Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

TIMO BERTHOLD[1]
AMBROS M. GLEIXNER

UNDERCOVER – a primal heuristic for MINLP based on sub-MIPs generated by set covering

Zuse Institute Berlin, Department of Optimization, Takustr. 7, 14195 Berlin, Germany, {berthold,gleixner}@zib.de

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

ZIB-Report 09-40 (December 2009)

Undercover – a primal heuristic for MINLP based on sub-MIPs generated by set covering

Timo Berthold
Ambros M. Gleixner

Abstract. We present Undercover, a primal heuristic for mixed-integer nonlinear programming (MINLP). The heuristic constructs a mixed-integer *linear* subproblem (sub-MIP) of a given MINLP by fixing a subset of the variables. We solve a set covering problem to identify a minimal set of variables which need to be fixed in order to linearise each constraint. Subsequently, these variables are fixed to approximate values, e.g. obtained from a linear outer approximation. The resulting sub-MIP is solved by a mixed-integer linear programming solver. Each feasible solution of the sub-MIP corresponds to a feasible solution of the original problem.

Although general in nature, the heuristic seems most promising for mixed-integer quadratically constrained programmes (MIQCPs). We present computational results on a general test set of MIQCPs selected from the MINLPLib [12].

1. Introduction

For mixed-integer programming (MIP) it is well-known that, apart from complete solving methods, general-purpose primal heuristics like the feasibility pump [2, 14, 16] are able to find high-quality solutions for a wider range of problems. Over the years, primal heuristics have become a substantial ingredient of state-of-the-art MIP solvers [5, 8]. For mixed-integer nonlinear programming (MINLP) there have only been a few publications on general-purpose primal heuristics [7, 10, 20, 11].

At the heart of many recently proposed primal MIP heuristics, such as Local Branching [15], RINS [13], DINS [17], and RENS [6], lies large neighbourhood search, hence the paradigm of solving a small sub-MIP which promises to contain good solutions. In this paper, we introduce *Undercover* which is a large neighbourhood search start heuristic that constructs and solves a sub-MIP of a given MINLP. We demonstrate its effectiveness on a general test set of mixed-integer quadratically constrained programs (MIQCPs) taken from the MINLPLib [12].

An MINLP is an optimization problem of the form

$$\begin{aligned} \min \quad & d^T x \\ \text{s.t.} \quad & g_i(x) \leqslant 0 && \text{for } i = 1, \dots, m, \\ & L_k \leqslant x_k \leqslant U_k && \text{for } k = 1, \dots, n, \\ & x_k \in \mathbb{Z} && \text{for } k \in \mathcal{I}, \end{aligned} \tag{1.1}$$

where $\mathcal{I} \subseteq \{1, \dots, n\}$ is the index set of the integer variables, $d \in \mathbb{R}^n$, $g_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots, m$, and $L \in (\mathbb{R} \cup \{-\infty\})^k$, $U \in (\mathbb{R} \cup \{+\infty\})^k$ are lower and upper bounds on the variables, respectively. Note that a nonlinear objective function can always be reformulated by introducing one additional constraint, hence form (1.1) is general.

2. A generic algorithm

Given a mixed-integer programme, the paradigm of fixing a subset of its variables to obtain subproblems which are easier to solve has proven successful in many primal MIP heuristics such as RINS [13], DINS [17], and RENS [6]. The core difficulty in MIP solving are the integrality constraints, thus in MIP context “easy to solve” usually takes the meaning of few integer variables. While in MINLP integralities do contribute to the complexity of the problem, a specific difficulty are the nonlinearities. Hence, “easy” in MINLP can be understood as few nonlinear constraints.

This research was partially funded by the DFG Research Center Matheon, Project B20. We thank GAMS Development Corp. for providing us with evaluation licenses for BARON and Couenne. Thanks to Stefan Vigerske for his valuable comments and to Mark Wallace for his proofreading.
Keywords: MINLP, MIQCP, primal heuristic, large neighborhood search, set covering.

Our heuristic is based on the simple observation that by fixing certain variables (to some value within their bounds) any given mixed-integer *nonlinear* programme can be reduced to a mixed-integer *linear* subproblem (sub-MIP). Every feasible solution of this sub-MIP is then a feasible solution of the original MINLP.

