Math. Prog. Comp. (2011) 3:103–163
DOI 10.1007/s12532-011-0025-9

FULL LENGTH PAPER

MIPLIB 2010

Mixed Integer Programming Library version 5

Thorsten Koch · Tobias Achterberg · Erling Andersen · Oliver Bastert · Timo Berthold · Robert E. Bixby · Emilie Danna · Gerald Gamrath · Ambros M. Gleixner · Stefan Heinz · Andrea Lodi · Hans Mittelmann · Ted Ralphs · Domenico Salvagnin · Daniel E. Steffy · Kati Wolter

Received: 4 April 2011 / Accepted: 2 May 2011 / Published online: 7 June 2011

Abstract This paper reports on the fifth version of the Mixed Integer Programming Library. The MIPLIB 2010 is the first MIPLIB release that has been assembled by a large group from academia and from industry, all of whom work in integer programming. There was mutual consent that the concept of the library had to be expanded in order to fulfill the needs of the community. The new version comprises 361 instances sorted into several groups. This includes the main *benchmark* test set of 87 instances, which

T. Koch (✉) · T. Berthold · G. Gamrath · A. M. Gleixner · S. Heinz · D. E. Steffy · K. Wolter
Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
e-mail: koch@zib.de

T. Berthold
e-mail: berthold@zib.de

G. Gamrath
e-mail: gamrath@zib.de

A. M. Gleixner
e-mail: gleixner@zib.de

S. Heinz
e-mail: heinz@zib.de

D. E. Steffy
e-mail: steffy@zib.de

K. Wolter
e-mail: wolter@zib.de

T. Achterberg
IBM Deutschland, Böblingen, Germany
e-mail: achterberg@de.ibm.com

E. Andersen
MOSEK, Copenhagen, Denmark
e-mail: e.d.andersen@mosek.com

are all solvable by today's codes, and also the *challenge* test set with 164 instances, many of which are currently unsolved. For the first time, we include scripts to run automated tests in a predefined way. Further, there is a solution checker to test the accuracy of provided solutions using exact arithmetic.

Keywords Mixed Integer Programming · Problem instances · IP · MIP · MIPLIB

Mathematics Subject Classification (2000) 90C11 · 90C10 · 90C90

1 Introduction

The MIPLIB is now going into its fifth incarnation. Starting in 1992 with the first two versions by Bixby, Boyd, and Indovina [23], the update to MIPLIB 3 by Bixby, McZeal, Ceria, and Savelsbergh [24] in 1996, and the compilation of MIPLIB 2003 by Achterberg, Koch, and Martin [3], we have finally arrived at MIPLIB 2010. The motivation for this update is the same as in the previous versions: the continuous progress in the field of mixed integer programming.

A *mixed integer (linear) program* (MIP) is an optimization problem in which a linear objective function is minimized subject to linear constraints over real- and integer-valued variables. For details on mixed integer programming, see, e.g., [69, 106]. The MIPLIB is a diverse collection of challenging real-world MIP instances from various academic and industrial applications suited for benchmarking and testing of MIP solution algorithms.

O. Bastert
FICO, Munich, Germany
e-mail: oliverbastert@fico.com

R. E. Bixby
Gurobi, Houston, USA
e-mail: bixby@gurobi.com

E. Danna
Google, Mountain View, USA
e-mail: edanna@google.com

A. Lodi
University of Bologna, Bologna, Italy
e-mail: andrea.lodi@unibo.it

H. Mittelmann
Arizona State University, Tempe, USA
e-mail: mittelmann@asu.edu

T. Ralphs
Lehigh University, Bethlehem, USA
e-mail: ted@lehigh.edu

D. Salvagnin
Università degli Studi di Padova, Padua, Italy
e-mail: salvagni@dei.unipd.it

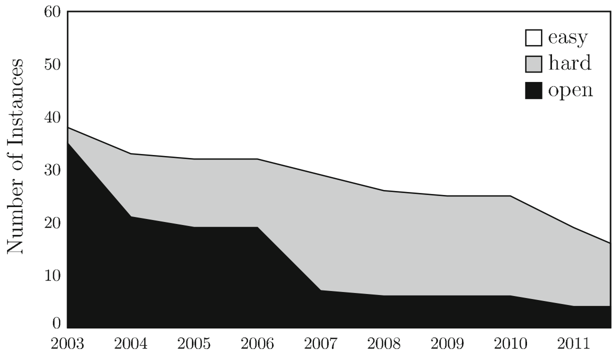


Fig. 1 Number of MIPLIB 2003 instances classified as *easy*, *hard*, and *open* over time

In this paper, we provide a detailed description of the instances in MIPLIB 2010, including their origin, information on the coefficient matrices, and the types of constraints and variables they contain. The complete library is available online at http://miplib.zib.de, where we have collected further information, such as references to papers that have used the MIPLIB as a test set for their algorithms. If available, we also provide a problem description in an algebraic modeling language, such as AMPL [57] or ZIMPL [70]. In addition to the role of MIPLIB as a test suite for integer programming algorithms, we strongly encourage investigation of alternative models for the given problems.

MIPLIB classifies instances into three categories: *easy* for those that can be solved within an hour on a contemporary PC with a state-of-the-art solver, *hard* for those that are solvable but take a longer time or require specialized algorithms, and finally *open* instances for which the optimal solution is not known. This classification is kept up-to-date on the website, and we ask for notification whenever an optimal solution is determined for an *open* instance. The website also contains details if specific settings have been used to solve an instance.

The progress in solving real-world MIP instances has been exceptional over the last decades. It is recorded in various articles [22,72,77] and numerous talks. One example is the solvability of the MIPLIB 2003 instances. As shown in Fig. 1, at the start of MIPLIB 2003 there were 22 easy, 3 hard, and 35 open instances. By the end of 2010, 41 were classified as easy, 15 as hard, and only 4 open instances remained.

Another showcase is the speedup of commercial MIP solvers. Figure 2 depicts the progress made by two of the commercial solver vendors with long traditions, CPLEX and XPRESS. These figures are based on their internal test sets and record purely the speedup due to algorithmic improvements. For CPLEX the geometric mean of the speedup is drawn and for XPRESS the reduction in total solution time. Though these measures cannot be compared directly, they both show the impressive performance improvements gained during the last years.

Combining a pure algorithmic speedup of 55,000 with the speedup in computing machinery, we see that solving MIPs has become something like 100 million times faster in the last 20 years. This easily translates into the difference between considering an instance to be trivial versus unsolvable.

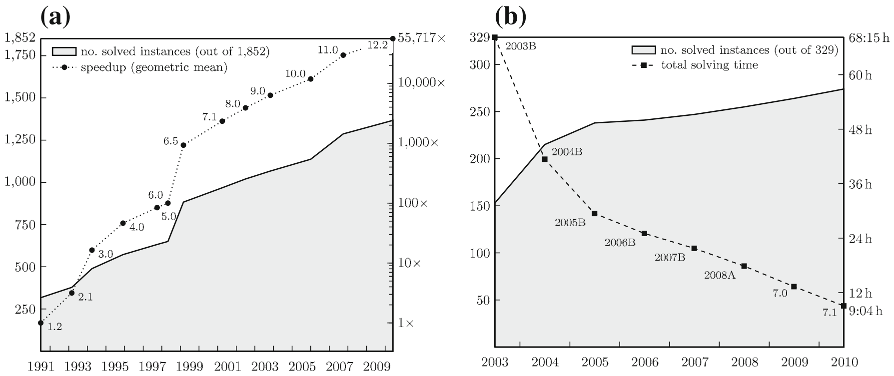


Fig. 2 Performance improvements of CPLEX and XPRESS. **a** Geometric mean speedup and number of instances solved by CPLEX versions 1.2 to 12.2 for an internal test set of 1,852 instances. **b** Total solution time and number of instances solved by XPRESS versions 2003B to 7.1 for an internal test set of 329 instances

This might give the misleading impression that all mixed integer programs are easy to solve nowadays. However, keep in mind that practitioners often experiment with tractable instances. In this sense, MIP codes tend to be tuned to more efficiently solve those models that are already known to be solvable. In order to compensate for this, we added a large set of instances to MIPLIB 2010 that are out-of-scope for today's solvers. As of this writing, the 361 instances of MIPLIB 2010 are classified as follows: 185 *easy*, 42 *hard*, and 134 *open*.

2 The test sets

During the initial discussions among the authors, it became evident that a single test set that also includes many very hard or even unsolved instances was not going to be sufficient. Researchers have often focused their attention on subsets of the MIPLIB 2003 instances that were suited to their particular topic of study. This often resulted in inadequate test sets because the full library only contained 60 instances, and added restrictions further reduced this size.

Therefore, we identified several areas for which dedicated test sets should be made available. Please note that a particular instance can be part of more than one test set. Table 1 gives an overview of all test sets.

2.1 What are the sources of the instances?

We started a call for instances on March 26, 2010. Submission was closed in October 2010. In total we received 1,108 instances from 57 contributors. Thanks to all of them! All contributed instances, unchanged, can be found at the MIPLIB homepage.[1] Additionally, we had access to a large number of instances from the NEOS server. Problems submitted to the solvers SCIP, FEASPUMP, QSOPT_EX in various formats were

[1] http://miplib.zib.de/contrib/submission2010.

Table 1 Overview of different test sets and the number of instances contained in each of these test sets

B	BENCHMARK (87 instances) contains only instances that can be solved to optimality by at least one solver within 2 h on a high-end PC. Except for the test sets CHALLENGE and UNSTABLE, at least one instance from each of the other test sets is included in this one
I	INFEASIBLE (20 instances) contains instances that are infeasible
P	PRIMAL (40 instances) contains instances for which the solution of the root LP relaxation has the same objective value as the optimal solution, i.e., the solver "only" has to find an optimal solution; the proof of optimality comes for free
X	XXL (11 instances) contains very large instances with respect to the number of variables, constraints, and non-zeros.
R	REOPTIMIZE (66 instances) contains instances for which the reoptimization of the sub-LPs takes an unusually long time.
T	TREE (52 instances) contains instances that (empirically) lead to large enumeration trees
U	UNSTABLE (21 instances) contains instances that have bad numerical properties and are likely to cause numerical troubles in the solver. This set is intended to test solver robustness
C	CHALLENGE (164 instances) is a compilation of hard-to-solve instances as well as instances that to our knowledge have not been solved to optimality. There are 21 instances in this set for which we have not yet been able to compute any feasible solution. Some of these may of course be infeasible

Table 2 Origin of instances

MIPLIB	2
MIPLIB 3 (excluding those from MIPLIB)	7
MIPLIB 2003 (excluding those from MIPLIB 3)	17
other publicly available	180
new contributions	155

translated to MPS format. For problems originally stated as AMPL model files, these files are preserved on the MIPLIB website.

Finally, we collected publicly available instances from the internet including instances available from the Berkeley Computational Optimization Lab [108], the Computational Optimization Research at Lehigh test set (COR@L) [109], and the DEIS—Operations Research Group Library of Instances [111]. Table 2 shows the distribution of instances according to their origin.

2.2 How were the instances selected?

In the following sections, we often mention that instances can be solved within certain time limits. Unless mentioned otherwise, this refers to experiments that were performed on dual core Intel Xeon E5420 2.5 GHz computers with 4 MB cache and 16 GB of main memory, running Linux in 64 bit mode.

After the submission of contributed instances was closed and the data mining of the public domain was finished, the initial candidate set contained about 2,000 instances.

Exclusion of trivial instances and (near) duplicates. In a first filtering step, duplicates were eliminated, homogeneous problem subsets were reduced, and instances which were too easy, i.e., could be solved by either SCIP or CBC in less than 1 min, were removed. The typical case for homogeneous test sets was that many variations of the same model with different data were submitted. When comprising these sets of similar problems, we tried to keep the variety while reducing the set to less than ten instances.

Examination and pre-selection by eight groups. The 659 remaining instances were examined independently by eight teams formed by the authors. Two instances entered the final test sets on short notice, as it was realized that certain classes of problems were not well represented in the candidate set. These were unitcal_7, a unit commitment model, and cov1075, a highly symmetric problem. Except for these two, all instances contained in the final test sets were part of this list of 659.

After the responses from the different groups had been evaluated, a first proposal of 100 instances plus 45 potential substitutes for the BENCHMARK set was compiled and circulated again. The minimal requirements for the BENCHMARK set were that first every instance could be solved to proven optimality within 24 h by at least one solver and second that each of the other test sets except the UNSTABLE and the CHALLENGE set, were represented in the BENCHMARK set by at least one instance.

Final refinement of the benchmark set. Starting from this proposal, several instances had to be exchanged or eliminated for one of the following reasons:

- problem classes were over- or underrepresented,
- instances proved to be too easy or too hard when checking them on different machines with the current developer versions of the involved software, or
- instances were discovered to be numerically difficult.

In cases where we had to choose between two instances, newly contributed ones were favored and instances for which the application was unknown, e.g., instances that have been provided via the NEOS server, were disfavored. Also, instances that showed a stable performance (see also Sect. 5) were favored. Finally, we tried to keep the overall running time of the BENCHMARK set reasonable.

It was far easier to exclude instances for the reasons named above than to include new ones that fulfilled all hard and soft requirements. We are satisfied to have arrived at a final BENCHMARK set with 87 instances. We believe that this test set is well-suited for benchmarking purposes: nearly all instances can be solved in less than 1 h at least one solver, but only a few of them can be solved within less than a minute by any current solver.

Most of the other test sets were created by predefined rules rather than dynamically, as with the BENCHMARK set. Because many criteria at least partially depend on the computational environment, e.g., number of simplex iterations, the numbers listed in the following paragraphs are only guiding values. The different test sets were double checked with different software packages on different machines to minimize the risk that some instance entered a particular test set just by chance.

The REOPTIMIZE *test set.* Warm-starting the simplex algorithm is one of the key requirements for efficiency in LP-based branch-and-bound algorithms. Often, reoptimizing an LP after changing a bound can be done in less than ten simplex iterations, whereas the initial LP solve takes thousands of iterations. However, there are a few cases when this advantage of the simplex vanishes. The REOPTIMIZE set is a collection of them. It contains instances for which LP reoptimization takes at least 500 simplex iterations on average. This does not include simplex iterations that have been performed at the root node. Furthermore, instances were only included if they required at least 100 nodes to solve.

The TREE *test set.* In the development of MIP solvers, many algorithmic improvements aim at reducing the size of the branch-and-bound tree. As a consequence, MIP solvers are often able to solve instances to proven optimality in the root node or after only a few hundred nodes. The test set TREE, however, comprises instances for which state-of-the-art MIP solvers perform a large amount of enumeration. The criterion applied for this test set was that the branch-and-bound tree created when solving the instance contains at least 1,000,000 nodes.

The XXL *test set.* Predicting the performance of a MIP solver just by knowing the dimensions of an instance is impossible. The markshare examples [1] show that problems with 60 variables and 6 constraints may be extremely hard to solve for state-of-the-art software packages. On the other hand, there are instances that contain 100,000 or more variables that can be solved in less than a minute. The test set XXL is dedicated to MIPs that are large-scale, but not too easy. It consists of problems with either more than a million variables, more than a million constraints, or more than 10 million non-zeros. Instances for which the majority of the variables and constraints can be eliminated by standard presolving methods were not considered.

The PRIMAL *and* INFEASIBLE *test sets.* A typical experience in solving a MIP is that finding an optimal solution is often much easier than proving its optimality. The test set PRIMAL is set up for instances for which the reverse holds. We included problems for which the bound given by the initial LP relaxation after presolving is equal to the MIP optimum. The only thing left for the solver to do is to find an optimal solution—the proof of optimality comes for free. We only included instances for which this appeared to be challenging.

Most MIP solvers are tuned for optimization problems, typically not for feasibility problems, such as the PRIMAL instances, and to our knowledge not at all for infeasible problems. The test set INFEASIBLE should help to investigate the behavior of MIP solvers for this kind of problem setting.

The CHALLENGE *and* UNSTABLE *test sets.* The new submissions contained several instances that are currently rather hard to solve. The CHALLENGE set contains problems that we could not solve in less than 2 h with any of the solvers. For some of them, the optimal solution is known; for many, it is not. As of this writing, no feasible solution is known, if one exists, for 21 of these instances.

Finally, we compiled the numerics test set UNSTABLE. MIPs are identified as numerically unstable if they exhibit some of the following properties: ill-conditioned basis matrices, large coefficients close to the internal numerical infinity values of the solvers, drastic performance changes with different choices of numerical tolerances, or different results reported by different solvers. See also the paragraph 'Condition numbers and numerical reliability' in Sect. 6.

3 The solution checker

Together with the MIPLIB, a consistency checker is provided to validate the answers produced by floating-point based MIP solvers. It tries to recognize incorrect results, while, to a certain extent taking into account the different feasibility policies of the codes.

3.1 Floating-point arithmetic and tolerances

Most MIP solvers (in particular, all codes considered in this paper) are based on floating-point arithmetic and work with tolerances to check solutions for feasibility and to decide on optimality. In their feasibility tests, solvers typically consider absolute tolerances for the integrality constraints and relative ones for linear constraints. Some normalize the activity of linear constraints individually, others directly scale the constraint matrix.

The tolerances affect solution times and solution accuracy, normally in opposite ways, and the solvers apply different strategies here. As an example, consider the instance rocII-4-11. Some solvers compute an optimal objective function value of -6.6556387, while others report -6.65275574. If one fixes all integer variables from the reported solution to the closest integer value and recomputes the continuous variables by solving the resulting LP with exact arithmetic, some of these post-processed solutions turn out to be infeasible with respect to exact arithmetic and zero tolerances.

It should be clear that this does *not* mean that any of the solvers made a mistake. It only means that the computed solution lies outside the feasible area described by the input file, but inside the extended feasible area created by reading in the problem and introducing tolerances. It is only solutions that are feasible in the latter sense that solvers attempt to deliver, and those are the solutions the checker checks. More precisely, the operation that rounds the reported value of the integer variables to the closest integer is *only* applied to compute fully reliable primal bounds for the MIPs as described in the paragraph 'MIP and LP solution values' in Sect. 6.

3.2 What do MIP solvers actually try to solve?

A binary double-precision number is represented in the form

$$(-1)^{sign} \cdot \left(1 + \sum_{i=1}^{52} 2^{-i} b_i\right) \cdot 2^e$$

where *sign* is a single bit indicating the sign, b_i are 52 binary digits used to represent the significant figures of the number i digits after the first binary digit and e defines the exponent, taking integer values in $e \in [-1022, \ 1023]$. The IEEE [117] standard defines which results should be returned by the basic arithmetic operations in order to maintain consistent behavior across different platforms.

Floating-point computations can be performed quickly on computers but the limited size of this representation has its disadvantages. The error incurred by a single operation is usually small but algorithms requiring many operations can accumulate and propagate these small errors, leading to errors of significant magnitude. A survey of the issues that can arise in floating-point computation can be found in [60]. MIP solvers implement techniques to handle these issues. We now discuss those techniques relevant to the solution checker for that phase of the process that precedes the actual solution of the problem.

