Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

TIMO BERTHOLD[†]
AMBROS M. GLEIXNER[†]
STEFAN HEINZ[†]
STEFAN VIGERSKE[‡]

# ON THE COMPUTATIONAL IMPACT OF MIQCP SOLVER COMPONENTS[*]

[†] Zuse Institute Berlin, Department of Optimization, Takustr. 7, 14195 Berlin, Germany, {berthold,gleixner,heinz}@zib.de
[‡] Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, stefan@math.hu-berlin.de
[*] Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

ZIB-Report 11-01 (January 2011)

# ON THE COMPUTATIONAL IMPACT OF MIQCP SOLVER COMPONENTS*

TIMO BERTHOLD†, AMBROS M. GLEIXNER†, STEFAN HEINZ†, AND STEFAN VIGERSKE‡

**Abstract.** We provide a computational study of the performance of a state-of-the-art solver for nonconvex mixed-integer quadratically constrained programs (MIQCPs). Since successful general-purpose solvers for large problem classes necessarily comprise a variety of algorithmic techniques, we focus especially on the impact of the individual solver components. The solver SCIP used for the experiments implements a branch-and-cut algorithm based on linear outer approximation to solve MIQCPs to global optimality. Our analysis is based on a set of 86 publicly available test instances.

**1. Introduction.** Recent years have seen a strong interest in algorithms for *Mixed-Integer Nonlinear Programming* (MINLP). Advances in research are also reflected by the development and computational progress of several general-purpose solvers for MINLP or specific sub-classes, such as convex MINLP or *Mixed-Integer Quadratically Constrained Programming* (MIQCP) [1, 5, 9, 10, 11, 13, 16, 17, 21].

State-of-the-art solvers for MINLP comprise a variety of algorithmic techniques from several related fields such as *Nonlinear Programming*, *Mixed-Integer Linear Programming* (MILP), *Global Optimization* (in case nonconvex functions are present), and *Constraint Programming (CP)*. The overall computational performance of a solver crucially depends on its single constituents *and* the mutual interplay of these features. The aim of this paper is to provide a detailed computational study that investigates the impact of single MINLP solver components.

In our study, we focus on the important subclass of MIQCPs, i.e. optimization problems of the form

$$\begin{array}{lll} \min & x^T A_0 x + {b_0}^T x + c_0 & \\ \text{s.t.} & x^T A_j x + {b_j}^T x + c_j \leqslant 0 & \text{for } j = 1, \ldots, m, \\ & x_k^L \leqslant x_k \leqslant x_k^U & \text{for } k = 1, \ldots, n, \\ & x_k \in \mathbb{Z} & \text{for all } k \in \mathcal{I}, \end{array}$$

where $\mathcal{I} \subseteq \{1, \ldots, n\}$ is the index set of integer variables, $A_j \in \mathbb{R}^{n \times n}$, $b_j \in \mathbb{R}^n$, $c_j \in \mathbb{R}$ for $j = 0, \ldots, m$, and $x_k^L \in \mathbb{R} \cup \{-\infty\}$ and $x_k^U \in \mathbb{R} \cup \{+\infty\}$ are the lower and upper bounds of variable $x_k$ for $k = 1, \ldots, n$, respectively. Note that we do not require the matrices $A_i$ to be positive semidefinite, thus we allow for nonconvex constraints. If $\mathcal{I} = \emptyset$, we have a *Quadratically Constrained Quadratic Programming* problem (QCQP).

Recently, the *Constraint Integer Programming* framework SCIP [2, 4] has been extended to solve nonconvex MIQCPs to global optimality [9]. Computational results have shown the competitiveness of the solver with the current state-of-the-art. The plugin-based architecture of SCIP is particularly suited to analyze the impact of individual components. We use this solver for our computational experiments.

The remainder of this paper is organized as follows: In Section 2, we briefly outline the general solution algorithm of SCIP and the specific algorithmic techniques used for MIQCPs. Section 3 describes our selection of publicly available test problems and the results of our experiments. In Section 4, we summarize and discuss the computational results.

**2. Algorithm.** SCIP employs a branch-and-bound algorithm to solve MIQCPs to global optimality. The problem is recursively split into smaller subproblems, thereby creating a branching tree. At each subproblem, domain propagation is applied to exclude further values from the variables' domains and a linear outer approximation is solved to achieve a local lower bound (assuming minimization problems). The relaxation may be strengthened by adding further valid inequalities. In case of an infeasible subproblem, conflict analysis is performed to learn no-goods,

*Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

†Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, {berthold,gleixner,heinz}@zib.de

‡Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, stefan@math.hu-berlin.de

see, e.g., [**?**]. Primal heuristics are used as supplementary methods to improve the upper bound. Figure 1 provides a flowchart of the main solving loop of SCIP. In the following, we present a brief overview over the SCIP plugins essential for solving MIQCPs. For further details see [2, 6, 9, 24].