The approach is motivated by our experience that for several practically relevant MIQCPs, fixing only a comparatively small subset of the variables already suffices to linearise the problem. The computational effort, however, is usually greatly reduced since we can apply the full strength of state-of-the-art MIP solving to the subproblem. Before formulating a first generic algorithm for our heuristic, consider the following definitions.

Definition 1 (cover of a function). *Let a function $g : D \to \mathbb{R}$, $x \mapsto g(x)$ on a domain $D \subseteq \mathbb{R}^n$ and a point $x^\star \in D$ be given. We call a set $\mathcal{C} \subseteq \{1, \dots, n\}$ of variable indices an $x^\star$-*cover* of g if and only if the set*

$$\{(x, g(x)) \mid x \in D, x_k = x_k^\star \text{ for all } k \in \mathcal{C}\} \tag{2.1}$$

is affine. We call $\mathcal{C}$ a (global) cover *of g if and only if $\mathcal{C}$ is an $x^\star$-cover of g for all $x^\star \in D$.*

Definition 2 (cover of an MINLP). *Let P be an MINLP of form* (1.1)*, let $x^\star \in [L, U]$, and $\mathcal{C} \subseteq \{1, \dots, n\}$ be a set of variable indices of P. We call $\mathcal{C}$ an $x^\star$-*cover* of P if and only if $\mathcal{C}$ is an $x^\star$-*cover* for $g_1, \dots, g_m$. We call $\mathcal{C}$ a* (global) cover *of P if and only if $\mathcal{C}$ is an $x^\star$-*cover* for all $x^\star \in [L, U]$.*

A first generic algorithm for our heuristic is given in Figure 1. The hinge of the algorithm is clearly found in Line 5: finding a suitable cover of the given MINLP. Section 3 elaborates on this in detail with special emphasis on the case of MIQCPs.

Figure 1: Generic algorithm

1 **Input**: MINLP P as in (1.1)
2 **begin**
3 compute a solution $x^\star$ of an approximation or relaxation of P
4 round $x_k^\star$ for $k \in \mathcal{I}$
5 determine an $x^\star$-cover $\mathcal{C}$ of P
6 solve the sub-MIP of P given by $x_k = x_k^\star$, $k \in \mathcal{C}$
7 **end**

To obtain suitable fixing values for the selected variables, an approximation or relaxation of the original MINLP is used. For integer variables the approximate values are rounded. Most exact solvers for MINLP are based on branch-and-bound. If the heuristic is embedded within a branch-and-bound solver, using its (linear or nonlinear) relaxation appears as a natural choice for obtaining approximate variable values.

Large neighbourhood search heuristics which rely on fixing variables typically have to trade off between eliminating many variables in order to make the sub-MIP tractable versus leaving enough degrees of freedom such that the sub-MIP is still feasible and contains good solutions. Often their implementation inside a MIP solver demands a sufficiently large percentage of variables to be fixed to arrive at an easy to solve sub-MIP [5, 6, 13, 17].

For our heuristic, the situation is different since we do not aim at eliminating integrality constraints, but nonlinearities. In order to linearise a given MINLP, in general we may be forced to fix integer and continuous variables. Especially the fixation of continuous variables in an MINLP can introduce a significant error, even rendering the subproblem infeasible. Thus our heuristic will aim at fixing as *few* variables as possible to obtain as large a linear subproblem as possible.

Remark 1. Note that in general a minimum cover does not necessarily yield a dimension-wise largest sub-MIP that can be obtained by fixing variables in a given MINLP. First, this is because in our definition we do not look at the feasible region given by a constraint, but at the graph of its left hand side. Second, we do not take into account the interrelation of constraints with each other and with variable bounds and integrality constraints. Through these interrelations, fixing one variable may lead to further domain reductions and variable fixations which can not immediately be foreseen when looking at each nonlinearity separately.

Hence, searching for a minimum cover may be understood as an approximate method for determining dimension-wise maximal sub-MIPs. Propagation routines using these interrelations can, however, be very effectively integrated within the heuristic, see "fix-and-propagate" in Section 4.