The MPS file format which is used as a standard to define MIP instances requires the input numbers to be written in base 10 ASCII representation. Furthermore, the definition of the MPS file format specifies that each entry uses only 12 characters, so if a problem cannot be expressed exactly in this format, even the input file will be an approximation of the intended problem. Suppose a problem is defined in an MPS file as having the feasible region

$$\{x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^n\}.$$

As this problem is read in by the solver, the entries in A, b will be transformed to a binary representation, possibly modifying their values and changing the feasible region to $\{x : \widetilde{A}x \leq \widetilde{b}, x \geq 0, x \in \mathbb{Z}^n\}$.

In addition, due to inexact floating-point computation, the solvers need to introduce tolerances, hence relaxing (introducing a perturbation of) the feasible region. Typically relative tolerances are used. In order to do this efficiently and to improve the numerical properties of the model, the constraint matrix is usually scaled. As a result, solvers operate on something similar to

$$\{x : (\widetilde{Q\widetilde{A}})x \leq \widetilde{Q\widetilde{b}} + \mathbf{1}\varepsilon, x \geq -\mathbf{1}\varepsilon, x \in (\mathbb{Z} + [-\delta, \delta])^n\},$$

where ε and δ are tolerances for feasibility and integrality and $\mathbf{1}$ is the vector of all ones. As we can see, even the steps of parsing and scaling the problem can change its description. The entire solution procedure is then applied to this transformed problem.

Note that we have not even mentioned other preprocessing techniques that are applied by the solver in order to simplify the problem, such as removing redundant constraints and variables, tightening bounds and coefficients, or aggregating variables. All of this may lead to further modifications of the problem *before* the branch-and-bound and cutting plane phases are actually started.

In this regard, it is important to note that even though an instance is infeasible, it can become feasible by the extension of the feasible area due to the tolerances. This is one of the reasons why the solution checker does not try to test optimality and infeasibility but makes all tests by using solutions as returned by the solvers and with respect to the tolerances. Some different practice-oriented tests performed *outside the checker* are described in Sect. 4.

3.3 What does the solution checker test?

Given a linear programming problem, the combinatorial nature of the bases provides an efficient way to check both the optimality and the feasibility of a given solution: this is not the case for mixed integer programming problems. In the MIP setting, we cannot expect polynomially sized certificates of optimality to exist in general (assuming $\mathcal{P} \neq \mathcal{NP}$). However, it is possible to check the feasibility of a given solution in polynomial time and this is exactly the goal of our solution checker. Although conceptually simple, the implementation of such a checker still poses some questions:

- In which kind of arithmetic should we perform our calculations?
- Should we take our data from the internal floating-point representation of a solver or should we use the original text-based one?
- What tolerances should we use?

Concerning the first two questions, we decided to perform all of our computation with an arbitrary precision arithmetic package, namely GMP [114], taking the problem specification from a text-based file. In particular, we decided to use the industry standard MPS format. Note that reading the input from the MPS file, we perform no risky or potentially unsafe simplification of the coefficients; for example, if the input file contains the coefficient 0.3333333 we do not try to infer the "nicer" rational representation $1/3$, but we stick to the original 3,333,333/10,000,000. As far as the solution to check is concerned, in order to keep the solution checker independent of the particular solvers and the language in which they are implemented, we decided also to read the solution from a text-based input file, whose very simple format is described in the MIPLIB package.

The solution checker performs two tests: a feasibility test of the computed solution and a consistency check of the corresponding objective value. For the feasibility test, we followed the common habit of using two, possibly different, absolute tolerances. One tolerance is used to check the satisfaction of linear constraints, including bounds on the variables, and the other tolerance is used to check integrality constraints. For its default settings, the code uses an absolute tolerance of 10^{-4} for both cases.

In addition to the solution computed by the solver, the solution file contains the computed objective value. Checking its precision is conceptually trivial, at least for pure integer programming problems. However, given the potentially large absolute value of the objective function coefficients, sometimes larger than 10^{10}, it quickly becomes a delicate matter, if one insists on using only absolute tolerances. Therefore, the checker accepts the objective value if the absolute value of the difference between the objective value computed by the checker and that read from the input file is less than 10^{-4} or if the relative error, i.e., the absolute error divided by the maximum of the absolute value of the checker's result and 1, is less than 10^{-7}.

Finally, the solution checker reports the maximum violation of the linear and integrality constraints and the absolute value of the difference between the solver's and checker's objective values.

4 How to run a test, add a solver, and what the scripts do

For the first time, MIPLIB comes with a test engine to run different solvers in a defined way, to check the answers for consistency, and to generate a table summarizing the results. In this section, we briefly explain how this engine works, enabling users to incorporate new solvers, and to adjust the scripts to their needs.

In order to allow meaningful benchmarks, the test engine runs each solver in deterministic mode[2] if possible and applies comparable termination criteria, i.e., the relative gap cutoff is set to 0. Concerning the performance evaluation, the wall-clock time is measured externally and solution times are reported rounded up to the second. This means the time for parsing the input file and creating the model is included. The primal-dual gap at termination is computed, assuming minimization, as

$$\text{gap} = \frac{pb - db}{\inf\{|z|, z \in [db, pb]\}}$$

where pb and db are primal bound and dual bound, respectively. In case either no feasible solution was found, the problem was reported as infeasible, or $db \leq 0 \leq pb$, the gap is not computed and is marked as "`--`". In case $db = pb = 0$ a gap of 0 is reported. We chose this gap calculation since it is monotonically decreasing if the dual bound increases and the primal bound decreases. Moreover, this measure gives a worst case bound on the relative gap normalized by the optimal solution value.

The resulting table presents for each instance the name, the primal and dual bound along with the gap at termination, the number of branch-and-bound nodes, and the solution time. Additionally, the status of the MIP solver and the result of the solution check are displayed. The solver status is `ok` for instances where the solver claims optimality, `stopped` if the solving process was stopped due to imposed limits, and `abort` otherwise, e.g., for termination due to errors in the MIP solver. Each solver is expected to write the best solution found into a solution file. If a solution file was generated, the solution gets checked and the solution status is `ok` or `fail` depending on whether or not the solution passed all tests of the solution checker (see Sect. 3). In case no solution was found, e.g., for infeasible problems, we report "`--`" as the solution status.

[2] That is, given the same machine and the same settings the solver should perform identical runs. Especially in a multi-thread environment special care has to be taken to achieve this since all dependencies on the relative timing of the threads have to be removed.

Optionally, a file with known primal bounds can be given. If such a file is provided and only after the solution passed the tests of the solution checker, the primal bound reported by the solver will be compared with the value given in the file. If a finite primal bound is stated in the file and the solver reports either infeasibility or an optimal solution with a significantly worse objective value, the solution status is set to `mismatch`. In case the solution file reports infeasibility, but the solver returns a solution within the tolerances of the solution checker, no mismatch is reported. Finally, if the solution checker code encountered internal problems, the solution status is set to `error`.

It is important to note that the comparison with respect to a primal solution from a file is independent of the solution checker, and requires a primal bound that is fully reliable. Here we computed the primal bounds by rounding the integer variables to the closest integer and then used exact arithmetic for solving the resulting LPs as described in paragraph 'MIP and LP solution values' of Sect. 6.

At the end, we state how many instances were solved to optimality and passed the solution tests (`solved`), the number of `failed` instances (solution process was aborted or solution test failed), and the remaining number of `stopped` instances.

We now briefly sketch how the test engine is organized. The engine assumes a Linux environment, adoption to other UNIX-like operating systems should be possible with minor modifications. The main folder contains the files "Makefile" and "README" and the folders "bin", "checker", "instances", "scripts", and "testsets". After running the first automated test, there will be a folder named "results". The "Makefile" provides all the functionality of the test engine and the "README" is an introduction to the environment. The folder "bin" should contain the solver binaries or a (soft) link to them. The environment will check this folder for the requested solver binary. The sources for the solution checker, mentioned in Sect. 3, are stored in the folder "checker". The solution checker can be compiled via the provided "Makefile" of the test engine using the command `make checker`. All MIPLIB 2010 instances are stored in "instances". The folder "testsets" includes, for each test set described in Sect. 2, a file with the location of its instances, and for the BENCHMARK set a file with known primal bounds that are believed to be optimal. Log files and solution files of test runs and corresponding tables will be stored in a folder called "results" and all relevant scripts can be found in the folder "scripts".

The complete evaluation process is triggered by the `test` target in "Makefile". For example,

```
make SOLVER=xyz TEST=benchmark TIME=3600 test
```

runs solver XYZ on the benchmark test set with a time limit of 1 h. More precisely, in this way, the main driver "run.sh" is invoked. It calls the MIP solver for every instance listed in the file "benchmark.test", asks for the best solution, and applies the solution checker. However, the solver-specific part, such as setting the time limit and getting the solution, is encapsulated in a separate script "run_xyz.sh". After running the solver, "run.sh" starts "parse.awk", which parses the generated log file and produces the summary table. The evaluation, including primal-dual gap computation, time measurements, and determination of the solution status, is standardized and directly implemented in "parse.awk", whereas the information from the solver-specific log files, e.g., dual bound and node count, is actually obtained via auxiliary scripts such as "parse_xyz.awk".

Currently, we support CBC, CPLEX, GUROBI, MOSEK, SCIP, and XPRESS. In order to include other MIP solvers, only the solver-specific information for running the solver and evaluating its log files have to be provided by "run_mysolver.sh" and "parse_mysolver.awk". The template files "parse_xyz.awk" and "run_xyz.sh" can be used as a basis. They are located in the "scripts" folder. In order to test instances that are not shipped with MIPLIB 2010 a file "mytestset.test" with their locations has to be added to the "testsets" folder.

5 Variability of MIP Solver performance

When running computational experiments, we often experience differences in behavior on different platforms when optimizing the same model with the same solver. The input format, e.g., the order of the constraints, can also change the solution process. We use the term *performance variability* [42] to denote such changes in performance measures for the same problem that are caused by seemingly performance-neutral changes in the environment or the input format. Loosely speaking, performance variability comprises unexpected changes in performance.

Note that other changes, such as minor variations in the model formulation, also affect performance in a way that is difficult to predict. Adding or removing redundant constraints or variable bounds can have a major impact on the performance of modern MIP solvers [4]. Some of these changes are also automatically applied by preprocessing, in which case their effects are compounded with other sources of variability.

5.1 Reasons for performance variability

One root cause of performance variability is imperfect tie-breaking. When solving a MIP, most decisions are taken by computing a score for several candidates and choosing the candidate with the highest score. For example, a score based on pseudocosts may be computed for several variables that are candidates for branching and the variable with the highest score is chosen [20]. If there is a tie among the top candidates, then a series of secondary criteria should be used to make the final decision [2]. If tie-breaking is imperfect, selections may be made arbitrarily, based on the order in which the candidates are considered or influenced by rounding errors in the score computation, which will differ from platform to platform. Similarly, some decisions consider a subset of candidates: for example, strong branching [76] is typically applied to the top N branching variable candidates according to some criterion. If there are more than N candidates tied for the best score, then the order in which candidates are considered will determine the variables for which strong branching is performed. Another factor is that compiler optimizations may reorder arithmetic operations and influence the outcome of floating-point computations; this might introduce a tie-breaking effect for values that should be equal or it might lead to tied values that should not be the same.[3]

Once the path in the branch-and-cut tree diverges, the entire subsequent resolution is affected. Altering the choice of a single branching variable may cause entirely different subtrees to be explored; the LP solutions at the child nodes could be different, leading to different cuts and different starting points for primal heuristics, which in turn leads to different integer solutions being found, the objective cutoff being updated differently, different nodes being pruned, etc. As we see from such a cascade of events, even a small divergence may lead to a completely different behavior of the solver and hence to a significant performance difference.

Also, for many models, the optimal basis of the root LP is not unique. If the LP basis is different, then cuts and LP-based primal heuristics applied at the nodes to find additional feasible solutions will be applied differently, and consequently the whole solution process will change. It is currently unknown whether the variability is correlated in any way with the number of alternate optimal LP bases for the root node.

Performance variability also depends on the intrinsic characteristics of the model; some structures may create more variability than others. It also depends on the characteristics of the MIP solver; some solvers may have more robust tie-breaking mechanisms or more sophisticated algorithms to recover from a bad decision. Many questions are open. In particular, what is the importance of each factor to variability and how should we change MIP solvers and MIP models so as to reduce variability. It is not even clear whether performance variability should exclusively be seen as a disturbing factor. For massive parallel MIP solvers, see, e.g., [97], it can even be exploited.

5.2 Generating and measuring performance variability

In order to study performance variability, we need a way to generate a large number of observations for the same model. Clearly, it is neither sufficient nor practical to run experiments on many different computers. Previous studies [42] have shown that a good variability generator is to permute columns and rows in the original model. This generator affects all types of problems and all components of a typical MIP solver. For instances of practically relevant size, many different permutations can be applied to create as many observations as needed.

For comparison reasons, it is desirable to summarize the performance variability into one number per model and solver. We choose as the *performance variability score* the coefficient of variation of X, where X is the random variable representing the performance of the given solver for the given model. For example, the performance can be measured as the solution time to optimality or the time to the first solution. The coefficient of variation is simply the standard deviation divided by the average: it captures the variations in performance and its normalization allows us to compare the variability independent of the computational difficulty of the model. This definition

[3] While it can be expected that the same binary will run identically on similar CPUs in general, modern compilers, e.g., the Intel C/C++ compilers [116], can detect the precise type of the CPU and use different instructions depending on the particular CPU. Since the internal precision of x86 type CPUs can differ depending on the instruction set used, the rounding of the last digit can be different.

is quite natural and allows us to interpret the variability score in relationship with the probability of a given performance degradation.

In practice, the coefficient of variation is estimated on a sample of observations provided by solving multiple permutations of the model: the quality of the estimate increases with the number of permutations. Given n permutations and corresponding running times t_i, $i = 1, \ldots, n$, we compute the variability score VS as follows:

$$VS = \frac{1}{\sum_{i=1}^{n} t_i} \cdot \sqrt{\sum_{i=1}^{n} \left(t_i - \frac{\sum_{i=1}^{n} t_i}{n} \right)^2}.$$

In Column VS of Table 7 the performance variability score for time to optimality with SCIP/SPX is shown. It is sampled from 100 different permutations for each model. However, we calculated the score only for models that SCIP/SPX could solve in less than 4 h, using the original formulation. In addition, we imposed a time limit of 10 h, which, of course, influences the variability score whenever a permutation hits this limit. In Table 7 those scores that are affected by the time limit are printed in *italic*.

We present variability for solution time rather than number of simplex iterations, as in [42], because time is easier to compare across solvers. Indeed, solvers may count iterations differently; for example, iterations in sub-MIPs may or may not be counted, which makes the comparison difficult. Besides, not all the work done in a MIP solver is based on the simplex algorithm, so counting iterations gives a less comprehensive picture than measuring time.

Note that the variability score is an estimator over a limited sample of the true variability. Therefore, small differences in variability scores should not be given too much consideration.

5.3 Results on performance variability

Performance variability due to permutations can be observed for all tested instances. There was no instance for which all 100 permutations showed the same behavior. The minimum observed impact was for the instance mik-250-1-100-1, where the ratio of the maximal and the minimal solution times was 1.29. The largest ratio was 915 for the instance tanglegram2; the fastest permutation took a few seconds, the slowest nearly 2 h.

The total running time of all 67 original, unpermuted instances for which we performed this test was 45.1 h. If we had an oracle that, hypothetically, always selected the best permutation for each individual instance, the total running time would be reduced to 25.5 h, which corresponds to a speedup factor of 1.77. On the other hand, always using the worst permutation would increase the running time by a factor of 3.82. This indicates that the negative outliers are more “extreme”, i.e., the distribution is not symmetric.

It seems to be a natural question whether the original formulation has an advantage over the permuted ones. Indeed, if we compare the solution time of the unpermuted instances to the average solution time taken over the 100 permutations, we observe a speedup of 14%. There is evidence that this might be mainly due to the more extreme behavior of negative outliers. If we compare the performance of the original formulations to the median, this advantage shrinks to 4% improvement. Said differently, the probability of improving performance by permuting the instance is about as high as the probability to deteriorate performance, but the average improvement is smaller than the average deterioration.

Overall, the range of variability scores is between 0.05 for the instance mik-250-1-100-1 and 2.23 for neos-916792, as can be seen in Column *VS* of Table 7. Figure 3a–e shows the distribution of performance for specific instances. For the instances ex9, pg5_34, neos13, bnatt350, and enlight13, each of the dots in the corresponding diagram depicts the performance of one permutation when being solved with SCIP/SPX. They have been sorted by non-decreasing solution time; the black dot corresponds to the performance of the original formulation.

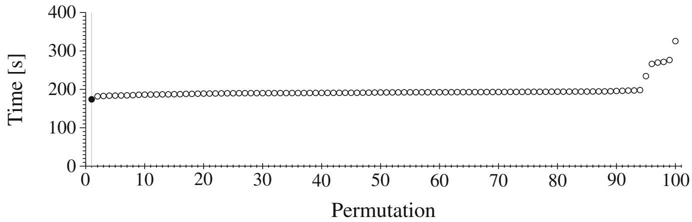
In Fig. 3a, the vast majority of the permuted instances perform very similar, with only a few outliers. The original formulation is superior to the others. The most common case, however, is illustrated in Fig. 3b. The different solution times are nearly uniformly distributed around the median value. There are a few negative outliers. The original formulation is "somewhere in the middle". Figure 3c shows the interesting case of clusters. Most of the permutations need around 12 min, but there is a significant accumulation point at 1:40 h. In order to have a smaller scale to make the accumulation point better visible in the picture, all permutations that took 3 h or longer are treated as if they hit a time limit of 3 h. Figure 3d and 3e depict bad cases for which permuting the model can lead to nearly arbitrary changes in the performance. For the instance enlight13, the best ten permutations need less than 3 min, but nearly 20% hit the time limit of 10 h.

Another source of variability is depicted in Fig. 4, which shows the performance variability of a single instance depending on the number of threads. The computations where done on a 32 core machine with 8 AMD Opteron 8384 processors at 2.7 GHz. As can be seen, both performance measures go up and down quite arbitrary with no visible pattern. If we compare neighboring bars, we see that adding one additional thread leads to fewer evaluated nodes in 17 cases, in 14 cases it results in more search nodes. The same holds for the solution time: in 14 cases the solver got faster when using one more thread, in 17 cases it slowed down.

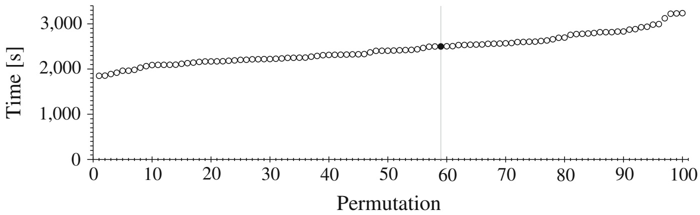
5.4 Consequences for benchmarking

Performance variability affects all standard objectives of benchmarking, such as comparing different solvers, comparing different parameter settings for the same solver, or comparing a new algorithm to an existing algorithm. In particular, it is important to take performance variability into account during the analysis of experimental results when the performance difference is small, the variability of the models is high, or the stopping criterion is highly variable. Small test sets are more prone to disturbances caused by performance variability. This is one reason why MIPLIB 2010 provides more instances than its predecessors.

