LaGO - a (heuristic) Branch and Cut algorithm for nonconvex MINLPs

Ivo Nowak* and Stefan Vigerske†

Submitted 20.09.2006

Abstract

We present a Branch and Cut algorithm of the software package LaGO to solve nonconvex mixed-integer nonlinear programs (MINLPs). A linear outer approximation is constructed from a convex relaxation of the problem. Since we do not require an algebraic representation of the problem, reformulation techniques for the construction of the convex relaxation cannot be applied, and we are restricted to sampling techniques in case of nonquadratic nonconvex functions. The linear relaxation is further improved by mixed-integer-rounding cuts. Also box reduction techniques are applied to improve efficiency. Numerical results on medium size test problems are presented to show the efficiency of the method.

Keywords: global optimization, branch and bound, branch and cut, outer approximation, mixed-integer nonlinear programming

1 Introduction

To find a global optimal point of a MINLP, local search methods are not sufficient, since the presence of discrete variables and nonconvex constraints introduces local minimal points that are not globally optimal. Mainly two approaches exist for the deterministic global optimization of a (nonconvex) MINLP [12, 20, 23]. In successive outer-approximation algorithms [10, 11, 29] an initial relaxation is iteratively solved and improved until a feasible point of the MINLP is found, which is then also globally optimal. In branching methods the feasible set is subdivided into smaller subregions (branching). If an optimal point of a subproblem is not found or global optimality (for the subproblem) cannot be verified, the subregion is further subdivided. In Branch and Bound algorithms [1, 2], lower bounds for each subregion are compared with an upper bound on the global optimum. Lower bounds from a linear relaxation that is generated by cutting planes lead to Branch and Cut algorithms [26, 27, 28].

This paper presents the Branch and Cut algorithm that is implemented in the MINLP solver LaGO (**La**grangian **G**lobal **O**ptimizer) [25]. The development on LaGO started in 2000 as a solver for nonconvex MIQQPs, based on Lagrangian decomposition and semidefinite relaxations [21, 22]. In a project funded by the German Science Foundation it was extended to a MINLP solver (2001–2004) [23, 24]. LaGO is now a COIN-OR project [16] and available as open source code at [25].

We consider MINLPs that are given in the following form:

$$\begin{array}{rll} \min & b_0^T x & \\ \text{such that} & h(x) \le 0, & \\ & x \in [\underline{x}, \overline{x}], & \\ & x_j \in \mathbb{Z}, & j \in B, \end{array} \tag{P}$$

*Lufthansa Systems Berlin

†Humboldt University, Department of Mathematics, 10099 Berlin, Germany, stefan@math.hu-berlin.de

where $B \subseteq \{1, \ldots, n\}$, b_0, $\underline{x}$, $\overline{x} \in \mathbb{R}^n$, and $h \in C^2(\mathbb{R}^n, \mathbb{R}^m)$. For the sake of simplicity we assume that the objective function is linear and equality constraints were replaced by two inequalities. Note, that to handle a nonlinear objective function $h_0(x)$, one can minimize a new variable y under the additional constraint $h_0(x) \leq y$.

We assume to have procedures for evaluating function values, gradients, and Hessians of the functions $h_i(x)$. The restriction to black-box functions has the advantage that we can handle very general functions, but has the disadvantage that advanced reformulation and box reduction techniques (as in [2, 20, 28]) cannot be used. Hence, in some components of the proposed algorithm we are restricted to sampling methods, by what we are departing from the area of deterministic global optimization. However, as our numerical experiments show, the obtained results are comparable with those from deterministic solvers which can utilize insights into the functions $h_i(x)$. For some advanced components, we assume further that we know which variables occur linearly and nonlinearly in a function, and that we can evaluate a function over an interval. All these information and functions are provided by the GAMS interface [13].

The proposed algorithm follows a Branch and Bound scheme to search for a global optimum of (P). It starts by considering the original problem with its complete feasible region, which is called the root problem. A lower bound on the global optimum of (P) is computed by solving a linear outer-approximation of (P). An upper bound $\overline{v}$ is computed by finding a local optimum of (P). If the bounds match, a global optimal solution has been found and the procedure terminates. Otherwise, two new problems are constructed by division of the feasible region of (P) using a subdivision of the box $[\underline{x}, \overline{x}]$ (branching). The new problems become children of the root problem, and the algorithm is applied recursively on each subproblem. This process constructs a tree of subproblems, the Branch and Bound tree.

