J Glob Optim (2012) 52:447–469
DOI 10.1007/s10898-011-9757-3

Reduced RLT representations for nonconvex polynomial programming problems

Hanif D. Sherali · Evrim Dalkiran · Leo Liberti

Received: 1 July 2011 / Accepted: 20 July 2011 / Published online: 31 July 2011

Abstract This paper explores equivalent, reduced size Reformulation-Linearization Technique (RLT)-based formulations for polynomial programming problems. Utilizing a basis partitioning scheme for an embedded linear equality subsystem, we show that a strict subset of RLT defining equalities imply the remaining ones. Applying this result, we derive significantly reduced RLT representations and develop certain coherent associated branching rules that assure convergence to a global optimum, along with static as well as dynamic basis selection strategies to implement the proposed procedure. In addition, we enhance the RLT relaxations with v-semidefinite cuts, which are empirically shown to further improve the relative performance of the reduced RLT method over the usual RLT approach. We present computational results for randomly generated instances to test the different proposed reduction strategies and to demonstrate the improvement in overall computational effort when such reduced RLT mechanisms are employed.

Keywords Reformulation-Linearization Technique (RLT) · Reduced basis techniques · Polynomial programs · Global optimization · Semidefinite cuts · BARON

1 Introduction

The *Reformulation-Linearization Technique* **(RLT)** offers a unified framework for solving nonconvex discrete and continuous optimization problems [14]. In this paper, we focus on applying the RLT to general polynomial programming problems, and explore the generation

H. D. Sherali · E. Dalkiran (✉)
Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA 24061, USA
e-mail: dalkiran@vt.edu

H. D. Sherali
e-mail: hanifs@vt.edu

L. Liberti
CNRS LIX, École Polytechnique, 91128 Palaiseau, France
e-mail: liberti@lix.polytechnique.fr

of equivalent, reduced size RLT representations, augmented with additional valid inequalities, in order to enhance the solvability of such problems via a branch-and-bound algorithm. Accordingly, consider the following polynomial program **PP** of order $\delta \geq 2$, where notationally, given any set $\mathscr{S} \subseteq \mathscr{N} \equiv \{1, \ldots, n\}$, we let $\mathscr{S}^d$ denote a *multi-set of order d*, which is comprised of distinct combinations of indices (arranged in nondecreasing order) that belong to the Cartesian product $\mathscr{S} \times \cdots \times \mathscr{S}$, where the latter involves d repetitions of $\mathscr{S}$. Hence, in particular, $\mathscr{S}^1 \equiv \mathscr{S}$.

$$\textbf{PP}: \text{Minimize } \phi_0(x) \tag{1a}$$

subject to

$$\phi_r(x) \geq \beta_r, \quad \forall r = 1, \ldots, R_1 \tag{1b}$$

$$\phi_r(x) = \beta_r, \quad \forall r = R_1 + 1, \ldots, R \tag{1c}$$

$$Ax = b \tag{1d}$$

$$x \in \Omega \equiv \{0 \leq l_j \leq x_j \leq u_j < \infty, \quad \forall j \in \mathscr{N}\}, \tag{1e}$$

where

$$\phi_r(x) \equiv \sum_{t \in T_r} \alpha_{rt} \Big[\prod_{j \in J_{rt}} x_j \Big], \text{ for } r = 0, \ldots, R,$$

and where T_r is an index set for the terms defining $\phi_r(.)$, with α_{rt} being real coefficients associated with the monomials $\prod_{j \in J_{rt}} x_j$, $\forall t \in T_r, r = 0, \ldots, R$, where each $J_{rt} \subseteq \cup_{d=1}^{\delta} \mathscr{N}^d$. Here, we have especially identified the presence of a linear equality system (1d), where A is $m \times n$ of rank $m < n$, and where (1c) then accommodates any other nonlinear equality restrictions defining the model formulation.

