*
atlanta-ip	3GHz Intel Core 2 Duo	10 hours	90.00987861
msc98-ip	3GHz Intel Core 2 Duo	1.5 hours	19 839 497.005874
protfold	Xeon 3.0 GHz, 2 CPU, 4 GB RAM	9 days	−31
rd-rplusc-21	Xeon 3.0 GHz, 2 CPU, 4 GB RAM	3.6 hours	165 395.2753

The following subsections describe the individual approaches that were used to solve these problems.

5.1 *atlanta-ip*

Problem *atlanta-ip* is a min-cost network problem with side constraints. Although the integrality gap is quite small it is very difficult to close it completely. To be able to solve this problem it was necessary to analyze the structure of the model, to understand why the problem was so difficult. The conclusions of this examination resulted in the following procedure:

1. First, we drop most of the redundant constraints from the problem. There are many of them and they seem to be cuts. Several hours are needed to identify those cuts.

2. Second, we split the objective function into two parts, one in the variables which have very large cost, and the remaining variables, which have cost around $[10^{-6}, 10^{-5}]$. By dropping the second part of the cost function, Xpress 2006B finds a solution that is optimal to within 0.1 absolute difference. The time needed for this step was roughly 4 hours (on a 3GHz Intel Core 2 Duo computer) and 95 000 nodes were computed during this stage.

3. Finally, using this near-optimal solution as the cutoff value, Xpress 2006B is able to find and prove optimality of the solution in 4.5 hours, with heavy cutting and strong branching. 66 000 nodes were computed by Xpress at this stage.

Problem *atlanta-ip* has an optimal objective value of 90.0098786144 ([30]).

5.2 *msc98-ip*

The problem *msc98-ip* is a min-cost network problem similar to *atlanta-ip* (see Section 5.1) but arising from the design of a nation-wide communication network.

To find this optimal solution we have taken the same approach as the one used to solve problem *atlanta-ip*. Problem *msc98-ip* is somewhat easier to solve than *atlanta-ip*.

Xpress 2006B needs about 8 minutes of computing time (on a 3GHz Intel Core 2 Duo computer) for the second stage, from which a near-optimal feasible solution to the problem is found.

For the final stage, Xpress 2006B needs about 1 hour of computing time on the same computer, in which it computed almost 4 000 nodes using (as before) heavy cutting and strong branching.

Problem *msc98-ip* has an optimal objective value of 19 839 497.005874 ([30]).

5.3 *protfold*

The problem *protfold* is a protein folding model. The degeneracy in the model makes it particularly difficult to solve to optimality and although it is easy to find poor quality solutions it becomes increasingly difficult to find better quality solutions.

The solution of this model requires the extensive use of heuristics (by setting *heurstrategy*= 3). Using Xpress parallel MIP, we were able to prove optimality for *protfold* in about 9 days of (elapsed) computing time.

The computer used in this experiment is a Xeon 3.0 GHz with 2 CPU, Hyper-Threading, 4 GB of RAM and runs Windows XP. The number of threads that we have assigned for parallel computing was equal to four (by setting *mipthreads*= 4). There was no particular reason in assigning this many threads, but it was clear from our previous experience on this problem that the four threads choice would cooperate well in increasing the lower bound faster than if just one individual thread had been selected instead.

The optimal solution was found by Xpress at node 133 930 of the search tree after almost 8 hours of computing time. At this particular node the bound was precisely −35. To close this gap completely Xpress computed nearly 2.4 million additional nodes.

Problem *protfold* has an optimal objective value of −31.

5.4 *rd-rplusc-21*

The problem *rd-rplusc-21* is a linearized relaxation of a non-linear model for a distillation column with external reactor. The problem has a large number of constraints compared to variables and these constraints are activated and deactivated by branching on the binary variables. So, the key to solve this problem is to find good estimates for the variable branching selection. Xpress 2006B can solve this problem by *not* applying three standard procedures,

i.e. by not generating cuts (*cutstrategy*=0), by not applying heuristics (*heurstrategy*=0), and by not using strong branching (*sbiterlimit*=0). In addition, the solver has to pick the most promising candidates for variable selection using a method based on probing with dual estimates (*sbestimate*=3), and from the selected candidates choose the one variable having the largest sum of the pseudo-cost degradation in both branches (*varselection*=2). With these settings, Xpress 2006B solves *rd-rplusc-21* in 3.6 hours (using a dual Xeon 3.0 GHz, 4GB RAM and XP), computing about 335 000 nodes. We finally remark the fact that using *constraint branching*, instance *rd-rplusc-21* can be solved in 14 minutes computing nearly 80 000 nodes.

Problem *rd-rplusc-21* has an optimal objective value of 165 395.2753 ([30]).

6 First feasible solution of *stp3d*

The MIP *stp3d* is possibly the hardest problem in MIPLIB 2003. The problem is a steiner tree problem for which the LP relaxation and some of the tree node re-optimizations can be difficult. The first feasible solution of problem *stp3d* has been found by Xpress using a maximum degradation strategy (i.e. by setting *varselection*= 4). This solution was found in about 5 hours using Xpress 2006B specially designed to apply fast node re-optimizations.