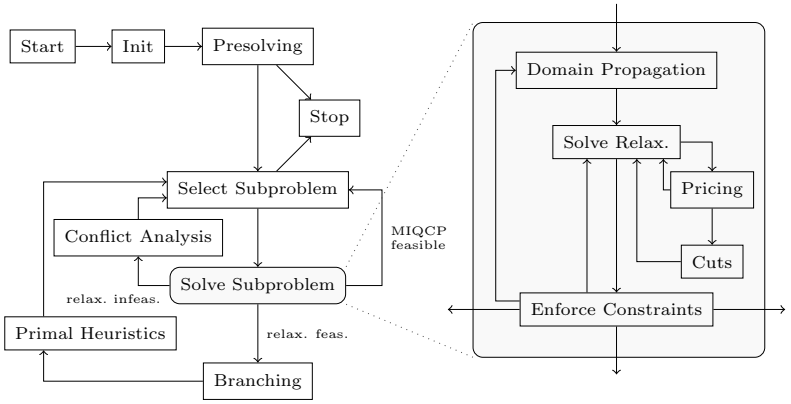


FIGURE 1. *Flowchart of the main solving loop of* SCIP

**Presolving.** During the presolving phase, a set of reformulations and simplifications are tried. Further, the domain propagation routines are used to tighten the bounds on the variables.

*MILP presolving.* Many MIQCPs contain a large linear and discrete part, for which SCIP's default MILP presolving routines [2] are applied.

*Products containing binary variables.* Products of a binary variable with a linear term, i.e. $x \sum_{i=1}^{k} a_i y_i$, where $x$ is a binary variable, $y_i$ are variables with finite bounds, and $a_i \in \mathbb{R}$, $i = 1, \ldots, k$, are replaced by new variables $z \in \mathbb{R}$ and linear inequalities guaranteeing $z = x \sum_{i=1}^{k} a_i y_i$ for $x \in \{0, 1\}$.

*Second-order cone (SOC) constraints.* Constraints of the form $\gamma + \sum_{i=1}^{k} (\alpha_i (x_i + \beta_i))^2 \leqslant (\alpha_0 (y + \beta_0))^2$ with $k \geqslant 2$, $\alpha_i, \beta_i \in \mathbb{R}$ for $i = 0, \ldots, k$, $\gamma \in \mathbb{R}_+$, and $y^L \geqslant -\beta_0$ are automatically recognized during presolving and handled by a specialized SOC constraint handler.

*Convexity check.* After the presolving phase, each quadratic function is checked for convexity by computing the sign of the minimum eigenvalue of the coefficient matrix $A$. For instances with bilinear terms, this information is essential for separation.

**Separation.** If the current solution $\tilde{x}$ of the LP outer approximation violates a constraint, we may add valid cutting planes in order to strengthen the formulation.

*MILP cutting planes.* To cut off fractional LP solutions, SCIP's standard MILP separators are used at the root node. They comprise general techniques like Gomory cuts and problem specific separation routines like knapsack cover cuts, for an overview and a computational study see [24].

*Separation of quadratic and SOC constraints.* If a violated constraint is known to be convex, it is always possible to linearize the constraint function at $\tilde{x}$. In SCIP, a quadratic constraint is recognized as convex if it is either a SOC constraint or its coefficient matrix is positive semidefinite. For a violated nonconvex quadratic constraint, each term of a quadratic function $\sum_{i,j} a_{i,j} x_i x_j$ is individually underestimated by a linearization for $a_{i,i} x_i^2$ with $a_{i,i} > 0$, a secant for $a_{i,i} x_i^2$ with $a_{i,i} < 0$, and a McCormick underestimator [18] for a bilinear term, respectively. If a linear inequality generated by this method does not cut off the current LP solution $\tilde{x}$, the infeasibility is resolved by branching.

**Domain propagation.** In domain propagation, deductions of the variables' local domains are inferred. These can yield stronger linear underestimators in the separation procedures, they may cut off nodes due to infeasibility of a constraint, and can result in further domain deductions on other constraints. For quadratic constraints, we implemented an interval-arithmetic based method similar to [14]. Domain propagation for linear constraints, special cases like knapsack constraints, and global MILP propagators are described in [2].

**Conflict Analysis.** If domain propagation routines or the LP solver detect infeasibility of a subproblem, a (preferably small) set of domain reductions is determined which are sufficient to prove the infeasibility. This gives rise to globally valid no-goods, which may help to prune the tree in the remaining search. In the current implementation of SCIP, quadratic constraints do not take part in conflict analysis. Analysis of the linear core of an MIQCP might, however, still generate short no-goods.