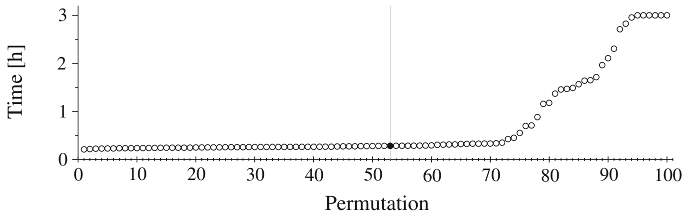
Performance is not deterministic. Instead, it is a random variable that can be sampled with computational experiments. In order to isolate the signal from the noise and



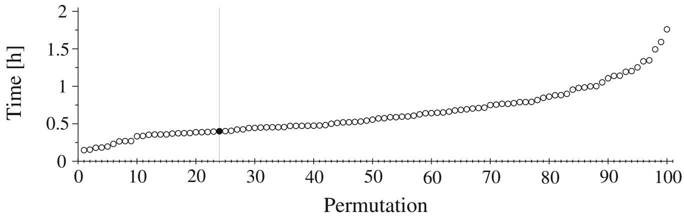
(a) Instance ex9



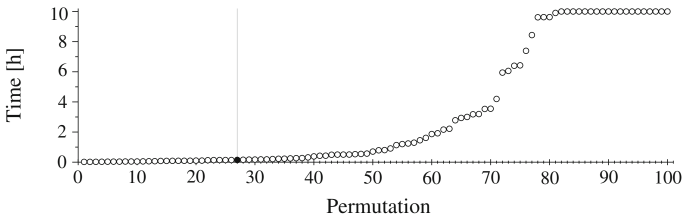
(b) Instance pg5_34



(c) Instance neos13



(d) Instance bnatt350



(e) Instance enlight13

Fig. 3 Solution times for 100 permutations

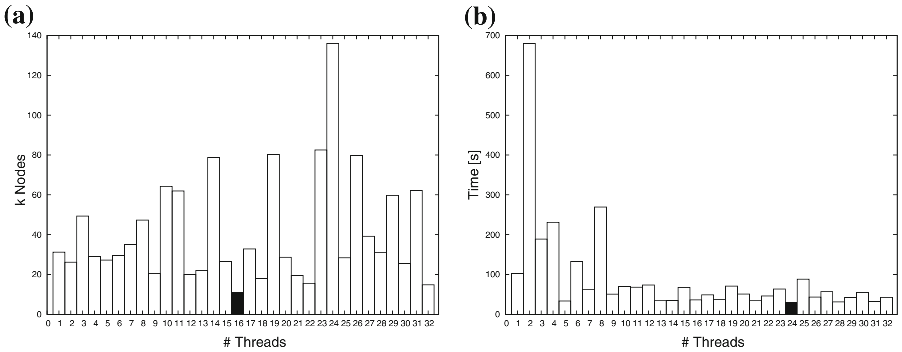


Fig. 4 Example of performance variability depending on the number of threads. Instance roll3000 on a 32 core computer. Filled bar indicates minimum. **a** Total number of nodes explored (CPLEX). **b** Wall clock solution time (GUROBI)

draw robust conclusions, performance needs to be studied with appropriate statistical tools. First, it is useful to obtain a large data sample, by running experiments on a large set of models or by artificially multiplying the number of observations by running experiments on permuted models for multiple permutations. Secondly, descriptive statistics such as the average or the geometric mean give limited insights. Instead, robust indicators such as truncated averages and rank statistics are more resistant to outliers. Performance profiles [47] are also useful. Finally, inferential statistics, such as statistical tests and confidence intervals, give the most insight and allow questions such as "how likely is it that the performance change is created by variability rather than by genuine algorithmic changes?" to be answered.

Variability should be taken into account not only when studying performance, but also when studying the correctness of computation. A different path in the branch-and-cut tree often leads the MIP solver to find a different integral solution because most models have several alternate optimal solutions. In all cases, the integral solution returned should have the expected objective value and respect all tolerances. It is a good idea to verify this on several permutations of the same model.

6 The instance catalog

In this section, we give an overview of all MIPLIB 2010 instances. Table 3 provides information about their origin and application. For each instance, we list originators or submitters, a short description of the application and references to publications in which the instance was used. Additionally, there are a large number of instances from NEOS and COR@L for which, unfortunately, no information is available to us. Instances with the same originator and application are grouped in one row. The 26 instances from previous versions of MIPLIB are marked by *italic* names.

Problem statistics. Table 4 gives statistics about the instances. In particular, for each instance, we give the number of *Rows*, *Columns*, and *Non-zeros*. Furthermore,

Table 3 Descriptions and references for MIPLIB instances. Instances coming from a previous version of MIPLIB are listed in *italic*

Name	Originator and description
30n20b8	E. Coughlan, M. Lübbecke, J. Schulz [40] Multi-mode resource leveling with availability constraints; precedence and resource constrained scheduling problem
30_70_45_095_100	J. Walser [102,103] Geographic radar station allocation
50v-10	S. Bisaillon Network loading instance
a1c1s1, b2c1s1	M. Vyve, Y. Pochet [55,91] Lot sizing instances
acc-tight4, acc-tight5, acc-tight6	J. Walser [83,104] ACC basketball scheduling instances
aflow40b	T. Achterberg [89] Arborescence flow problem on a graph with 40 nodes and edge density 0.9
air04	G. Astfalk [18] Airline crew scheduling set partitioning problem
app1-2	E. Danna Undisclosed industrial application from Google
ash608gpia-3col	M. Pfetsch Infeasible and highly symmetric graph 3-coloring assignment formulation
atlanta-ip, *msc98-ip*	E-Plus, D. Bienstock, A. Bley, R. Wessäly [26] Min-cost network dimensioning problems with a finite set of link capacities for each bidirected link, unsplittable shortest path routing, path restoration for single node failures, and routing path length restrictions
atm20-100	M. Galati [59] ATM cash management problem
bab1	E. Swarat, L. Traverso, J. Buwaya Integrated vehicle routing and crew scheduling of toll inspectors on German highways
bab3, bab5	E. Swarat Vehicle routing with profits and an integrated crew scheduling problem formulated by two coupled multi-commodity flow problems
beasleyC3, g200x740i, k16x240, mc11, p80x400b, p100x588b, r80x800	F. Ortega, L. Wolsey [85] Fixed cost network flow problems
berlin_5_8_0, usAbbrv-8-25_70	G. Klau [54] Railway optimization problems
bg512142, dg012142	A. Miller [8,54,98] Multilevel lot-sizing instances

Table 3 continued

Name	Originator and description
biella1, dc1c, dc1l, dolom1, nsr8k, siena1	Double-Click SAS [54,55]
	Crew scheduling instances
bienst2, binkar10_1	H. Mittelmann
	Relaxed versions of problems bienst and binkar10
bley_xl1	A. Bley
	Min-cost network dimensioning problem with finite sets of link capacities and unsplittable flow routing
blp-ar98, blp-ic97	M. Lübbecke [34,54]
	Railway line planning instances
bnatt350, bnatt400	T. Akutsu [9]
	Model to identify a singleton attractor in a Boolean network, applications in computational systems biology
buildingenergy	K. Pruitt
	Model to determine the minimum cost design and dispatch of a distributed generation system for a commercial building
cdma	S. Bisaillon
	3G wireless multiplexing communication model
circ10-3	[122]
	Instance from the 2010 SAT conference pseudo-Boolean competition
co-100	A. Werner [21]
	Model from optical access network planning
core2536-691, core4872-1529	A. Caprara, M. Fischetti, P. Toth [35]
	Set covering instances coming from Italian railway models
cov1075	F. Margot [37,79]
	Problem of selecting a minimum collection of 7-subsets containing all 5-subsets of a ground set of 10 elements
csched007, csched008, csched010	T. Yunes [107]
	Cumulative scheduling problem instances
d10200, d20200, leo1, leo2	COR@L test set [109]
	Instances coming from the COR@L test set with unknown origin
dano3mip, *danoint*, newdano	D. Bienstock [65]
	Telecommunications applications
datt256	J. Dattorro [43,110]
	Model to find solution to the "Eternity II" puzzle [112]
dfn-gwin-UUM, germany50-DBM, janos-us-DDM, nobel-eu-DBE, zib54-UUE	C. Raack [5,93] Network design, link dimensioning models for problems in the SNDlib [84,125]
ds-big	R. Borndörfer [64]
	Bus driver duty scheduling problem

Table 3 continued

Name	Originator and description
eil33-2, eilA101-2, eilB101	J. Linderoth [74,126] Set partitioning problem approximations for capacitated vehicle routing problem instances from TSPLIB
enlight9, enlight13, enlight14, enlight15, enlight16	A. Zymolka Model to solve instances of a combinatorial game "EnLight"
ex9, ex10	I. Ober Formulations of Boolean SAT instances
ex1010-pi	[122] Logic synthesis problem from the 2010 SAT conference pseudo-Boolean competition
f2000, hanoi5	[122] Reformulated SAT instances from the 2010 SAT conference pseudo-Boolean competition
ger50_17_trans	C. Raack[27] Multi-layer network design problem using a link-flow formulation over a path-flow formulation
germanrr	Q. Chen [109] Model from a German railroad company
glass4	I. Luzzi [78] Nesting instance
gmu-35-40, gmu-35-50, gmut-75-50, gmut-77-40	N. Konnyu Timber harvest scheduling models
go19	R. Miyashiro, Y. Yano, M. Muramatsu [82] Instance of the maximum string problem in the Go board game: to find a position of stones that maximizes the number of live "strings" on the board
harp2	M. Savelsbergh [24] Unknown application
hawaiiv10-130	J. van Dinter Unit commitment model
ic97_potential	L. Peeters [88] A model for cyclic railway timetable optimization
iis-100-0-cov, iis-bupa-cov, iis-pima-cov	M. Pfetsch [11,90] Set covering instances arising from irreducible infeasible subsystem covering problems
in	A. Fügenschuh Packing of paths, multicommodity flow formulation
ivu06-big, ivu52, rvb-sub	S. Weider [32,105] Set partitioning instances resulting from a column generation algorithm used for duty scheduling in public transportation

Table 3 continued

Name	Originator and description
lectsched-1, lectsched-1-obj, lectsched-2, lectsched-3, lectsched-4-obj	H. Schilly [95] University lecture scheduling instances
liu	X. Liu [3]
	Floorplan and placement problem in the physical design of VLSI circuits
lotsize	D. Gade, S. Küçükyavuz [58]
	Multi-item lot sizing with service level constraints
lrsa120	M. Atlihan
	Model to break a 120 bit RSA key
m100n500k4r1	L. Torres [100]
	Set packing problem with 4 ones per column
macrophage, methanosarcina, tanglegram1, tanglegram2, toll-like	F. Hüffner [28,68] Balanced subgraph instances coming from applications in bio-informatics: finding monotone subsystems in gene regulatory networks and finding optimal layouts of tanglegrams
map06, map10, map14, map18, map20	K. Ahmadizadeh [6,96]
	Land parcel selection problems motivated by Red-Cockaded Woodpecker conservation problem
markshare_5_0	G. Cornuéjols, M. Dawande [1,39]
	Market sharing problem
maxgasflow, transportmoment	G. Gamrath
	Transport momentum maximization in a capacitated gas network, transportmoment forbids cycles by pseudo pressures
mcsched, npmv07	Q. Chen [109]
	Unknown application
mik-250-1-100-1	A. Atamtürk [14,108]
	Problem with mixed integer knapsack constraints
mine-90-10, mine-166-5, reblock67, reblock166, reblock354, reblock420	A. Bley [25] Multi-period mine production scheduling instances
mining	K. Eurek
	Unspecified mining application
mkc	J. Kalagnanam, M. Dawande [45,56]
	Multiple knapsack problem with color constraints
momentum1, *momentum2*, *momentum3*	T. Koch [51]
	Snapshot based UMTS planning problems, having a very wide dynamic range in the matrix coefficients and tending to be numerically unstable

Table 3 continued

Name	Originator and description
mspp16	P. Troubil, P. Holub, M. Liška, H. Rudová [67,101] Media Streams Planning Problem—A network optimization problem regarding routing of multiple concurrent multimedia streams with bandwidth close to capacity of network links
mzzv11	S. Lukac [94] Railway slot allocation problems
n3-3, n4-3, n9-3, n15-3	A. Atamtürk [13,108] Capacitated network design problems
n370a, n3700, n3705, ran14x18, ran14x18-disj-8, ran16x16	J. Aronson [99] Fixed charge transportation problems
n3div36, n3seq24	R. Meirich [81] Static line planning models on the Dutch IC network
nag	N. Shenoy [109] Unknown Application
nb10tb	S. Bisaillon Forestry industry model
net12	P. Belotti [19,55] Network design instance
netdiversion	C. Cullenbine [41] Directed network diversion problem
noswot	J. Gregory, L. Schrage [23] Unknown application
ns4-pr3, ns4-pr9, nu60-pr9, nu120-pr3	A. Atamtürk, D. Rajan [15,108] Multicommodity flow capacitated network design problems
ns1111636, ns1158817	H. Mittelmann, NEOS Server [121] Network routing problems
ns1631475, ns2137859	H. Mittelmann, NEOS Server [121] Traveling salesman problem models
ns1702808, ns1905797, ns1905800, ns2118727	H. Mittelmann, NEOS Server [121] Vehicle routing problems
ns1853823, ns1854840	H. Mittelmann, NEOS Server [121] Network flow problems
ns1856153, ns1904248	H. Mittelmann, NEOS Server [121] Sensor placement problems
ns2081729	H. Mittelmann, NEOS Server [121] Short-term scheduling in a multiproduct sequence dependent facility
ofi	L. Poderico Natural gas supply portfolio optimization

Table 3 continued

Name	Originator and description
opm2-z7-s2, opm2-z10-s2, opm2-z11-s8, opm2-z12-s7, opm2-z12-s14	D. Espinoza Problems coming from precedence constrained knapsacks arising in mining applications
p2m2p1m1p0n100	B. Krishnamoorthy, G. Pataki A 0-1 knapsack problem constructed to be difficult
p6b	B. Borchers Maximum independent set problem on a component of the graph *1et.2048* from the collection of N. Sloane [118]
pb-simp-nonunif	[122] Retrieving haplotype information from DNA samples using Haplotype Inference by Pure Parsimony
pg, pg5_34	M. Dawande [44] Multiproduct partial shipment models
pigeon-10, pigeon-11, pigeon-12, pigeon-13, pigeon-19	S. Allen [10] Instances of 3D packing (container loading) problems
probportfolio	S. Ahmed, S. Dey, F. Qiu Sample average approximation formulation of a probabilistic portfolio optimization problem
protfold	A. Fügenschuh [46] Protein folding instance
pw-myciel4	A. Koster Model to compute the pathwidth of Mycielski-4 instance from DIMACS graph coloring database
qiu	Y. Chiu, J. Eckstein [48–50] Fiber-optic network design, logical SONET ring level
queens-30	A. Mahajan [124] Models the problem of placing as many queens on a 30 by 30 chess board as possible so that each queen threatens at most one other queen
rail01, rail02, rail03	T. Schlechte [33] Track allocation problem modeled as arc coupling problem
rail507	A. Caprara, M. Fischetti, P. Toth [35,55] Railway crew scheduling
ramos3	F. Ramos Set covering problem from a product manufacturing application
rmatr100-p5, rmatr100-p10, rmatr200-p5, rmatr200-p10, rmatr200-p20	D. Krushinsky [61] Instances coming from a formulation of the p-Median problem using square cost matrices
rmine6, rmine10, rmine14, rmine21, rmine25	D. Espinoza Set of instances coming from open pit mining over a cube considering multiple time periods and two knapsack constraints per period

Table 3 continued

Name	Originator and description
rococoB10-011000, rococoC10-001000, rococoC11-011100, rococoC12-111000	A. Chabrier, E. Danna, C. Le Pape, L. Perron [36] Models for dimensioning the arc capacities in a telecommunication network
rocII-4-11, rocII-7-11, rocII-9-11	J. Rambau Optimal control of opinion dynamics
roll3000	L. Kroon [55] Rolling stock and line planning instances
satellites1-25, satellites2-60, satellites2-60-fs, satellites3-40, satellites3-40-fs	A. Ceselli, R. He Satellite scheduling instances
sct1, sct5, sct32	Siemens Assembly line balancing for printed circuit board production
set3-10, set3-15, set3-20	K. Akartunalı, A. Miller [7,8,120] Multi-item lot-sizing with backlogging
seymour	W. Cook, P. Seymour A set-covering problem that arose from work related to the proof of the 4-color theorem
seymour-disj-10	M. Ferris, G. Pataki, S. Schmieta [53] The *seymour* instance after adding 10 rounds of disjunctive cuts
shipsched	E. Günther, M. Lübbecke A ship scheduling problem on the Kiel Canal
shs1023	C. Helmberg [66,115] Joint online truck scheduling and inventory management for multiple warehouses
sing2, sing161, sing245, sing359, uc-case3, uc-case11	D. Espinoza Unit commitment problems (electricity production planning problems) coming from the Chilean electricity system, they have either 1- or 2-week planning horizons, and include constraints on minimum on-off time for the power plants, ensure some *reserve* energy in the system, and minimize global operation costs
sp97ar, sp98ic, sp98ir	J. Goessens, S. v. Hoessel, L. Kroon [55,63] Railway line planning instances
splan1	C. Helmberg, A. Lau [71] University course timetabling instances
stockholm	L. Bai, P. Rubin [16,17] Toll booth placement problem
stp3d	T. Koch [70] Steiner tree packing instance in a 3 dimensional grid-graph, LP relaxation is highly degenerate

Table 3 continued

Name	Originator and description
sts405, sts729	J. Linderoth [86]
	Steiner triple system problems
swath	D. Panton [87]
	Model arising from the defense industry, involves planning missions for radar surveillance
t1717, t1722	R. Borndörfer [29,30]
	Vehicle scheduling set partitioning problems from Berlin's Telebus handicapped people's transportation system
timtab1	C. Liebchen, R. Möhring [73]
	Public transport scheduling problems
triptim1, triptim2, triptim3	R. Borndörfer [31] Trip timetable optimization problems
tw-myciel4	A. Koster
	Model to compute the treewidth of the Mycielski-4 instance from the DIMACS graph coloring database
uct-subprob	G. Lach
	Subproblem of a university course timetabling problem
umts	C. Polo [55,92]
	Telecommunications network model
unitcal_7	R. O'Neill
	California 7-day unit commitment problem
van	C. Mannino, E. Parrello [55]
	Telecommunications network model
vpphard, vpphard2	C. Cardonha
	Vehicle positioning problem instances
wachplan	S. Orlowski
	Shift planning model to assign crew members to shifts for a sail training trip
wnq-n100-mw99-14	[122]
	Weighted n-queens problem with an additional separation constraint
zib01, zib02	T. Koch
	Group channel routing on a 3D grid graph

neos6, neos13, neos15, neos16, neos18, neos-476283, neos-506422, neos-506428, neos-520729, neos-555424, neos-631710, neos-686190, neos-693347, neos-738098, neos-777800, neos-785912, neos788725, neos-799711, neos-807456, neos808444, neos-820146, neos-820157, neos-824661, neos-824695, neos-826650, neos-826694, neos-826812, neos-826841, neos-847302, neos-849702, neos858960, neos-859770, neos-885086, neos-885524, neos-911880, neos-916792, neos-932816, neos-933638, neos-933966, neos-934278, neos-935627, neos-935769, neos-937511, neos-937815, neos-941262, neos-941313, neos-942830, neos-948126, neos-952987, neos-957389, neos-984165, neos-1109824, neos-1112782, neos-1112787, neos-1140050, neos-1171692, neos-1171737, neos-1224597, neos-1225589, neos-1311124, neos-1337307, neos-1396125, neos-1426635, neos-1426662, neos-1429212, neos-1436709, neos-1440225, neos-1440460, neos-1442119, neos-1442657, neos-1601936, neos-1605061, neos-1605075, neos-1616732, neos-1620770, ns894236, ns894244, ns894786, ns894788, ns903616, ns930473, ns1116954, ns1208400, ns1456591, ns1606230, ns1644855, ns1663818, ns1685374, ns1686196, ns1688347, ns1696083, ns1745726, ns1758913, ns1766074, ns1769397, ns1778858, ns1830653, ns1952667, ns2124243, ns2017839
Instances coming from the NEOS Server [121] with unknown applications