The first known solution of problem *stp3d* has an objective value of 529.778190.

Using the next generation of local search heuristics of Xpress, the first solution of *stp3d* was quickly improved to a solution with objective value of 500.736 (see Section 7).

It should be remarked that this solution defines a 3.3% relative gap.

So far the best known solution for *stp3d* is 500.736.

7 Reducing the gap of all open problems

Using the next generation of Xpress heuristics, based on local search procedures built on top of Xpress 2006B, improved solutions were found for all of the remaining (seven) open problems from MIPLIB 2003. Table 8 provides these improvements along with the previously best know objectives, to our best knowledge.

Before this study, the best know objectives of the open problems were mostly taken from the discussion list of the MIPLIB 2003 website [5]. Several recent papers ([2, 10, 11, 20, 27]) referencing MIPLIB 2003 were also considered to define the best objectives.

From Table 8 it can be seen that the new local search heuristics were capable of considerably improving the best know solutions. The most notable achievement occurred with the *ds* problem, which consists of a 58.9% gain over the old best know solution.

Each one of these solutions was found in a few hours on a standard computer on an average case. The precise solution times are not available since the improved solutions were found in an incremental way, and because the parameters would usually be different when applied between two consecutive rounds of heuristics.

Table 8: Improved objective values of the MIPLIB 2003 unsolved problems.

Problem	*Old Best Known Objective* ($z^{\star\star}$)	*Xpress Improved Objective* ($z^{\star}$)	*Gain* $\left\lvert 1 - \frac{(z^{\star})}{(z^{\star\star})} \right\rvert$
stp3d	unknown	500.736	n/a
ds	283.4425	116.59	58.9%
momentum3	370 177.036	236 426.335	36.1%
t1717	193 221	170 195	11.9%
liu	1 172	1 138	2.9%
dano3mip	691.2	687.733333	0.5%
sp97ar	664 565 103.76	661 670 441.4	0.4%

Table 9: Best known lower and upper bounds of the MIPLIB 2003 unsolved problems.

Problem	*Lower Bound* (z^{+})	*Upper Bound* ($z^{\star}$)	*Gap* $\left\lvert 1 - \frac{z^{\star}}{z^{+}} \right\rvert$
sp97ar	657 862 912	661 670 441.4	0.6%
stp3d	484.71817	500.736	3.3%
dano3mip	578.05603	687.733333	19.0%
t1717	136 538.4219	170 195	24.6%
ds	76.32504272	116.59	52.8%
liu	613	1 138	85.6%
momentum3	94 824.16406	236 426.335	149.3%

Table 9 lists the remaining unsolved problems in MIPLIB 2003, and for each instance presents the best upper bound (i.e. objective) and lower bound to the optimum.

It can be seen that instances *momentum3*, *liu* and *ds* have the largest relative gaps, having respectively gaps of 149.3%, 85.6% and 52.8%. These problems still offer a challenge for Xpress and it seems unlikely that they will be solved without making use of information associated with the problem logic or further breakthroughs in MIP technology.

On the other end, instance *sp97ar* has a relatively small gap (0.6%), and consequently we foresee that this will be the next challenging MIP to leave the list of open problems from MIPLIB 2003. Although *stp3d* has a small gap the rate at which the gap can be closed by branching or cutting is relatively slow so it may be a while before this problem is solved.

8 Conclusions

In this paper we have shown that it is possible to solve some of the hardest problems in MIPLIB 2003 using current technology. In particular, the solution methodology and technology considered, lead us to achieve the following results: (i) four MIP problems (*atlanta-ip*, *msc98-ip*, *protfold* and *rd-plusc-21*) were solved for the first time; (ii) two MIP problems (*a1c1s1* and *swath*) were solved for the first time using a single standard computer, without the use of logical cuts derived from the knowledge of the model formulation; (iii) the first feasible solution for the *stp3d* problem has been found; (iv) improved solutions were found for the remaining open problems from MIPLIB 2003.

Although there have been great advances in MIP technology, tuning and modeling still play an important role when solving hard optimization problems.

References

[1] Achterberg, T.: Conflict analysis in mixed integer programming. International Symposium on Mathematical Programming (ISMP) , July 30 - August 4, Rio de Janeiro, Brazil (2006)

[2] Achterberg, T., Berthold, T.: Improving the feasibility pump. ZIB-Report 05-42, Konrad-Zuse-Zentrum fr Informationstechnik Berlin, Takustrae 7 D-14195 Berlin-Dahlem (2005)

[3] Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Operations Research Letters **33**, 42–54 (2005)

[4] Achterberg, T., Koch, T., Martin, A.: Miplib 2003. Operations Research Letters **34**, 1–12 (2006)

[5] Achterberg, T., Koch, T., Martin, A.: Mixed Integer Problem Library (MIPLIB) 2003 (October 2006). URL `http://miplib.zib.de/`

[6] Ashford, R.: Mixed integer programming: A historical perspective with Xpress-MP. Annals of Operations Research (forthcoming) (2006)