**Branching.** If an integer-infeasible LP relaxation solution $\tilde{x}$ cannot be cut off by separation or domain propagation, an integer variable with fractional value is selected for branching. SCIP's default branching variable selection rule is "hybrid branching" [3], which combines pseudo cost based reliability branching with *VSIDS* and inference/impact based branching.

Only if $\tilde{x}$ is integer feasible but violates a nonconvex quadratic constraint, we perform a spatial branching operation. Therefore, we use a pseudo-cost based branching rule as suggested in [5] to select the branching variable. Note that feasibility of a convex quadratic constraint can always be enforced by separation.

**Primal heuristics.** When solving MIQCPs, we still make use of all default MILP primal heuristics of SCIP [6]. Even if solutions suggested by MILP heuristics are infeasible for the quadratic part of the problem, they might serve as starting points for nonlinear repair and improvement heuristics. Additionally, three *large neighborhood search* heuristics specific to MIQCPs are implemented in SCIP, a QCQP based local search [9], an extended form of the RENS heuristic [7], and the novel Undercover heuristic [8].

**3. Computational experiments.** For our experiments, we compiled a test set of 86 publicly available MIQCP instances from different sources: *constrained layout problems* (`clay*`) and *safety layout problems* (`SLay*`) from [20], Hans Mittelmann's MIQP benchmark instances (`i*`) [19], *portfolio optimization problems* (`classical*`, `robust*`, `shortfall*`) from [22], *truss structure design problems* (`*bar*`) from [25], *uncapacitated facility location problems* (`uflquad*`) from [15], and selected instances from the MINLPLib [12]. Problem statistics for the original instances and after default presolving are given in the Appendix (Table 2).

Initially, we ran all instances with default settings as outlined in Section 2. To measure the impact of individual components, we compare the default run to the performance with a feature disabled or switched to a simpler strategy. Since many MIQCP instances contain a considerable linear and discrete part, we also investigate the effect of the classical MILP components. All in all, we compared 11 alternative settings against the SCIP default: we disabled linear presolving, binary reformulations, the detection of SOC constraints, convexity checks, domain propagation, cutting plane separation, usage of a linear outer approximation, and primal heuristics; we further altered the variable selection strategy to random and the node selection to depth first search.

Our experiments were conducted on a 2.66 GHz Intel® Xeon® 5150 with 4 MB cache and 8 GB RAM. We used SCIP 1.2.1.1 with CPLEX 12.1.0 [16] as LP solver, Ipopt 3.8.1 [23] as QCQP solver for the heuristics, and LAPACK 3.1.0 to compute eigenvalues. The optimality tolerance was set to zero, the relative feasibility tolerance to $10^{-6}$. We imposed a time limit of one hour and a memory limit of 4 GB for SCIP and 4 GB for the underlying solvers.

With default settings, SCIP could solve 55 of the 86 instances within the time limit of one hour. For 4 of the unsolved instances, no feasible solution was found. Table 1 shows the impact if a particular component of SCIP is switched off or changed to a simpler mode. Obviously, some of the features may only have an effect on a certain subset of the test set, e.g., disabling upgrading of SOC constraints is only applied if such constraints are present in the model. For those tests, we split the test set in a "relevant" and a "control" group, expecting no change in performance for the control group. Column "size" gives the number of instances in the respective test group.

All performance measures are w.r.t. the default settings of SCIP. We count the absolute number of instances for which a particular setting was more than 10% faster or slower, reported in columns "better" and "worse", respectively; for instances that could not be solved within the time limit, we compare the bounds at termination. Further, we compare the shifted geometric mean of the overall running time, the number of branch-and-bound nodes, and the time until the

TABLE 1

*Impact of implemented MIQCP methods. Column "size" gives the number of instances in the test group. Performance measures are absolute/relative differences compared to* SCIP *with default settings.*