Table 4 Problem statistics

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
30_70_45_095_100	12,526	10,976	46,640	10,975		1	3	3		P	Easy
30n20b8	576	18,380	109,706	11,036	7,344		302	1.566408		B	Easy
50v-10	233	2,013	2,745	1,464	183	366	3311.179984123	2879.065687		C	Hard
a1c1s1	3,312	3,648	10,178	192		3,456	11503.444125	997.529583		C	Hard
acc-tight4	3,285	1,620	17,073	1,620			0	0		RP	Easy
acc-tight5	3,052	1,339	16,134	1,339			0	0		BRP	Easy
acc-tight6	3,047	1,335	16,108	1,335			0	0	0.001	RP	Easy
aflow40b	1,442	2,728	6,783	1,364		1,364	1168	1005.664817		B	Easy
air04	823	8,904	72,965	8,904			56137	55535.436388		B	Easy
app1-2	53,467	26,871	199,175	13,300		13,571	−41	−264.601651	0.001	B	Easy
ash608gpia-3col	24,748	3,651	74,244	3,651			*Infeasible*	2		BI	Easy
atlanta-ip	21,732	48,738	257,532	46,667	106	1,965	90.009878614	81.243199		C	Hard
atm20-100	4,380	6,480	58,878	2,220		4,260	?	2141734.684878	0.006	C	Open
b2c1s1	3,904	3,872	11,408	288		3,584	?	4034.218333		C	Open
bab1	60,680	61,152	854,392	61,152			?	−286923.824223		C	Open
bab3	23,069	393,800	3,301,838	393,800			?	−733091.180443		C	Open
bab5	4,964	21,600	155,520	21,600			−106411.8401	−124657.641413		B	Easy
beasleyC3	1,750	2,500	5,000	1,250		1,250	754	40.426829		B	Easy
berlin_5_8_0	1,532	1,083	4,507	794		289	?	52		C	Open
bg512142	1,307	792	3,953	240		552	?	144364.073815		C	Open
biella1	1,203	7,328	71,489	6,110		1,218	3065005.78	3060037.430763		B	Easy
bienst2	576	505	2,184	35		470	54.6	11.724138		B	Easy
binkar10_1	1,026	2,298	4,496	170		2,128	6742.200024	6637.188027		B	Easy
bley_xl1	175,620	5,831	869,391	5,831			190	154.3902		B	Easy

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
blp-ar98	1,128	16,021	200,601	15,806		215	6205.2147104	5891.22658		T	Easy
blp-ic97	923	9,845	118,149	9,753		92	?	3846.358667		C	Open
bnatt350	4,923	3,150	19,061	3,150			0	0		BRP	Easy
bnatt400	5,614	3,600	21,698	3,600			1	0		CR	Hard
buildingenergy	277,594	154,978	788,969		26,287	128,691	?	*33246.2*		C	Open
cdma	9,095	7,891	168,227	4,235		3,656	?	*−6.38289e+16*	0.194	CU	Open
circ10-3	42,620	2,700	307,320	2,700			?	140	0.001	CR	Open
co-100	2,187	48,417	1,995,817	48,417			2639942.06	917102.214427		C	Hard
core2536-691	2,539	15,293	177,739	15,284		9	689	688.476034		B	Easy
core4872-1529	4,875	24,656	218,762	24,645		11	?	1509.718561		C	Open
cov1075	637	120	14,280	120			20	17.142857		B	Easy
csched007	351	1,758	6,379	1,457		301	351	269.251587		T	Easy
csched008	351	1,536	5,687	1,284		252	173	171		RT	Easy
csched010	351	1,758	6,376	1,457		301	408	332.422727		B	Easy
d10200	947	2,000	57,637	733	1,267		?	12425.583005		C	Open
d20200	1,502	4,000	189,389	3,181	819		?	12229.625788		C	Open
dano3mip	3,202	13,873	79,655	552		13,321	?	576.23162	0.005	CR	Open
danoint	664	521	3,232	56		465	65.666666667	62.63728		B	Easy
datt256	11,077	262,144	1,503,732	262,144			?	*256*	–	C	Open
dc1c	1,649	10,039	121,158	8,380		1,659	?	1754946.863638		C	Open
dc1l	1,653	37,297	448,754	35,638		1,659	?	1744591.692942		C	Open
dfn-gwin-UUM	158	938	2,632		90	848	38752	27467.257235		B	Easy
dg012142	6,310	2,080	14,795	640		1,440	?	757818.480114		C	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
dolom1	1,803	11,612	190,413	9,720		1,892	?	6556066.068315		C	Open
ds-big	1,042	174,997	4,623,442	174,997			?	86.820068		CR	Open
eil33-2	32	4,516	44,243	4,516			934.007916	811.278996		B	Easy
eilA101-2	100	65,832	959,373	65,832			880.920108	803.373888		C	Hard
eilB101	100	2,818	24,120	2,818			1216.920174	1075.247691		B	Easy
enlight13	169	338	962	169	169		71	0		B	Easy
enlight14	196	392	1,120	196	196		*Infeasible*	0		BI	Easy
enlight15	225	450	1,290	225	225		69	0		T	Easy
enlight16	256	512	1,472	256	256		*Infeasible*	0		IT	Easy
enlight9	81	162	450	81	81		*Infeasible*	0		I	Easy
ex1010-pi	1,468	25,200	102,114	25,200			?	220.670087		C	Open
ex10	69,608	17,680	1,162,000	17,680			100	100	0.001	P	Easy
ex9	40,962	10,404	517,112	10,404			81	81		BP	Easy
f2000	10,500	4,000	29,500	4,000			?	1331		CR	Open
g200x740i	940	1,480	2,960	740		740	?	2292.465		C	Open
ger50_17_trans	499	22,414	172,035		18,062	4,352	?	6850.628623	0.01	CU	Open
germanrr	10,779	10,813	175,547	5,288	5,286	239	?	45980135.416399		C	Open
germany50-DBM	2,526	8,189	24,479		88	8,101	?	438028		C	Open
glass4	396	322	1,815	302		20	1200012600	800002400		BT	Easy
gmu-35-40	424	1,205	4,843	1,200		5	−2406733.3688	−2406943.556343		BT	Easy
gmu-35-50	435	1,919	8,643	1,914		5	−2607958.33	−2608070.315743		T	Easy
gmut-75-50	2,565	68,865	571,475	68,859		6	?	−14182312.661731	0.001	C	Open
gmut-77-40	2,554	24,338	159,902	24,332		6	?	−14173396.636852	0.002	C	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
go19	441	441	1,885	441			84	76.530222		CT	Hard
hanoi5	16,399	3,862	39,718	3,862			?	1467		CR	Open
harp2	112	2,993	5,840	2,993			−73899798	−74353341.5023		U	Easy
hawaiiv10-130	1,388,052	685,130	183,263,061	578,444		106,686	?	52281537.594958	0.24	CXU	Open
ic97_potential	1,046	728	3,138	450	73	205	3942	3868		C	Hard
iis-100-0-cov	3,831	100	22,986	100			29	16.666667		B	Easy
iis-bupa-cov	4,803	345	38,392	345			36	26.497217		B	Easy
iis-pima-cov	7,201	768	71,941	768			33	26.620389		B	Easy
in	1,526,202	1,449,074	6,811,639	1,489		1,447,585	?	?	–	CRX	Open
ivu06-big	1,177	2,277,736	23,125,770	2,277,736			?	*135.428*		CRX	Open
ivu52	2,116	157,591	2,179,476	157,591			?	480.250438	0.001	CR	Open
janos-us-DDM	760	2,184	6,384		84	2,100	?	1488134.75		C	Open
k16x240	256	480	960	240		240	10674	2769.838		T	Easy
lectsched-1	50,108	28,718	310,792	28,236	482		0	0		P	Easy
lectsched-1-obj	50,108	28,718	310,792	28,236	482		?	0		C	Open
lectsched-2	30,738	17,656	186,520	17,287	369		0	0		P	Easy
lectsched-3	45,262	25,776	279,967	25,319	457		0	0		P	Easy
lectsched-4-obj	14,163	7,901	82,428	7,665	236		4	0		B	Easy
leo1	593	6,731	131,218	6,730		1	404227536.16	388573315.509608		T	Easy
leo2	593	11,100	219,959	11,099		1	404077441.12	386421293.208919		C	Hard
liu	2,178	1,156	10,626	1,089		67	?	346		C	Open
lotsize	1,920	2,985	6,565	1,195		1,790	1480195	348385.346551	0.009	C	Hard
lrsa120	14,521	3,839	39,956	120	119	3,600	?	29		C	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
m100n500k4r1	100	500	2,000	500			*−25*	−25		BP	Hard
macrophage	3,164	2,260	9,492	2,260			374	0		B	Easy
map06	328,818	164,547	549,920	146		164,401	−289	−406.181173		R	Easy
map10	328,818	164,547	549,920	146		164,401	−495	−602.176181		R	Easy
map14	328,818	164,547	549,920	146		164,401	−674	−778.549324		R	Easy
map18	328,818	164,547	549,920	146		164,401	−847	−932.782685		BR	Easy
map20	328,818	164,547	549,920	146		164,401	−922	−998.836419		B	Easy
markshare_5_0	5	45	203	40		5	1	0		T	Easy
maxgasflow	7,160	7,437	19,717	2,456		4,981	?	−70929535.9	0.01	CT	Open
mc11	1,920	3,040	6,080	1,520		1,520	11689	608.84434		T	Easy
mcsched	2,107	1,747	8,088	1,731	14	2	211913	193774.753707		B	Easy
methanosarcina	14,604	7,930	43,812	7,930			?	0		C	Open
mik-250-1-100-1	151	251	5,351	100	150	1	−66729	−79842.423635		B	Easy
mine-166-5	8,429	830	19,412	830			−566395707.87083	−821763677.673139		B	Easy
mine-90-10	6,270	900	15,407	900			−784302337.633173	−887165318.510226		B	Easy
mining	661,133	348,921	3,844,879	348,920		1	?	−949724584.696851	–	C	Open
mkc	3,411	5,325	17,038	5,323		2	−563.846	−611.85		C	Hard
momentum1	42,680	5,174	103,198	2,349		2,825	*109143*	72793.345255	0.017	CT	Hard
momentum2	24,237	3,732	349,695	1,808	1	1,923	*12314.1*	7225.44067	0.069	U	Easy
momentum3	56,822	13,532	949,495	6,598	1	6,933	?	91952.392314	0.014	CR	Open
msc98-ip	15,850	21,143	92,918	20,237	53	853	19839497.005874302	19520966.151661		BR	Easy
mspp16	561,657	29,280	27,678,735	29,280			363	341		BX	Easy
mzzv11	9,499	10,240	134,603	9,989	251		−21718	−22945.239631	0.001	B	Easy

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
n15-3	29,494	153,140	611,000		780	152,360	?	23703.941176	0.007	CR	Open
n3-3	2,425	9,028	35,380		366	8,662	?	7465.294118		C	Open
n3700	5,150	10,000	20,000	5,000		5,000	?	972305.748043	0.008	C	Open
n3705	5,150	10,000	20,000	5,000		5,000	?	973361.017012	0.008	C	Open
n370a	5,150	10,000	20,000	5,000		5,000	?	979219.544813	0.007	C	Open
n3div36	4,484	22,120	340,740	22,120			130800	114333.374741		B	Hard
n3seq24	6,044	119,856	3,232,340	119,856			52200	52000		B	Easy
n4-3	1,236	3,596	14,036		174	3,422	8993	4080.882353		B	Easy
n9-3	2,364	7,644	30,072		252	7,392	14409	7889.705882		C	Hard
nag	5,840	2,884	26,499	1,350	35	1,499	?	465		C	Open
nb10tb	150,495	73,340	1,172,289	14,124	2,756	56,460	?	12986061164.423624	0.34	CU	Open
neos-1109824	28,979	1,520	89,528	1,520			378	278		B	Easy
neos-1112782	2,115	4,140	8,145	2,070		2,070	*5.72103e+11*	499999999923.7446	0.246	U	Easy
neos-1112787	1,680	3,280	6,440	1,640		1,640	*5.65032e+11*	499999999913.1923	0.066	U	Easy
neos-1140050	3,795	40,320	808,080	38,640		1,680	?	5071593.661704	0.299	CU	Open
neos-1171692	4,239	1,638	42,945	819		819	−273	−273		P	Easy
neos-1171737	4,179	2,340	58,620	1,170		1,170	−195	−195		P	Easy
neos-1224597	3,276	3,395	25,090	3,150	245		−428	−428		P	Easy
neos-1225589	675	1,300	2,525	650		650	1231065191.85	200000000	0.186	U	Easy
neos-1311124	1,643	1,092	7,140	546		546	?	−182		C	Open
neos-1337307	5,687	2,840	30,799	2,840			−202319	−203123.973856		B	Easy
neos-1396125	1,494	1,161	5,511	129		1,032	3000.045337302	388.5524		B	Easy
neos13	20,852	1,827	253,842	1,815		12	−95.474806559	−126.178378		B	Easy

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
neos-1426635	796	520	3,400	260		260	−176	−178		T	Easy
neos-1426662	1,914	832	8,048	416		416	−44	−52		T	Easy
neos-1429212	58,726	416,040	1,855,220	54,756		361,284	?	30		C	Open
neos-1436709	1,417	676	6,214	338		338	−128	−129		T	Easy
neos-1440225	330	1,285	14,168	1,285			36	36	0.003	P	Easy
neos-1440460	989	468	4,302	234		234	−179.25	−180		T	Easy
neos-1442119	1,524	728	6,692	364		364	−181	−182		T	Easy
neos-1442657	1,310	624	5,736	312		312	−154.5	−156		T	Easy
neos15	552	792	1,766	160		632	80598.430096861	29624.693694		T	Easy
neos-1601936	3,131	4,446	72,500	3,906		540	3	1		BR	Easy
neos-1605061	3,474	4,111	93,483	3,570		541	12	6.150735		R	Easy
neos-1605075	3,467	4,173	91,377	3,633		540	9	3.214461		R	Easy
neos-1616732	1,999	200	3,998	200			159	100		T	Easy
neos-1620770	9,296	792	19,292	792			9	1		T	Hard
neos16	1,018	377	2,801	336	41		446	429		T	Easy
neos18	11,402	3,312	24,614	3,312			16	7		B	Easy
neos-476283	10,015	11,915	3,945,693	5,588		6,327	406.363206984	406.244708		B	Easy
neos-506422	6,811	2,527	31,815	63		2,464	0	0		P	Easy
neos-506428	129,925	42,981	343,466	42,981			583780	145945		R	Easy
neos-520729	31,178	91,149	322,203	30,708		60,441	−1385000	−1391291.666667	0.185	U	Easy
neos-555424	2,676	3,815	15,667	3,800	15		1286800	1196312.54944		P	Easy
neos-631710	169,576	167,056	834,166	167,056			?	188.25		CR	Open
neos-686190	3,664	3,660	18,085	3,600	60		6730	5134.81383		B	Easy

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
neos-693347	3,192	1,576	113,472	1,405		171	234	234		P	Easy
neos6	1,036	8,786	251,946	8,340		446	83	83		P	Easy
neos-738098	25,849	9,093	101,360	8,946		147	−1099	−1099		P	Easy
neos-777800	479	6,400	32,000	6,400			−80	−80		P	Easy
neos-785912	1,714	1,380	16,610	1,380			*Infeasible*	42		I	Easy
neos788725	433	352	4,912	352			*Infeasible*	−44.65		I	Easy
neos-799711	59,218	41,998	147,164	910		41,088	−11170211.73363777	−11228065.153797	0.056	U	Easy
neos-807456	840	1,635	4,905	1,635			?	280	0.005	C	Open
neos808444	18,329	19,846	120,512	19,846			0	0		P	Easy
neos-820146	830	600	3,225	600			*Infeasible*	0		IT	Easy
neos-820157	1,015	1,200	4,875	1,200			*Infeasible*	0		IT	Easy
neos-824661	18,804	45,390	138,890	15,640		29,750	33	33		P	Easy
neos-824695	9,576	23,970	72,590	8,500		15,470	31	31		P	Easy
neos-826650	2,414	5,912	20,440	5,792		120	29	28		T	Easy
neos-826694	6,904	16,410	59,268	16,290		120	58	58		P	Easy
neos-826812	6,844	15,864	53,808	10,350		5,514	58.011	58.011		P	Easy
neos-826841	2,354	5,516	18,460	3,488		2,028	29.0082	28.0082		T	Easy
neos-847302	609	737	9,566	729		8	4	0		T	Hard
neos-849702	1,041	1,737	19,308	1,737			0	0	0.001	BP	Easy
neos858960	132	160	2,770	160			*Infeasible*	1		IT	Easy
neos-859770	2,065	2,504	880,736	2,504			*Infeasible*	4500		I	Easy
neos-885086	11,574	4,860	248,310	2,430		2,430	−243	−243		P	Easy
neos-885524	65	91,670	258,309	91,670			12320.092	11754.885		P	Easy