[7] Atamtürk, A., Savelsbergh, M.W.P.: Integer-programming software systems. Annals of Operations Research **140**, 67–124 (2005)

[8] Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Mathematical Programming **58**, 295–324 (1993)

[9] Balas, E., Perregaard, M.: Lift-and-project for mixed 0-1 programming: Recent progress. Discrete Applied Mathematics **123**, 129–154 (2002)

[10] Balas, E., Saxena, A.: Optimizing over the split closure. Research Report MSRR-674 (2005)

[11] Balas, E., Saxena, A.: Optimizing over the split closure. Mathematical Programming, Series A (2006). DOI 10.1007/s10107-006-0049-5

[12] Bertacco, L., Fischetti, M., Lodi, A.: A feasibility pump heuristic for general mixed-integer problems. Discrete Optimization (forthcoming) (2006)

[13] Cordier, C., Marchand, H., Laundy, R., Wolsey, L.: bc-opt: A branch-and-cut code for mixed integer programs. Mathematical Programming **86**, 335 (1999)

[14] Cornuéjols, G.: Valid inequalities for mixed integer linear programs. Mathematical Programming (2006)

[15] Cornuéjols, G., Karamanov, M., Li, Y.: Early estimates of the size of branch-and-bound trees. INFORMS Journal on Computing **18**, 86–96 (2006)

[16] Dash, S., Günlük, O.: On the strength of gomory mixed-integer cuts as group cuts. IBM Research Report RC23967, Optimization Online (2006)

[17] Dash Optimization: Xpress-MP Optimizer Reference Manual. Dash Optimization Ltd., Blisworth House, Church Lane, Blisworth, Nothants NN7 3BX, UK (2005)

[18] Ferris, M.: GAMS, Condor and the grid: Solving hard optimization problems in parallel. Industrial and Systems Engineering, Lehigh University, Pennsylvania (2006)

[19] Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Mathematical Programming, Series A **104**, 91–104 (2005)

[20] Fischetti, M., Lodi, A.: Local branching. Mathematical Programming, Series B **98**, 23–47 (2003)

[21] Fischetti, M., Lodi, A.: Optimizing over the first Chvàtal closure. Mathematical Programming, Series B (2006). DOI 10.1007/s10107-006-0054-8

[22] Fischetti, M., Lodi, A.: Repairing mip infeasibility through local branching. Computers and Operations Research (2006). DOI 10.1016/j.cor.2006.08.004

[23] Fischetti, M., Lodi, A.: MIPping closures: An instant survey. Computers and Operations Research (forthcoming) (2007)

[24] Fischetti, M., Monaci, M.: How tight is the corner relaxation? Discrete Optimization (forthcoming) (2007)

[25] Glankwamdee, W., Linderoth, J.: Lookahead branching for mixed integer programming. Research Report 06T-004, Industrial and Systems Engineering, Lehigh University (2006)

[26] Guéret, C., Prins, C., Sevaux, M., Heipcke (Translator), S.: Applications of Optimization with XpressMP. Dash Optimization Ltd., Blisworth House, Church Lane, Blisworth, Nothants NN7 3BX, UK (2002)

[27] Hansen, P., Mladenović, N., Uroević, D.: Variable neighborhood search and local branching. Computers and Operations Research **33**, 3034–3045 (2006)

[28] Kroon, L.: Personal communication (2002)

[29] Laundy, R.: Implementation of parallel branch-and-bound algorithms in Xpress-MP. in Ciriani, T., Gliozzi, S., Johnson, E. and Tadei R. (Editors), Operational Research in Industry, Macmillan Press Ltd (1999)

[30] Laundy, R., Perregaard, M., Tavares, G., Vazacopoulos, A.: State-of-the-optimization using Xpress-MP v2006. INFORMS Annual Meeting, Nov 5-8, Pittsburgh, USA (2006)

[31] Laundy, R., Tavares, G., Vazacopoulos, A.: State-of-the-art optimization using Xpress-MP 2006A. International Symposium on Mathematical Programming (ISMP), July 30 - August 4, Rio de Janeiro, Brazil (2006)

[32] Luzzi, I.: Exact and heuristic methods for nesting problems. Ph.D. thesis, University of Padova (2002)

[33] Marchand, H., Wolsey, L.A.: The 0-1 knapsack problem with a single continuous variable. Mathematical Programming **85**, 15–33 (1999)

[34] Padberg, M.W., Roy, T.J.V., Wolsey, L.A.: Valid inequalities for fixed charge problems. OR **33**, 842–861 (1985)

[35] Patel, J., Chinneck, J.W.: Active-constraint variable ordering for faster feasibility of mixed integer linear programs. Mathematical Programming (forthcoming) (2006)

[36] Perregaard, M.: A practical implementation of lift-and-project cuts: A computational exploration of lift-and-project cuts with Xpress-MP. International Symposium on Mathematical Programming (ISMP), August 18-22, Copenhagen, Denmark (2003)

[37] Wolsey, L.: Integer Programming. Wiley (1998)