The gap between the lower bound $\underline{v}(U)$ of a node U and the global upper bound $\overline{v}$ is diminished by improving the linear outer-approximation on each node and by computation of local optimal points. If such a point is found and the upper bound $\overline{v}$ is improved, nodes of the tree whose lower bound exceeds $\overline{v}$ are pruned. The process of branching and bounding is performed until no unprocessed nodes are left or the gap has been sufficiently reduced.

The outer-approximation is improved by cutting planes that are derived from a (nonlinear) convex outer-approximation of (P) and by mixed-integer-rounding cuts derived from the linear outer-approximation itself. The efficiency of the algorithm is improved by box reduction techniques that allow to tighten the box $[\underline{x}, \overline{x}]$ (or a subbox) and can discover infeasibility.

The components of LaGO are explained in more detail in the next sections. We start with a reformulation of (P) into a block-separable form (Section 2). Section 3 depicts the steps to the linear outer-approximation of (P). Box reduction algorithms are explained in Section 4. The components are brought together in a Branch and Cut algorithm (Section 5). Finally, a comparison with the MINLP solver BARON [26] on models from the GAMS test problem libraries is presented in Section 6.

2 Block-separable reformulation

Many real-world optimization problems have a natural separable structure, which is often related to components of the underlying model. This structure allows all functions of (P) to be represented as a sum of sub-functions which depend on a small number of variables. Functions having such a property are called *block-separable*. LaGO automatically identifies a block-separable structure of the black-box functions of (P) and reformulates them to

$$h_i(x) = c_i + b_i^T x + \sum_{k=1}^{q_i} x_{Q_{i,k}}^T A_{i,k} x_{Q_{i,k}} + \sum_{k=1}^{p_i} h_{i,k}(x_{N_{i,k}}), \tag{1}$$

where the index sets $Q_{i,k}$ and $N_{i,k}$ of quadratic and nonlinear nonquadratic variables, respectively, are a partition of the set V_i of variables that appear nonlinear in $h_i(x)$, i.e., $V_i = \dot{\bigcup}_{k=1}^{q_i} Q_{i,k} \dot{\bigcup}_{k=1}^{p_i} N_{i,k}$. The sets $Q_{i,k}$ and $N_{i,k}$ are also refered as *blocks* of the function $h_i(x)$. Furthermore, the sparsity graph E_i^{sparse} of the Hessian for each function is computed. This graph has V_i as nodes and there is an edge between j and j' if there is a point $x \in [\underline{x}, \overline{x}]$ such that $(\nabla^2 h_i(x))_{j,j'} \neq 0$, i.e., the variables x_j and $x_{j'}$ are *coupled* in $h_i(x)$.

The block-separable structure allows to distinguish between linear, quadratic, and nonquadratic parts of a function, and to treat each block separately if advantageous. Unification of the sparsity graphs E_i^{sparse}, $i = 1, \dots, m$, allows to identify a block-separable structure for the whole problem, in which blocks are coupled by linear constraints only.

The graph E_i^{sparse} is constructed by evaluation of the Hessian $\nabla^2 h_i(\hat{x})$ at sample points $\hat{x} \in [\underline{x}, \overline{x}]$ and adding the edge $\{j, j'\}$ to E_i^{sparse} if $(\nabla^2 h_i(\hat{x}))_{j,j'}$ is nonzero for at least one sample point $\hat{x}$. The blocks of $h_i(x)$ are then the connected components of E_i^{sparse}. If for all variables j and j' from one block the Hessian entries $(\nabla^2 h_i(\hat{x}))_{j,j'}$ are constant (over all considered sample points), the block contains only quadratic variables, thus it yields a set $Q_{i,k}$.

Observe, that only the information whether some entry of the Hessian is constant and nonzero is used, but not the actual values. Thus, for functions that are common in practice, this sampling approach yields correct results.

3 Relaxations

We now describe the steps which lead to a polyhedral relaxation of (P).