| | | | primal bound | | dual bound | | running time | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| disabled feature | size | solved | better | worse | better | worse | better | worse | mean | to first sol. | to opt sol. | nodes |
| linear presolving | 86 | **−1** | **3** | **4** | **5** | **9** | **15** | **19** | **+3%** | **+14%** | **+12%** | −1% |
| binary var. reform. | 86 | 0 | **2** | **6** | **1** | **6** | **4** | **16** | **+4%** | **+28%** | **+7%** | −9% |
| relevant | 27 | 0 | **1** | **5** | **0** | **5** | **2** | **11** | **+11%** | **+75%** | **+37%** | −31% |
| control | 59 | 0 | 1 | 1 | 1 | 1 | **2** | **5** | **+1%** | **+7%** | −9% | **+2%** |
| SOC upgrades | 86 | **−9** | **1** | **3** | **1** | **13** | **3** | **13** | **+41%** | **+28%** | −1% | **+48%** |
| relevant | 12 | **−8** | **0** | **2** | **0** | **9** | **0** | **9** | **+1098%** | **+316%** | 0% | **+2048%** |
| control | 74 | **−1** | 1 | 1 | **1** | **4** | **3** | **4** | −1% | **+2%** | −1% | −4% |
| convexity check | 86 | **−4** | 3 | 3 | **2** | **8** | **5** | **7** | **+29%** | −3% | 0% | **+59%** |
| relevant | 13 | **−3** | 2 | 2 | **1** | **4** | **2** | **6** | **+478%** | −18% | −6% | **+3005%** |
| control | 73 | **−1** | 1 | 1 | **1** | **4** | 3 | 1 | −1% | **+1%** | −2% | −4% |
| domain propagation | 86 | **−2** | 4 | 4 | **4** | **13** | 18 | 14 | **+7%** | **+31%** | **+36%** | **+34%** |
| MIP cuts | 86 | **−1** | 3 | 2 | **5** | **6** | **15** | **19** | **+3%** | **+1%** | **+14%** | **+12%** |
| nonlin. separation | 86 | **−19** | **2** | **15** | **0** | **33** | **3** | **30** | **+140%** | **+26%** | **+60%** | **+680%** |
| relevant | 68 | **−19** | **2** | **14** | **0** | **33** | **1** | **30** | **+202%** | **+31%** | **+189%** | **+1300%** |
| control | 18 | 0 | **0** | **1** | 0 | 0 | 2 | 0 | −3% | **+5%** | −6% | −15% |
| hybrid branching | 86 | **−8** | **1** | **5** | **1** | **21** | **5** | **29** | **+62%** | **+10%** | **+37%** | **+106%** |
| relevant | 82 | **−8** | **1** | **5** | **1** | **20** | **5** | **29** | **+67%** | **+10%** | **+41%** | **+112%** |
| control | 4 | 0 | 0 | 0 | **0** | **1** | 0 | 0 | **+1%** | 0% | 0% | 0% |
| best est. nodesel. | 86 | **−2** | **2** | **12** | **2** | **18** | **13** | **18** | **+3%** | **+25%** | **+48%** | **+5%** |
| relevant | 82 | **−2** | **2** | **12** | **2** | **18** | **13** | **18** | **+3%** | **+26%** | **+53%** | **+5%** |
| control | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% | −1% | 0% | 0% |
| primal heuristics | 86 | **−3** | **0** | **19** | **4** | **11** | 17 | 13 | **+2%** | **+347%** | **+9%** | **+16%** |
| conflict analysis | 86 | **−2** | 4 | 4 | **3** | **7** | **8** | **10** | **+5%** | **+1%** | −2% | **+5%** |
| relevant | 82 | **−2** | 4 | 4 | **3** | **7** | **8** | **10** | **+5%** | **+1%** | −2% | **+5%** |
| control | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0% | −1% | **+1%** | 0% |

first and the optimal solution were found.

Table 1 shows that disabling a feature always leads to an increase in overall computation time and, except for linear presolving and binary variable reformulation, the number of branch-and-bound nodes. Even more importantly, except for binary variable reformulation, there is always at least one instance which could not be solved after disabling a certain component. We further see that the features specific to nonlinear optimization, like using an outer approximation, specialized algorithms for SOCs, or convexity detection, have by far the biggest impact. Using a sophisticated branching rule also reduces the computational effort tremendously. MILP specific features like linear presolving, cuts, and conflict analysis on the linear part are less successful than in pure MILP, but still slightly reduce the running time and the number of branch-and-bound nodes. The results for domain propagation surprised us. It gives a clear benefit w.r.t. the number of branching nodes and the dual bounds at termination, but this is not reflected by the computation time. Primal heuristics slightly improve the overall computation time, but very much help to find a first feasible solution and to obtain a good primal bound if a run has to be terminated due to a time limit.

**4. Conclusion and Outlook.** In this paper, we gave a brief overview over different algorithmic parts of the branch-and-cut framework SCIP and discussed their relevance for solving MIQCPs. The main focus was the last section, which presented and discussed computational results on the individual impact of those components. All 11 features proved to be beneficial for the overall performance. The parts specific to nonlinear optimization clearly made the biggest difference, but they often only operate on a small subset of instances.

The results for the MILP and CP components suggest that these techniques do not unfold their full potential for MINLP, yet. For some techniques like conflict analysis, this is probably due to a theoretical gap between MILP/CP and MINLP.

Finally, our experiments were only performed for one specific solver, which employs an LP-based branch-and-cut approach. It would be interesting to see the outcome of similar experiments for other solvers and algorithms.

