Efficient Separation of RLT Cuts for Implicit and Explicit Bilinear Products

Ksenia Bestuzheva[1(✉)], Ambros Gleixner[1,2], and Tobias Achterberg[3]

[1] Zuse Institute Berlin, Berlin, Germany
{bestuzheva,gleixner}@zib.de
[2] HTW Berlin, Berlin, Germany
[3] Gurobi GmbH, Frankfurt am Main, Germany
achterberg@gurobi.com

Abstract. The reformulation-linearization technique (RLT) is a prominent approach to constructing tight linear relaxations of non-convex continuous and mixed-integer optimization problems. The goal of this paper is to extend the applicability and improve the performance of RLT for bilinear product relations. First, a method for detecting bilinear product relations implicitly contained in mixed-integer linear programs is developed based on analyzing linear constraints with binary variables, thus enabling the application of bilinear RLT to a new class of problems. Our second contribution addresses the high computational cost of RLT cut separation, which presents one of the major difficulties in applying RLT efficiently in practice. We propose a new RLT cutting plane separation algorithm which identifies combinations of linear constraints and bound factors that are expected to yield an inequality that is violated by the current relaxation solution. A detailed computational study based on implementations in two solvers evaluates the performance impact of the proposed methods.

Keywords: Reformulation-linearization technique · Bilinear products · Cutting planes · Mixed-integer programming

1 Introduction

The reformulation-linearization technique (RLT) was first proposed by Adams and Sherali [1–3] for bilinear problems with binary variables, and has been applied to mixed-integer [18–20], general bilinear [21] and polynomial [24] problems. RLT constructs valid polynomial constraints, then linearizes these constraints by using nonlinear relations given in the problem and applying relaxations when such relations are not available. If relations used in the linearization step are violated by a relaxation solution, this procedure may yield violated cuts. By increasing the degree of derived polynomial constraints, hierarchies of

A. Del Pia and V. Kaibel (Eds.): IPCO 2023, LNCS 13904, pp. 14–28, 2023.
https://doi.org/10.1007/978-3-031-32726-1_2

relaxations can be constructed, which were shown to converge to the convex hull representation of MILPs and mixed-integer polynomial problems where continuous variables appear linearly [18–20].

RLT has been shown to provide strong relaxations [21,23], but this comes at the cost of excessive numbers of cuts. To address this, Sherali and Tuncbilek [25] proposed a technique to add a subset of RLT cuts, depending on signs of coefficients of monomial terms in the original constraints and the RLT constraints. Furthermore, the reduced RLT technique [12–14,22] yields equivalent representations with fewer nonlinear terms for polynomial problems containing linear equality constraints.

We focus on RLT for bilinear products, which is of particular interest due to the numerous applications whose models involve nonconvex quadratic nonlinearities [3,5,7–9,17]. Even in the bilinear case, large numbers of factors to be multiplied and of RLT cuts that are generated as a result remain an issue that can lead to considerable slowdowns, both due to the cost of cut separation and the large sizes of resulting LP relaxations.

The first contribution of this paper is a new approach to applying RLT to MILPs. Unlike the approaches that only introduce multilinear relations via multiplication [18,19], this approach detects and enforces bilinear relations that are already implicitly present in the model. A bilinear product relation where one multiplier is a binary variable and the other multiplier is a variable with finite bounds can be equivalently written as two linear constraints. We identify such pairs of linear constraints that implicitly encode a bilinear product relation, then utilize this relation in the generation of RLT cuts.

The second contribution of this paper addresses the major bottleneck for applying RLT successfully in practice, which stems from prohibitive costs of separating RLT cuts, by proposing an efficient separation algorithm. This algorithm considers the signs of bilinear relation violations in a current LP relaxation solution and the signs of coefficients in linear constraints in order to ignore combinations of factors that will not produce a violated inequality. Furthermore, we propose a technique which projects the linear constraints onto a reduced space and constructs RLT cuts based on the resulting much smaller system.

The rest of the paper is organized as follows. In Sect. 2, RLT for bilinear products is explained. In Sect. 3, we describe the technique for deriving bilinear product relations from MILP constraints. Section 4 presents the new cut separation algorithm, and computational results are presented in Sect. 5.

2 RLT for Bilinear Products

We consider mixed-integer (nonlinear) programs (MI(N)LPs) of the extended form where auxiliary variables w are introduced for all bilinear products:

$$\min \mathbf{c}^{\mathrm{T}}\mathbf{x} \tag{1a}$$
$$\text{s.t. } A\mathbf{x} \leqslant \mathbf{b}, \tag{1b}$$
$$g(\mathbf{x}, \mathbf{w}) \leqslant 0, \tag{1c}$$
$$x_i x_j \lesseqgtr w_{ij} \text{ for all } (i,j) \in \mathcal{I}^w, \tag{1d}$$
$$\underline{\mathbf{x}} \leqslant \mathbf{x} \leqslant \overline{\mathbf{x}},\ \underline{\mathbf{w}} \leqslant \mathbf{w} \leqslant \overline{\mathbf{w}}, \tag{1e}$$
$$x_j \in \mathbb{R} \text{ for all } j \in \mathcal{I}^c,\ x_j \in \{0,1\} \text{ for all } j \in \mathcal{I}^b, \tag{1f}$$

