Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

AMBROS M. GLEIXNER AND STEFAN WELTGE

Learning and Propagating Lagrangian Variable Bounds for Mixed-Integer Nonlinear Programming

This paper is to appear in the Proceedings of the 10th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR 2013) held May 18–22, 2013, in Yorktown Heights, NY, USA. The final publication is available at http://www.springerlink.com/.

ZIB-Report 13-04 (January 2013)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Learning and Propagating Lagrangian Variable Bounds for Mixed-Integer Nonlinear Programming*

Ambros M. Gleixner† and Stefan Weltge‡

January 2013

Abstract

Optimization-based bound tightening (OBBT) is a domain reduction technique commonly used in nonconvex mixed-integer nonlinear programming that solves a sequence of auxiliary linear programs. Each variable is minimized and maximized to obtain the tightest bounds valid for a global linear relaxation. This paper shows how the dual solutions of the auxiliary linear programs can be used to learn what we call *Lagrangian variable bound constraints*. These are linear inequalities that explain OBBT's domain reductions in terms of the bounds on other variables and the objective value of the incumbent solution. Within a spatial branch-and-bound algorithm, they can be learnt a priori (during OBBT at the root node) and propagated within the search tree at very low computational cost. Experiments with an implementation inside the MINLP solver SCIP show that this reduces the number of branch-and-bound nodes and speeds up solution times.

1 Introduction

Mixed-integer nonlinear programming studies the large class of mathematical programs specified by a nonlinear objective function, nonlinear constraints, and integrality requirements on some of the variables. It comprises the special cases of mixed-integer linear programming and nonlinear programming and provides a flexible modelling tool for a wide range of academic and industrial applications. For a detailed discussion, see, e.g., [20].

*This article is to appear in the Proceedings of the 10th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR 2013) held May 18–22, 2013, in Yorktown Heights, NY, USA.

†Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, gleixner@zib.de.

‡Otto-von-Guericke-Universität Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany, weltge@ovgu.de.

We consider *mixed-integer nonlinear programs* (MINLPs) of the form

$$\min\{\, c^{\mathsf{T}}x : x \in \mathcal{X}, x \in [\ell, u], x_j \in \mathbb{Z} \text{ for } j \in \mathcal{I}\,\}, \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$, ℓ and u are the vectors of lower bounds $\ell_j \in \mathbb{R} \cup \{-\infty\}$ and upper bounds $u_j \in \mathbb{R} \cup \{+\infty\}$, and $\mathcal{I} \subseteq \{1, \ldots, n\}$ is the index set of integer variables. Without loss of generality, we assume a linear objective, since for a nonlinear objective function $f(x)$, we can append the constraint $f(x) \leqslant x_0$ and minimize x_0. The feasible region $\mathcal{X}$ is specified by a list of linear and nonlinear constraints $g_i(x) \leqslant 0$, where the g_i (and hence $\mathcal{X}$) may be nonconvex.

Many complete algorithms for solving nonconvex MINLPs to (ε-)global optimality rely on spatial branch-and-bound combined with a convex relaxation. Domain reduction procedures have become a crucial element of state-of-the-art MINLP solvers because they not only reduce the size of the search space (as in mixed-integer or constraint programming), but specifically because smaller domains allow for tighter convex relaxations of the nonconvex constraints.

This paper is concerned with a specific domain reduction technique often referred to as *optimization-based bound tightening* (OBBT). Given a linear relaxation $\mathcal{R} \supseteq \mathcal{X}$, OBBT computes the tightest bounds valid for all relaxation solutions by in turn minimizing and maximizing each variable over $\mathcal{R}$,

$$\min / \max\{\, x_k : x \in \mathcal{R}, x \in [\ell, u]\,\}. \tag{2}$$

Its first appearance in the literature we are aware of is an application to heat exchanger networks by Quesada and Grossmann [17] from 1993. Subsequently, it became a component of generic global optimization algorithms, see, e.g., [18, 14, 19].