3. **Finding minimum covers**

This section describes our method for determining a minimum cover of an MINLP, i.e. a minimal subset of variables to fix in order to linearise each constraint. The idea for Undercover originated from our work on solving MIQCPs. Since its application also appears most promising for this class of problems, we start by presenting conditions for covers of quadratic constraints.

Covering quadratic functions. Suppose we are given a quadratic function $g : \mathbb{R}^n \to \mathbb{R}, x \mapsto x^T Q x$, with $Q \in \mathbb{R}^{n\times n}$ symmetric. Let $x^\star \in \mathbb{R}^n$ and $\mathcal{C} \subseteq \{1,\dots,n\}$. Fixing $x_k = x^\star_k$ for all $k \in \mathcal{C}$ transforms $x^T Q x$ into $y^T \tilde{Q} y + \tilde{q}^T y + \tilde{c}$ with variable vector $y = (x_k)_{k\notin\mathcal{C}} \in \mathbb{R}^{n-|\mathcal{C}|}$, the restricted matrix $\tilde{Q} = (Q_{uv})_{u,v\notin\mathcal{C}}$ of dimension $(n-|\mathcal{C}|)\times(n-|\mathcal{C}|)$, the vector $\tilde{q} = (2\sum_{u\in\mathcal{C}} Q_{uk} x^\star_u)_{k\notin\mathcal{C}} \in \mathbb{R}^{n-|\mathcal{C}|}$, and offset $\tilde{c} = \sum_{u,v\in\mathcal{C}} Q_{uv} x^\star_u x^\star_v$. Thus, the set (2.1) is affine if and only if $\tilde{Q}$ vanishes, i.e. if $q_{uv} = 0$ for all $u, v \notin \mathcal{C}$. In reverse, this means that for $\mathcal{C}$ to be a cover of g, it needs to contain at least one out of u or v for all nonzero matrix entries Q_{uv}.

This can be interpreted as a *set covering problem*, where items are given by those $(u, v) \in \{1,\dots,n\}\times\{1,\dots,n\}$ with nonzero Q_{uv}, and sets are given by $S(k) := \{(u, v) \mid u = k \text{ or } v = k\}$ for each variable index $k = 1,\dots,n$.

Remark 2. Note that in the quadratic case, any $x^\star$-cover is already a global cover, thus the distinction made in Definitions 1 and 2 is void.

Covering MIQCPs. An MIQCP is an MINLP as in (1.1) where each constraint $i = 1,\dots,m$ takes the form $g_i(x) = x^T A_i x + {b_i}^T x + c_i \leqslant 0$ with $A_i \in \mathbb{R}^{n\times n}$ symmetric, $b_i \in \mathbb{R}^n$, and $c_i \in \mathbb{R}$. Matrices A_i are not required to be positive semidefinite, i.e. we allow for nonconvex constraints.

In order to find a cover of a given MIQCP P, we solve the set covering problem outlined above, taking into account all constraints. We introduce auxiliary binary variables α_k, $k = 1,\dots,n$, equal to 1 if and only if x_k is fixed in P. As explained above, $\mathcal{C}(\alpha) := \{k \mid \alpha_k = 1\}$ is a cover of P if and only if

$$\alpha_k = 1 \qquad \text{for all } i \in \{1,\dots,m\}, k \in \{1,\dots,n\}, A^i_{kk} \neq 0, L_k \neq U_k, \tag{3.1}$$

$$\alpha_k + \alpha_j \geqslant 1 \qquad \text{for all } i \in \{1,\dots,m\}, k \neq j \in \{1,\dots,n\}, A^i_{kj} \neq 0, L_k \neq U_k, L_j \neq U_j, \tag{3.2}$$

i.e. we require all square terms and one variable in each bilinear term to be fixed. Our heuristic tries to identify as large a linear subproblem as possible. Therefore, we solve the binary programme

$$\min\Big\{\sum_{k=1}^{n} \alpha_k : (3.1), (3.2), \alpha \in \{0,1\}^n\Big\}. \tag{3.3}$$

The following lemma summarises our discussion from above:

Lemma 1. *Let an MIQCP P be given. Then $\alpha \mapsto \mathcal{C}(\alpha) = \{k \mid \alpha_k = 1\}$ gives a one-to-one correspondence between the feasible solutions of* (3.3) *and covers of P. A solution $\alpha^\star$ of* (3.3) *is optimal if and only if $\mathcal{C}(\alpha^\star)$ is a minimum cardinality cover of P.*

Remark 3. Note that the above covering problem is an optimisation version of 2-SAT, hence there is a polynomial-time algorithm for its solution. In our computational experiments the binary programme (3.3) could always be solved within a fraction of a second by a general MIP solver, hence we did not employ a specialised polynomial-time algorithm. Moreover, in the case of general MINLP, a cover generating problem contains more involved constraints.