First, for each function $h_{i,k}(x_{N_{i,k}})$ and $x_{Q_{i,k}}^T A_{i,k} x_{Q_{i,k}}$ (as given by (1)) it is checked whether it is convex over $[\underline{x}, \overline{x}]$ or not. For a function $h_{i,k}(x_{N_{i,k}})$ the minimal eigenvalue of $\nabla^2 h_{i,k}(x_{N_{i,k}})$ is evaluated at sample points. Observe that only the sign of the eigenvalue is of interest, so that even for curvaceous functions a sufficiently rich set of sampling points yields correct results. However, since the functions $h_{i,k}(x_{N_{i,k}})$ can be of any (twice-differentiable) form, their eigenvalues can depend very irregular on x, thus the sampling should be finer than in Section 2. For a function $x_{Q_{i,k}}^T A_{i,k} x_{Q_{i,k}}$, we can check convexity by evaluating the minimal eigenvalue of $A_{i,k}$.

Next, convex underestimators are constructed in a two-step approach. First, nonconvex functions $h_{i,k}(x_{N_{i,k}})$ are underestimated by (possibly nonconvex) quadratic functions (Section 3.1). Second, quadratic nonconvex functions are replaced by convex α-underestimators (Section 3.2) [3]. The direct application of the α-underestimator technique to a function $h_{i,k}(x_{N_{i,k}})$ would also give a convex underestimator. However, the proposed quadratic underestimator is often tighter because the α-convexification depends only on the curvature of the function and not on the function behavior.

Finally, nonlinear functions of the convex relaxation are linearized to obtain a polyhedral relaxation.

3.1 Pre-convex relaxation

The relaxation (Q) of (P) is obtained by replacing nonconvex functions $h_{i,k}(x_{N_{i,k}})$ by quadratic underestimators $p(x_{N_{i,k}})$ and dropping the integrality restrictions on the variables x_j, $j \in B$.

Let $f : \mathbb{R}^r \to \mathbb{R}$ be a nonconvex functions $h_{i,k}$ from (1), $r := |N_{i,k}|$. The coefficients A, b, and c of a quadratic underestimator $p(x) = x^T A x + b^T x + c$ of $f(x)$ over a box $[\underline{x}, \overline{x}] \subset \mathbb{R}^r$ are computed by the following linear program:

$$
\begin{array}{rll}
\min\limits_{A,b,c} & \sum\limits_{x \in S} f(x) - p(x) & \\
\text{such that} & p(x) \leq f(x), & x \in \hat{S}, \\
 & p(\hat{x}) = f(\hat{x}), &
\end{array}
\tag{2}
$$

where $\hat{S} \subset [\underline{x}, \overline{x}]$ is a set of sample points, $S \subseteq \hat{S}$, $\hat{x} \in \hat{S}$ is a reference point, and the sparsity pattern of the matrix A and the vector b are chosen according to that of f.

This method can be applied to black-box functions for which no analytic expressions are known. The quality of the quadratic underestimator depends thereby strongly on the sample set

$\hat{S}$. Therefore, we implemented an adaptive procedure to improve $\hat{S}$. It locally maximizes the error $p(x) - f(x)$ over $[\underline{x}, \overline{x}]$ and, if the error becomes larger then a prescribed tolerance $\delta_{\mathrm{tol}} > 0$, that is, $p(x)$ overestimates $f(x)$ by at least δ_{tol} in some point, $\hat{S}$ is enlarged and $p(x)$ recomputed, c.f. Algorithm 1.

Algorithm 1 Computation of a quadratic underestimator

1. Set $\hat{S} := S := \mathrm{vert}([\underline{x}, \overline{x}]) \cup \{\hat{x}, (\underline{x} + \overline{x})/2\} \cup M$, where $\mathrm{vert}([\underline{x}, \overline{x}])$ are the vertices of the box $[\underline{x}, \overline{x}]$ (or a subset of them if r is large), $\hat{x}$ is one local minimum point of f, and $M \subset [\underline{x}, \overline{x}]$ a set of randomly generated points.
2. Compute $p(x)$ by solving (2).
3. For all $\tilde{x} \in \hat{S}$ with $f(\tilde{x}) = p(\tilde{x})$:
   Maximize the error $p(x) - f(x)$ for $x \in [\underline{x}, \overline{x}]$ by a local search starting from $\tilde{x}$. If a point x^* with $p(x^*) - f(x^*) > \delta_{\mathrm{tol}}$ is found, add x^* to $\hat{S}$ and go to 2.
4. Let $\delta_{\max}$ be the maximal error found by the local searches in Step 3. If $\delta_{\max} > 0$, lower $p(x)$ by setting $c := c - \delta_{\max}$.