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
neos-911880	83	888	2,568	840		48	54.76	23.26		T	Easy
neos-916792	1,909	1,474	134,442	717		757	31.870398371	26.203596	0.001	B	Easy
neos-932816	30,823	21,007	484,926	20,566		441	15376	2285.5		P	Easy
neos-933638	13,658	32,417	187,173	28,637		3,780	276	276		P	Easy
neos-933966	12,047	31,762	180,618	27,982		3,780	318	318		P	Easy
neos-934278	11,495	23,123	125,577	19,955		3,168	260	259.5		B	Easy
neos-935627	7,859	10,301	40,476	7,522		2,779	2598	2598		RP	Easy
neos-935769	6,741	9,799	36,447	7,020		2,779	3010	3010		P	Easy
neos-937511	8,158	11,332	44,237	8,562		2,770	3510	3510		P	Easy
neos-937815	9,251	11,646	48,013	8,876		2,770	?	2837		CR	Open
neos-941262	6,703	9,480	35,659	6,710		2,770	2791	2790.5		R	Easy
neos-941313	13,189	167,910	484,080	167,910			9361	9361		P	Easy
neos-942830	803	882	13,290	834		48	16	12		T	Easy
neos-948126	7,271	9,551	38,219	6,965		2,586	2607	2602		R	Hard
neos-952987	354	31,329	90,384	31,329			?	1327.33714		C	Open
neos-957389	5,115	6,036	355,372	6,036			1.5	1.5		P	Easy
neos-984165	6,962	8,883	36,742	6,478		2,405	?	2186		CR	Open
net12	14,021	14,115	80,384	1,603		12,512	214	17.249479		BR	Easy
netdiversion	119,589	129,180	615,282	129,180			242	230.8		B	Easy
newdano	576	505	2,184	56		449	65.666666667	11.724138		B	Easy
nobel-eu-DBE	879	3,771	11,313	1,639		2,132	608910	570687.5		T	Hard
noswot	182	128	735	75	25	28	−41	−43		BT	Easy
npmv07	76,342	220,686	859,614	1,880		218,806	*1.0481e+11*	104809667051.7079	0.007	U	Hard

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
ns1111636	13,895	360,822	568,444	13,200		347,622	?	96.25	0.015	CR	Open
ns1116954	131,991	12,648	410,582	7,482		5,166	0	0		P	Easy
ns1158817	68,455	1,804,022	2,842,044	66,022		1,738,000	Infeasible	Infeasible	0.01	I	Easy
ns1208400	4,289	2,883	81,746	2,880		3	*2*	0		B	Easy
ns1456591	1,997	8,399	199,862	8,000	19	380	?	361.939022	0.001	CT	Open
ns1606230	3,503	4,173	92,133	3,633		540	21	13.225		R	Easy
ns1631475	24,496	22,696	116,733	22,470	211	15	?	817.193748		CR	Open
ns1644855	40,698	30,200	2,110,696	10,000		20,200	−1524.333333333	−1524.333333	0.039	R	Hard
ns1663818	172,017	124,626	20,433,649	124,626			?	1	–	CX	Open
ns1685374	44,121	10,000	220,859	10,000			−13	−51.800913		R	Hard
ns1686196	4,055	2,738	68,529	2,738			*Infeasible*	2		I	Easy
ns1688347	4,191	2,685	66,908	2,685			27	2		B	Easy
ns1696083	11,063	7,982	384,129	7,982			?	2	0.003	CR	Open
ns1702808	1,474	804	5,856	666		138	*Infeasible*	1700		I	Easy
ns1745726	4,687	3,208	90,278	3,208			*Infeasible*	3		I	Easy
ns1758913	624,166	17,956	1,283,444	17,822		134	−1454.671755	−1501.183256		B	Easy
ns1766074	182	100	666		90	10	*Infeasible*	5833.8		BIT	Easy
ns1769397	5,527	3,772	117,383	3,772			*Infeasible*	3	0.001	I	Easy
ns1778858	10,666	4,720	32,673	4,720			?	−28931833.446612		C	Open
ns1830653	2,932	1,629	100,933	1,458		171	20622	6153		B	Easy
ns1853823	224,526	213,440	1,489,480	213,440			?	*60182.9*	0.008	C	Open
ns1854840	143,616	135,754	856,994	135,280	474		?	*122000*		C	Open
ns1856153	35,407	11,998	105,882	11,956		42	?	0		CR	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
ns1904248	149,437	38,458	378,770	38,416		42	?	0	0.007	C	Open
ns1905797	51,884	18,192	239,700	17,676	4	512	?	11.75098		C	Open
ns1905800	8,289	3,228	38,100	3,030	3	195	?	6.4366		C	Open
ns1952667	41	13,264	335,643		13,264		0	0		P	Easy
ns2017839	54,510	55,224	317,840	12		55,212	*7.70305e+13*	77025639567520.05	0.01	U	Easy
ns2081729	1,190	661	5,680	600		61	9	4.6		T	Easy
ns2118727	163,354	167,440	646,864	159,514		7,926	*Infeasible*	260.751793	0.002	IR	Easy
ns2122603	24,754	19,300	77,044	7,588		11,712	*Infeasible*	0	1	IU	Hard
ns2124243	139,280	156,083	429,032	16,447		139,636	?	53998.333333		C	Open
ns2137859	206,726	103,361	923,682	103,041		320	?	24629		C	Open
ns4-pr3	2,210	8,601	25,986		61	8,540	?	36073		C	Open
ns4-pr9	2,220	7,350	22,176		42	7,308	?	35175		C	Open
ns894236	8,218	9,666	41,067	9,666			?	12.305229		CR	Open
ns894244	12,129	21,856	90,864	21,856			15	12.326616	0.006	R	Easy
ns894786	16,794	27,278	113,575	27,278			?	6.080533	0.001	CR	Open
ns894788	2,279	3,463	14,381	3,463			7	6.304802		T	Easy
ns903616	18,052	21,582	91,641	21,582			?	14.750147	0.001	CR	Open
ns930473	23,240	11,328	121,764	11,176		152	?	0	0.001	CR	Open
nsr8k	6,284	38,356	371,608	32,040		6,316	?	17500809.511815	0.001	CR	Open
nu120-pr3	2,210	8,601	25,986	8,540	61		28130	21306.442308		C	Hard
nu60-pr9	2,220	7,350	22,176	7,308	42		24940	22850		C	Hard
ofi	422,587	420,434	1,778,754	18,632	11,073	390,729	*6.15538e+09*	6131846109.623997	0.168	U	Hard
opm2-z10-s2	160,633	6,250	371,243	6,250			?	−49308.278601	0.001	C	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
opm2-z11-s8	223,082	8,019	510,283	8,019			?	−62971.930395		CR	Open
opm2-z12-s14	319,508	10,800	725,376	10,800			?	−91524.54224	0.001	CR	Open
opm2-z12-s7	319,508	10,800	725,385	10,800			?	−90514.285842	0.001	CR	Open
opm2-z7-s2	31,798	2,023	79,762	2,023			−10280	−12879.686897		B	Easy
p100x588b	688	1,176	2,352	588		588	?	5554.011111		C	Open
p2m2p1m1p0n100	1	100	100	100			*Infeasible*	80424		IT	Easy
p6b	5,852	462	11,704	462			?	−231		C	Open
p80x400b	480	800	1,600	400		400	39667	6418.8		T	Easy
pb-simp-nonunif	1,451,912	23,848	4,366,648	23,848			?	*6*	0.001	CX	Open
pg5_34	225	2,600	7,700	100		2,500	−14339.353446926	−16646.586017		B	Easy
pg	125	2,700	5,200	100		2,600	−8674.342607117	−11824.657382		T	Easy
pigeon-10	931	490	8,150	400		90	−9000	−10000		BT	Easy
pigeon-11	1,123	572	9,889	473		99	−10000	−11000		T	Easy
pigeon-12	1,333	660	11,796	552		108	−11000	−12000		CT	Hard
pigeon-13	1,561	754	13,871	637		117	?	−13000		C	Open
pigeon-19	3,307	1,444	29,849	1,273		171	?	−19000		C	Open
probportfolio	302	320	6,620	300		20	16.734246764	5		C	Hard
protfold	2,112	1,835	23,491	1,835			*−31*	−41.957447	0.01	R	Hard
pw-myciel4	8,164	1,059	17,779	1,058	1		10	0		B	Easy
qiu	1,192	840	3,432	48		792	−132.873136947	−931.638845		B	Easy
queens-30	960	900	93,440	900			?	−70.912689		C	Open
r80x800	880	1,600	3,200	800		800	?	3651.48		C	Open
rail01	46,843	117,527	392,086	117,527			−70.5699643	−92.0873	0.002	R	Easy

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
rail02	95,791	270,869	756,228	270,869			−200.449907667	−206.6102	0.01	CR	Hard
rail03	253,905	758,775	1,728,451	758,775			?	*−920.274*	0.025	CR	Open
rail507	509	63,019	468,878	63,009		10	174	172.145567		B	Easy
ramos3	2,187	2,187	32,805	2,187			?	145.8		CR	Open
ran14x18-disj-8	447	504	10,277	252		252	*3735*	3444.421066		T	Easy
ran14x18	284	504	1,008	252		252	3712	3016.944354		T	Easy
ran16x16	288	512	1,024	256		256	3823	3116.429512		B	Easy
reblock166	17,024	1,660	39,442	1,660			*−6.00052e+08*	−886882876.965829		T	Easy
reblock354	19,906	3,540	52,901	3,540			-39280521.2281657	−39641692.452292		C	Hard
reblock420	62,800	4,200	138,670	4,200			*−5.17793e+08*	−886535897.522137		C	Hard
reblock67	2,523	670	7,495	670			−34630648.43833169	−39339910.923037		B	Easy
rmatr100-p10	7,260	7,359	21,877	100		7,259	423	360.593308		B	Easy
rmatr100-p5	8,685	8,784	26,152	100		8,684	976	762.040054		B	Easy
rmatr200-p10	35,055	35,254	105,362	200		35,054	?	1550.620783		C	Open
rmatr200-p20	29,406	29,605	88,415	200		29,405	837	688.357943		C	Hard
rmatr200-p5	37,617	37,816	113,048	200		37,616	4521	3283.653831		C	Hard
rmine10	65,274	8,439	162,264	8,439			?	−1926.86382		C	Open
rmine14	268,535	32,205	660,346	32,205			?	−4310.694704	0.002	C	Open
rmine21	1,441,651	162,547	3,514,884	162,547			?	*−10679.2*	0.01	CX	Open
rmine25	2,953,849	326,599	7,182,744	326,599			?	*−15667.9*	–	CX	Open
rmine6	7,078	1,096	18,084	1,096			−457.18614	−462.305727		B	Easy
rocII-4-11	21,738	9,234	243,106	9,086		148	−6.652756	−11.937162		B	Easy
rocII-7-11	37,215	16,101	423,661	15,851		250	?	−11.964093		C	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
rocII-9-11	47,533	20,679	544,031	20,361		318	?	−11.972072		C	Open
rococoB10-011000	1,667	4,456	16,517	4,320	136		19449	8350.199468		C	Hard
rococoC10-001000	1,293	3,117	11,751	2,993	124		11460	7515.271029		B	Easy
rococoC11-011100	2,367	6,491	30,472	6,325	166		?	9024.205406		CR	Open
rococoC12-111000	10,776	8,619	48,920	8,432	187		?	27337.357381		C	Open
roll3000	2,295	1,166	29,386	246	492	428	12890	11097.127677		B	Easy
rvb-sub	225	33,765	984,143	33,763		2	?	11.206514		C	Open
satellites1-25	5,996	9,013	59,023	8,509		504	−5	−20	0.016	B	Easy
satellites2-60-fs	16,516	35,378	125,048	34,324		1,054	−19	−30	0.051	R	Easy
satellites2-60	20,916	35,378	283,668	34,324		1,054	−19	−30	0.143	CRU	Hard
satellites3-40-fs	35,553	81,681	291,161	79,961		1,720	?	−39	0.292	CRU	Open
satellites3-40	44,804	81,681	698,176	79,961		1,720	?	−39	0.3	CRU	Open
sct1	12,154	22,886	105,571	9,044	1,268	12,574	?	−218.139003		C	Open
sct32	5,440	9,767	109,654	6,396	1,332	2,039	?	−62.990833		C	Open
sct5	13,304	37,265	147,037	20,702	2,302	14,261	?	−228.176552		CR	Open
set3-10	3,747	4,019	13,747	1,424		2,595	?	788.890655		C	Open
set3-15	3,747	4,019	13,747	1,424		2,595	?	8364.040745		C	Open
set3-20	3,747	4,019	13,747	1,424		2,595	?	10347.381087		C	Open
seymour	4,944	1,372	33,549	1,372			423	403.846474		C	Hard
seymour-disj-10	5,108	1,209	64,704	1,209			?	280.817818		C	Open
shipsched	45,554	13,594	121,571	10,549		3,045	?	0		C	Open
shs1023	133,944	444,625	1,044,725	1,296	440,899	2,430	?	12121.358113	0.001	C	Open
siena1	2,220	13,741	258,915	11,775		1,966	?	10163179.229002		C	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
sing161	455,631	770,102	2,072,500	733,244		36,858	?	19071259.113927		C	Open
sing245	143,161	235,146	652,817	220,692		14,454	?	25024015.754083	0.001	CR	Open
sing2	28,891	31,630	149,712	23,377		8,253	?	17150914.71694		C	Open
sing359	437,116	713,762	1,975,605	674,643		39,119	?	22534478.962498	0.001	CR	Open
sp97ar	1,761	14,101	290,968	14,101			660705645.76	652560391.247564		C	Hard
sp98ic	825	10,894	316,317	10,894			449144758.4	444277568.934152		B	Easy
sp98ir	1,531	1,680	71,704	871	809		219676790.4	216663444.589936		B	Easy
splan1	572,800	1,317,382	5,233,840	90,810	1,978	1,224,594	?	*−201899*	–	CXU	Open
stockholm	57,346	20,644	171,076	962		19,682	?	0.755613	0.001	C	Open
stp3d	159,488	204,880	662,128	204,880			493.71965	481.877786	0.006	CR	Hard
sts405	27,270	405	81,810	405			?	135		C	Open
sts729	88,452	729	265,356	729			?	243	0.001	CR	Open
swath	884	6,805	34,965	6,724		81	467.407491	334.496858		C	Hard
t1717	551	73,885	325,689	73,885			?	134531.021428		C	Open
t1722	338	36,630	133,096	36,630			?	98815.407611		C	Open
tanglegram1	68,342	34,759	205,026	34,759			5182	0		B	Easy
tanglegram2	8,980	4,714	26,940	4,714			443	0		B	Easy
timtab1	171	397	829	64	107	226	764772	28694		BT	Easy
toll-like	4,408	2,883	13,224	2,883			?	0		C	Open
transportmoment	9,616	9,685	29,541	2,456		7,229	?	−70929535815.37999	0.325	CU	Open
triptim1	15,706	30,055	515,436	20,451	9,597	7	22.8681	22.868088	0.01	BP	Easy
triptim2	14,427	27,326	521,898	20,771	6,548	7	?	10.866359	0.01	CR	Open
triptim3	14,939	28,440	524,124	21,621	6,812	7	?	13.511741	0.011	CR	Open

Table 4 continued

Name	Rows	Columns	Non-zeros	Binary	Integer	Continuous	MIP Solution	LP Solution	Att.-level	Sets	Status
tw-myciel4	8,146	760	27,961	759	1		?	3.838028		C	Open
uc-case11	51,438	34,134	202,042	3,898	302	29,934	?	611267.013004	0.001	C	Open
uc-case3	52,003	37,749	273,618	11,256		26,493	?	7181.264177		C	Open
uct-subprob	1,973	2,256	10,147	379		1,877	314	242		C	Hard
umts	4,465	2,947	23,016	2,802	72	73	30090328	29129565.161344		T	Easy
unitcal_7	48,939	25,755	127,595	2,856		22,899	19635558.2440195	19387553.381271	0.002	B	Easy
usAbbrv-8-25_70	3,291	2,312	9,628	1,681		631	?	95		C	Open
van	27,331	12,481	487,296	192		12,289	?	1.72352		C	Open
vpphard2	198,450	199,999	648,340	199,999			?	0		C	Open
vpphard	47,280	51,471	372,305	51,471			5	0		B	Easy
wachplan	1,553	3,361	89,361	3,360	1		−8	−9		T	Easy
wnq-n100-mw99-14	656,900	10,000	1,333,400	10,000			?	185.263158		CT	Open
zib01	5,887,041	12,471,400	49,877,768	12,471,400			?	*64697*	–	CX	Open
zib02	9,049,868	37,709,944	146,280,582	37,709,944			*Infeasible*	*Infeasible*	–	CIX	Hard
zib54-UUE	1,809	5,150	15,288	81		5,069	10334015.82	3875862.862589		B	Easy

Table 5 Distribution of instances base upon variable types

Type	Binary	Integer	Continuous	Instances
Binary Programs	X			116
Integer Programs		X		1
	X	X		29
Mixed Integer Programs	X		X	173
		X	X	12
	X	X	X	30

the number of variables is divided into the number of *Binary*, general *Integer*, and *Continuous* variables. Table 5 shows a distribution of instances based upon variables types. Only 13 instances do not have any binary variables, while only 72 of 361 instances have general integer variables.

MIP *and* LP *solution values.* If known, the objective values of an optimal MIP *Solution* and the LP relaxation (LP *Solution*) are listed. The rational LP solver QSOPT_EX [12, 52, 123] was used to solve the LP relaxation of the instances with exact arithmetic. We report up to 16 digits, last digit rounded, in the table. For those instances that could not be solved by QSOPT_EX within a reasonable amount of time, we print the optimal solution value of the LP relaxation reported by CPLEX in *italic*. Note that we used the pure LP relaxation without applying MIP presolving, so the root LP bound computed during MIP solution might be better, even before cutting planes are added.

Although first steps have been taken in developing exact MIP solvers [12, 38], current implementations are not yet capable of computing provably optimal solutions for most of the instances in MIPLIB 2010. Therefore, we post-processed the MIP solutions generated by the inexact solvers. The solution values of all integer variables were rounded and fixed to the nearest integer. The remaining LP was solved with QSOPT_EX. This does not give an optimality proof, but can provide a MIP solution that is truly feasible. Again, if this LP could not be solved by QSOPT_EX, we state in *italic* the objective value of an inexact optimal solution that passed the test of the solution checker. Instances where no optimal MIP solution is known, i.e., *open* instances, are labeled with a question mark '?'.

For the instances in the challenge set, we invested limited time to try to solve them with one of the available solvers. Four instances, 50v-10, probportfolio, reblock354, and rmatr200-p20, could be solved by UG[SCIP/SPX] [97], a distributed massively parallel version of SCIP run on 2,000 cores at the HLRN- II super computer facility. To finally solve reblock354 about 42 billion nodes had to be processed. This took approximately 36 CPU years.

Condition numbers and numerical reliability. Column ATT.- LEVEL lists the numerical *attention level* calculated using the `set mip strategy kappastats 2` command of CPLEX 12.2.0.2. This gathers statistics about the condition numbers of sub-problem basis matrices during CPLEX's solution process and is explained below in more detail. Vaguely speaking, a higher attention level indicates larger condition numbers of LP basis matrices, which in turn points to a higher probability of numerical instabilities during the solution process. A node limit of 1,000 and a time limit of 2 days was used to get a reliable data base for this computation. Those few instances that did not finish the root node within this time limit, are marked by "—", for those instances with attention level 0.0, the value is omitted.

The condition number $\kappa(A_B)$ is a well-known measure of how errors in the input of a linear system of equations $A_B x = b$ are propagated to the solution x; for further details see, e.g., [62]. The condition numbers $\kappa(A_B)$ of the basis matrices A_B that are used to calculate optimal solutions of the sub-problem LP relaxations provide some insight into the origin of potential numerical inaccuracies that may have arisen during the solution process.