with $\mathcal{I} = \mathcal{I}^c \cup \mathcal{I}^b$ being a disjoint partition of variables $\mathbf{x}$ and $\mathbf{x}$ having dimension $|\mathcal{I}| = n$. In the above formulation, $\underline{\mathbf{x}}, \overline{\mathbf{x}} \in \overline{\mathbb{R}}^n$, $\underline{\mathbf{w}}, \overline{\mathbf{w}} \in \overline{\mathbb{R}}^{|\mathcal{I}^w|}$ ($\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$), $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^{m^{(l)}}$ are constant vectors and $A \in \mathbb{R}^{m^{(l)} \times n}$ is a coefficient matrix, and the function g defines the nonlinear constraints. Constraint (1d) defines the bilinear product relations in the problem and allows for inequalities and equations. Let $\mathcal{I}^p$ denote the set of indices of all variables that participate in bilinear product relations (1d).

Solvers typically employ McCormick inequalities [16] to construct an LP relaxation of constraints (1d). These inequalities describe the convex hull of the set given by the relation $x_i x_j \lesseqgtr w_{ij}$:

$$\underline{x}_i x_j + x_i \underline{x}_j - \underline{x}_i \underline{x}_j \leqslant w_{ij}, \quad \overline{x}_i x_j + x_i \overline{x}_j - \overline{x}_i \overline{x}_j \leqslant w_{ij}, \tag{2a}$$
$$\underline{x}_i x_j + x_i \overline{x}_j - \underline{x}_i \overline{x}_j \geqslant w_{ij}, \quad \overline{x}_i x_j + x_i \underline{x}_j - \overline{x}_i \underline{x}_j \geqslant w_{ij}, \tag{2b}$$

where (2a) is a relaxation of $x_i x_j \leqslant w_{ij}$ and (2b) is a relaxation of $x_i x_j \geqslant w_{ij}$.

In the presence of linear constraints (1b), this relaxation can be strengthened by adding RLT cuts. Consider a linear constraint: $\sum_{k=1}^{n} a_{1k} x_k \leqslant b_1$. Multiplying this constraint by nonnegative bound factors $(x_j - \underline{x}_j)$ and $(\overline{x}_j - x_j)$, where $\underline{x}_j$ and $\overline{x}_j$ are finite, yields valid nonlinear inequalities. We will derive the RLT cut using the lower bound factor. The derivation is analogous for the upper bound factor. The multiplication, referred to as the reformulation step, yields:

$$\sum_{k=1}^{n} a_{1k} x_k (x_j - \underline{x}_j) \leqslant b_1 (x_j - \underline{x}_j).$$

This nonlinear inequality is then linearized in order to obtain a valid linear inequality. The following linearizations are applied to each nonlinear term $x_k x_j$:

- $x_k x_j$ is replaced by w_{kj} if the relation $x_k x_j \leqslant w_{kj}$ exists in the problem and $a_{1k} \leqslant 0$, or if the relation $x_k x_j \geqslant w_{kj}$ exists and $a_{1k} \geqslant 0$, or if the relation $x_k x_j = w_{kj}$ exists in the problem,
- if $k = j \in \mathcal{I}^b$, then $x_k x_j = x_j^2 = x_j$,
- if $k = j \notin \mathcal{I}^b$, then $x_k x_j = x_j^2$ is outer approximated by a secant from above or by a tangent from below, depending on the sign of the coefficient,
- if $k \neq j$, $k, j \in \mathcal{I}^b$ and one of the four clique constraints is implied by the linear constraints (1b), then: $x_k + x_j \leqslant 1 \Rightarrow x_k x_j = 0$; $x_k - x_j \leqslant 0 \Rightarrow x_k x_j = x_k$; $-x_k + x_j \leqslant 0 \Rightarrow x_k x_j = x_j$; $-x_k - x_j \leqslant -1 \Rightarrow x_k x_j = x_j + x_j - 1$,
- otherwise, $x_k x_j$ is replaced by its McCormick relaxation.

The key step is the replacing of products $x_k x_j$ with the variables w_{kj}. When a bilinear product relation $x_k x_j \lesseqgtr w_{kj}$ does not hold for the current relaxation solution, this substitution may lead to an increase in the violation of the inequality, thus possibly producing a cut that is violated by the relaxation solution.

In the case that we have a linear equation constraint $\sum_{k=1}^{n} a_{1k} x_k = b_1$ *and* all nonlinear terms can be replaced using equality relations, then RLT produces an equation cut. Otherwise, the equation constraint is treated as two inequalities $\sum_{k=1}^{n} a_{1k} x_k \leqslant b_1$ and $\sum_{k=1}^{n} a_{1k} x_k \geqslant b_1$ to produce inequality cuts.