Covering general MINLPs. Our approach for computing covers of quadratic constraints can be applied to more general nonlinearities. The immediate generalisations are multilinear and polynomial constraints. Sufficient conditions for a global cover of a monomial $x_1^{p_1} \cdots x_n^{p_n}$, $p_1, \ldots, p_n \in \mathbb{N}_0$, are $\alpha_k = 1$ for all $k \in \{1, \ldots, n\}, p_k \geqslant 2, L_k \neq U_k$, and $\sum_{k:p_k=1, L_k \neq U_k} (1-\alpha_k) \leqslant 1$, similar to (3.1) and (3.2).

As can be seen from this example, with more and more general nonlinearities present, more and more variables need to be fixed to arrive at a linear subproblem. However, note that now the notion of an $x^\star$-cover may be much weaker than that of a global cover.[1]

4. Variants and extensions

The generic algorithm in Figure 1 can be extended and modified in several ways in order to make Undercover more efficient in practise. This section outlines a few of them, with main focus on avoiding infeasibility of the sub-MIPs.

Fix-and-propagate. Fixing a variable can have great impact on the original problem and the approximation we use. Therefore, we do not fix the variables simultaneously, but sequentially one by one, propagating the bound changes after each fixing. If by that, the approximation solution falls outside the feasible domain of a variable, we instead fix it to the closest bound.[2] This fix-and-propagate method resembles a method described in [16]. Additionally, we apply it for continuous variables and apply backtracking in case of infeasibility.

Backtracking. If the fix-and-propagate procedure deduces some variable domain to be empty, hence the subproblem to be infeasible, we apply a one-level backtracking, i.e. we undo the last bound change and try other fixing values instead.[3]

Recovering. During the fix-and-propagate routine, variables outside the precomputed cover may also be fixed. In this case, the fixing of some of the yet unfixed variables in the cover might become redundant and recomputing the cover may yield a smaller number of variable fixings still necessary.

Different covers. Our initial motivation for fixing as few variables as possible was to minimise the impact on the orginal MINLP. Other measures for the impact of fixing a variable could be used in the objective function of (3.3), such as domain size, appearance in nonlinear terms or nonlinear constraints violated by the approximation solution, variable type, or hybrid measures. In particular, if a minimum cardinality cover yields an infeasible sub-MIP, we may try a cover minimising a different impact measure.

NLP postprocessing. In the spirit of the QCP local search heuristic described in [7], we try to further improve the best sub-MIP solution $\tilde{x}$ by solving the NLP which results from fixing all integer variables to their values in $\tilde{x}$.

Convexification. The main idea of Undercover is to reduce the computational effort by finding easier to solve subproblems. While here we have focused on sub-MIPs, for nonconvex MINLPs, already a convex sub-MINLP may be significantly easier to solve and contain more and better solutions than a sub-MIP. This modification of Undercover merely requires to weaken the constraints in the cover generating problem (3.3) suitably.

Domain reduction. Instead of fixing the variables in a cover, we could also only reduce their domains. Especially on continuous variables this leaves significantly more freedom to the subproblem. Since domain propagation is an essential ingredient in MINLP solvers, this might still reduce the computational effort significantly on some problems.

[1] As a simple example consider the multilinear term $x_1 \cdots x_n$ with no variable bounds. The minimum cardinality of an $x^\star$-cover is 1 as soon as $x_k^\star = 0$ for some k. In contrast, the smallest global cover has size $n-1$.

[2] Alternatively, we could recompute our approximation to obtain values within the current bounds.

[3] In our implementation, we try the bounds of the variable, if finite, and the midpoint between the approximation value and each finite bound. For unbounded variables, we try zero and twice the approximation value.