3.2 Convex relaxation

The relaxation (C) of (Q) is obtained by replacing all nonconvex forms (which are quadratic terms due to the construction of (Q)) by α-underestimators as introduced by Adjiman and Floudas [3] (note also recent improvements of this technique in [7, 18]). An α-underestimator of a function $f(x) = x^T Ax$ over the box $[\underline{x}, \overline{x}]$ is the function

$$\breve{f}(x) = f(x) + \sum_{i=1}^{r} \frac{\max\{0, -\lambda_1(WAW)\}}{(\overline{x}_i - \underline{x}_i)^2}(x_i - \underline{x}_i)(x_i - \overline{x}_i), \tag{3}$$

where $\lambda_1(D)$ denotes the minimal eigenvalue of a matrix D, the diagonal matrix W has the box-width $\overline{x} - \underline{x}$ on its diagonal and has been introduced for scaling reasons. It is clear that $\breve{f}$ is convex and $\breve{f}(x) \leq f(x)$ for all $x \in [\underline{x}, \overline{x}]$ [3].

The convex relaxation takes now the form

$$\begin{array}{rl} \min & c^T x \\ \text{such that} & \breve{h}(x) \leq 0, \\ & x \in [\underline{x}, \overline{x}], \end{array} \tag{C}$$

where $\breve{h}_i(x) \equiv h_i(x)$ for convex functions $h_i(x)$ in (Q).

3.3 Linear relaxation by cut-generation

The linear relaxation (R) of (P) is generated by linearization of each nonlinear function $\breve{h}_i(x)$ in (C) at an optimal point of (C). In the Branch and Cut algorithm, (R) is augmented by further linearizations at, e.g., candidates for optimal points of (P). Observe, that the linearization of an α-underestimator (3) can easily be updated after a change of the box.

Since the linearization of functions from (C) does not allow to inherit information about integrality requirements into (R) and we do not want to admit the computational burden of a mixed-integer linear relaxation, we additionally add mixed-integer-rounding (MIR) cuts to (R). These cuts have their origin in mixed-integer linear programming [17, 19]. An MIR cut is derived from the following disjunctive argument [17], which can be extended to general linear inequalities: Let $X := \{(x, y) \in \mathbb{Z} \times \mathbb{R}_+ | x - y \leq b\}$. Then the inequality

$$x - \frac{1}{1 - (b - \lfloor b \rfloor)} y \leq \lfloor b \rfloor$$

is valid for both the sets $X \cap \{(x,y)|x \le \lfloor b \rfloor\}$ and $X \cap \{(x,y)|x \ge \lfloor b \rfloor + 1\}$, see also Figure 1 for an illustration. MIR cuts are constructed after (R) has been solved. They cut off a solution point with nonintegral values for some x_j, $j \in B$, from the feasible set of (R), cf. [14, 17].

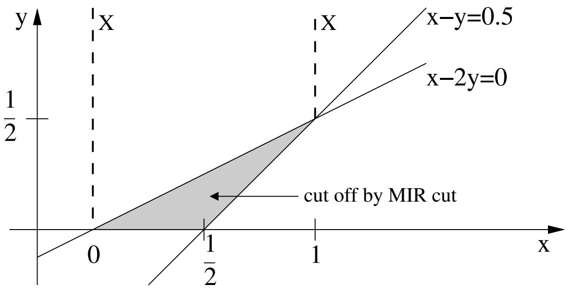

Figure 1: The MIR cut $x - 2y \le 0$ derived from $X = \{(x,y) \in \mathbb{Z} \times \mathbb{R}_+ | x - y \le 0.5\}$.

For a box $U \subseteq [\underline{x}, \overline{x}]$, we denote by (R[$U$]) the linear relaxation where the variables are restricted to take values in U:

$$\begin{array}{lll} \min & b_0^T x & \\ \text{s.t.} & \breve{h}_i(x) \le 0, & \text{if } \breve{h}_i \text{ is linear}, \\ & \breve{h}_i(x^*) + \nabla \breve{h}_i(x^*)(x - x^*) \le 0, \quad x^* \in X^*, & \text{if } \breve{h}_i \text{ is nonlinear}, \\ & x \in U, & i \in \{1, \ldots, m\}, \\ & d^T x \le e, & (d,e) \in C_{\text{MIR}}, \end{array} \tag{R[U]}$$

where X^* are reference points generated in the preprocessing and Branch and Cut (e.g., local minimizers of (P) and (C)), and C_{MIR} is a set of MIR cuts.