3 Detection of Implicit Products

Consider a product relation $w_{ij} = x_i x_j$, where x_i is binary. It can be equivalently rewritten as two implications: $x_i = 0 \Rightarrow w_{ij} = 0$ and $x_i = 1 \Rightarrow w_{ij} = x_j$. With the use of the big-M technique, these implications can be represented as linear constraints, provided that the bounds of x_j are finite:

$$w_{ij} - \overline{x}_j x_i \leqslant 0, \;\; w_{ij} - x_j - \underline{x}_j x_i \leqslant -\underline{x}_j \tag{3a}$$

$$-w_{ij} + \underline{x}_j x_i \leqslant 0, \;\; -w_{ij} + x_j + \overline{x}_j x_i \leqslant \overline{x}_j. \tag{3b}$$

Linear constraints with binary variables can be analyzed in order to detect constraint pairs of the forms (3). The method can be generalized to allow for bilinear relations of the following form, with $A, B, C, D \in \mathbb{R}$:

$$A x_i + B w_{ij} + C x_j + D \lesseqgtr x_i x_j \tag{4}$$

Theorem 1. *Consider two linear constraints depending on the same three variables x_i, x_j and w_{ij}, where x_i is binary:*

$$a_1 x_i + b_1 w_{ij} + c_1 x_j \leqslant d_1, \tag{5a}$$

$$a_2 x_i + b_2 w_{ij} + c_2 x_j \leqslant d_2. \tag{5b}$$

If $b_1 b_2 > 0$ and $\gamma = c_2 b_1 - b_2 c_1 \neq 0$, then these constraints imply the following product relation:

$$(1/\gamma)((b_2(a_1 - d_1) + b_1 d_2)x_i + b_1 b_2 w_{ij} + b_1 c_2 x_j - b_1 d_2) \leqslant x_i x_j \quad \text{if } b_1/\gamma \geqslant 0,$$

$$(1/\gamma)((b_2(a_1 - d_1) + b_1 d_2)x_i + b_2 b_2 w_{ij} + b_1 c_2 x_j - b_1 d_2) \geqslant x_i x_j \quad \text{if } b_1/\gamma \leqslant 0.$$

Proof. We begin by writing the bilinear relation (4), treating its coefficients and inequality sign as unknown, and reformulating it as two implications:

$$x_i = 1 \quad \Rightarrow \; B w_{ij} + (C-1) x_j \lesseqgtr -D - A, \tag{6a}$$

$$x_i = 0 \quad \Rightarrow \; B w_{ij} + C x_j \lesseqgtr -D, \tag{6b}$$

where the inequality sign must be identical in both implied inequalities. Similarly, we rewrite constraints (5) with scaling parameters α and β:

$$x_i = 1 \quad \Rightarrow \ \alpha b_1 w_{ij} + \alpha c_1 x_j \lesseqgtr \alpha(d_1 - a_1), \tag{7a}$$

$$x_i = 0 \quad \Rightarrow \ \beta b_2 w_{ij} + \beta c_2 x_j \lesseqgtr \beta d_2, \tag{7b}$$

where the inequality signs depend on the signs of α and β.

The goal is to find the coefficients A, B, C and D and the inequality sign. We require that coefficients and inequality signs in implications (6) and (7) match. Solving the resulting system yields:

$$b_1 b_2 > 0, \ A = (1/\gamma)(b_2(a_1 - d_1) + b_1 d_2)$$
$$B = b_1 b_2/\gamma, \ C = b_1 c_2/\gamma, \ D = -b_1 d_2/\gamma, \ \gamma \neq 0,$$

where $\gamma = c_2 b_1 - b_2 c_1$ and the inequality sign is '$\leqslant$' if $b1/\gamma \geqslant 0$, and '$\geqslant$' if $b1/\gamma \leqslant 0$. Thus, the bilinear relation stated in this theorem is obtained. □

Although the conditions of the theorem are sufficient for the bilinear product relation to be implied by the linear constraints, in practice more conditions are checked before deriving such a relation. In particular:

- At least one of the coefficients a_1 and a_2 must be nonzero. Otherwise, the product relation is always implied by the linear constraints, including when $0 < x_i < 1$.
- The signs of the coefficients of the binary variable x_i must be different, that is, one linear relation is more restrictive when $x_i = 1$ and the other when $x_i = 0$. While this is not necessary for the non-redundancy of the derived product relation, by requiring this we focus on stronger implications (for instance, for a linear relation $a_1 x_i + b_1 w_{ij} + c_1 x_j \leqslant d_1$ with $a_1 > 0$, we use the more restrictive implication $x_i = 1 \ \Rightarrow b_1 w_{ij} + c_1 x_j \leqslant d_1 - a_1$ rather than the less restrictive implication $x_i = 0 \ \Rightarrow b_1 w_{ij} + c_1 x_j \leqslant d_1$).

In separation, the product relation (4) is treated similarly to product relations $w_{ij} \lesseqgtr x_i x_j$, with the linear left-hand side $Ax_i + Bw_{ij} + Cx_j + D$ being used instead of the individual auxiliary variable w_{ij}.

The detection algorithm searches for suitable pairs of linear relations and derives product relations from them. Let x_i, as before, be a binary variable. The following relation types are considered as candidates for the first relation in such a pair: implied relations of the form $x_i = \xi \ \Rightarrow \ \tilde{b}_1 w_{ij} + \tilde{c}_1 x_j \leqslant \tilde{d}_1$, where $\xi = 0$ or $\xi = 1$; and implied bounds of the form $x_i = \xi \ \Rightarrow \ w_{ij} \leqslant \tilde{d}_1$.