An optimization algorithm may exclude suboptimal parts of the feasible region as long as at least one optimal solution remains. In OBBT, this can be exploited by adding an *objective cutoff* constraint $c^{\mathsf{T}}x \leqslant z$ to $\mathcal{R}$, where $z = c^{\mathsf{T}}\hat{x}$ is the objective value of the current incumbent solution $\hat{x}$. Zamora and Grossmann [24] have used this idea in a "branch-and-contract" algorithm, which employs OBBT aggressively at every node of the search tree.

Examples of MINLP solvers implementing OBBT are αBB [2, 3], Couenne [4, 10], GloMIQO [15, 12], LaGO [16, 11], and SCIP [1, 22, 25]. Since applying a full round of OBBT amounts to solving $2n$ linear programs—an expensive algorithm compared to the average amount of work performed at a branch-and-bound node—it is typically applied at the root node and within the search tree only with limited frequency or based on its success rate. For a recent theoretical study of an iterated version of OBBT see the paper by Caprara and Locatelli [7].

Contribution. Our paper presents a new idea for how to benefit from the potentially expensive solution of (2) beyond simply obtaining tighter bounds on variable x_k. To this end, we observe that the proof of optimality given by a dual solution of (2) can be used to learn globally valid inequalities whose propagation gives a local approximation of OBBT. These inequalities, which we call *Lagrangian variable bound constraints*, are redundant since they are obtained merely as an aggregation of the rows of the relaxation $\mathcal{R}$. Nevertheless, we demonstrate that propagating them during the tree search helps to speed up the solution process significantly.

In the remainder of the paper, Sec. 2 explains the derivation and propagation of Lagrangian variable bounds in detail. Section 3 presents computational results analyzing their effect on instances from MINLPLib and summarizes our findings.

2 Lagrangian variable bounds

Besides valid bounds for variable x_k, solving (2) yields dual multipliers for the constraints of $\mathcal{R}$ that prove that for no $x \in \mathcal{R}$—and by that for no feasible solution of (1)—variable x_k can lie outside these bounds. The following lemma uses basic LP duality to motivate our approach. For clarity, we restrict the presentation to upper bounds.

Lemma 1. *Let $\mathcal{R} = \{x \in \mathbb{R}^n : a_i^\mathsf{T} x \leqslant b_i,\, i = 1, \dots, m\} \supseteq \mathcal{X}$ be given, where $a_i \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. Let x^* be an optimal solution of*

$$\max\{\, x_k : x \in \mathcal{R},\, c^\mathsf{T} x \leqslant z,\, x \in [\ell, u] \,\}, \tag{3}$$

with $z \in \mathbb{R} \cup \{\infty\}$ an upper bound on the optimal objective value of (1)*. Further, let $\lambda_1, \dots, \lambda_m, \mu \geqslant 0$ be feasible dual multipliers with reduced costs*

$$r_j := \begin{cases} 1 - \sum_i \lambda_i a_{ij} - \mu c_j & \text{if } j = k, \\ \quad - \sum_i \lambda_i a_{ij} - \mu c_j & \text{otherwise.} \end{cases} \tag{4}$$

Then

$$U(\ell, u, z) := \sum_{j : r_j < 0} r_j \ell_j + \sum_{j : r_j > 0} r_j u_j + \mu z + \lambda^\mathsf{T} b \tag{5}$$

is a valid upper bound for x_k. If $\lambda_1, \dots, \lambda_m, \mu$ are optimal multipliers then $U(\ell, u, z) = x_k^*$, otherwise $U(\ell, u, z) > x_k^*$.

Proof. Multiplying the rows of (3) with their dual values and aggregating them gives the valid inequality $(\sum_i \lambda_i a_i + \mu c)^\mathsf{T} x \leqslant \lambda^\mathsf{T} b + \mu z$. Using (4), this becomes

$$x_k \leqslant \textstyle\sum_j r_j x_j + \mu z + \lambda^\mathsf{T} b, \tag{6}$$

which for $x \in [\ell, u]$ is at most $U(\ell, u, z)$. Optimal multipliers are complementary slack with x^*, yielding the relation of $U(\ell, u, z)$ and the OBBT bound x_k^*.

We will refer to bounds of type (5) as well as their lower bound counterparts as *Lagrangian variable bounds* (LVBs). Figure 1 provides an illustrative example, which shows that LVBs can be learnt even when OBBT fails to tighten the bound.