4 Box reduction

In practice, the bounding box $[\underline{x}, \overline{x}]$ of a given MINLP may be quite large. In this case, the quality of the convex underestimators and cuts may be bad. This drawback might be prevented if a box reduction procedure is applied in the preprocessing. Also during the Branch and Cut algorithm, a branching operation might facilitate possible reductions of variable bounds, and even detect infeasibility for a subregion or fix binary variables.

Two box reduction techniques are currently implemented in LaGO. The first one utilizes the whole set of constraints of the linear relaxation (R) at once, where the second one is a simple constraint propagation method which utilizes one constraint at a time, but works on the original formulation (P).

4.1 Box reduction by enclosing the feasible set of the linear relaxation

This procedure minimizes (maximizes) one variable over the feasible set of the linear relaxation to obtain a new lower (upper) bound on that variable. The feasible set is further restricted by a level cut that cuts off all points which objective function value exceeds the incumbent upper bound $\overline{v}$.

Let $U \subseteq [\underline{x}, \overline{x}]$ be a box and denote by $\Omega[U]$ the feasible set of (R[U]). The new lower (upper) bound on a variable x_j is computed by solving

$$\begin{array}{rl} \min\,(\max) & x_j \\ \text{such that} & x \in \Omega[U] \\ & c^T x \le \overline{v} \end{array} \tag{B$_j$[U]}$$

If (B$_j[U]$) is infeasible, no point with a better optimal value than $\overline{v}$ exists in U. Hence, the subregion U does not need further investigation.

Solving (B$_j[U]$) for all variables is quite expensive. Hence, it's solved for variables which seem promising for a box reduction only, cf. Algorithm 2.

4.2 Box reduction by interval arithmetic

This algorithm does not depend on the quality of the relaxation (R[U]). Instead it applies interval arithmetic techniques to the constraints of the original formulation (P) to tighten the box.

Let $U \subseteq [\underline{x}, \overline{x}]$ be a box and write $h_i(x) = g(x) + bx_j$ with $b \neq 0$. By $g(U)$ we denote an interval in $\mathbb{R} \cup \{\pm\infty\}$ such that $g(x) \in g(U)$ for all $x \in U$. Let $[\underline{y}_j, \overline{y}_j] = -g(U)/b$. If $b > 0$, $\overline{x}_j$ can be updated to $\min(\overline{x}_j, \overline{y}_j)$, and if $b < 0$, $\underline{x}_j$ can be updated to $\max(\underline{x}_j, \underline{y}_j)$. In case that the new bounds define an empty box, infeasibility of the subproblem with box U is detected.

After reducing the box of a variable x_j, other constraints depending on x_j might yield further box reductions. This information is stored in a graph G with the variables as nodes. The box reduction algorithm is given a list J of variables. It takes an index j from J and considers all neighbors of j in G. If the box of some neighbor can be reduced considerably, it is added to J. This process iterates until J is empty or infeasibility is detected.

5 Branch and Cut algorithm

The Branch and Cut algorithm is outlined in Algorithm 2. Even though we assumed that a bounded box on the variable values is given within the problem formulation (P), this is not the case for many models like those from the libraries we used in Section 6. While the first three steps of the preprocessing do not necessarily require a bounded box, the computation of the relaxation (Q) relies strongly on it. Hence, if after the (maybe expensive) solution of convex nonlinear problems in step 4 some variable is unbounded, we have the choice to stop the algorithm and ask the user to modify its model, or to "guess". We decided to guess using the heuristic in step 5.

If the lower bounds $\underline{v}(U)$ are correct and tight, Algorithm 2 converges to a global optimum of (P). Even though linearizations of α-underestimators are updated after branching operations, quadratic underestimators in the relaxation (Q) are not. Hence, the relaxations (Q), (C), and (R) might not be tight for problems that have nonconvex nonquadratic functions, and convergence to a global optimum cannot be ensured. Another problem occurs when the quadratic underestimator of a function $h_{i,k}(x_{N_{i,k}})$ is not rigorous and a wrong lower bound leads to a mistaken pruning of a node. Hence, in case of nonconvex functions $h_{i,k}(x_{N_{i,k}})$, the proposed algorithm is only heuristic. However, as we show next, it performs well on many examples.