The second relation in a pair can be: an implied relation of the form $x_i = \overline{\xi} \ \Rightarrow \ \tilde{b}_2 w_{ij} + \tilde{c}_2 x_j \leqslant \tilde{d}_2$, where $\overline{\xi}$ is the complement of ξ; if w_{ij} is non-binary, an implied bound of the form $x_i = \overline{\xi} \ \Rightarrow \ w_{ij} \leqslant \tilde{d}_2$; if w_{ij} is binary, a clique containing the complement of x_i if $\xi = 1$ or x_i if $\xi = 0$, and w_{ij} or its complement; a constraint on x_j and w_{ij}; or a global bound on w_{ij}. Cliques are constraints of the form: $\sum_{k \in \mathcal{J}} x_k + \sum_{k \in \overline{\mathcal{J}}} (1 - x_k) \leqslant 1$, where $\mathcal{J} \subseteq \mathcal{I}^b$, $\overline{\mathcal{J}} \subseteq \mathcal{I}^b$ and $\mathcal{J} \cap \overline{\mathcal{J}} = \varnothing$.

4 Separation Algorithm

We present a new algorithm for separating RLT cuts within an LP-based branch-and-bound solver. The branch-and-bound algorithm builds LP relaxations of problem (1) by constructing linear underestimators of functions g in the constraint $g(\mathbf{x}, \mathbf{w}) \leqslant 0$ and McCormick inequalities for constraints (1d).

Let $(\mathbf{x}^*, \mathbf{w}^*)$ be the solution of an LP relaxation at a node of the branch-and-bound tree, and suppose that $(\mathbf{x}^*, \mathbf{w}^*)$ violates the relation $x_i x_j \lesseqgtr w_{ij}$ for some $i, j \in \mathcal{I}^w$. Separation algorithms generate cuts that separate $(\mathbf{x}^*, \mathbf{w}^*)$ from the feasible region, and add those cuts to the solver's cut storage.

The standard separation algorithm, which will serve as a baseline for comparisons, iterates over all linear constraints. For each constraint, it iterates over all variables x_j that participate in bilinear relations and generates RLT cuts using bound factors of x_j. Violated cuts are added to the MINLP solver's cut storage.

4.1 Row Marking

Let the bound factors be denoted as $f_j^{(\ell)}(\mathbf{x}) = x_j - \underline{x}_j$ and $f_j^{(u)}(\mathbf{x}) = \overline{x}_j - x_j$. Consider a linear constraint multiplied by a bound factor:

$$f_j^{(\cdot)}(\mathbf{x})\mathbf{a}_r\mathbf{x} \leqslant f_j^{(\cdot)}(\mathbf{x})b_r. \tag{8}$$

The ith nonlinear term is $a'_{ri}x_i x_j$, where $a'_{ri} = a_{ri}$ when multiplying by $(x_j - \underline{x}_j)$ and $a'_{ri} = -a_{ri}$ when multiplying by $(\overline{x}_j - x_j)$. Following the procedure described in Sect. 2, RLT may replace the product $x_i x_j$ with w_{ij}. The product can also be replaced with a linear expression, but this does not change the reasoning, and we will only use w_{ij} in this section.

If $w^*_{ij} \neq x^*_i x^*_j$, then such a replacement will change the violation of (8). The terms whose replacement will increase the violation are of interest, that is, the terms where:

$$a'_{ri}x^*_i x^*_j \leqslant a'_{ri}w^*_{ij}.$$

This determines the choice of bound factors to multiply with:

$$x^*_i x^*_j < w^*_{ij} \;\Rightarrow\; \begin{array}{l}\text{multiply by } (x_j - \underline{x}_j) \text{ if } a_{ri} > 0,\\ \text{multiply by } (\overline{x}_j - x_j) \text{ if } a_{ri} < 0,\end{array}$$

$$x^*_i x^*_j > w^*_{ij} \;\Rightarrow\; \begin{array}{l}\text{multiply by } (\overline{x}_j - x_j) \text{ if } a_{ri} > 0,\\ \text{multiply by } (x_j - \underline{x}_j) \text{ if } a_{ri} < 0.\end{array}$$

The separation algorithm is initialized by creating data structures to enable efficient access to 1) all variables appearing in bilinear products together with a given variable and 2) the bilinear product relation involving two given variables.

For each variable x_i, linear rows are marked in order to inform the separation algorithms which bound factors of x_i they should be multiplied with, if any. The algorithm can work with inequality rows in both '$\leqslant$' and '$\geqslant$' forms as well as equation rows. For each bilinear product $x_i x_j$, the row marking algorithm iterates over all linear rows that contain x_j with a nonzero coefficient. These rows are stored in a sparse array and have one of the following marks:

– MARK_LT: the row contains a term $a_{rj}x_j$ such that $a_{rj}x_i^*x_j^* < a_{rj}w_{ij}^*$;
– MARK_GT: the row contains a term $a_{rj}x_j$ such that $a_{rj}x_i^*x_j^* > a_{rj}w_{ij}^*$;
– MARK_BOTH: the row contains terms fitting both cases above.

Row marks are represented by integer values 1, 2 and 3, respectively, and are stored in two sparse arrays, *row_idcs* and *row_marks*, the first storing sorted row indices and the second storing the corresponding marks. In the algorithm below, we use the notation *mark(r)* to denote accessing the mark of row r by performing a search in *row_idcs* and retrieving the corresponding entry in *row_marks*. We also define a sparse matrix W with entries w_{ij}.