5. Computational results

Only few solvers exist that handle general nonconvex MINLPs, such as BARON [21], Couenne [4], and LindoGlobal [19]. Others, such as BONMIN [9] and SBB [3], guarantee global optimality only for convex problems, but can be used as heuristic solvers for nonconvex problems. Recently, the solver SCIP [1] was extended to solve nonconvex MIQCPs to global optimality [7].

The target of our computational experiments is to demonstrate the potential of Undercover as a start heuristic for MINLPs applied at the root node. We implemented Undercover within the branch-cut-and-price framework SCIP [1] and incorporated the fix-and-propagate, backtracking, and NLP postprocessing features described in Section 4. To perform the fix-and-propagate procedure, we use the standard propagation engine of SCIP. Secondary SCIP instances were used to solve both the cover generating problem and the Undercover sub-MIP.

In our experiments, we ran SCIP with all heuristics other than Undercover switched off, set a node limit of 1, and deactivated cut generation. We set a node limit of 500 both for the covering problem and the sub-MIP. For solving the sub-MIP, we used "emphasis feasibility" and "fast presolving" settings. We used SCIP 1.2.0.4 with CPLEX 12.1 [18] as LP solver and Ipopt 3.7 [22] as NLP solver for the postprocessing. This configuration we refer to as UC.

For comparison, we ran SCIP 1.2.0.4 with CPLEX 12.1 and Ipopt 3.7 in default mode, which applies ten primal heuristics at the root node. We further compared with the state-of-the-art solvers BARON [21] (commercial) and Couenne [4] (open source). For all solvers, we used node limit 1.

As test set we used a selection of 35 MIQCP instances from MINLPLib [12]. We excluded `lop97ic`, `lop97icx`, `pb302035`, `pb351535`, `qap`, and `qapw`, which are linear after the default presolving of SCIP. The results are shown in Table 1. In column "% covered" we report the relative size of the cover used by UC as percentage of the total number of variables after preprocessing, and the objective value of the best solution found by Undercover. For all other solvers, we provide the objective value of the best solution found during root node processing. The best solution among the four solvers is marked bold.

UC finds a feasible solution for 21 test instances, whereas SCIP does in 14, BARON in 15, Couenne in 10. On those instances for which Undercover and one of the other solvers each find a

Table 1: Computational results on MIQCP instances.

instance	% covered	UC	SCIP	BARON	Couenne
du-opt	95.24	4233.8709	4233.8709	108.331477	**41.3038865**
du-opt5	94.74	3407.05415	**14.1684489**	–	1226.02232
ex1263	3.88	**30.1**	–	–	–
ex1264	4.26	**15.1**	–	–	–
ex1265	3.52	**15.1**	–	–	**15.1**
ex1266	3.03	**16.3**	–	–	–
fac3	78.26	130653857	–	**38328601.6**	–
feedtray2	3.26	–	0	0	–
meanvarx	23.33	16.9968975	**14.3692129**	14.3692321	18.701873
netmod_dol1	0.30	0	**-0.317294979**	0	–
netmod_dol2	0.38	-0.0780227488	**-0.50468289**	0	–
netmod_kar1	0.88	0	**-0.132809562**	0	–
netmod_kar2	0.88	0	**-0.132809562**	0	–
nous1	19.44	–	–	–	**1.567072**
nous2	19.44	–	1.38431729	**0.625967412**	1.38431741
nuclear14	–	–	–	–	**-1.12259354**
nuclear14a	6.43	–	–	–	–
nuclear14b	6.43	–	–	–	**-1.11054393**
nuclearva	–	–	–	–	–
nvs19	88.89	–	0	**-1098**	–
nvs23	90.00	–	0	**-1124.8**	–
product	30.87	–	–	–	–
product2	26.15	–	–	–	–
sep1	10.53	**-510.080984**	–	**-510.080984**	**-510.080984**
space25	1.04	–	–	–	–
space25a	5.84	–	–	–	–
spectra2	35.71	26.6076018	**23.2840887**	119.8743	–
tln5	9.09	15.1	–	–	**14.5**
tln6	7.69	**32.3**	–	–	–
tln7	6.67	**30.3**	–	–	–
tln12	3.99	–	–	–	–
tloss	7.89	**27.3**	–	–	–
tltr	12.50	**61.1333333**	–	–	–
util	3.13	**999.578743**	1000.48517	1006.50609	–
waste	5.65	693.392795	**693.290675**	712.301232	–

solution, the solution found by Undercover is better than the one found by SCIP in 4 cases (worse in 8), better than BARON 6 times (worse 3), better than Couenne in 1 case (worse 3 cases). On `nuclear14` and `nuclearva`, UC could not run, since the root LP relaxation is unbounded due to unbounded variables in nonconvex terms of the constraints.