6 Performance on examples from MINLPLib and GlobalLib

We now summarize results that show the performance of LaGO on examples from the GAMS model libraries MINLPLib [8] and GlobalLib [9] which have at most 1000 variables. To allow a comparision with BARON [26], we further excluded models that contain sin, cos, or errorf functions, since these cannot be handled by BARON. Hence, our test set consists of 77 MIQQPs (22 thereof convex), 127 (nonquadratic) MINLPs (59 thereof convex), and 162 QQPs (11 thereof convex). Nonquadratic NLPs are not considered.

LaGO was run on these examples with a time limit of one hour and a final gap tolerance of 1%. Because of the above mentioned limitations in the improvement of underestimators, branching for nonquadratic MINLPs was restricted to discrete variables only. Hence, subproblems where all binary variables are fixed are discarded even when the gap is not closed yet.

CONOPT 3.14P [13] was used as local optimizer for (P) (with fixed x_B) and IPOPT 3.2 [30] to solve (C) and (B$_j$[U]) in step 4 of Algorithm 2 and for the local searches in Algorithm 1. LPs were solved with CPLEX 10.0 [15], and MIR cuts were generated by the CGL [14, 16].

Table 1 compares the results obtained by LaGO with the best known optimal point from the MINLPLib. The GlobalLib does not contain such points for many models, so we excluded the QQPs from this table.

Table 2 summarizes the results of a competition of LaGO with the state-of-the-art solver BARON 7.8.1 [26], c.f. [25] for details. If LaGO did not solve a model then because the time limit was exceeded, often because the Branch and Bound method was overburdened by the combinatorial

Algorithm 2 Branch and Cut algorithm

Preprocessing:

1. Reformulate all functions of (P) into the form (1) and compute sparsity graphs.
2. Perform box reduction by interval arithmetic and by enclosing the polyhedron defined by linear functions $h_i(x)$ in (P) (if any) and the box $[\underline{x}, \overline{x}]$.
3. Check the functions $h_{i,k}(x_{N_{i,k}})$ and $x^T_{Q_{i,k}} A_{i,k} x_{Q_{i,k}}$ for convexity.
4. For all missing bounds of a variable x_j, solve ($\mathrm{B}_j[U]$) with Ω defined by the convex (and possibly nonlinear) constraints of (P).
5. For all j with $\overline{x}_j = \infty$, set $\overline{x}_j := \max(10000, 10\max\{\overline{x}_i | \overline{x}_i < \infty\})$. For all j with $\underline{x}_j = -\infty$, set $\underline{x}_j := \min(-10000, 10\min\{\underline{x}_i | \underline{x}_i > -\infty\})$.
6. Construct pre-convex relaxation (Q).
7. Construct convex relaxation (C).
8. Solve convex relaxation (C). Let x^* be a solution point of (C).
9. Construct linear relaxation (R) using $X^* = \{x^*\}$ and $C_{\mathrm{MIR}} = \emptyset$.
10. Perform box reduction by enclosing the feasible set of the linear relaxation.
11. Perform box reduction by interval arithmetic with J the set of variables which box has been updated in the last step.

Main loop:

Initialize the Branch and Bound tree $\mathcal{U}$ with the node $[\underline{x}, \overline{x}]$. Set $\overline{v} = \infty$.
Set $\underline{v}([\underline{x}, \overline{x}])$ to the optimal value of (C) and $\hat{x}_{[\underline{x},\overline{x}]}$ to the solution point x^* of (C).
Repeat the following steps as long as the there are unprocessed nodes in $\mathcal{U}$ and the gap between $\min_{U\in\mathcal{U}} \underline{v}(U)$ and $\overline{v}$ is too large.

1. Node selection: Take a node U with lowest lower bound from $\mathcal{U}$.
2. Upper bounds: If none of the local optimizers found so far lie in U, start a local search from $\hat{x}_U$ (with rounded discrete variables) in (P) where the discrete variables are fixed. If a new local optimizer x^* is found, update $\overline{v}$ and construct linearization cuts by adding x^* to X^*.
3. Branching: select a variable x_j
   - whose integrality condition is mostly violated by $\hat{x}_U$ (if $j \in B$),
   - or: where $(\overline{x}_j - (\hat{x}_U)_j)((\hat{x}_U)_j - \underline{x}_j)/(\overline{x}_j - \underline{x}_j)^2$ is largest for a variable x_j that appears nonlinearly in a by $\hat{x}_U$ most violated constraint of (Q)
   - or: whose box $[\underline{x}_j, \overline{x}_j]$ is least reduced.