Input: x^*, w^*, W
$marks := \varnothing$
for $i \in \mathcal{I}^p, j \in nnz(\boldsymbol{w}_i)$ **do**
  for r *such that* $j \in nnz(\boldsymbol{a}_r)$ **do**
    if $r \notin marks$ **then**
      $marks \leftarrow r$
      $mark(r) := 0$
    if $a_{rj}x_i^*x_j^* < a_{rj}w_{ij}^*$ **then**
      $mark(r) \mid=$ MARK_LT
    else
      $mark(r) \mid=$ MARK_GT

The algorithm iterates over the sparse array of marked rows and generates RLT cuts for the following combinations of linear rows and bound factors:

– If $mark$ = MARK_LT, then "$\leqslant$" constraints are multiplied with the lower bound factor and "$\geqslant$" constraints are multiplied with the upper bound factor;
– If $mark$ = MARK_GT, then "$\leqslant$" constraints are multiplied with the upper bound factor and "$\geqslant$" constraints are multiplied with the lower bound factor;
– If $mark$ = MARK_BOTH, then both "$\leqslant$" and "$\geqslant$" constraints are multiplied with both the lower and the upper bound factors;
– Marked equality constraints are always multiplied with x_i itself.

4.2 Projection Filtering

If at least one of the variables x_i and x_j has a value equal to one of its bounds, then the McCormick relaxation (2) is tight for the relation $w_{ij} = x_ix_j$. Therefore, if x_i or x_j is at a bound and the McCormick inequalities are satisfied, then the product relation is also satisfied. We describe the equality case here, and the reasoning is analogous for the inequality case of $x_ix_j \lesseqgtr w_{ij}$.

Consider the linear system $A\mathbf{x} \leqslant \mathbf{b}$ projected onto the set of variables whose values are not equal to either of their bounds.

$$\sum_{k \in \mathcal{J}^1} a_{rk} x_k \leqslant b_r - \sum_{k \in \mathcal{J}^2} a_{rk} x_k^*, \ \forall r \in 1, \ldots, m^{(l)},$$

where $\mathcal{J}^1 \subseteq \mathcal{I}$ is the set of all problem variables whose values in the solution $\mathbf{x}^*$ of the current LP relaxation are not equal to one of their bounds, and $\mathcal{J}^2 = \mathcal{I} \setminus \mathcal{J}^1$.

Violation is then first checked for RLT cuts generated based on the projected linear system. Only if such a cut, which we will refer to as a projected RLT cut, is violated, then the RLT cut for the same bound factor and the corresponding constraint in the original linear system will be constructed. Since $\mathbf{x}^*$ is a basic LP solution, in practice either $x_k^* = \underline{x}_k$ or $x_k^* = \overline{x}_k$ holds for many of the variables, and the projected system often has a considerably smaller size than the original system.

In the projected system multiplied with a bound factor $f_j^{(.)}(\mathbf{x})$:

$$f_j^{(.)}(\mathbf{x}) \cdot \sum_{k \in \mathcal{J}^1} a_{rk} x_k \leqslant f_j^{(.)}(\mathbf{x})(b_r - \sum_{k \in \mathcal{J}^2} a_{rk} x_k^*), \ \forall r \in 1, \ldots, m^{(l)},$$

the only nonlinear terms are $x_j x_k$ with $k \in \mathcal{J}^1$, and therefore, no substitution $x_i x_k \rightarrow w_{ik}$ is performed for $k \in \mathcal{J}^2$. If the McCormick inequalities for x_i, x_k and w_{ik} hold, then $x_i^* x_k^* = w_{ik}^*$ for $k \in \mathcal{J}^2$, and checking the violation of a projected RLT cut is equivalent to checking the violation of a full RLT cut.

Depending on the solver, McCormick inequalities may not be satisfied at $(\mathbf{x}^*, \mathbf{w}^*)$. Thus, it is possible that $x_i^* x_k^* \neq w_{ik}^*$ for some $k \in \mathcal{J}^2$, but these violations will not contribute to the violation of the projected RLT cut. In this case, projection filtering has an additional effect: for violated bilinear products involving variables whose values in $\mathbf{x}^*$ are at bound, the violation of the product will be disregarded when checking the violation of RLT cuts. Thus, adding McCormick cuts will be prioritized over adding RLT cuts.

5 Computational Results

5.1 Setup

We tested the proposed methods on the MINLPLib[1] [6] test set and a test set comprised of instances from MIPLIB3, MIPLIB 2003, 2010 and 2017 [10], and Cor@l [15]. These test sets consist of 1846 MINLP instances and 666 MILP instances, respectively. After structure detection experiments, only those instances were chosen for performance evaluations that either contain bilinear products in the problem formulation, or where our algorithm derived bilinear products. This resulted in test sets of 1357 MINLP instances and 195 MILP instances.

[1] https://www.minlplib.org.

The algorithms were implemented in the MINLP solver SCIP [4]. We used a development branch of SCIP (githash `dd6c54a9d7`) compiled with SoPlex 5.0.2.4, CppAD 20180000.0, PaPILO 1.0.0.1, bliss 0.73p and Ipopt 3.12.11. The experiments were carried out on a cluster of Dell Poweredge M620 blades with 2.50GHz Intel Xeon CPU E5-2670 v2 CPUs, with 2 CPUs and 64GB memory per node. The time limit was set to one hour, the optimality gap tolerance to 10^{-4} for MINLP instances and to 10^{-6} for MILP instances, and the following settings were used for all runs, where applicable:

- The maximum number of unknown bilinear terms that a product of a row and a bound factor can have in order to be used was 20. Unknown bilinear terms are those terms $x_i x_j$ for which no w_{ij} variable exists in the problem, or its extended formulation which SCIP constructs for the purposes of creating an LP relaxation of an MINLP.
- RLT cut separation was called every 10 nodes of the branch-and-bound tree.
- In every non-root node where separation was called, 1 round of separation was performed. In the root node, 10 separation rounds were performed.
- Unless specified otherwise, implicit product detection and projection filtering were enabled and the new separation algorithm was used.