## REFERENCES


[1] K. Abhishek, S. Leyffer, and J. T. Linderoth, *FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs*, Tech. Rep. ANL/MCS-P1374-0906, Argonne National Laboratory, Mathematics and Computer Science Division, 2006.

[2] T. Achterberg, *Constraint Integer Programming*, PhD thesis, Technische Universität Berlin, 2007.

[3] T. Achterberg and T. Berthold, *Hybrid branching*, in Proc. of CPAIOR 2009, W. J. van Hoeve and J. N. Hooker, eds., vol. 5547 of LNCS, Springer, May 2009, pp. 309–311.

[4] T. Achterberg, T. Berthold, T. Koch, and K. Wolter, *Constraint integer programming: A new approach to integrate CP and MIP*, in Proc. of CPAIOR 2008, L. Perron and M. Trick, eds., vol. 5015 of LNCS, Springer, 2008, pp. 6–20.

[5] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter, *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods and Software, 24 (2009), pp. 597–634.

[6] T. Berthold, *Primal heuristics for mixed integer programs*, Master's thesis, Technische Universität Berlin, 2006.

[7] ———, *RENS – relaxation enforced neighborhood search*, ZIB-Report 07-28, Zuse Institute Berlin, 2007.

[8] T. Berthold and A. M. Gleixner, *Undercover – a primal heuristic for MINLP based on sub-MIPs generated by set covering*, ZIB-Report 09-40, Zuse Institute Berlin, 2009.

[9] T. Berthold, S. Heinz, and S. Vigerske, *Extending a CIP framework to solve MIQCPs*, ZIB-Report 09-23, Zuse Institute Berlin, 2009.

[10] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. W. Sawaya, and A. Wächter, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optim., 5 (2008), pp. 186–204.

[11] P. Bonami, M. Kilinç, and J. Linderoth, *Algorithms and software for convex mixed integer nonlinear programs*. available at Optimization Online, `http://www.optimization-online.org/DB_HTML/2009/10/2429.html`, 2009.

[12] M. R. Bussieck, A. S. Drud, and A. Meeraus, *MINLPLib - a collection of test models for mixed-integer nonlinear programming*, INFORMS J. Comput., 15 (2003), pp. 114–119.

[13] M. R. Bussieck and S. Vigerske, *MINLP solver software*, Tech. Rep. 691, Matheon, 2010. `http://www.matheon.de`.

[14] F. Domes and A. Neumaier, *Constraint propagation on quadratic constraints*, Constraints, to appear (2010). available online at `http://www.mat.univie.ac.at/~dferi/publications.html`.

[15] O. Günlük, J. Lee, and R. Weismantel, *Minlp strengthening for separable convex quadratic transportation-cost ufl*, IBM Research Report RC23771, 2007.

[16] IBM, *CPLEX*. `http://ibm.com/software/integration/optimization/cplex`.

[17] Y. Lin and L. Schrage, *The global solver in the LINDO API*, Optimization Methods and Software, 24 (2009), pp. 657–668.

[18] G. P. McCormick, *Computability of global solutions to factorable nonconvex programs: Part I-Convex Underestimating Problems*, Math. Program., 10 (1976), pp. 147–175.

[19] H. Mittelmann, *MIQP test instances*. `http://plato.asu.edu/ftp/miqp.html`.

[20] N. Sawaya, *Reformulations, relaxations and cutting planes for generalized disjunctive programming*, PhD thesis, Carnegie Mellon University, 2006.

[21] M. Tawarmalani and N. V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer Academic Publishers, 2002.

[22] J. P. Vielma, S. Ahmed, and G. L. Nemhauser, *A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs*, INFORMS J. Comput., 20 (2008), pp. 438–450.

[23] A. Wächter and L. T. Biegler, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2006), pp. 25–57.

[24] K. Wolter, *Implementation of cutting plane separators for mixed integer programs*, Master's thesis, Technische Universität Berlin, 2006.

[25] T. Yunes, I. D. Aron, and J. N. Hooker, *An integrated solver for optimization problems*, tech. rep., University of Miami, 2008.

## APPENDIX

Table 2 presents statistics on the size and structure of the 86 MIQCP instances in our test set. We show each instance before and after the default presolving of SCIP. Columns "bin", "int", and "vars" give the number of binary variables, general integers, and the total number (including continuous) of variables, respectively. Columns "soc", "quad", and "linear" show the number of second order cone, general quadratic, and linear constraints, respectively. If all quadratic constraints of an instance were recognized as convex or concave, a checkmark is set in column "conv". Note that this is different from the test set relevant for the convexity check in Table 1, which consists of all instances that have convex constraints with bilinear terms.