Following the RLT procedure described in [19], given the lower and upper bounds in (1e), we define the *bound-factors* $(x_j - l_j) \geq 0$ and $(u_j - x_j) \geq 0, \forall j \in \mathscr{N}$, and generate the corresponding *bound-factor product (RLT) constraints* composed by taking the products of the bound-factors δ at a time (including repetitions) as follows:

$$\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \geq 0, \quad \forall (J_1 \cup J_2) \subseteq \mathscr{N}^{\delta}. \tag{2}$$

These implied $\binom{2n+\delta-1}{\delta}$ restrictions are then appended to Problem PP, along with any other (optionally generated) RLT constraints obtained by multiplying the original inequality constraints in (1b) with themselves as well as with bound-factors, while preserving the degree of the resulting augmented polynomial program as δ. Similarly, the equality constraints (1c) and (1d) can be utilized to generate RLT constraints by multiplying each of them with distinct monomials of the form $\prod_{j \in J} x_j$ such that the resulting restrictions are of degree less than or equal to δ. It is worthwhile noting that whereas appending Problem PP with such RLT constraints other than (2) potentially tightens the relaxation, they are not required for establishing the convergence of the algorithm proposed in [19] to solve Problem PP to global optimality. Moreover, we limit the generation of RLT polynomial constraints to degree δ only to curtail the size of the resulting relaxation, although higher order restrictions can further tighten the derived representation.

Next, we linearize the augmented polynomial program by replacing each distinct monomial $\prod_{j \in J} x_j$ with a new (RLT) variable w_J, for $J \subseteq \cup_{d=2}^{\delta} \mathscr{N}^d$. Note that this produces $\binom{n+\delta}{\delta} - (n+1)$ such additional so-called *RLT variables*. The corresponding RLT-based *linear programming* (LP) *relaxation* is constructed via this substitution process upon dropping

from the formulation the defining identities $w_J = \prod_{j \in J} x_j, \forall J$. At each node of the branch-and-bound tree, a lower bound is thus computed via the RLT-based LP relaxation associated with the revised sub-hyperrectangle Ω that defines the current node problem. Accordingly, at the top of the loop in this branch-and-bound process, we select a node having the least lower bound for further partitioning. Denoting $(\bar{x}, \bar{w})$ as the LP relaxation solution obtained for this node, we select a branching variable index as

$$j^* \in \arg\max_{j \in \mathcal{N}} \theta_j,$$

where θ_j is computed as follows:

$$\theta_j \equiv \sum_{d=1}^{\delta-1} \sum_{J \in \mathcal{N}^d} \left| \bar{w}_{J \cup j} - \bar{w}_J \bar{x}_j \right|, \quad \forall j \in \mathcal{N}.$$

We then partition the selected interval $[l_{j^*}, u_{j^*}]$ at the current LP relaxation solution value $\bar{x}_{j^*}$ according to the dichotomy that $\{x_{j^*} \in [l_{j^*}, \bar{x}_{j^*}]\} \vee \{x_{j^*} \in [\bar{x}_{j^*}, u_{j^*}]\}$ provided that $\min\{\bar{x}_{j^*} - l_{j^*}, u_{j^*} - \bar{x}_{j^*}\} \geq 0.05(u_{j^*} - l_{j^*})$; else, we split the selected interval at its mid-point. This particular choice of branching variable selection and partitioning strategy preserves the arguments of the convergence proof presented in [19]. Whenever the lower bound LB computed for any node subproblem satisfies $(v^* - LB) \leq \varepsilon |v^*|$, where v^* is the incumbent solution value and $\varepsilon \geq 0$ is a specified optimality gap tolerance, we fathom the corresponding node. The branch-and-bound algorithm terminates when no active node exists.