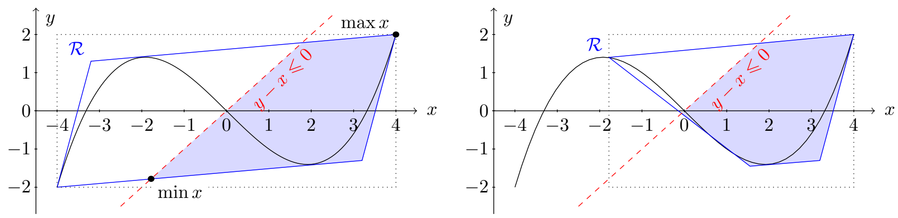

Figure 1: Example $\min\{y - x : y = 0.1x^3 - 1.1x, x \in [-4, 4], y \in [-2, 2]\}$. On the left, the shaded region over which OBBT is performed is defined by the relaxation $\mathcal{R}$ and the dashed objective cutoff resulting from the zero solution. Minimizing x gives a lower bound of $-\frac{16}{9}$ and the LVB $x \geqslant -\frac{10}{9}z - \frac{16}{9}$. Maximizing x does not tighten its upper bound, still the LVB $x \leqslant \frac{10}{37}y + \frac{128}{37}$ can be learnt. In this two variable example, this is only the rightmost facet of $\mathcal{R}$, but in higher dimensions it may be nontrivial. On the right is the resulting, tighter relaxation.

Remark 1. If μ is nonzero then $U(\ell, u, z)$ depends on the primal bound; if some r_j, $j \neq k$, is nonzero, it depends on x_j. Hence, whenever an improving solution is found or $[\ell_j, u_j]$ is reduced, the LVB may tighten the bounds of x_k further.

Additionally, in stark contrast to OBBT, LVBs can be propagated very efficiently. This motivates the application of LVBs within a spatial branch-and-bound algorithm for nonconvex MINLP in the following scheme.

1. Learn LVB constraints while performing OBBT once during the root node.
2. Propagate them locally at the nodes of the search tree whenever bounds appearing on the right-hand side are reduced by branching or propagation.
3. Propagate them globally whenever an improving solution is found.

Already in [21] it has been observed that any dual feasible solution encountered during the solution process may be used to construct a one-row relaxation of the LP at hand and that this inequality can be used to tighten the bounds of each variable involved. Applied unconditionally, however, this idea appears too expensive. In this paper, we suggest to specifically select the dual solutions from OBBT and propagate LVBs only towards the left-hand side variable.

Remark 2. The main purpose of the LVB constraints (6) is to identify bounds already implied by the relaxation and, by making them explicit, allow for improving the relaxations of nonconvex nonlinear constraints. Note that, unlike MIP cutting planes, they are not designed to cut off the LP optimum. Since they are redundant inequalities, it is not beneficial to add them to the LP relaxation.

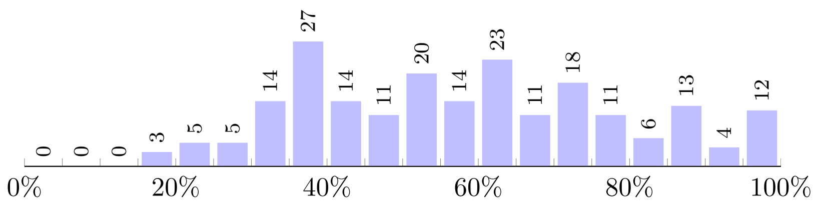

Figure 2: Rate of generated LVBs per OBBT LP as distributed over 211 instances from MINLPLib for which OBBT was applied at the root node.

3 Computational results

Experimental setup. The aim of our experiments was two-fold: first, to quantify how many *nontrivial LVBs* can be generated during OBBT, i.e., LVBs with $\mu \neq 0$ or $r_j \neq 0$ for some $j \neq k$; second, to evaluate the effect of propagating them during the solution process. Within the MINLP solver SCIP 3.0 [1, 5, 22, 25] we have implemented an OBBT scheme that minimizes and maximizes each variable once subject to the LP relaxation after the first separation loop. We consider only nonbinary variables that appear in nonlinear constraints. By slightly relaxing the bounds on the variable that is currently minimized or maximized, we increase the chance to generate nontrivial LVBs when the bound is not tightened by OBBT. The generated LVB constraints are stored and propagated efficiently in a suitable topological order whenever their right-hand side improves.