TABLE 2
*Problem statistics before and after default presolving*

| | original problem | | | | | | | presolved problem | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| instance | vars | int | bin | linear | quad | soc | conv | vars | int | bin | linear | quad | soc | conv |
| 108bar | 1872 | 0 | 1188 | 1500 | 216 | 0 | | 939 | 0 | 263 | 484 | 216 | 0 | |
| 10bar2 | 176 | 0 | 110 | 56 | 20 | 0 | | 154 | 0 | 110 | 34 | 20 | 0 | |
| 200bar | 7850 | 0 | 6000 | 4570 | 600 | 0 | | 4532 | 0 | 2880 | 1175 | 600 | 0 | |
| SLay05H | 231 | 0 | 40 | 290 | 1 | 0 | ✓ | 231 | 0 | 40 | 290 | 1 | 0 | ✓ |
| SLay05M | 71 | 0 | 40 | 90 | 1 | 0 | ✓ | 71 | 0 | 40 | 90 | 1 | 0 | ✓ |
| SLay07M | 141 | 0 | 84 | 189 | 1 | 0 | ✓ | 141 | 0 | 84 | 189 | 1 | 0 | ✓ |
| SLay10M | 291 | 0 | 180 | 405 | 1 | 0 | ✓ | 291 | 0 | 180 | 405 | 1 | 0 | ✓ |
| classical_200_0 | 601 | 0 | 200 | 403 | 1 | 0 | ✓ | 600 | 0 | 200 | 402 | 1 | 0 | ✓ |
| classical_200_1 | 601 | 0 | 200 | 403 | 1 | 0 | ✓ | 600 | 0 | 200 | 402 | 1 | 0 | ✓ |
| classical_20_0 | 61 | 0 | 20 | 43 | 1 | 0 | ✓ | 60 | 0 | 20 | 42 | 1 | 0 | ✓ |
| classical_20_1 | 61 | 0 | 20 | 43 | 1 | 0 | ✓ | 60 | 0 | 20 | 42 | 1 | 0 | ✓ |
| classical_50_0 | 151 | 0 | 50 | 103 | 1 | 0 | ✓ | 150 | 0 | 50 | 102 | 1 | 0 | ✓ |
| classical_50_1 | 151 | 0 | 50 | 103 | 1 | 0 | ✓ | 150 | 0 | 50 | 102 | 1 | 0 | ✓ |
| clay0205m | 81 | 0 | 50 | 96 | 40 | 0 | ✓ | 75 | 0 | 45 | 90 | 40 | 0 | ✓ |
| clay0305m | 86 | 0 | 55 | 96 | 60 | 0 | ✓ | 81 | 0 | 51 | 93 | 60 | 0 | ✓ |
| du-opt | 21 | 13 | 0 | 9 | 1 | 0 | ✓ | 21 | 13 | 0 | 5 | 1 | 0 | ✓ |
| ex1263 | 93 | 0 | 72 | 52 | 4 | 0 | | 91 | 0 | 71 | 47 | 4 | 0 | |
| ex1264 | 89 | 0 | 68 | 52 | 4 | 0 | | 82 | 0 | 62 | 47 | 4 | 0 | |
| ex1265 | 131 | 0 | 100 | 70 | 5 | 0 | | 122 | 0 | 92 | 65 | 5 | 0 | |
| ex1266 | 181 | 0 | 138 | 90 | 6 | 0 | | 168 | 0 | 126 | 81 | 6 | 0 | |
| fac3 | 67 | 0 | 12 | 33 | 1 | 0 | ✓ | 67 | 0 | 12 | 33 | 1 | 0 | ✓ |
| feedtray2 | 88 | 0 | 36 | 137 | 147 | 0 | | 300 | 0 | 12 | 1001 | 147 | 0 | |
| iair04 | 8905 | 0 | 8904 | 823 | 1 | 0 | | 12858 | 0 | 7363 | 17483 | 0 | 0 | ✓ |