The overall time for SCIP preprocessing, solving the root LP and applying Undercover was always less than two seconds. Thereof, the time for applying Undercover was always less than 0.2 seconds, except for the instance `waste`, where Undercover ran for 1.1 seconds. The major amount of time was usually spent in solving the sub-MIP. Although the polytope described by (3.3) is not integral, the covering instance could always be solved to optimality in the root node by SCIP's default heuristics and cutting plane algorithms. In 10 out of the 14 cases where the resulting sub-MIP was infeasible, the infeasibility was already detected during the fix-and-propagate stage.

Conclusion. Altogether, Undercover seems to be a fast start heuristic, that often produces feasible solutions of reasonable quality. On the chosen test set, the experiments comfirmed our expectation that a low fixing rate often suffices to obtain a feasible linear subproblem which is easy to solve. The computational results indicate, that it complements nicely with existing root node heuristics in different solvers.

Bibliography

[1] T. Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.
[2] T. Achterberg and T. Berthold. Improving the Feasibility Pump. *Disc. Opt.*, Special Issue 4(1):77–86, 2007.
[3] ARKI Consulting & Development A/S and GAMS Inc. SBB. `http://www.gams.com/solvers/solvers.htm#SBB`.
[4] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Waechter. Branching and bounds tightening techniques for non-convex MINLP. Research Report RC24620, IBM, 2008.
[5] T. Berthold. Primal heuristics for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.
[6] T. Berthold. RENS – Relaxation Enforced Neighborhood Search. ZIB-Report 07-28, Zuse Institute Berlin, 2007.
[7] T. Berthold, S. Heinz, and S. Vigerske. Extending a CIP framework to solve MIQCPs. ZIB-Report 09-23, Zuse Institute Berlin, 2009.
[8] R.E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice – closing the gap. In M.J.D. Powell and S. Scholtes, editors, *Systems Modelling and Optimization: Methods, Theory, and Applications*, pages 19–49. Kluwer Academic Publisher, 2000.
[9] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Disc. Opt.*, 5:186–204, 2008.
[10] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Math. Prog.*, 119(2):331–352, 2009.
[11] P. Bonami and J.P.M. Gonçalves. Primal heuristics for mixed integer nonlinear programs. Research Report RC24639, IBM, 2008.
[12] M.R. Bussieck, A.S. Drud, and A. Meeraus. MINLPLib – A Collection of Test Models for Mixed-Integer Nonlinear Programming. *INFORMS Journal on Computing*, 15(1):114–119, 2003.
[13] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Prog. A*, 102(1):71–90, 2004.
[14] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Math. Prog. A*, 104(1):91–104, 2005.
[15] M. Fischetti and A. Lodi. Local branching. *Math. Prog. B*, 98(1-3):23–47, 2003.
[16] M. Fischetti and D. Salvagnin. Feasibility pump 2.0. *Math. Prog. C*, 1:201–222, 2009.
[17] S. Ghosh. DINS, a MIP improvement heuristic. In *Proc. of 12th IPCO*, pages 310–323, 2007.
[18] ILOG, Inc. CPLEX. `http://www.ilog.com/products/cplex`.
[19] Lindo Systems, Inc. LindoGlobal. `http://www.lindo.com`.
[20] G. Nannicini, P. Belotti, and L. Liberti. A local branching heuristic for MINLPs. *ArXiv e-prints*, 2008.
[21] M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Math. Prog.*, 99:563–591, 2004.
[22] 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. Prog.*, 106(1):25–57, 2006.

Timo Berthold & Ambros M. Gleixner
Zuse Institute Berlin
Takustraße 7, 14195 Berlin, Germany
`{berthold,gleixner}@zib.de`