5.2 Impact of RLT Cuts

In this subsection we evaluate the performance impact of RLT cuts. The following settings were used: *Off* - RLT cuts are disabled; *ERLT* - RLT cuts are added for products that exist explicitly in the problem; *IERLT* - RLT cuts are added for both implicit and explicit products. The setting *ERLT* was used for the MINLP test set only, since MILP instances contain no explicitly defined bilinear products.

We report overall numbers of instances, numbers of solved instances, shifted geometric means of the runtime (shift 1 s), and the number of nodes in the branch-and-bound tree (shift 100 nodes), and relative differences between settings. Additionally, we report results on subsets of instances. Affected instances are instances where a change of setting leads to a difference in the solving process, indicated by a difference in the number of LP iterations. [x,timelim] denotes the subset of instances which took the solver at least x seconds to solve with at least one setting, and were solved to optimality with at least one setting. All-optimal is the subset of instances which were solved to optimality with both settings.

Table 1 shows the impact of RLT cuts on MILP performance. We observe a slight increase in time when RLT cuts are enabled, and a slight decrease in number of nodes. The difference is more pronounced on 'difficult' instances: a 9% decrease in number of nodes on subset [100,timelim] and 28% on subset [1000,timelim], and a decrease of 21% in the mean time on subset [1000,timelim].

Table 2 reports the impact of RLT cuts derived from explicitly defined bilinear products. A substantial decrease in running times and tree sizes is observed across all subsets, with a 15% decrease in the mean time and a 19% decrease in

Table 1. Impact of RLT cuts: MILP instances

		Off			*IERLT*			*IERLT* / *Off*	
Subset	instances	solved	time	nodes	solved	time	nodes	time	nodes
All	971	905	45.2	1339	909	46.7	1310	1.03	0.98
Affected	581	571	48.8	1936	575	51.2	1877	1.05	0.97
[0,tilim]	915	905	34.4	1127	909	35.6	1104	1.04	0.98
[1,tilim]	832	822	47.2	1451	826	49.0	1420	1.04	0.98
[10,tilim]	590	580	126.8	3604	584	133.9	3495	1.06	0.97
[100,tilim]	329	319	439.1	9121	323	430.7	8333	0.98	0.91
[1000,tilim]	96	88	1436.7	43060	92	1140.9	31104	0.79	0.72
All-optimal	899	899	31.9	1033	899	34.1	1053	1.07	1.02

the number of nodes on all instances, and a 87% decrease in the mean time and a 88% decrease in the number of nodes on the subset [1000,timelim]. 223 more instances are solved with *ERLT* than with *Off*.

Table 3 evaluates the impact of RLT cuts derived from implicit bilinear products. Similarly to MILP instances, the mean time slightly increases and the mean number of nodes slightly decreases when additional RLT cuts are enabled, but on MINLP instances, the increase in the mean time persists across different instance subsets and is most pronounced (9%) on the subset [100,timelim], and the number of nodes increases by $6-7\%$ on subsets [100,timelim] and [1000,timelim].

Table 2. Impact of RLT cuts derived from explicit products: MINLP instances

		Off			*ERLT*			*ERLT* / *Off*	
Subset	instances	solved	time	nodes	solved	time	nodes	time	nodes
All	6622	4434	67.5	3375	4557	57.5	2719	0.85	0.81
Affected	2018	1884	18.5	1534	2007	10.6	3375	0.57	0.51
[0,timelim]	4568	4434	10.5	778	4557	8.2	569	0.78	0.73
[1,timelim]	3124	2990	28.3	2081	3113	20.0	1383	0.71	0.67
[10,timelim]	1871	1737	108.3	6729	1860	63.6	3745	0.59	0.56
[100,tilim]	861	727	519.7	35991	850	196.1	12873	0.38	0.36
[1000,tilim]	284	150	2354.8	196466	273	297.6	23541	0.13	0.12
All-optimal	4423	4423	8.6	627	4423	7.5	518	0.87	0.83

Table 3. Impact of RLT cuts derived from implicit products: MINLP instances

		ERLT			IERLT			*ERLT*/*IERLT*	
Subset	instances	solved	time	nodes	solved	time	nodes	time	nodes
All	6622	4565	57.0	2686	4568	57.4	2638	1.01	0.98
Affected	1738	1702	24.2	1567	1705	24.8	1494	1.02	0.95
[0,timelim]	4601	4565	8.5	587	4568	8.6	576	1.01	0.98
[1,timelim]	3141	3105	21.1	1436	3108	21.4	1398	1.01	0.97
[10,timelim]	1828	1792	74.1	4157	1795	75.4	4012	1.02	0.97
[100,tilim]	706	670	359.9	22875	673	390.4	24339	1.09	1.06
[1000,tilim]	192	156	1493.3	99996	159	1544.7	107006	1.03	1.07
All-optimal	4532	4532	7.7	540	4532	7.8	529	1.02	0.98

Table 4 reports numbers of instances for which a change in the root node dual bound was observed, where the relative difference is quantified as $\frac{\gamma_2-\gamma_1}{\gamma_1}$, where γ_1 and γ_2 are root node dual bounds obtained with the first and second settings, respectively. The range of the change is specified in the column 'Difference', and each column shows numbers of instances for which one or the other setting provided a better dual bound, within given range.