Double-precision arithmetic, as employed in the solvers at hand, provides roughly 16 significant decimal digits. Given that we used a feasibility tolerance of 10^{-6}, condition numbers larger than 10^{10} suggest that the errors encountered when calculating optimal LP solutions of the sub-problems may be larger than the feasibility tolerance. CPLEX classifies the condition numbers by the following four categories and outputs a histogram of the condition number distribution of optimal LP bases encountered during the MIP search:

- stable if $\kappa(A_B) < 10^7$,
- suspicious if $\kappa(A_B) \in [10^7, 10^{10})$,
- unstable if $\kappa(A_B) \in [10^{10}, 10^{14})$, and
- ill-posed if $\kappa(A_B) \geq 10^{14}$.

The attention level $AL \in [0, 1]$ summarizes this histogram as

$$AL = p_{\text{ill-posed}} + 0.3 \cdot p_{\text{unstable}} + 0.01 \cdot p_{\text{suspicious}}$$

with $p_{\dots}$ being the relative frequencies of the sampled condition number categories. Thus, the attention level provides an estimate of the probability that the solver has encountered numerical issues during the solution process and their severity.

The attention level is not a property inherent to the model, but a measure for a *specific algorithm applied to a certain instance* and will even vary for one solver when different parameter settings are used. Since all solvers tested are based on the same paradigm of branch-and-bound-and-cut, we still believe that the attention level is a measure that can be of use in predicting numerical difficulties during the solution of an instance, though by no means the only one.

Category and status. The last two columns denote the test sets in which an instance is contained by giving the first letter of BENCHMARK, CHALLENGE, INFEASIBLE, PRIMAL, UNSTABLE, RESOLVE, TREE, XXL, as well as the status of the instance as of this writing, i.e., *easy* (solvable within 1 h), *hard* (solvable in a longer time or with a specialized algorithm), or *open* (optimal solution unknown).

Size, structure and sparsity. The coefficient matrices vary largely in size and proportion. Figure 5 shows their distribution according to number of rows and columns.

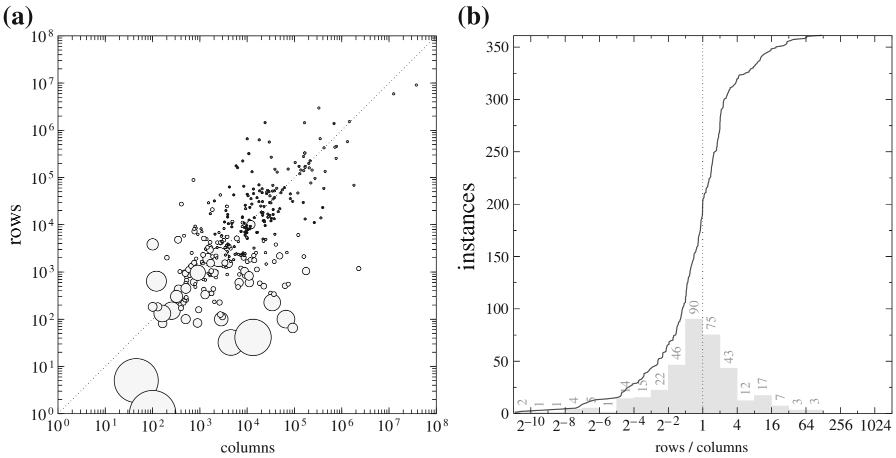


Fig. 5 Size distribution of instances in MIPLIB 2010. **a** Instances as scattered by number of rows vs. number of columns. The area of each bubble corresponds to the density of the instance. Densities smaller than 0.05 are drawn in uniform size. **b** Cumulative distribution function of number of rows divided by number of columns and corresponding histogram over a $\log_2$ scale

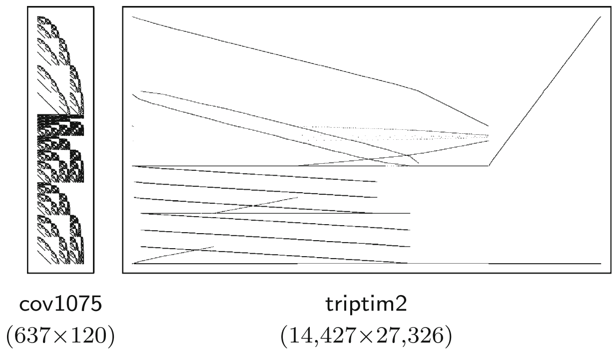


Fig. 6 Two examples of sparsity patterns

It can be seen that instances with more columns than rows are slightly in the majority (201 vs. 160 instances).

Apart from all the randomness involved, the instances typically show a strong structure, see, e.g., [80]. To underline this, we provide pictures representing the non-zero distribution in the coefficient matrices on the MIPLIB website. Figure 6 shows the sparsity patterns for instances cov1075 and triptim2.

Note that the structure shown in the pictures is intensified, i.e., the instances are usually sparser than they appear here. This is especially true for large instances where the ratio of zero to non-zero elements of the matrix is as small as $2.3 \cdot 10^6$ to one (for instance zib02). Only 11 instances show a density larger than 10% and the average density over all 361 instances is 1.6%; see also Fig. 5a.

Table 6 Solvers used for the computations

Solver	Version	Status	Website
CBC	2.6.4 R1630	Release	http://projects.coin-or.org/Cbc
CPLEX	12.2.0.2	Release	http://www.cplex.com
GUROBI	4.5 beta0	Beta	http://www.gurobi.com
SCIP/SPX	2.0.1.3/1.5.0.3	Internal Version	http://zibopt.zib.de
XPRESS	7.2 RC	Release Candidate	http://www.fico.com/xpress

7 State-of-the-art MIP solving

In this section, we try to provide a snapshot of the state-of-the art in MIP solving as of March 2011. We emphasize that the goal of this section is *not* to compare the performance of the solvers, but rather to introduce the instances of the MIPLIB 2010 test set. We hope to reveal characteristics of the problems and give an impression of what is possible with today's codes, giving a point of reference for future research and investigation.

All of the solvers are in continuous development, often experiencing significant improvements from release to release. After having assembled a preliminary version of the test sets between December 2010 and mid January 2011, each development group was asked to send the latest version of their code, even if it was not yet officially released. While it is relatively easy for CBC and SCIP to make new releases on short notice, the commercial vendors require more thorough testing. For GUROBI, we used the very latest beta and from XPRESS, the first release candidate. Both will be officially released soon. CPLEX provided its most recent public version, which was released in January 2011. Table 6 lists the solvers used.

We also investigated other solvers like GLPK [113] and LP_SOLVE [119], but since neither of them was able to solve more than 20% of the instances within the time limit and no multithreaded versions were available, we omitted the results. For a recent survey of MIP software we refer to [75].

For computations in this section, a computer with two Intel Xeon X5680 CPUs, each providing 6 cores at 3.33 GHz, was used. Hyperthreading and Turboboost were disabled. The machine has 32 GB RAM. All computations were done using the BENCHMARK test set.

Table 7 lists the results for computations using one thread with a time limit of 1 h and a memory limit of 8 GB. The solvers are ordered by release date, commercial ones first, non-commercial ones in a second block. For each solver, the number of branch-and-bound nodes used and the wall clock time in seconds are listed. If the time limit was reached before the solver finished, the optimality gap is given instead. If the gap at termination is infinite, i.e., if no primal solution was found or if $0 \in [db, pb]$ and $db \neq pb$, we write *inf%*. *Italics* indicate that the solution (as printed in the output file by the solver) did not pass the tolerances for the solution checker described in Sect. 3. If a solver aborted, we write *abort* across the columns for nodes and time. In the latter two cases, a footnote gives details, if available.

Table 7 Results for the Benchmark-Set using 1 thread, time limit 1 h. Time is given in seconds, if the time limit was reached the optimality gap is given instead. *Italic* indicates that the solution did not pass the feasibility check. *Best* lists the minimum number of nodes and time over all solvers providing a feasible solution. Nodes and Time may have *not* been taken from the same solver. *VS* lists the variablility score of the instance as described in Section 5.2. Higher numbers indicate higher performance variability. The additional test sets containing each instance, other than BENCHMARK, are listed in *Sets*

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		SCIP/SPX		Best		VS
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	
30n20b8		96.7k	20.3%	2.3M	74.8%	87.7k	804	213.1k	60.2%	8.2k	332.5%	87654	804	
acc-tight5	RP	9	5	487	75	1.4k	121	1.2k	303	497	137	9	5	0.72
aflow40b		98.3k	376	134.8k	399	166.7k	738	367.9k	5.3%	278.7k	2400	98349	376	0.39
air04		225	6	121	12	125	10	456	55	76	65	76	6	0.14
app1-2		30.0k	1420	33.6k	125	64	65	2.7k	inf%	1	inf%	64	65	
ash608gpia-3col	I	9	53	15	180	5	221	1	567	7	47	1	47	0.20
bab5		55.1k	1722	27.3k	0.7%	9.5k	662	22.7k	inf%	7.5k	1.4%	9470	662	
beasleyC3		544.9k	1.1%	11.2k	89	729	13	170.1k	21.6%	504.8k	22.3%	729	13	
biella1		11.2k	899	5.5k	424	1.2k	238	16.0k	0.0%	1.7k	682	1162	238	0.29
bienst2		70.1k	63	101.8k	121	97.6k	33	90.8k	1312	89.6k	290	70054	33	0.21
binkar10_1		8.7k	15	5.5k	14	51.5k	35	59.0k	481	199.4k	406	5461	14	0.50
bley_xl1		43	17	1	8	1	17	50	891	38	305	1	8	0.26
bnatt350	RP	16.5k	3408	82.9k	inf%	36.1k	inf%	7.4k	inf%	6.6k	1191	6567	1191	0.50
core2536-691		748	57	183	81	771	194	869	275	308	377	183	57	0.67
cov1075		4.7k	19	185.9k	8.7%	342	6	138.6k	8.9%	529.5k	8.9%	342	6	0.06
csched010		400.3k	3.9%	136.3k	1176	621.6k	2.8%	549.1k	18.9%	466.4k	5.2%	136309	1176	0.26
danoint		528.9k	2.4%	514.7k	3398	819.9k	3.0%	210.1k	2.7%	507.6k	3.1%	514711	3398	0.09
dfn-gwin-UUM		36.4k	72	150.9k	264	242.3k	225	332.5k	1132	14.5k	54	14491	54	0.23
eil33-2		10.8k	58	15.7k	125	10.1k	110	8.8k	307	11.1k	180	8792	58	0.15
eilB101		10.5k	160	2.3k	80	22.0k	231	23.8k	1570	3.0k	680	2305	80	0.23
enlight13		84.2k	397	39.1k	13	62.0k	307	237.6k	inf%	622.6k	413	39107	13	*1.22*
enlight14	I	663.8k	inf%	142.0k	42	571.5k	inf%	183.1k	inf%	738.2k	665	142035	42	*0.82*
ex9	P	1	2102	1	17	1	227	1	inf%	1	85	1	17	0.11

Table 7 continued

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		SCIP/SPX		Best		VS
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	
glass4	T	4.4M	1537	1.7M	26.6%	496.3k	280	221.8k	47.5%	4.0M	100.0%	496349	280	
gmu-35-40	T	1.6M	893	3.6M	0.0%	13.0M	1898	1.8M	0.0%	4.9M	0.0%	1578834	893	
iis-100-0-cov		386.8k	3233	83.4k	1882	144.0k	1515	30.1k	12.1%	103.9k	1808	83425	1515	0.09
iis-bupa-cov		154.1k	8.6%	54.1k	7.5%	164.2k	5.9%	25.2k	11.3%	85.0k	10.0%	–	–	0.06
iis-pima-cov		27.0k	869	17.2k	1014	7.9k	318	15.9k	10.2%	11.1k	864	7917	318	0.21
lectsched-4-obj		1.5k	9	1	1	108	5	8.4k	1544	9.4k	265	1	1	1.04
m100n500k4r1	P	1.8M	4.2%	1.6M	4.2%	4.3M	4.2%	1.0M	4.2%	3.9M	4.2%	–	–	
macrophage		8.6k	131	641	43	418.2k	1.4%	6.1k	20.6%	308.1k	27.1%	641	43	
map18	R	809	190	513	147	1.1k	203	1.2k	1485	544	764	513	147	0.15
map20		575	138	521	177	1.2k	177	1.0k	1339	550	580	521	138	0.20
mcsched		111.8k	479	427.7k	2769	28.6k	198	117.4k	2.4%	23.1k	337	23111	198	0.16
mik-250-1-100-1		118.1k	14	79.7k	37	97.2k	18	441.8k	0.5%	415.1k	176	79739	14	0.05
mine-166-5		5.4k	24	249	7	6.7k	24	4.7k	157	6.1k	76	249	7	0.24
mine-90-10		345.9k	1350	142.5k	216	1.1M	1144	130.8k	1965	56.2k	264	56177	216	0.75
msc98-ip	R	2.4k	5.0%	21.0k	1.3%	4.7k	700	294	inf%	118	inf%	4734	700	
mspp16	X	53	635	1	253	1	1840	abort		abort[a]		1	253	
mzzv11		869	51	69	8	6	17	739	187	2.4k	591	6	8	0.21
n3div36		286.3k	6.0%	190.3k	1.2%	501.7k	3358	120.4k	0.5%	84.7k	11.2%	501686	3358	
n3seq24		491	61	1	181	40.8k	677	268	1495	348	0.4%	1	61	
n4-3		8.5k	328	47.6k	1979	120.6k	2110	293.2k	18.7%	81.5k	1052	8467	328	0.34
neos-1109824		8.7k	30	2.1k	14	10.5k	109	22.0k	428	10.8k	119	2071	14	0.30
neos-1337307		67.5k	0.1%	439.5k	0.0%	220.7k	0.0%	12.4k	0.1%	167.4k	0.0%	–	–	
neos-1396125		26.4k	82	12.6k	79	8.2k	135	54.9k	24.0%	49.8k	3338	8175	79	0.25

Table 7 continued

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		SCIP/SPX		Best		VS
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	
neos-1601936	R	6.3k	1759	695	31	333	56	8.9k	996	8.9k	66.7%	333	31	*0.81*
neos-476283		1.0k	243	777	149	515	142	710	1561	453	322	453	142	0.16
neos-686190		4.3k	29	6.7k	52	5.3k	47	5.5k	154	5.6k	95	4298	29	0.27
neos-849702	P	44.9k	inf%	43.9k	409	104.6k	1066	152.6k	inf%	244.3k	inf%	43941	409	*1.59*
neos-916792		122.8k	569	42.1k	208	67.2k	1525	106.0k	4.3%	67.4k	367	42060	208	*2.23*
neos-934278		101	230	199	82	1	82	400	649	2.1k	2.9%	1	82	
neos13		4.1k	58	65	26	711	47	28.1k	2.7%	283	657	65	26	*1.72*
neos18		13.7k	27	20.0k	55	13.7k	65	abort		9.9k	68	9854	27	0.39
net12	R	2.5k	22.1%	2.5k	92	1.3k	290	1.4k	35.3%	5.0k	57.6%	1348	92	0.33
netdiversion		28	74	9	76	123	628	847	1.4%	51	inf%	9	74	
newdano		170.6k	34.7%	372.4k	4.8%	683.3k	11.0%	81.9k	23.7%	1.5M	31.0%	–	–	0.21
noswot	T	6.9M	950	1.7M	344	566.3k	72	1.0M	2194	605.0k	210	566273	72	0.34
ns1208400		2.9k	88	221.6k	inf%	623	81	57.8k	inf%	3.0k	526	623	81	0.70
ns1688347		7.3k	139	2.5k	25	868	31	3.6k	inf%	4.2k	1027	868	25	0.79
ns1758913		24	158	1	37	1	44	1	inf%	1	inf%	1	37	
ns1766074	IT	870.3k	133	520.8k	141	1.1M	110	189.9k	592	945.4k	668	189900	110	0.12
ns1830653		10.9k	206	31.9k	394	26.9k	427	20.7k	95.0%	47.0k	638	10915	206	0.42
opm2-z7-s2		2.0k	282	2.5k	501	639	67	3.5k	1039	1.5k	660	639	67	0.17
pg5_34		107.8k	244	88.6k	376	187.0k	439	97.3k	1787	257.0k	1247	88573	244	0.13
pigeon-10	T	5.8M	985	7.8M	2561	4.0M	558	685.7k	11.1%	3.7M	11.1%	3984370	558	
pw-myciel4		33.1k	159	108.6k	1121	4.7M	11.1%	35.5k	66.7%	309.9k	42.9%	33124	159	0.51
qiu		2.2k	11	8.3k	55	3.4k	18	13.1k	222	14.5k	98	2157	11	0.19
rail507		3.5k	207	2.8k	911	1.5k	123	7.6k	1458	1.3k	1252	1259	123	

Table 7 continued

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		SCIP/SPX		Best		VS
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	
ran16x16		64.3k	73	69.9k	88	81.8k	76	195.6k	785	344.3k	291	64322	73	0.20
reblock67		588.1k	766	234.5k	342	636.6k	651	305.1k	3275	139.6k	373	139635	342	0.32
rmatr100-p10		2.0k	76	917	56	1.0k	28	2.9k	157	863	322	863	28	0.08
rmatr100-p5		1.2k	94	681	107	1.2k	107	1.4k	167	419	991	419	94	0.12
rmine6		917.2k	1979	416.7k	1032	404.4k	476	459.9k	0.1%	546.9k	0.1%	404406	476	0.19
rocII-4-11		175.6k	271	47.3k	318	33.8k	205	19.2k	0.3%	27.6k	381	27610	205	0.32
rococoC10-001000		45.5k	597	35.9k	420	1.7M	0.4%	36.3k	1345	462.5k	2493	35937	420	0.42
roll3000		31.3k	138	1.9k	16	6.8k	68	34.5k	2284	593.9k	3399	1947	16	*0.54*
satellites1-25		3.5k	200	2.3k	120	634	102	14.5k	*1582*[b]	8.6k	197.0%	634	102	*0.53*
sp98ic		196.3k	1166	23.3k	563	32.8k	289	86.5k	0.3%	35.0k	3.5%	23317	289	
sp98ir		6.5k	39	2.6k	38	6.9k	126	6.9k	103	4.8k	78	2641	38	0.20
tanglegram1		31	1791	35	157	23	287	203	inf%	1	9954.4%	23	157	
tanglegram2		3	59	3	2	7	17	302	216	15	1306	3	2	0.41
timtab1	T	855.6k	797	307.8k	373	3.4M	1568	233.4k	24.8%	700.0k	415	307773	373	0.43
triptim1	P	1	39	33	60	1	59	1	123	118	0.1%	1	39	0.48
unitcal_7		2.0k	108	39	18	3.1k	189	3.3k	1511	12.3k	1479	39	18	*1.05*
vpphard		13.6k	2481	149.3k	inf%	12.8k	inf%	6.4k	inf%	4.3k	inf%	13580	2481	
zib54-UUE		22.1k	1818	164.0k	1527	31.3k	2397	79.3k	6.5%	431.0k	9.9%	22093	1527	0.28

[a] Out-of-memory
[b] Constraint violation, absolute error: 0.0003

The column *Best* lists the minimum number of nodes and the best time taken over all solvers that solved the instance to optimality and provided a solution accepted by the solution checker. In this respect, minimum tree size and solution times may have been taken from *different* solvers. The column *VS* lists the variability score of the instance as described in Sect. 5. Higher numbers indicate higher performance variability. In order to indicate special properties of the instances, the additional test sets containing an instance, other than BENCHMARK, are listed in column *Sets*.