| iair05 | 7196 | 0 | 7195 | 426 | 1 | 0 | | 10571 | 0 | 6117 | 14202 | 0 | 0 | ✓ |
| ibc1 | 1752 | 0 | 252 | 1913 | 1 | 0 | | 865 | 0 | 252 | 1436 | 0 | 0 | ✓ |
| ibell3a | 123 | 29 | 31 | 104 | 1 | 0 | ✓ | 130 | 29 | 31 | 164 | 1 | 0 | ✓ |
| ibienst1 | 506 | 0 | 28 | 576 | 1 | 0 | ✓ | 473 | 0 | 28 | 592 | 0 | 0 | ✓ |
| icap6000 | 6001 | 0 | 6000 | 2171 | 1 | 0 | ✓ | 7301 | 0 | 5865 | 6307 | 0 | 0 | ✓ |
| icvxqp1 | 10001 | 10000 | 0 | 5000 | 1 | 0 | ✓ | 10003 | 9998 | 2 | 5006 | 1 | 0 | ✓ |
| ieilD76 | 1899 | 0 | 1898 | 75 | 1 | 0 | | 2686 | 0 | 1898 | 3170 | 0 | 0 | ✓ |
| ilaser0 | 1003 | 151 | 0 | 1000 | 1 | 0 | ✓ | 1003 | 151 | 0 | 1000 | 1 | 0 | ✓ |
| imas284 | 152 | 0 | 150 | 68 | 1 | 0 | | 228 | 0 | 150 | 299 | 0 | 0 | ✓ |
| imisc07 | 261 | 0 | 259 | 212 | 1 | 0 | | 360 | 0 | 238 | 583 | 0 | 0 | ✓ |
| imod011 | 10958 | 1 | 96 | 4480 | 1 | 0 | ✓ | 8962 | 1 | 96 | 2727 | 1 | 0 | ✓ |
| inug08 | 1633 | 0 | 1632 | 912 | 1 | 0 | ✓ | 2223 | 0 | 1632 | 3096 | 0 | 0 | ✓ |
| iportfolio | 1201 | 192 | 775 | 201 | 1 | 0 | ✓ | 1201 | 192 | 775 | 201 | 1 | 0 | ✓ |
| iqiu | 841 | 0 | 48 | 1192 | 1 | 0 | ✓ | 871 | 0 | 48 | 1285 | 0 | 0 | ✓ |
| iran13x13 | 339 | 0 | 169 | 195 | 1 | 0 | | 469 | 0 | 169 | 588 | 0 | 0 | ✓ |
| iran8x32 | 513 | 0 | 256 | 296 | 1 | 0 | | 649 | 0 | 256 | 707 | 0 | 0 | ✓ |
| isqp | 1001 | 50 | 0 | 249 | 1 | 0 | ✓ | 1001 | 50 | 0 | 249 | 1 | 0 | ✓ |
| iswath2 | 6405 | 0 | 2213 | 483 | 1 | 0 | | 8007 | 0 | 2213 | 5632 | 0 | 0 | ✓ |
| itointqor | 51 | 50 | 0 | 0 | 1 | 0 | ✓ | 51 | 50 | 0 | 0 | 1 | 0 | ✓ |
| ivalues | 203 | 202 | 0 | 1 | 1 | 0 | | 203 | 202 | 0 | 1 | 1 | 0 | |
| lop97ic | 1754 | 831 | 831 | 52 | 40 | 0 | | 5228 | 708 | 708 | 11521 | 0 | 0 | ✓ |
| lop97icx | 987 | 831 | 68 | 48 | 40 | 0 | | 488 | 68 | 68 | 1138 | 0 | 0 | ✓ |
| meanvarx | 36 | 0 | 14 | 44 | 1 | 0 | ✓ | 30 | 0 | 12 | 36 | 1 | 0 | ✓ |
| netmod_dol1 | 1999 | 0 | 462 | 3137 | 1 | 0 | ✓ | 1993 | 0 | 462 | 3131 | 1 | 0 | ✓ |
| netmod_dol2 | 1999 | 0 | 462 | 3080 | 1 | 0 | ✓ | 1592 | 0 | 454 | 2637 | 1 | 0 | ✓ |
| netmod_kar1 | 457 | 0 | 136 | 666 | 1 | 0 | ✓ | 453 | 0 | 136 | 662 | 1 | 0 | ✓ |