The results of comparisons *Off*/*IERLT* for MILP instances and *Off*/*ERLT* for MINLP instances are consistent with the effect of RLT cuts on performance observed in Tables 1 and 2. Interestingly, *IERLT* performs better than *ERLT* in terms of root node dual bound quality. Thus, RLT cuts derived from implicit products in MINLP instances tend to improve root node relaxations.

5.3 Separation

In Table 5, the setting *Marking-off* employs the standard separation algorithm, and *Marking-on* enables the row marking and projection filtering algorithms described in Sect. 4. Row marking reduces the running time by 63% on MILP instances, by 70% on affected MILP instances, by 12% on MINLP instances and by 22% on affected MINLP instances. The number of nodes increases when row marking is enabled because, due to the decreased separation time, the solver can

Table 4. Root node dual bound differences

	MILP	MINLP	
Difference	*Off* / *IERLT*	*Off* / *ERLT*	*ERLT* / *IERLT*
0.01-0.2	54 / 62	224 / 505	379 / 441
0.2-0.5	2 / 4	23 / 114	44 / 48
0.5-1.0	0 / 3	40 / 150	19 / 30
>1.0	0 / 2	4 / 182	4 / 23

explore more nodes before reaching the time limit: this is confirmed by the fact that on the subset All-optimal, the number of nodes remains nearly unchanged.

Table 5. Separation algorithm comparison

			Marking-off			*Marking-on*			*M-on/M-off*	
Test set	subset	instances	solved	time	nodes	solved	time	nodes	time	nodes
MILP	All	949	780	124.0	952	890	45.2	1297	0.37	1.37
	Affected	728	612	156.6	1118	722	46.4	1467	0.30	1.31
	All-optimal	774	774	58.4	823	774	21.2	829	0.36	1.01
MINLP	All	6546	4491	64.5	2317	4530	56.4	2589	0.88	1.12
	Affected	3031	2949	18.5	1062	2988	14.3	1116	0.78	1.05
	All-optimal	4448	4448	9.1	494	4448	7.4	502	0.81	1.02

Table 6 analyzes the percentage of time that RLT cut separation takes out of overall running time, showing the arithmetic mean and maximum over all instances, numbers of instances for which the percentage was within a given interval, and numbers of failures. The average percentage is reduced from 54.2% to 2.8% for MILP instances and from 15.1% to 2.4% for MINLP instances, and the maximum percentage is reduced from 99.6% to 71.6% for MILP instances, but remains at 100% for MINLP instances. The numbers of failures are reduced with *Marking-on*, mainly due to avoiding failures that occur when the solver runs out of memory.

Table 6. Separation times

Test set	Setting	avg %	max %	N(< 5%)	N(5-20%)	N(20-50%)	N(50-100%)	fail
MILP	*Marking-off*	54.2	99.6	121	117	169	552	16
	Marking-on	2.8	71.6	853	87	31	4	0
MINLP	*Marking-off*	15.1	100.0	3647	1265	1111	685	77
	Marking-on	2.4	100.0	6140	376	204	49	16

Projection filtering has a minor impact on performance. When comparing the runs where projection filtering is disabled and enabled, the relative difference in time and nodes does not exceed 1% on both MILP and MINLP instances, except for affected MILP instances where projection filtering decreases the number of nodes by 4%. This is possibly occurring due to the effect of prioritizing McCormick inequalities to RLT cuts when enforcing derived product relations. The number of solved instances remains almost unchanged, with one less instance being solved on both MILP and MINLP test sets when projection filtering is enabled.

5.4 Experiments with Gurobi

In this subsection we present results obtained by running the mixed-integer quadratically-constrained programming solver Gurobi 10.0 beta [11]. The algorithms for implicit product detection and RLT cut separation are the same as in SCIP, although implementation details may differ between the solvers.

The internal Gurobi test set was used, comprised of models sent by Gurobi customers and models from public benchmarks, chosen in a way that avoids overrepresenting any particular problem class. Whenever RLT cuts were enabled, so was implicit product detection, row marking and projection filtering. The time limit was set to 10000 s.

Table 7 shows, for both MILP and MINLP test sets, the numbers of instances in the test sets and their subsets, and the ratios of shifted geometric means of running time and number of nodes of the runs with RLT cuts enabled, to the same means obtained with RLT cuts disabled. The last row shows the numbers of instances solved with one setting and unsolved with the other, that is, for example, "RLT off: +41" means that 41 instances were solved with the setting "off" that were not solved with the setting "on".

While the results cannot be directly compared to those obtained with SCIP due to the differences in the experimental setup, we observe the same tendencies. In particular, RLT cuts yield small improvements on MILP instances which become more pronounced on subsets [100,timelim] and [1000,timelim], and larger improvements are observed on MINLP instances both in terms of geometric means and numbers of solved instances. Relative differences are comparable to those observed with SCIP, but the impact of RLT cuts is larger in Gurobi, and no slowdown is observed with Gurobi on any subset of MILP instances.