As a test set, we used MINLPLib [6]. We excluded 18 instances which cannot be parsed or handled by SCIP 3.0.[1] Further 41 instances were linear after presolving or solved at the root node before OBBT was applied.[2] After removing `gear4` and `nvs22`, for which SCIP 3.0 returned a wrong solution value, we were left with 211 instances. The experiments were conducted on a cluster of 64bit Intel Xeon X5672 CPUs with 3.2 GHz, 12 MB cache, and 48 GB main memory. SCIP used CPLEX 12.4 [13] as LP solver, CppAD 20120101.3 [8], and Ipopt 3.10.2 [23, 9].

Results. First, we measured the percentage of OBBT LPs solved that lead to a nontrivial LVB. The histogram in Fig. 2 shows the distribution of this success rate over the test set. For all instances, LVBs were generated from at least 15% of the OBBT LPs. For 132 out of 211 instances, the rate was above 50%.

Second, we compared SCIP with OBBT only and SCIP with OBBT and LVB propagation in a performance run with a time limit of one hour. To reduce distorting side effects from heuristic components of the tree search we

[1] `blendgap`, `deb{6,7,8,9,10}`, `dosemin{2,3}d`, `prob10`, `var_con{5,10}`, `water{3,ful2,s,sbp,sym1,sym2}`, and `windfac`.

[2] `ex{1221,1222,1223a,1225}`, `feedtray2`, `gbd`, `hmittelman`, `lop97ic`, `lop97icx`, `mbtd`, `nvs{03,07,10}`, `pb*`, `prob{02,03}`, `qap`, `qapw`, `st_e{13,15,27}`, `st_miqp{1,2,3,4,5}`, `st_test{1,2,3,4,5,6,8}`, and `tln2`.

deactivated primal heuristics in the tree, turned off conflict analysis, and used a simple first index branching rule with depth first node selection.

In this setting, two more instances could be solved with LVB propagation, while one instance solved before then hit the time limit of one hour. On 94 instances both solvers timed out; 109 instances were solved by both. Disregarding ten easy instances that were solved at the root by both variants, on the remaining 99 instances LVB propagation reduced the shifted geometric mean[3] of the number of branch-and-bound nodes by 14% and the solving time by 7%. Detailed results are shown in Tab. 1.

For validation, we performed a control experiment using SCIP's default parameters as base setting. Here, for the instances solved by both solvers the number of nodes was reduced by 12% and the total solving time by 6%. Note that for a single propagation algorithm the achieved savings are substantial, in particular when considering its low computational overhead. Except for two easy instances, LVB propagation never took more than 2% of the total running time.

The fact that the solving time was reduced by less than the tree size is mostly explained by the longer processing time of the root. This general phenomenon is intensified by our experimental setup, since we applied a full round of OBBT without controlling the effort spent, e.g., by limiting LP iterations.

Conclusion. In this paper, we have introduced the notion of Lagrangian variable bound constraints, which are linear inequalities that can be learnt during OBBT and exploited during a spatial branch-and-bound algorithm. They can be propagated efficiently and give an approximation of reapplying OBBT where it may be overly expensive. Our experiments showed that on affected instances from MINLPLib this reduces the average number of branch-and-bound nodes by more than 10% and speeds up the solution process.

Future research should investigate whether LVB success correlates, for instance, with the tightness of the generating OBBT LP and how LVB propagation behaves in combination with a more sophisticated OBBT implementation.

Acknowledgments. The authors want to thank Timo Berthold, Pietro Belotti, and Domenico Salvagnin for the fruitful discussions and valuable comments on this paper.

---

[3]The shifted geometric mean of values $x_1, \ldots, x_n \geqslant 0$ with shift $s > 0$ is defined as $\left(\prod_i (x_i + s)\right)^{1/n} - s$. We use a shift of five seconds and 100 nodes, respectively. This reduces the bias from outliers with large values as well as from very easy instances.