Notwithstanding the standard RLT constraint generation process, the resulting LP relaxation can be further tightened by utilizing *v-semidefinite cuts* (or simply *SDP cuts*) as described in [16,17]. These cuts are generated by imposing positive semidefinite restrictions on dyadic variable-product matrices composed as $M = [vv^t]_L \succeq 0$, where $[.]_L$ denotes the linearization of [.] via the RLT variable substitution process, and where $v \equiv [1, \{\prod_{j \in J} x_j : J \subseteq \mathcal{J}\}]$ for certain appropriately defined sets $\mathcal{J}$ involving selected monomials of order up to $\lfloor \delta/2 \rfloor$ (see [16]). In this strategy, having solved an RLT-based LP relaxation, and letting $\bar{M}$ denote the matrix M evaluated at the resulting solution, in case $\bar{M}$ is not positive semidefinite, a suitable unit-norm vector $\bar{\gamma}$ is derived (in polynomial time) having $\bar{\gamma}^T \bar{M} \bar{\gamma} < 0$, which yields an associated *v-semidefinite cut* $[(\bar{\gamma}^T v)^2]_L \geq 0$. Incorporating particular classes of such cuts can significantly enhance the performance of the RLT-based algorithm for solving polynomial programming programs as demonstrated in [16].

The present paper makes the following specific contributions. First, we introduce equivalent, reduced size RLT formulations for polynomial programming problems that *directly* utilize bases of the matrix A as opposed to bases of substantially larger companion systems as adopted in [8] and [2] (see the literature review in Sect. 2). Second, we design and test several static and dynamic basis selection methods to implement the proposed reduction strategy. Third, we enhance the relaxations via v-semidefinite cuts along with the developed basis reduction mechanisms within a branch-and-bound algorithm to solve polynomial programming problems to global optimality, and we provide extensive computational results that demonstrate a significant performance improvement using the proposed approach.

The remainder of this paper is organized as follows. Section 2 provides a brief literature review on existing RLT constraint reduction strategies. To set the stage and facilitate the presentation, Sect. 3 first considers quadratic polynomial programming problems having an embedded linear equality system, and presents the proposed equivalent, reduced RLT representation for such problems. Section 4 then extends the theory to address more general

higher order polynomial programs, which is the main focus of this paper, and Sect. 5 describes several static and dynamic basis selection methods for constructing reduced RLT representations. Computational results are presented in Sect. 6, and Sect. 7 concludes the paper with a summary and recommendations for future research.

2 Literature review

The RLT methodology generates a hierarchy of increasingly tighter relaxations, which is, however, accompanied with an increase in problem size. Accordingly, this approach can be potentially enhanced by suitably imposing only a subset of certain identified key RLT constraints. In order to accelerate RLT-based algorithms for solving polynomial programming problems, certain constraint filtering techniques are proposed in [22] that select only a subset of the RLT bound-factor restrictions to incorporate within the model formulation based on the signs of the coefficients of the original monomials in the objective function or constraints. For example, in an inequality constraint (1b), if α_{rt} is positive (negative), then the associated RLT variable $w_{J_{rt}}$ might have a tendency to be larger (smaller) than $\prod_{j \in J_{rt}} x_j$ in the RLT-based LP relaxation, and so, restrictions that provide upper (lower) bounds on $w_{J_{rt}}$ are more likely to be relevant in tightening the relaxation. Rules of this type for retaining only such bound-factor product constraints along with certain additional restrictions that provide significant support for the monomials involved in the retained constraints are developed in [22]. However, for the sake of theoretical convergence, the algorithm described in [22] ultimately considers the entire set of bound-factor constraints (2). More recently, in this same vein, certain theoretical filtering strategies for bound-factor constraints have been devised and tested in [5], which guarantee convergence of the RLT-based branch-and-bound algorithm to a global optimal solution, while yet using only a subset of the bound-factor restrictions (2).

Another RLT constraint reduction strategy is explored in [18] for first-level RLT representations (RLT-1) of 0–1 linear mixed-integer programs. Employing optimal dual multipliers to the LP relaxation $\overline{\text{RLT-1}}$ of RLT-1, it is shown that there exists an *equivalent* representation using only one of each identified pair of RLT restrictions within RLT-1, which provides the same lower bound as that obtained via $\overline{\text{RLT-1}}$. Promising computational results are presented for a proposed constraint selection strategy that attempts to a priori predict such an existing equivalent reduced size RLT representation.