   Construct the nodes $U_1 := \{x \in U | x_j \le (\hat{x}_U)_j\}$, $U_2 := \{x \in U | x_j \ge (\hat{x}_U)_j\}$.
4. Lower bounds for both nodes U_t, $t = 1, 2$: Let $J = \{j\}$.
   (a) Update linearizations of an α-underestimator to the new box U_t in ($\mathrm{R}[U_t]$).
   (b) Reduce the box U_t by interval arithmetic starting with the variable set J.
   (c) Reduce the box U_t by solving problem ($\mathrm{B}_k[U_t]$) for all variables x_k that are in a block (i.e., a connected component of the sparsity graph of (P)) which box has been reduced by at least 20% in step (b). Let J be the variables which box has been reduced by at least 20%.
   (d) Solve the linear relaxation ($\mathrm{R}[U_t]$), update $\underline{v}(U_t)$, and let $\hat{x}_{U_t}$ be a minimizer of ($\mathrm{R}[U_t]$). Generate MIR cuts and add $\hat{x}_{U_t}$ to X^*. Repeat this step several times, if $\hat{x}_{U_t}$ is not feasible for (C) or MIR cuts were generated.
   (e) If infeasibility of U_t has been proven, proceed with U_2 (if $t = 1$) or step 5.
   (f) If $J \ne \emptyset$, go back to step (a). Otherwise, add U_t to $\mathcal{U}$.
5. Pruning: Prune nodes $U \in \mathcal{U}$ with $\underline{v}(U) > \overline{v}$.

	MIQQPs	MINLPs
number of models	77	127
best known optimal solution found	60	68
nonoptimal but feasible solution found	1	18
no feasible solution found	16	41

Table 1: Performance of LaGO on MINLPLib models.

part of the model. Exceptions are a few MINLPs: In 2 cases CONOPT reported a solution as feasible which was in fact infeasible, in 4 cases LaGO failed due to too many domain violations in function evaluations, in 2 cases a guess on variable bounds was wrong, and in another 2 cases other numerical errors appeared. For the QQP `bayes2_10` LaGO found a better point because BARON computed a wrong lower bound in the root node. Failures of BARON for MIQQPs and MINLPs are in most cases due to the time limit (e.g., models `nuclear*`), and in 5 cases presumably because of missing variable bounds.

			better optimal value by		
		Total	LaGO	same	BARON
QQPs	BARON fail, LaGO not	0			
	LaGO faster	10		10	
	both solvers the same	86		86	
	BARON faster	65	1	64	
	LaGO fail, BARON not	1			1
	Total	162	1	160	1
MIQQPs	BARON fail, LaGO not	2	2		
	LaGO faster	13		13	
	both perform the same	22		21	1
	BARON faster	24		24	
	LaGO fail, BARON not	9			9
	LaGO and BARON fail	7		7	
	Total	77	2	65	10
MINLPs	BARON fail, LaGO not	9	9		
(nonquadratic)	LaGO faster	12	1	7	4
	both solvers the same	11		5	6
	BARON faster	54		46	8
	LaGO fail, BARON not	34			34
	LaGO and BARON fail	7		7	
	Total	127	10	65	52

Table 2: Comparison of LaGO and BARON.

We also mention that LaGO was applied to a model of an energy conversion system [4, 5, 6]. The goal was to find a design of a combined-cycle-based cogeneration plant with minimum levelized total costs. Hereby, the structure and process variables of the plant were simultaneously optimized, so that LaGO had to deal with combinatorial decisions concerning the presence and connection of components and nonconvex functions describing thermodynamic behaviours. The GAMS model (super1,2,3 in MINLPLib) has 1308 variables, whereof 44 are binary, 1659 constraints, and is block-separable. We refer to [5] for a detailed discussion of this model and results.

Acknowledgments

We thank Arnold Neumaier for proposing the sample set improvement in Algorithm 1 and GAMS Corporation for their support and provision of a GAMS evaluation license. Further, we thank two anonymous referees whose reports helped to improve an earlier version of this paper.