continued on next page

continued from previous page

| | original problem | | | | | | | presolved problem | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| instance | vars | int | bin | linear | quad | soc | conv | vars | int | bin | linear | quad | soc | conv |
| netmod_kar2 | 457 | 0 | 136 | 666 | 1 | 0 | ✓ | 453 | 0 | 136 | 662 | 1 | 0 | ✓ |
| nous1 | 51 | 0 | 2 | 15 | 29 | 0 | | 47 | 0 | 2 | 11 | 29 | 0 | |
| nous2 | 51 | 0 | 2 | 15 | 29 | 0 | | 47 | 0 | 2 | 11 | 29 | 0 | |
| nuclear14a | 993 | 0 | 600 | 50 | 584 | 0 | | 1568 | 0 | 600 | 2377 | 560 | 0 | |
| nuclear14b | 1569 | 0 | 600 | 1226 | 560 | 0 | | 1568 | 0 | 600 | 1225 | 560 | 0 | |
| nvs19 | 9 | 8 | 0 | 0 | 9 | 0 | | 9 | 8 | 0 | 0 | 9 | 0 | |
| nvs23 | 10 | 9 | 0 | 0 | 10 | 0 | | 10 | 9 | 0 | 0 | 10 | 0 | |
| pb351535 | 526 | 0 | 525 | 50 | 1 | 0 | | 1049 | 0 | 525 | 1622 | 0 | 0 | ✓ |
| product | 1554 | 0 | 107 | 1794 | 132 | 0 | | 446 | 0 | 92 | 450 | 82 | 0 | |
| product2 | 2843 | 0 | 128 | 2598 | 528 | 0 | | 480 | 0 | 128 | 338 | 128 | 0 | |
| qap | 226 | 0 | 225 | 30 | 1 | 0 | | 448 | 0 | 225 | 699 | 0 | 0 | ✓ |
| qapw | 451 | 0 | 225 | 255 | 1 | 0 | | 675 | 0 | 225 | 930 | 0 | 0 | ✓ |
| robust_100_0 | 404 | 0 | 101 | 305 | 2 | 0 | | 403 | 0 | 101 | 304 | 1 | 1 | ✓ |
| robust_100_1 | 404 | 0 | 101 | 305 | 2 | 0 | | 403 | 0 | 101 | 304 | 1 | 1 | ✓ |
| robust_200_0 | 804 | 0 | 201 | 605 | 2 | 0 | | 803 | 0 | 201 | 604 | 1 | 1 | ✓ |
| robust_20_0 | 84 | 0 | 21 | 65 | 2 | 0 | | 83 | 0 | 21 | 64 | 1 | 1 | ✓ |
| robust_50_0 | 204 | 0 | 51 | 155 | 2 | 0 | | 203 | 0 | 51 | 154 | 1 | 1 | ✓ |
| robust_50_1 | 204 | 0 | 51 | 155 | 2 | 0 | | 203 | 0 | 51 | 154 | 1 | 1 | ✓ |
| sep1 | 30 | 0 | 2 | 26 | 6 | 0 | | 19 | 0 | 2 | 15 | 6 | 0 | |
| shortfall_100_0 | 405 | 0 | 101 | 306 | 2 | 0 | | 404 | 0 | 101 | 305 | 0 | 2 | ✓ |
| shortfall_100_1 | 405 | 0 | 101 | 306 | 2 | 0 | | 404 | 0 | 101 | 305 | 0 | 2 | ✓ |
| shortfall_200_0 | 805 | 0 | 201 | 606 | 2 | 0 | | 804 | 0 | 201 | 605 | 0 | 2 | ✓ |
| shortfall_20_0 | 85 | 0 | 21 | 66 | 2 | 0 | | 84 | 0 | 21 | 65 | 0 | 2 | ✓ |
| shortfall_50_0 | 205 | 0 | 51 | 156 | 2 | 0 | | 204 | 0 | 51 | 155 | 0 | 2 | ✓ |
| shortfall_50_1 | 205 | 0 | 51 | 156 | 2 | 0 | | 204 | 0 | 51 | 155 | 0 | 2 | ✓ |
| space25 | 894 | 0 | 750 | 211 | 25 | 0 | | 767 | 0 | 716 | 118 | 25 | 0 | |
| spectra2 | 70 | 0 | 30 | 65 | 8 | 0 | | 68 | 0 | 30 | 30 | 8 | 0 | |
| tln12 | 169 | 156 | 12 | 61 | 12 | 0 | | 180 | 144 | 24 | 85 | 11 | 0 | |
| tln5 | 36 | 30 | 5 | 26 | 5 | 0 | | 35 | 30 | 5 | 20 | 5 | 0 | |
| tln6 | 49 | 42 | 6 | 31 | 6 | 0 | | 48 | 42 | 6 | 24 | 6 | 0 | |
| tln7 | 64 | 56 | 7 | 36 | 7 | 0 | | 63 | 56 | 7 | 28 | 7 | 0 | |
| tltr | 49 | 36 | 12 | 52 | 3 | 0 | | 56 | 27 | 20 | 73 | 2 | 0 | |
| uflquad-15-60 | 916 | 0 | 15 | 960 | 1 | 0 | ✓ | 916 | 0 | 15 | 960 | 1 | 0 | ✓ |
| uflquad-20-50 | 1021 | 0 | 20 | 1050 | 1 | 0 | ✓ | 1021 | 0 | 20 | 1050 | 1 | 0 | ✓ |
| uflquad-30-100 | 3031 | 0 | 30 | 3100 | 1 | 0 | ✓ | 3031 | 0 | 30 | 3100 | 1 | 0 | ✓ |
| uflquad-40-80 | 3241 | 0 | 40 | 3280 | 1 | 0 | ✓ | 3241 | 0 | 40 | 3280 | 1 | 0 | ✓ |
| waste | 2485 | 0 | 400 | 624 | 1368 | 0 | | 1238 | 0 | 400 | 516 | 1230 | 0 | |