For quadratic or bilinear programs having an embedded linear equality system of the form $Ax = b$, where A is $m \times n$ of rank m, a different technique was developed in [8] to reduce the number of bilinear terms in the RLT formulation while maintaining an equivalent representation. Specifically, consider the RLT constraints obtained by multiplying the linear equality restrictions with the variable $x_k, \forall k \in \mathcal{N}$, to yield $Aw^k - bx_k = 0, \forall k \in \mathcal{N}$, where the RLT variable vector $w^k \equiv (w^k_j, \forall j \in \mathcal{N})$ represents the product terms $(x_k x_j, \forall j \in \mathcal{N})$. Accordingly, examining a so-called *companion system* $Az^k = 0, \forall k \in \mathcal{N}$, where $z^k_j \equiv w^k_j - x_k x_j, \forall j \in \mathcal{N}$, and where this system has a rank between $\frac{m(m+1)}{2}$ and $(n-m)m$, it was shown that such a companion system with rank r implies r of the defining RLT substitution identities, which can therefore be dropped from the formulation without weakening it. Based on this result, basic feasible solutions were explored to the companion system of size $mn \times n^2$ to determine which bilinear terms to explicitly retain in the formulation, where the latter correspond to the nonbasic variables for this expanded system. The efficiency and applicability of the foregoing method for solving sparse problems in which the number of w-variables might be considerably larger than the rank of the associated companion system was also

discussed in [8]. To handle such cases more effectively, certain graph theoretical algorithms were introduced in [10] to select valid sets of RLT constraints that contain a reduced number of w-variables. For quadratic programming problems, an algorithm was proposed in [9] to construct a basis for the companion system by selecting a set of variables that maximizes the convexity gap, where for each RLT variable, the convexity gap was formulated as the area (or volume) between the corresponding RLT substitution constraint and the associated bound-factor product constraints. The concepts developed in [8,9] were further extended in [2] to address polynomial programming programs, wherein each linear equality restriction was multiplied with monomials up to degree $\delta - 1$ in order to generate a similar companion system of size $m[\binom{n+\delta-1}{\delta-1} - 1] \times [\binom{n+\delta}{\delta} - (n+1)]$, and then a basis of this companion system was utilized to derive a reduced RLT representation. A technique for deriving a convexity gap estimate for each candidate basis was also proposed, where the recommended procedure then selected a basis having the maximum such value. Using randomly generated continuous quadratic knapsack problems having up to 10 variables, it was demonstrated that the computational effort decreased substantially when using the reduced RLT approach in lieu of the full RLT representation in six out of the nine test cases.

We now proceed to present our proposed reduced RLT representations, which directly utilize bases of the matrix A itself, as opposed to bases of some expanded companion systems as in [2,8–10].

3 Quadratic polynomial programming problems

Consider the following nonconvex quadratic polynomial programming problem having an embedded equality subsystem:

$$\mathbf{P}: \text{Minimize } c_0^T x + x^T H_0 x \tag{3a}$$

subject to

$$c_i^T x + x^T H_i x \le h_i, \quad \forall i = 1, \ldots, Q \tag{3b}$$

$$Ax = b \tag{3c}$$

$$0 \le l_j \le x_j \le u_j < \infty, \quad \forall j \in \mathcal{N} \equiv \{1, \ldots, n\}, \tag{3d}$$

where A is $m \times n$ of rank $m < n$, and where $c_i \in \mathbb{R}^n$ and $H_i \in \mathbb{R}^{n \times n}, \forall i = 0, 1, \ldots, Q$. Problem P arises in several applications and is of interest in its own right (see [6]), but such a representation might also be an equivalent quadratic reformulation of a more general polynomial program (see [23] and [21]).