While each solver hits the time limit for at least 12 instances, there are only four BENCHMARK instances that none of the solvers was capable of solving within 1 h, using only one thread. These are iis-bupa-cov, m100n500k4r1, neos-1337307, and newdano. Furthermore, for each solver there is at least one instance for which this solver is faster than all the other solvers. The geometric mean of the times in the *Best* column actually is less than half of the geometric mean of the numbers in any of the solvers *Time* columns.

Table 8 gives the results for computations using 12 threads with a time limit of 1 h and a memory limit of 24 GB. Except for the last column, we list the same numbers as for the single thread computations. The last column lists the speedup between the *Best Time* column from the single thread and from the 12 thread results.

Using 12 threads, all instances with the exception of m100n500k4r1 can be solved by at least one of the solvers within 1 h. The speedup between the best of all solvers and any particular solver is now even more significant. The average overall speedup for going from 1 to 12 threads is approximately a factor of 3. In some of the cases where the speedup is less than 1 this is due to rounding the time up to the second on instances which do not benefit from using multiple threads. This rounding is also the reason for the super linear speedup reported for binkar10_1.

We purposely refrained from directly comparing the individual solvers based on measures of average solution times. There are a number of reasons for this:

- As stated in Sect. 5, we would have to sample over a sufficient number of permuted instances to decrease the impact of performance variability.
  When redoing the single thread computations on a 14% faster machine with a different memory/cache system, the speedup of the geometric mean solution time was between 8.5% and nearly 20%, depending on the particular solver.
  When redoing the 12 thread computations, even though the codes ran in deterministic mode on an empty machine, differences in wall clock time up to 73 s between two runs of the same instance could be observed.
- The result depends on several quite arbitrary choices, like the time limit, the number of threads, and the particular computer system. If we changed any of them, the results would change.
- The fact that each solver was the single fastest on some instances and only a few instances could not be solved by any solver, strongly indicates that if one wants to know which solver is the fastest for a particular problem, the only way to find out is to try them all.
- While the geometric mean of the solution times in the 12 thread setting for the commercial solvers is the same within measurement precision, the ratio between the slowest and the fastest of the three solvers can be over 1,000 on an individual

Table 8 Results for the Benchmark-Set using 12 cores, time limit 1 h. Time is given in seconds, if the time limit was reached the optimality gap is given instead. *Italic* indicates that the solution did not pass the feasibility check. *Best* lists the minimum number of nodes and time over all solvers providing a feasible solution. Nodes and Time may have *not* been taken from the same solver. *SU* gives the speedup as the ratio between the best solution times for 1 thread and 12 threads. The additional test sets containing each instance, other than BENCHMARK, are listed in *Sets*

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		UG[SCIP/SPX]		Best		
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	SU
30n20b8		158.7k	401	7.9M	19.4%	61.2k	173	abort[a]		361	200.0%	61204	173	4.6
acc-tight5	RP	53	6	743	13	235	13	395	45	151	66	53	6	0.8
aflow40b		67.1k	45	218.8k	139	535.9k	159	abort[a]		28.9k	233	28947	45	8.4
air04		765	9	122	11	214	9	1.1k	34	35	70	35	9	0.7
app1-2		378.0k	1545	330.3k	276	36	61	10.9k	3443	282	490.3%	36	61	1.1
ash608gpia-3col	I	1	14	15	203	5	222	1	571	7	94	1	14	3.4
bab5		230.5k	1048	350.6k	0.1%	25.1k	277	465.7k	8.6%	2.0k	0.9%	25086	277	2.4
beasleyC3		1.6M	1088	31.7k	29	2.2k	11	1.5M	21.5%	557.4k	22.3%	2248	11	1.2
biella1		10.8k	101	17.0k	128	2.8k	178	22.0k	371	1.5k	723	1496	101	2.4
bienst2		72.0k	9	99.5k	14	102.6k	5	104.7k	104	115.5k	128	71998	5	6.6
binkar10_1		16.5k	6	9.5k	4	2.5k	1	36.6k	67	28.3k	35	2516	1	14.0
bley_xl1		18	12	1	6	1	17	21	563	1	178	1	6	1.3
bnatt350	RP	12.5k	357	256.0k	1237	128.7k	1650	108.3k	inf%	1.4k	344	1398	344	3.5
core2536-691		1.1k	47	639	49	3.6k	122	1.1k	125	54	359	54	47	1.2
cov1075		8.7k	6	1.6M	3415	342	3	748.2k	7.0%	1.6M	1601	342	3	2.0
csched010		590.4k	391	280.4k	319	2.3M	1158	4.8M	14.1%	1.0M	1307	280429	319	3.7
danoint		472.3k	239	479.8k	336	1.6M	598	abort[a]		754.8k	599	472286	239	14.2
dfn-gwin-UUM		31.2k	12	181.2k	65	245.7k	23	91.1k	73	12.2k	57	12202	12	4.5
eil33-2		10.6k	16	10.7k	60	8.4k	38	9.5k	175	8.1k	163	8145	16	3.6
eilB101		6.2k	18	2.8k	34	41.7k	53	28.5k	231	2.2k	445	2202	18	4.4
enlight13		1.5M	273	106.6k	6	1.5M	274	abort[a]		3.1M	931	106585	6	2.2
enlight14	I	7.3M	2617	234.6k	13	14.7M	inf%	2.3M	inf%	114.7k	193	114658	13	3.2
ex9	P	1	376	1	16	1	226	1	inf%	1	87	1	16	1.1

Table 8 continued

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		UG[SCIP/SPX]		Best		
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	SU
glass4	T	2.2M	20.8%[b]	6.5M	718	1.6M	45	abort[a]		4.3M	570	1622146	45	6.2
gmu-35-40	T	1.2M	55	19.6M	0.0%	103.2M	3482	abort[a]		abort[c]		1240043	55	16.2
iis-100-0-cov		147.6k	149	88.2k	257	149.8k	215	113.2k	7.5%	50.0k	718	49991	149	10.2
iis-bupa-cov		770.5k	1576	300.3k	1246	224.1k	612	355.3k	5.6%	126.8k	1350	126850	612	>5.9
iis-pima-cov		30.8k	142	36.2k	244	20.3k	149	32.3k	695	7.3k	889	7304	142	2.2
lectsched-4-obj		5.7k	8	393	2	547	4	12.7k	170	697	121	393	2	0.5
m100n500k4r1	P	19.8M	4.2%	12.3M	4.2%	47.8M	4.2%	3.8M	4.2%	8.0M	4.2%	–	–	–
macrophage		248.1k	355	725	29	1.6M	1102	159.6k	15.8%	4.4k	27.0%	725	29	1.5
map18	R	1.1k	124	2.0k	78	1.5k	48	1.3k	733	375	1531	375	48	3.1
map20		1.1k	103	3.1k	113	1.4k	36	1.1k	625	332	1292	332	36	3.8
mcsched		117.8k	58	324.0k	263	26.7k	50	76.2k	389	15.5k	269	15487	50	4.0
mik-250-1-100-1		46.1k	2	77.1k	8	195.5k	5	256.5k	210	96.1k	19	46052	2	7.0
mine-166-5		6.2k	9	253	7	24.0k	18	2.0k	92	2.6k	67	253	7	1.0
mine-90-10		314.0k	124	172.2k	44	1.4M	270	120.0k	320	63.2k	154	63248	44	4.9
msc98-ip	R	4.7k	1342	169.9k	0.7%	6.6k	215	1.1k	inf%	29	57.0%	4737	215	3.3
mspp16	X	22	659	1	254	1	1844	abort		abort[c]		1	254	1.0
mzzv11		828	25	141	8	1	13	621	98	2.1k	850	1	8	1.0
n3div36		2.1M	3.6%	578.5k	3268	1.3M	1342	155.4k	1167	50.1k	11.0%	155362	1167	2.9
n3seq24		539	69	1	183	3.1k	187	599	1243	11	20.0%	1	69	0.9
n4-3		12.7k	116	75.7k	359	65.5k	224	808.6k	16.1%	20.1k	441	12681	116	2.8
neos-1109824		11.1k	9	2.2k	6	3.0k	3	19.7k	72	4.4k	169	2167	3	4.7
neos-1337307		252.2k	1024	818.7k	1094	638.9k	1051	175.5k	0.0%	1.2M	0.1%	252150	1024	>3.5
neos-1396125		96.6k	31	34.1k	50	28.2k	25	126.3k	521	23.9k	1278	23883	25	3.2
neos-1601936	R	1.6k	273	2.5k	19	4.8k	213	537.4k	200.0%	2.0k	1223	1558	19	1.6

Table 8 continued

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		UG[SCIP/SPX]		Best		
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	SU
neos-476283		832	143	537	104	905	106	408	1169	486	878	408	104	1.4
neos-686190		6.3k	11	17.2k	24	10.0k	20	4.6k	43	2.5k	81	2533	11	2.6
neos-849702	P	241.0k	1685	11.9k	19	1.8M	3160	107.2k	187	25.6k	412	11877	19	21.5
neos-916792		98.9k	83	205.5k	210	69.8k	183	467.9k	7.1%	898.4k	15.7%	69764	83	2.5
neos-934278		606	151	4.5k	119	1	82	abort		58	4.0%	1	82	1.0
neos13		3.9k	33	23	14	2.1k	55	8.5k	563	510	655	23	14	1.9
neos18		7.9k	5	24.8k	10	14.3k	20	28.4k	224	6.3k	65	6293	5	5.4
net12	R	3.5k	562	909	11	4.1k	229	2.2k	1010	1.7k	44.9%	909	11	8.4
netdiversion		27	96	17	67	54	550	229	1652	15	inf%	17	67	1.1
newdano		938.0k	1321	797.7k	506	950.0k	373	741.4k	1197	3.4M	924	741404	373	>9.7
noswot	T	1.6M	28	2.1M	94	382.4k	7	788.5k	182	1.5M	191	382411	7	10.3
ns1208400		8.6k	248	2.0M	inf%	3.1k	47	835.8k	inf%	141.3k	inf%	3142	47	1.7
ns1688347		2.5k	23	2.9k	6	1.3k	20	29.2k	34.7%	23.4k	671	1271	6	4.2
ns1758913		1	118	1	45	1	44	1	inf%	1	inf%	1	44	0.8
ns1766074	IT	845.6k	18	521.9k	22	1.1M	14	191.0k	75	925.5k	2366	191024	14	7.9
ns1830653		12.0k	35	20.0k	40	53.0k	284	22.8k	237	22.5k	308	12003	35	5.9
opm2-z7-s2		6.5k	236	2.8k	374	868	42	1.2k	459	1.2k	1572	868	42	1.6
pg5_34		108.1k	43	108.4k	62	156.0k	50	57.6k	190	87.9k	377	57638	43	5.7
pigeon-10	T	5.8M	118	9.2M	399	3.4M	51	abort[a]		1.6M	11.1%	3405798	51	10.9
pw-myciel4		10.9k	15	75.3k	93	711.5k	126	268.6k	3351	434.3k	1242	10893	15	10.6
qiu		5.6k	3	13.6k	11	4.1k	5	12.2k	36	10.0k	70	4143	3	3.7
rail507		3.0k	39	20.2k	685	4.2k	26	abort[a]		1.2k	3054	1250	26	4.7
ran16x16		56.4k	10	89.8k	19	238.1k	24	59.7k	52	134.6k	54	56408	10	7.3
reblock67		572.2k	84	485.1k	95	817.8k	160	114.3k	369	115.7k	86	114337	84	4.1

Table 8 continued

Name	Sets	CPLEX		XPRESS		GUROBI		CBC		UG[SCIP/SPX]		Best		
		Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	SU
rmatr100-p10		1.8k	15	791	12	1.8k	8	3.0k	39	809	397	791	8	3.5
rmatr100-p5		441	10	421	19	881	12	1.7k	63	419	1535	419	10	9.4
rmine6		944.8k	278	584.9k	220	487.9k	137	262.2k	585	474.8k	354	262168	137	3.5
rocII-4-11		250.8k	75	42.7k	45	29.7k	73	51.6k	1266	27.3k	*374*[d]	27259	45	4.6
rococoC10-001000		70.9k	91	69.1k	91	1.7M	309	40.5k	154	164.7k	370	40502	91	4.6
roll3000		32.0k	19	2.1k	5	69.4k	52	22.7k	233	672.0k	1440	2117	5	3.2
satellites1-25		4.2k	90	4.8k	45	2.5k	97	18.9k	*176*[e]	14.7k	240.0%	2496	45	2.3
sp98ic		213.5k	200	39.0k	215	14.2k	73	380.4k	2292	1.2M	0.4%	14212	73	4.0
sp98ir		6.3k	12	5.0k	18	9.9k	39	6.8k	16	4.4k	82	4376	12	3.2
tanglegram1		31	514	23	65	54	196	549	inf%	1	0.0%	23	65	2.4
tanglegram2		1	3	3	2	15	16	519	46	47	2153	1	2	1.0
timtab1	T	842.8k	62	403.8k	63	1.7M	106	abort[a]		1.5M	174	403805	62	6.0
triptim1	P	1	159	119	57	1	60	1	125	2	3011	1	57	0.7
unitcal_7		3.2k	97	45	16	3.7k	128	3.7k	664	4.5k	0.1%	45	16	1.1
vpphard		18.5k	1340	511.8k	inf%	78.0k	66.7%	abort[a]		409	inf%	18514	1340	1.9
zib54-UUE		16.2k	353	292.5k	317	40.9k	312	263.3k	1148	360.7k	811	16156	312	4.9

[a] Segmentation fault
[b] Early termination after 90 s due to bug in CPLEX 12.2.0.2, will be fixed in CPLEX 12.3
[c] Out-of-memory
[d] Objective function mismatch
[e] Constraint violation, absolute error: 0.0003

instance. The geometric mean of this ratio is over 5, and the median is more than 3, i.e., for more than half of the instances the fastest solver is at least three times faster than the slowest one. (We are not giving precise numbers here because due to the time limit there are several possibilities for computing the numbers or lower bounds.)

- We were "only" using default settings. Each of the solvers has at least 50 parameters to tune it. This often allows considerable improvements in performance, especially on instances were a certain solver has difficulties.
- Finally, there are many performance measures that could be used to decide which solver is best, like lowest number of timeouts, highest number of fastest solves, minimum total time, lowest geometric mean of solution time, etc. Interestingly, regarding the commercial solvers, depending on the measure and the number of threads used, each solver wins at least once.

8 Final remarks

Since the first release of MIPLIB, nearly 20 years ago, we have seen impressive advances in computational mixed integer programming. To support the ongoing progress, we have compiled MIPLIB 2010 to help provide a basis for evaluating future developments and to stimulate continuing algorithmic improvements.

The common efforts of collecting, experimenting with, selecting, and categorizing relevant problem instances have proven to be very insightful. Getting everybody involved and agreeing on common limits and tolerances for testing already led to beneficial discussions, code improvements, and new ideas.

Last but not least, one important insight after literally spending decades of CPU time is that one should not try to boil down the result tables to a single number. We hope that our detailed computational experiments provide an accurate snapshot of the current state-of-the-art in mixed integer programming.

To be continued...

Acknowledgments The authors wish to thank all contributors for providing us with such a variety of instances from which we could select. Thanks to: Adrian Zymolka, Alper Atamtürk, Andreas Bley, Andrew J. Miller, Arie Koster, Armin Fügenschuh, Ashutosh Mahajan, Axel Werner, Brian Borchers, Carlos Cardonha, Christian Raack, Christoph Helmberg, Christopher Cullenbine, Daniel Bienstock, Daniel Espinoza, Dinakar Gade, Dmitry Krushinsky, Elmar Swarat, F. Jordan Srour, Falk Hueffner, Federico Liberatore, Feng Qiu, François Margot, Gábor Pataki, Gerald Lach, Gerardo Gonzalez, Hamideh Anjomshoa, Harald Schilly, Iulian Ober, Jennifer Van Dinter, Jens Schulz, Joachim P. Walser, Jon Dattorro, Jonathan Ott, Jörg Rambau, Kelly Eurek, Kerem Akartunalı, Kiyan Ahmadizadeh, Kristopher A. Pruitt, Luigi Poderico, Marc Pfetsch, Marco Lübbecke, Martin Bergner, Martin Savelsbergh, Matthew Galati, Matteo Fischetti, Michael Winkler, Milind Dawande, Mustafa Atlihan, Nora Konnyu, Paul Rubin, Pavel Troubil, Ralf Borndörfer, Renjie He, Richard O'Neill, Robert Fourer, Ryuhei Miyashiro, Sam Allen, Sebastian Orlowski, Serge Bisaillon, Simge Küçükyavuz, Steffen Weider, Tally Yunes, Tatsuya Akutsu, Thomas Schlechte, William Cook, and all those which we missed. Our apologies that not all of the submitted instances could make it into one of the test sets.
We would like to thank Robert Ashford, Katsuki Fujisawa, Ed Klotz, Michael Perregaard, Edward Rothberg, and Yuji Shinano for their valuable feedback.
Thanks to Jeff Linderoth for his efforts in putting together the COR@L test suite from instances called from NEOS and elsewhere. This effort contributed directly to the construction of MIPLIB 2010.

Emilie Danna worked on performance variability in mixed integer programming from 2008 to 2010 while she was employed at ILOG then IBM.
Timo Berthold and Stefan Heinz are supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin. Kati Wolter is funded by DFG Priority Program 1307 “Algorithm Engineering”.