Table 7. Results obtained with Gurobi 10.0 beta

	MILP			MINLP		
Subset	instances	timeR	nodeR	instances	timeR	nodeR
All	5011	0.99	0.97	806	0.73	0.57
[0,timelim]	4830	0.99	0.96	505	0.57	0.44
[1,timelim]	3332	0.98	0.96	280	0.40	0.29
[10,timelim]	2410	0.97	0.93	188	0.29	0.20
[100,timelim]	1391	0.95	0.91	114	0.17	0.11
[1000,timelim]	512	0.89	0.83	79	0.12	0.08
Solved	RLT off: +41; RLT on: +37			RLT off: +2; RLT on: +35		

5.5 Summary

RLT cuts yield a considerable performance improvement for MINLP problems and a small performance improvement for MILP problems which becomes more

pronounced for challenging instances. The new separation algorithm drastically reduces the computational burden of RLT cut separation and is essential to an efficient implementation of RLT cuts, enabling the speedups we observed when activating RLT.

Acknowledgements. The work for this article has been conducted within the Research Campus Modal funded by the German Federal Ministry of Education and Research (BMBF grant numbers 05M14ZAM, 05M20ZBM).

References

1. Adams, W.P., Sherali, H.D.: A tight linearization and an algorithm for zero-one quadratic programming problems. Manage. Sci. **32**(10), 1274–1290 (1986)
2. Adams, W.P., Sherali, H.D.: Linearization strategies for a class of zero-one mixed integer programming problems. Oper. Res. **38**(2), 217–226 (1990)
3. Adams, W.P., Sherali, H.D.: Mixed-integer bilinear programming problems. Math. Program. **59**(1), 279–305 (1993)
4. Bestuzheva, K., et al.: Enabling research through the SCIP optimization suite 8.0. ACM Trans. Math. Softw. (2023). https://doi.org/10.1145/3585516
5. Buchheim, C., Wiegele, A., Zheng, L.: Exact algorithms for the quadratic linear ordering problem. INFORMS J. Comput. **22**(1), 168–177 (2010)
6. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib - a collection of test models for mixed-integer nonlinear programming. INFORMS J. Comput. **15**(1), 114–119 (2003). https://doi.org/10.1287/ijoc.15.1.114.15159
7. Castillo, I., Westerlund, J., Emet, S., Westerlund, T.: Optimization of block layout design problems with unequal areas: a comparison of MILP and MINLP optimization methods. Comput. Chem. Eng. **30**(1), 54–69 (2005)
8. Frank, S., Steponavice, I., Rebennack, S.: Optimal power flow: a bibliographic survey I. Energy syst. **3**(3), 221–258 (2012)
9. Frank, S., Steponavice, I., Rebennack, S.: Optimal power flow: a bibliographic survey II. Energy Syst. **3**(3), 259–289 (2012)
10. Gleixner, A., et al.: MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. Math. Program. Comput. **13**(3), 443–490 (2021). https://doi.org/10.1007/s12532-020-00194-3
11. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022). https://www.gurobi.com
12. Liberti, L.: Reduction constraints for the global optimization of NLPs. Int. Trans. Oper. Res. **11**(1), 33–41 (2004)
13. Liberti, L.: Reformulation and convex relaxation techniques for global optimization. Ph.D. thesis. Springer (2004)
14. Liberti, L.: Linearity embedded in nonconvex programs. J. Global Optim. **33**(2), 157–196 (2005)
15. Linderoth, J.T., Ralphs, T.K.: Noncommercial software for mixed-integer linear programming. Integer Programm. Theory Practice **3**, 253–303 (2005)
16. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I - convex underestimating problems. Math. Program. **10**(1), 147–175 (1976)
17. Misener, R., Floudas, C.A.: Advances for the pooling problem: modeling, global optimization, and computational studies. Appl. Comput. Math. **8**(1), 3–22 (2009)

18. Sherali, H.D., Adams, W.P.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J. Discret. Math. **3**(3), 411–430 (1990)
19. Sherali, H.D., Adams, W.P.: A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. Discret. Appl. Math. **52**(1), 83–106 (1994)
20. Sherali, H.D., Adams, W.P.: A reformulation-linearization technique (RLT) for semi-infinite and convex programs under mixed 0–1 and general discrete restrictions. Discret. Appl. Math. **157**(6), 1319–1333 (2009)
21. Sherali, H.D., Alameddine, A.: A new reformulation-linearization technique for bilinear programming problems. J. Global Optim. **2**(4), 379–410 (1992)
22. Sherali, H.D., Dalkiran, E., Liberti, L.: Reduced RLT representations for nonconvex polynomial programming problems. J. Global Optim. **52**(3), 447–469 (2012)
23. Sherali, H.D., Smith, J.C., Adams, W.P.: Reduced first-level representations via the reformulation-linearization technique: results, counterexamples, and computations. Discret. Appl. Math. **101**(1–3), 247–267 (2000)
24. Sherali, H.D., Tuncbilek, C.H.: A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. J. Global Optim. **2**(1), 101–112 (1992)
25. Sherali, H.D., Tuncbilek, C.H.: New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. Oper. Res. Lett. **21**(1), 1–9 (1997)