Given a basis B of A for the linear system of Eq. 3c, we adopt the familiar partitioning of x into basic and nonbasic variables x_B and x_N, respectively, to write (3c) in the form:

$$Bx_B + Nx_N = b, \tag{4}$$

where B represents the columns of the x_B-variables, and N represents the columns of the x_N-variables. Let J_B and J_N denote the respective index sets of the basic and nonbasic variables, where $J_B \cup J_N = \mathcal{N}$.

Now, following the RLT process, we multiply (4) by x_j for each $j \in \mathcal{N}$, and linearize the resulting system by using the substitution

$$w_{ij} = x_i x_j, \quad \forall i, j \in \mathcal{N} \quad \text{with} \quad i \le j, \tag{5}$$

where $w \equiv (w_{ij}) \in \mathbb{R}^{n(n+1)/2}$. We denote by $[.]_L$ the linearization of $[.]$ under (5). Also, for the sake of ease in presentation, we denote

$$w_{(ij)} \equiv \begin{cases} w_{ij} & \text{if} i \leq j \\ w_{ji} & \text{if} i > j \ , \end{cases} \tag{6a}$$

where the subscript notation (ij) is assumed to properly permute indices according to the definition of the w-variables in (5). Furthermore, let

$$w_{(Bj)} \equiv \left[x_B x_j\right]_L \ , \quad w_{(Nj)} \equiv \left[x_N x_j\right]_L \ , \quad \text{and} \quad w_{(.j)} \equiv \left[x x_j\right]_L \ , \quad \forall j \in \mathcal{N}. \tag{6b}$$

Then, the foregoing RLT constraint multiplication and linearization operation yields the following system:

$$B w_{(Bj)} + N w_{(Nj)} = b x_j, \quad \forall j \in \mathcal{N}. \tag{7}$$

Proposition 1 *Let the system $Ax = b$ be partitioned according to (4) for any basis B of A, and define*

$$Z = \{(x, w) : (3c), (7), \quad \textit{and} \quad w_{ij} = x_i x_j, \quad \forall i, j \in J_N \quad \textit{with } i \leq j\}. \tag{8}$$

Then, we have (5) holding true for any $(x, w) \in Z$.

Proof Let $(x, w) \in Z$. Then, from (7) and (8), we get

$$B w_{(Bj)} = (b - N x_N) x_j, \quad \forall j \in J_N. \tag{9}$$

But (4) implies that $x_B = B^{-1}(b - N x_N)$, which together with (9) yields

$$w_{(Bj)} = x_B x_j, \quad \forall j \in J_N, \ \text{i.e.,} \ w_{(ij)} = x_i x_j, \quad \forall i \in J_B, \quad j \in J_N. \tag{10}$$

Furthermore, from (7) written for $j \in J_B$, and noting (10), we get that,

$$B w_{(Bj)} = (b - N x_N) x_j, \quad \forall j \in J_B. \tag{11}$$

Again, together with (4), this implies that

$$w_{(Bj)} = x_B x_j, \quad \forall j \in J_B, \ \text{i.e.,} \ w_{(ij)} = x_i x_j, \quad \forall i, j \ \in J_B. \tag{12}$$

The result now follows by observing (8), (10), and (12). □

Remark 1 Noting that $|J_N| = (n - m)$ from (3c) and (4), Proposition 1 asserts that subject to (3c) (or (4)) and (7), if we enforce (5) for just the $(n - m)(n - m + 1)/2$ identities corresponding to $i, j \in J_N$ with $i \leq j$, then the remaining identities in (5) will automatically hold true. This concept can be used to curtail RLT relaxations and branching decisions, and improves upon the strategy expounded in [8,9] and [10] by permitting a reformulation based *directly* on the selection of suitable bases B of A, rather than requiring the examination of bases of an enlarged $(mn \times n^2)$ companion system as developed in the latter works. In the same spirit, reduced RLT relaxations for polynomial programs employing bases of a linear companion system of size $m[\binom{n+\delta-1}{\delta-1} - 1] \times [\binom{n+\delta}{\delta} - (n+1)]$ are proposed in [2]. A more compact generalization of Proposition 1 to polynomial programming problems is likewise presented in Sect. 4 below.