Table 1: SCIP with OBBT only vs. SCIP with OBBT and LVB propagation. Columns "diff" state the relative difference in percent. The time for LVB propagation in column "LVB time" is included in the solving time.

	nodes			solving time			
instance	LVB off	LVB on	diff [%]	LVB off	LVB on	LVB time	diff [%]
alan	3	3	0.0	0.10	0.10	0.00	0.0
csched1	98380	19565	−80.1	24.40	4.84	0.05	−80.2
du-opt	1349	1349	0.0	1.86	1.69	0.00	−9.1
du-opt5	141	141	0.0	0.49	0.49	0.01	0.0
elf	843	843	0.0	0.62	0.63	0.01	+1.6
eniplac	839	761	−9.3	1.25	1.18	0.01	−5.6
enpro48	133	133	0.0	0.79	0.99	0.00	+25.3
enpro48pb	127	115	−9.4	0.86	0.79	0.00	−8.1
enpro56	383	4831	+1161.4	1.29	2.45	0.00	+89.9
enpro56pb	360	360	0.0	1.21	1.22	0.00	+0.8
ex1223	5	5	0.0	0.10	0.10	0.00	0.0
ex1223b	5	5	0.0	0.10	0.10	0.00	0.0
ex1224	15	15	0.0	0.10	0.10	0.00	0.0
ex1226	3	3	0.0	0.10	0.10	0.00	0.0
ex1243	252	238	−5.6	0.83	0.93	0.01	+12.0
ex1244	50141	442	−99.1	37.31	1.20	0.00	−96.8
ex1252a	286870	1742660	+507.5	532.45	2027.65	4.65	+280.8
ex1263	5464	3677	−32.7	1.53	1.22	0.01	−20.3
ex1263a	223	149	−33.2	0.10	0.10	0.00	0.0
ex1264	24736	24550	−0.8	4.63	4.63	0.07	0.0
ex1264a	596	252	−57.7	0.18	0.10	0.00	−44.4
ex1265	1121	3357	+199.5	0.56	1.07	0.01	+91.1
ex1265a	238	69	−71.0	0.10	0.10	0.00	0.0
ex1266	149	66	−55.7	0.26	0.25	0.00	−3.8
ex1266a	11	11	0.0	0.10	0.10	0.00	0.0
ex3	3	3	0.0	0.16	0.15	0.00	−6.2
ex3pb	3	3	0.0	0.14	0.15	0.00	+7.1
ex4	31	31	0.0	0.54	0.59	0.00	+9.3
fac1	5	5	0.0	0.10	0.10	0.00	0.0
fac3	19	19	0.0	0.32	0.32	0.00	0.0
fo7	2339728	2334886	−0.2	1102.54	1098.52	9.51	−0.4
fo7_2	717141	717303	0.0	313.64	314.38	2.10	+0.2
fo7_ar2_1	488871	489807	+0.2	179.97	181.46	1.52	+0.8
fo7_ar3_1	1074346	1072910	−0.1	496.14	500.05	3.57	+0.8
fo7_ar4_1	2063037	1494929	−27.5	866.25	738.24	7.31	−14.8
fo7_ar5_1	518897	518065	−0.2	238.83	238.33	1.18	−0.2
fuel	3	3	0.0	0.10	0.10	0.00	0.0
gastrans	9	9	0.0	0.21	0.12	0.00	−42.9
gear	1772	2125	+19.9	0.55	0.65	0.00	+18.2
gear2	1717	1089	−36.6	0.68	0.47	0.00	−30.9
gear3	1772	2125	+19.9	0.51	0.46	0.00	−9.8
gkocis	3	3	0.0	0.10	0.10	0.00	0.0
m3	43	43	0.0	0.10	0.12	0.00	+20.0
m6	119341	119331	−0.0	40.26	40.25	0.36	−0.0
m7	1337679	1337511	−0.0	486.80	490.68	2.84	+0.8
m7_ar25_1	8387	8141	−2.9	3.20	3.09	0.00	−3.4
m7_ar2_1	21003	20541	−2.2	6.22	5.68	0.12	−8.7
m7_ar3_1	98854	98031	−0.8	40.62	40.11	0.26	−1.3
m7_ar4_1	163529	163459	−0.0	58.03	58.40	0.39	+0.6
m7_ar5_1	387403	387425	0.0	159.25	161.40	0.77	+1.4
meanvarx	7	7	0.0	0.13	0.13	0.00	0.0