References

1. Aardal, K., Bixby, R.E., Hurkens, C.A.J., Lenstra, A.K., Smeltink, J.W.: Market split and basis reduction: towards a solution of the Cornuéjols–Dawande instances. INFORMS J. Comput. **12**(3), 192–202 (2000)
2. Achterberg, T., Berthold, T.: Hybrid branching. In: van Hoeve, W.J., Hooker, J.N. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, vol. 5547 of Lecture Notes in Computer Science, pp. 309–311. Springer, Berlin (2009)
3. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. Oper. Res. Lett. **34**(4), 361–372 (2006)
4. Achterberg, T., Koch, T., Tuchscherer, A.: On the effect of minor changes in model formulations. Technical Report ZR 08-29. Zuse Institute Berlin (2008)
5. Achterberg, T., Raack, C.: The MCF-separator—detecting and exploiting multi-commodity flows in MIPs. Math. Program. Comput. **2**(2), 125–165 (2010)
6. Ahmadizadeh, K., Dilkina, B., Gomes, C.P., Sabharwal, A.: An empirical study of optimization for maximizing diffusion in networks. In: Principles and Practice of Constraint Programming, vol. 6308 of Lecture Notes in Computer Science, pp. 514–521 (2010)
7. Akartunalı, K., Miller, A.J.: Computational analysis of lower bounds for big bucket production planning problems. Technical Report, http://www.optimization-online.org/DB_HTML/2007/05/1668.html, Optimization Online (2007)
8. Akartunalı, K., Miller, A.J.: A heuristic approach for big bucket multi-level production planning problems. Eur. J. Oper. Res. **193**, 396–411 (2009)
9. Akutsu, T., Hayashida, M., Tamura, T.: Integer programming-based methods for attractor detection and control of Boolean networks. In: Proceedings of the Combined 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, pp. 5610–5617 (2009)
10. Allen, S.D., Burke, E.K., Marecek, J.: A space-indexed formulation of packing boxes into a larger box. Technical Report. University of Nottingham (2010)
11. Amaldi, E., Pfetsch, M.E., Trotter, L.E. Jr.: On the maximum feasible subsystem problem, IISs, and IIS-hypergraphs. Math. Program. **95**(3), 533–554 (2003)
12. Applegate, D.L., Cook, W., Dash, S., Espinoza, D.G.: Exact solutions to linear programming problems. Oper. Res. Lett. **35**, 693–699 (2007)
13. Atamtürk, A.: On capacitated network design cut-set polyhedra. Math. Program. **92**, 425–437 (2002)
14. Atamtürk, A.: On the facets of the mixed-integer knapsack polyhedron. Math. Program. **98**, 145–175 (2003)
15. Atamtürk, A., Rajan, D.: On splittable and unsplittable capacitated network design arc-set polyhedra. Math. Program. **92**, 315–333 (2002)
16. Bai, L., Rubin, P.A.: Combinatorial Benders cuts for the minimum tollbooth problem. Oper. Res. **57**(6), 1510–1522 (2009)
17. Bai, L., Stamps, M.T., Harwood, R.C., Kollmann, C.J.: A genetic algorithm for the minimum tollbooth problem. In: Proceedings of the 2006 Meeting of the Decision Sciences Institute (2006)
18. Barutt, J., Hull, T.: Airline crew scheduling: supercomputers and algorithms. SIAM News 23(6), 20–22 (1990)
19. Belotti, P., Malucelli, F.: A Lagrangian relaxation approach for the design of networks with shared protection. In: Proceedings of the 2003 International Network Optimization Conference, pp. 72–77 (2003)
20. Benichou, M., Gauthier, J., Girodet, P., Hentges, G., Ribiere, G., Vincent, O.: Experiments in mixed-integer programming. Math. Program. **1**, 76–94 (1971)
21. Bentz, W., Martens, M., Orlowski, S., Werner, A., Wessäly, R.: FTTx-PLAN: Optimierter Aufbau von FTTx-Netzen. In: Breitbandversorgung in Deutschland, vol. 220 of ITG-Fachbericht. VDE-Verlag, Berlin (2010)
22. Bixby, R.E.: Solving real-world linear programs: a decade and more of progress. Oper. Res. **50**(1), 3–15 (2002)

23. Bixby, R.E., Boyd, E.A., Indovina, R.R.: MIPLIB: a test set of mixed integer programming problems. SIAM News **25**, 16 (1992)
24. Bixby, R.E., Ceria, S., McZeal, C., Savelsbergh, M.: An updated mixed integer programming library: MIPLIB 3.0. Optima **58**, 12–15 (1998)
25. Bley, A., Boland, N., Fricke, C., Froyland, G.: A strengthened formulation and cutting planes for the open pit mine production scheduling problem. Comput. Oper. Res. **37**, 1641–1647 (2010)
26. Bley, A., Koch, T.: Integer programming approaches to access and backbone IP-network planning. Technical Report ZR 02-41. Zuse Institute Berlin (2002)
27. Bley, A., Menne, U., Klaehne, R., Raack, C., Wessaely, R.: Multi-layer network design—a model-based optimization approach. In: Proceedings of the 5th Polish-German Teletraffic Symposium, pp. 107–116 (2008)
28. Böcker, S., Hüffner, F., Truss, A., Wahlström, M.: A faster fixed-parameter approach to drawing binary tanglegrams. In: Chen, J., Fomin, F. (eds.) Parameterized and Exact Computation, vol. 5917 of Lecture Notes in Computer Science, pp. 38–49. Springer, Berlin (2009)
29. Borndörfer, R.: Aspects of Set Packing, Partitioning, and Covering. Ph.D. Thesis, Technische Universität Berlin. Shaker Verlag, Aachen (1998)
30. Borndörfer, R., Grötschel, M., Klostermeier, F., Küttner, C.: Telebus Berlin: vehicle scheduling in a dial-a-ride system. In: Wilson, N. (ed.) Proceedings of the 7th International Workshop on Computer-Aided Transit Scheduling, vol. 471 of Lecture Notes in Economics and Mathematical Systems, pp. 391–422. Springer, Berlin (1999)
31. Borndörfer, R., Liebchen, C.: When periodic timetables are suboptimal. In: Kalcsics, J., Nickel, S. (eds.) Operations Research Proceedings 2007, pp. 449–454. Springer, Berlin (2008)
32. Borndörfer, R., Löbel, A., Weider, S.: A bundle method for integrated multi-depot vehicle and duty scheduling in public transit. In: Hickman, M., Mirchandani, P., Vo, S. (eds.) Computer-aided Systems in Public Transport, vol. 600 of Lecture Notes in Economics and Mathematical Systems, pp. 3–24 (2008)
33. Borndörfer, R., Schlechte, T.: Models for railway track allocation. In: Liebchen, C., Ahuja, R.K., Mesa, J.A. (eds.) Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems. Dagstuhl Publishing, Germany (2007)
34. Bussieck, M.R., Lindner, T., Lübbecke, M.E.: A fast algorithm for near optimal line plans. Math. Methods Oper. Res. **59**(2), 205–220 (2004)
35. Caprara, A., Fischetti, M., Toth, P.: A heuristic method for the set covering problem. Oper. Res. **47**, 730–743 (1999)
36. Chabrier, A., Danna, E., Pape, C.L., Perron, L.: Solving a network design problem. Ann. Oper. Res. **130**, 217–239 (2004)
37. Colbourn, C., Dinitz, J.: Handbook of Combinatorial Designs, 2nd ed. Chapman & Hall/CRC, Boca Raton (2006)
38. Cook, W., Koch, T., Steffy, D., Wolter K.: An exact rational mixed-integer programming solver. Integer Program. Comb. Optim. (2011, in press)
39. Cornuéjols, G., Dawande, M.: A class of hard small 0-1 programs. INFORMS J. Comput. **11**(2), 205–210 (1999)
40. Coughlan, E., Lübbecke, M., Schulz, J.: A branch-and-price algorithm for multi-mode resource leveling. In: Festa, P. (ed.) Experimental Algorithms, vol. 6049 of Lecture Notes in Computer Science, pp. 226–238. Springer, Berlin (2010)
41. Curet, N.D.: The network diversion problem. Mil. Oper. Res. **6**(2), 35–44 (2001)
42. Danna, E.: Performance variability in mixed integer programming. Presentation at Workshop on Mixed Integer Programming (2008)
43. Dattorro, J.: Convex Optimization & Euclidean Distance Geometry. Meboo Publishing, USA (2011)
44. Dawande, M., Gavirneni, S., Tayur, S.: Effective heuristics for multiproduct partial shipment models. Oper. Res. **54**(2), 337–352 (2006)
45. Dawande, M., Kalagnanam, J.: The multiple knapsack problem with color constraints. Research Report RC 21138. IBM (1998)
46. Dittel, A., Fügenschuh, A., Martin, A.: Polyhedral aspects of self-avoiding walks. Technical Report ZR 11-11. Zuse Institute Berlin (2011)
47. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)

48. Eckstein, J.: Control strategies for parallel mixed integer branch and bound. In: Proceedings of Supercomputing 1994, pp. 41–48. IEEE Computer Society Press, Washington (1994)
49. Eckstein, J.: Parallel branch-and-bound methods for mixed integer programming. SIAM News **27**(1), 12–15 (1994)
50. Eckstein, J.: Parallel branch-and-bound methods for mixed integer programming on the CM-5. SIAM J. Optim. **4**(4), 794–814 (1994)
51. Eisenblätter, A., Fügenschuh, A., Fledderus, E., Geerdes, H.-F., Heideck, B., Junglas, D., Koch, T., Kürner, T., Martin, A.: Mathematical methods for automatic optimization of UMTS radio networks. Technical Report D4.3, IST-2000-28088 MOMENTUM (2003)
52. Espinoza, D.G.: On linear programming, integer programming and cutting planes. PhD Thesis. Georgia Institute of Technology (2006)
53. Ferris, M.C., Pataki, G., Schmieta, S.: Solving the seymour problem. Optima, **66**, 2–6 (2001)
54. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Program. **104**, 91–104 (2005)
55. Fischetti, M., Lodi, A.: Local branching. Math. Program. **98**, 23–47 (2003)
56. Forrest, J.J.H., Kalagnanam, J., Ladanyi, L.: A column-generation approach to the multiple knapsack problem with color constraints. INFORMS J. Comput. **18**(1), 129–134 (2006)
57. Fourer, R., Gay, D.M., Kernighan, B.W.: AMPL: A Modelling Language for Mathematical Programming, 2nd ed. Duxbury Press, Brooks/Cole Publishing Company, Monterey (2002)
58. Gaden, D., Küçükyavuz, S.: Deterministic lot sizing with service levels. Technical Report. http://www.optimization-online.org/DB_HTML/2010/12/2844.html, Optimization Online (2010)
59. Galati, M.: Decomposition Methods for Integer Linear Programming. PhD Thesis. Lehigh University (2010)
60. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. **23**(1), 5–48 (1991)
61. Goldengorin, B., Krushinsky, D.: Complexity evaluation of benchmark instances for the p-median problem. Math. Comput. Model. **53**(9–10), 1719–1736 (2011)
62. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd ed. Johns Hopkins University Press, Baltimore (1996)
63. Goossens, J.-W., van Hoesel, S., Kroon, L.G.: A branch-and-cut approach for solving railway line-planning problems. Transp. Sci. **38**(3), 379–393 (2004)
64. Grötschel, M., Borndörfer, R., Löbel, A.: Duty scheduling in public transit. In: Jäger, W., Krebs, H.-J. (eds.) MATHEMATICS—Key Technology for the Future, pp. 653–674. Springer, Berlin (2003)
65. Günlük, O., Bienstock, D.: Computational experience with a difficult mixed-integer multicommodity flow problem. Math. Program. **68**, 213–237 (1995)
66. Helmberg, C., Röhl, S.: A case study of joint online truck scheduling and inventory management for multiple warehouses. Oper. Res. **55**(4), 733–752 (2007)
67. Holub, P., Rudová, H., Liška, M.: Data transfer planning with tree placement for collaborative environments. Constraints (2011, in press)
68. Hüffner, F., Betzler, N., Niedermeier, R.: Separator-based data reduction for signed graph balancing. J. Comb. Optim. **20**, 335–360 (2010)
69. Jünger, M., Liebling, T., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.): 50 Years of Integer Programming 1958–2008. Springer, Berlin (2009)
70. Koch, T.: Rapid Mathematical Programming. PhD Thesis. Technische Universität Berlin (2004)
71. Lau, A.: Erstellen von wegeoptimierten Stundenplänen mit Diskreten Methoden. Diploma Thesis. Technische Universität Chemnitz (2008)
72. Laundy, R., Perregaard, M., Tavares, G., Tipi, H., Vazacopoulos, A.: Solving hard mixed integer programming problems with Xpress-MP: a MIPLIB 2003 case study. INFORMS J. Comput. **21**, 304–319 (2009)
73. Liebchen, C., Möhring, R.H.: Information on the MIPLIB's timetab-instances. Technical Report 2003/49. Technische Universität Berlin, Department of Mathematics (2003)
74. Linderoth, J.T., Lee, E.K., Savelsbergh, M.W.P.: A parallel, linear programming based heuristic for large scale set partitioning problems. INFORMS J. Comput. **13**, 191–209 (2001)
75. Linderoth, J.T., Lodi, A.: MILP software. In: Cochran, J. (ed.) Wiley Encyclopedia of Operations Research and Management Science, vol. 5, pp. 3239–3248. Wiley, New York (2011)
76. Linderoth, J.T., Savelsbergh, M.W.P.: A computational study of search strategies for mixed integer programming. INFORMS J. Comput. **11**, 173–187 (1999)

77. Lodi, A.: MIP computation. In: Jünger, M., Liebling, T., Naddef, D., Nemhauser, G., Pulleyblank, W., Reinelt, G., Rinaldi, G., Wolsey, L. (eds.) 50 Years of Integer Programming 1958–2008, pp. 619–645. Springer, Berlin (2009)
78. Luzzi, I.: Exact and Heuristic Methods for Nesting Problems. PhD Thesis. University of Padova (2002)
79. Margot, F.: Small covering designs by branch-and-cut. Math. Program. B **94**, 207–220 (2003)
80. Martin, A.: Integer Programs with Block Structure. Habilitations-Schrift, Technische Universität Berlin (1998)
81. Meirich, R.: Polyedrische Untersuchung eines Linienplanungsproblems. Diploma Thesis. Technische Universität Berlin (2010)
82. Miyashiro, R., Yano, Y., Muramatsu, M.: On the maximum number of strings in go. Trans. Inf. Proces. Soc. Jpn. **48**(11), 3463–3469 (2007)
83. Nemhauser, G.L., Trick, M.A.: Scheduling a major college basketball conference. Oper. Res. **46**(1), 1–8 (1998)
84. Orlowski, S., Pióro, M., Tomaszewski, A., Wessäly, R.: SNDlib 1.0—Survivable Network Design Library. Networks **55**(3), 276–286 (2010)
85. Ortega, F., Wolsey, L.: A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. Networks **41**(3), 143–158 (2003)
86. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Solving large Steiner triple covering problems. Oper. Res. Lett. **39**, 127–131 (2011)
87. Panton, D.M., Elbers, A.W.: Mission planning for synthetic aperture radar surveillance. Interfaces **29**(2), 73–88 (1999)
88. Peeters, L.: Cyclic Railway Timetable Optimization. PhD Thesis. Erasmus Universiteit Rotterdam (2003)
89. Pfender, T.: Arboreszenz-Flüsse in Graphen: polyedrische Untersuchungen. Diploma Thesis. Technische Universität Berlin (2000)
90. Pfetsch, M.E.: Branch-and-cut for the maximum feasible subsystem problem. SIAM J. Optim. **19**, 21–38 (2008)
91. Pochet, Y., Vyve, M.V.: A general heuristic for production planning problems. INFORMS J. Comput. **16**(3), 316–327 (2004)
92. Polo, C.: Algoritmi euristici per il progetto ottimo di una rete di interconnessione. Technical Report. Testi di laurea in Ingegneria Informatica, Universitità degli Studi di Padova (2002)
93. Raack, C., Koster, A.M.C.A., Orlowski, S., Wessäly, R.: On cut-based inequalities for capacitated network design polyhedra. Networks **57**(2), 141–156 (2011)
94. Reuter, A.: Kombinatorische Auktionen und ihre Anwendungen im Schienenverkehr. Diploma Thesis. Technische Universität Berlin (2005)
95. Schilly, H.: Modellierung und Implementation eines Vorlesungsplaners. Diploma Thesis. Universität Wien (2007)
96. Sheldon, D., Dilkina, B., Elmachtoub, A., Finseth, R., Sabharwal, A., Conrad, J., Gomes, C.P., Shmoys, D., Allen, W., Amundsen, O., Vaughan, B.: Maximizing spread of cascades using network design. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, pp. 517–526 (2010)
97. Shinano, Y., Achterberg, T., Berthold, T., Heinz, S., Koch, T.: ParaSCIP—a parallel extension of SCIP. Technical Report ZR 10-27. Zuse Institute Berlin (2010)
98. Stadtler, H.: Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. Oper. Res. **51**(3), 487–502 (2003)
99. Sun, M., Aronson, J.E., McKeown, P.G., Drinka, D.A.: A tabu search heuristic procedure for the fixed charge transportation problem. Eur. J. Oper. Res. **106**, 441–456 (1998)
100. Torres Carvajal, L.M.: Online Vehicle Routing. PhD Thesis. Technische Universität Berlin (2003)
101. Troubil, P., Rudová, H.: Integer programming for media streams planning problem. In: Matyska, L., Kozubek, M., Vojnar, T., Zemcík, P., Antos, D. (eds.) In: Proceedings of the Sixth Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, vol. 16 of Open Access Series in Informatics, pp. 116–123. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Germany (2011)
102. Walser, J.P.: Radar surveillance. http://www.ps.uni-saarland.de/~walser/radar/radar.html (1997)
103. Walser, J.P.: Solving linear pseudo-boolean constraint problems with local search. In: Proceedings of the 14th National Conference on Artificial Intelligence and 9th Conference on Innovative Applications of Artificial Intelligence, pp. 269–274. AAAI Press, California (1997)

104. Walser, J.P.: Solving the ACC basketball scheduling problem with integer local search. http://www.ps.uni-saarland.de/~walser/acc/acc.html (1998)
105. Weider, S.: Integration of Vehicle and Duty Scheduling in Public Transport. PhD Thesis. Technische Universität Berlin (2007)
106. Wolsey, L.A.: Integer Programming. Wiley-Interscience, New York (1998)
107. Yunes, T.: CuSPLIB 1.0: A library of single-machine cumulative scheduling problems. http://moya.bus.miami.edu/~tallys/cusplib/ (2009)
108. Berkeley Computational Optimization Lab-Data Sets. http://ieor.berkeley.edu/~atamturk/data/
109. COR@L MIP Instances. http://coral.ie.lehigh.edu/data-sets/mixed-integer-instances/
110. Convex Optimization of Eternity II. http://www.convexoptimization.com/wikimization/index.php/Dattorro_Convex_Optimization_of_Eternity_II
111. DEIS-Operations Research Group Library of Instances. http://www.or.deis.unibo.it/research_pages/ORinstances/MIPs.html
112. Eternity II puzzle. http://www.eternityii.com
113. GNU linear programming toolkit version 4.45. http://www.gnu.org/software/glpk
114. GMP, GNU multiple precision arithmetic library. http://gmplib.org
115. Management of Inter-Warehouse-Logistics for Stochastic Demand. http://www.tu-chemnitz.de/mathematik/discrete/projects/warehouse_trucks/index.html
116. ICC, Intel C++ compiler. http://software.intel.com/en-us/articles/intel-compilers/
117. IEEE standard 754-2008 for floating-point arithmetic (2008)
118. Challenge Problems: Independent Sets in Graphs. http://www2.research.att.com/~njas/doc/graphs.html
119. lp_solve 5.5.2. http://lpsolve.sourceforge.net
120. MULTILSB: Multi-Item Lot-Sizing with Backlogging. http://personal.strath.ac.uk/kerem.akartunali/research/multi-lsb/
121. NEOS Server for Optimization. http://www.neos-server.org
122. Pseudo-Boolean Competition 2010. http://www.cril.univ-artois.fr/PB10/
123. QSopt_ex. http://www.dii.uchile.cl/~daespino/ESolver_doc/main.html
124. IBM Ponder This-August 2008. http://domino.research.ibm.com/comm/wwwr_ponder.nsf/challenges/August2008.html
125. SNDlib. http://sndlib.zib.de
126. TSPLIB. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/