References

[1] C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. Global optimization of mixed-integer nonlinear problems. *Journal of the American Institute of Chemical Engineers*, 46:1769–1797, 2000.

[2] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances. *Computers and Chemical Engineering*, 22:1137–1158, 1998.

[3] C. S. Adjiman and C. A. Floudas. Rigorous convex underestimators for general twice-differentiable problems. *Journal of Global Optimization*, 9:23–40, 1997.

[4] T. Ahadi-Oskui. *Optimierung des Entwurfs komplexer Energieumwandlungsanlagen*, volume Fortschritts-Berichte VDI, Reihe 6, Nr. 543. VDI-Verlag, Düsseldorf, 2006.

[5] T. Ahadi-Oskui, I. Nowak, G. Tsatsaronis, and S. Vigerske. Optimizing the design of complex energy conversion systems by branch and cut. Preprint 07-11, Department of Mathematics, Humboldt-University Berlin. Available at `http://www.math.hu-berlin.de/publ/pre/2007/P-07-11.pdf`, 2007.

[6] T. Ahadi-Oskui and G. Tsatsaronis. Optimization of the design of a complex energy conversion system using mathematical programming and genetic algorithms. In *Proceedings of IMECE2006*, 2006.

[7] I. G. Akrotirianakis and C. A. Floudas. A new class of improved convex underestimators for twice differentiable constrained NLPs. *Journal of Global Optimization*, 30(4):367–390, 2004.

[8] 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.

[9] GAMS Development Corp. GLOBALLib. `http://www.gamsworld.org/global/globallib.htm`.

[10] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.

[11] R. Fletcher and S. Leyffer. Solving Mixed Integer Nonlinear Programs by Outer Approximation. *Mathematical Programming*, 66(3(A)):327–349, 1994.

[12] C. A. Floudas, I. G. Akrotirianakis, C. Caratzoulas, C. A. Meyer, and J. Kallrath. Global optimization in the 21st century: Advances and challenges. *Computers and Chemical Engineering*, 29(6):1185–1202, 2005.

[13] GAMS Development Corp. *GAMS - The Solver Manuals*. Washington DC, 2003.

[14] J.P.M. Gonçalves and L. Ladanyi. An implementation of a separation procedure for mixed integer rounding inequalities. Research Report RC23686, IBM Research Division, August 2005.

[15] ILOG, Inc. CPLEX. `http://www.ilog.com/products/cplex`.

[16] R. Lougee-Heimer. The Common Optimization INterface for Operations Research. *IBM Journal of Research and Development*, 47(1):57–66, 2003. `http://www.coin-or.org`.

[17] H. Marchand and L.A. Wolsey. Aggregation and mixed integer rounding to solve mips. *Operations Research*, 49(3):363–371, 2001.

[18] C. A. Meyer and C. A. Floudas. Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: Spline αBB underestimators. *Journal of Global Optimization*, 29(6):1185–1202, 2005.

[19] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, 1988.

[20] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. In *Acta Numerica*, volume 13, chapter 4, pages 271–370. Cambridge University Press, 2004.

[21] I. Nowak. A new semidefinite programming bound for indefinite quadratic forms over a simplex. *Journal of Global Optimization*, 14(4):357–364, 1999.

[22] I. Nowak. Lagrangian decomposition of block-separable mixed-integer all-quadratic programs. *Mathematical Programming*, 102(2):295–312, March 2005.

[23] I. Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming*. Birkhäuser, 2005.

[24] I. Nowak, H. Alperin, and S. Vigerske. LaGO - an object oriented library for solving MINLPs. In Ch. Bliek, Ch. Jermann, and A. Neumaier, editors, *Global Optimization and Constraint Satisfaction*, volume 2861 of *Lecture Notes in Computer Science*, pages 31–43. Springer, 2003.

[25] I. Nowak and S. Vigerske. LaGO - Lagrangian Global Optimizer. `https://projects.coin-or.org/LaGO`.

[26] N. Sahinidis and M. Tawarmalani. BARON. `http://archimedes.scs.uiuc.edu/baron/baron.html`, 2002.

[27] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.

[28] 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.

[29] J. Viswanathan and I. E. Grossmann. A Combined Penalty Function and Outer-Approximation Method for MINLP Optimization. *Computers and Chemical Engineering*, 14(7):769–782, 1990.

[30] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. `http://projects.coin-or.org/Ipopt`.