continued on next page

continued from previous page

instance	nodes			solving time			
	LVB off	LVB on	diff [%]	LVB off	LVB on	LVB time	diff [%]
meanvarxsc	7	7	0.0	0.14	0.14	0.00	0.0
netmod_dol1	115879	115733	−0.1	2906.79	2912.69	0.56	+0.2
netmod_dol2	583	583	0.0	28.46	28.40	0.01	−0.2
netmod_kar1	483	487	+0.8	1.77	1.82	0.00	+2.8
netmod_kar2	483	487	+0.8	1.65	1.65	0.00	0.0
no7_ar25_1	1071694	1071018	−0.1	564.64	570.99	5.60	+1.1
no7_ar3_1	2568064	2565390	−0.1	1305.33	1299.00	10.14	−0.5
nvs01	38	40	+5.3	0.10	0.10	0.00	0.0
nvs04	3	3	0.0	0.10	0.10	0.00	0.0
nvs06	21	21	0.0	0.10	0.11	0.00	+10.0
nvs11	5	5	0.0	0.10	0.10	0.00	0.0
nvs12	9	9	0.0	0.10	0.10	0.00	0.0
nvs13	29	29	0.0	0.10	0.10	0.00	0.0
nvs15	7	7	0.0	0.10	0.10	0.00	0.0
nvs16	5	5	0.0	0.10	0.10	0.00	0.0
nvs17	57	57	0.0	0.10	0.10	0.00	0.0
nvs18	39	37	−5.1	0.10	0.10	0.00	0.0
nvs19	105	107	+1.9	0.15	0.14	0.00	−6.7
nvs20	514107	255511	−50.3	165.62	89.98	1.43	−45.7
nvs21	37	37	0.0	0.10	0.10	0.00	0.0
nvs23	185	183	−1.1	0.26	0.27	0.00	+3.8
nvs24	367	367	0.0	0.46	0.47	0.00	+2.2
ortez	172258	81	−100.0	23.67	0.52	0.00	−97.8
pump	2079411	2117797	+1.8	2461.28	2397.94	6.04	−2.6
ravem	141	141	0.0	1.02	1.09	0.00	+6.9
ravempb	19	19	0.0	0.75	0.73	0.00	−2.7
risk2b	15624	3672	−76.5	3.32	1.43	0.00	−56.9
sep1	56485	45691	−19.1	8.55	7.53	0.14	−11.9
spectra2	39	39	0.0	0.90	0.69	0.00	−23.3
st_e14	5	5	0.0	0.10	0.10	0.00	0.0
st_e29	15	15	0.0	0.10	0.10	0.00	0.0
st_e31	863	863	0.0	0.84	0.90	0.01	+7.1
st_e36	139	139	0.0	0.27	0.19	0.01	−29.6
st_e38	3	3	0.0	0.12	0.12	0.00	0.0
st_e40	37	35	−5.4	0.10	0.10	0.00	0.0
st_testgr1	31	29	−6.5	0.10	0.10	0.00	0.0
st_testgr3	19	5	−73.7	0.10	0.10	0.00	0.0
synthes1	3	3	0.0	0.10	0.10	0.00	0.0
synthes2	3	3	0.0	0.10	0.10	0.00	0.0
synthes3	7	7	0.0	0.10	0.10	0.00	0.0
tln4	636	628	−1.3	0.46	0.46	0.00	0.0
tln5	10656	9333	−12.4	5.21	4.60	0.04	−11.7
tln6	35889	35612	−0.8	35.25	35.28	0.11	+0.1
tloss	11	11	0.0	0.10	0.10	0.00	0.0
tls2	27	27	0.0	0.10	0.10	0.00	0.0
tls4	2673344	2441601	−8.7	1578.22	1498.11	10.14	−5.1
tltr	16	16	0.0	0.10	0.10	0.00	0.0
util	507	265	−47.7	0.26	0.13	0.00	−50.0
shift. geo. mean	1423	1222	−14.1	8.27	7.69	0.47	−7.0

References

[1] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, TU Berlin, 2007. http://vs24.kobv.de/opus4-zib/frontdoor/index/index/docId/1018.

[2] Claire S. Adjiman, Ioannis P. Androulakis, and Christodoulos A. Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs—II. Implementation and computational results. *Computers & Chemical Engineering*, 22(9):1159–1179, 1998. doi:10.1016/S0098-1354(98)00218-X.

[3] Claire S. Adjiman, Ioannis P. Androulakis, and Christodoulos A. Floudas. Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 46(9):1769–1797, 2000. doi:10.1002/aic.690460908.

[4] Pietro Belotti, Jon Lee, Leo Liberti, Francois Margot, and Andreas Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods & Software*, 24:597–634, 2009. doi:10.1080/10556780903087124.

[5] Timo Berthold, Stefan Heinz, and Stefan Vigerske. Extending a CIP framework to solve MIQCPs. In Jon Lee and Sven Leyffer, editors, *Mixed-integer nonlinear optimization*, volume 154 of *The IMA volumes in Mathematics and its Applications*, pages 427–444. Springer, 2012. doi:10.1007/978-1-4614-1927-3_15.

[6] M.R. Bussieck, A.S. Drud, and A. Meeraus. MINLPLib – a collection of test models for mixed-integer nonlinear programming. *INFORMS J. on Comput.*, 15(1):114–119, 2003.

[7] Alberto Caprara and Marco Locatelli. Global optimization problems and domain reduction strategies. *Mathematical Programming*, 125:123–137, 2010. doi:10.1007/s10107-008-0263-4.

[8] COIN-OR. CppAD. A Package for Differentiation of C++ Algorithms. http://www.coin-or.org/CppAD.

[9] COIN-OR. Ipopt. Interior point optimizer. http://www.coin-or.org/Ipopt.

[10] COIN-OR. Couenne. Convex Over and Under ENvelopes for Nonlinear Estimation. http://www.coin-or.org/Couenne.

[11] COIN-OR. LaGO. Lagrangian Global Optimizer. http://www.coin-or.org/LaGO.

[12] Computer-Aided Systems Laboratory, Princeton University. GloMIQO. Global Mixed-Integer Quadratic Optimizer. http://helios.princeton.edu/GloMIQO.

[13] IBM. ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[14] Costas D. Maranas and Christodoulos A. Floudas. Global optimization in generalized geometric programming. *Computers & Chemical Engineering*, 21(4):351–369, 1997. doi:10.1016/S0098-1354(96)00282-7.

[15] Ruth Misener and Christodoulos A. Floudas. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, pages 1–48, 2012. doi:10.1007/s10898-012-9874-7.

[16] Ivo Nowak and Stefan Vigerske. LaGO: a (heuristic) branch and cut algorithm for nonconvex MINLPs. *Central European Journal of Operations Research*, 16(2):127–138, 2008. doi:10.1007/s10100-007-0051-x.

[17] Ignacio Quesada and Ignacio E. Grossmann. Global optimization algorithm for heat exchanger networks. *Industrial & Engineering Chemistry Research*, 32(3):487–499, 1993. doi:10.1021/ie00015a012.

[18] Ignacio Quesada and Ignacio E. Grossmann. A global optimization algorithm for linear fractional and bilinear programs. *Journal of Global Optimization*, 6:39–76, 1995. doi:10.1007/BF01106605.

[19] Edward M.B. Smith and Constantinos C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478, 1999. doi:10.1016/S0098-1354(98)00286-5.

[20] Mohit Tawarmalani and Nikolaos V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Dordrecht Boston London, 2002.

[21] Mohit Tawarmalani and Nikolaos V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004. doi:10.1007/s10107-003-0467-6.

[22] Stefan Vigerske. *Decomposition in Multistage Stochastic Programming and a Constraint Integer Programming Approach to MINLP*. PhD thesis, HU Berlin, 2012.

[23] Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Prog.*, 106(1):25–57, 2006.

[24] Juan M. Zamora and Ignacio E. Grossmann. A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. *Journal of Global Optimization*, 14:217–249, 1999. doi:10.1023/A:1008312714792.

[25] Zuse Institute Berlin, Department of Optimization. SCIP. Solving Constraint Integer Programs. http://scip.zib.de.