For the sake of interest, we note that the following more general result holds true, although we shall focus on Proposition 1 to achieve computational expediency.

Proposition 2 *There exists a partitioning of $w \in \mathbb{R}^{n(n+1)/2}$ according to $w = (w_{Q_1}, w_{Q_2})$, where $w_{Q_1} \in \mathbb{R}^{m(n-m)+m(m+1)/2}$ and $w_{Q_2} \in \mathbb{R}^{(n-m)(n-m+1)/2}$ are such that the columns of w_{Q_1} in (7) are linearly independent. Moreover, for any such partitioning of w, any feasible solution to the system*

$$Ax = b, \quad Aw_{(.j)} = bx_j, \quad \forall j \in \mathcal{N}, \tag{13}$$

along with the identities (5) for the w_{Q_2}-variables, will also automatically have (5) holding true for the w_{Q_1}-variables.

Proof First of all, note that a partitioning of w as stated in the proposition exists by Proposition 1 and Remark 1, as for example, by selecting any basis B of A and letting w_{Q_1} and w_{Q_2} be respectively composed of the sets of variables $\{w_{(Bj)} \text{for} j \in J_N; w_{(Bj)} \text{for} j \in J_B\}$, and $\{w_{(Nj)} \text{for} j \in J_N\}$. Now, for any given feasible solution x to (3c), let the w_{Q_2}-variables be determined according to (5), and consider the resulting residual system (7) in the w_{Q_1}-variables. Since the columns of w_{Q_1} in (7) are linearly independent, this system either has a unique solution or has no solution. But by construction, fixing the w_{Q_1}-variables according to (5) yields a feasible solution to (7), and therefore this must be the unique completion of the solution to the residual system in (7) or (13). □

Now, defining the variables $w \in \mathbb{R}^{n(n+1)/2}$ according to (5) and adopting the notation (6), we can rewrite (3) as follows, where $g_i \in \mathbb{R}^{n(n+1)/2}$ for $i = 0, 1, \ldots, Q$ are appropriate vectors that represent the pure quadratic forms appearing in (3a, 3b), and where we have included the RLT constraints (13) obtained by multiplying $Ax = b$ with $x_j, \forall j \in \mathcal{N}$, within (14d), as well as the RLT bound-factor constraints (2) within (14e). (Note that the algebraic notation in (14e) represents the quadruple bound-factor product relationships $[(x_i - l_i)(x_j - l_j)]_L \geq 0$, $[(x_i - l_i)(u_j - x_j)]_L \geq 0$, $[(u_i - x_i)(x_j - l_j)]_L \geq 0$, and $[(u_i - x_i)(u_j - x_j)]_L \geq 0$ linearized under (5), $\forall i, j \in \mathcal{N}$ with $i \leq j$, where it is understood that for $i = j$, we retain only one of the duplicating second and third product restrictions.)

$$\textbf{P1}: \text{Minimize } c_0^T x + g_0^T w \tag{14a}$$

subject to

$$c_i^T x + g_i^T w \leq h_i, \quad \forall i = 1, \ldots, Q \tag{14b}$$

$$Ax = b \tag{14c}$$

$$Aw_{(.j)} = bx_j, \quad \forall j \in \mathcal{N} \tag{14d}$$

$$\left[(l_i \leq x_i \leq u_i) * (l_j \leq x_j \leq u_j)\right]_L \geq 0, \quad \forall i,\ j \in \mathcal{N} \quad \text{with } i \leq j \tag{14e}$$

$$l \leq x \leq u \tag{14f}$$

$$w_{ij} = x_i x_j, \quad \forall i, j \in \mathcal{N} \quad \text{with } i \leq j. \tag{14g}$$

Next, following the concept discussed in Proposition 1, we partition $Ax = b$ according to (4), and correspondingly derive a reduced form of Problem P1 as stated below by enforcing (14g) only for $\forall i, j \in J_N$ with $i \leq j$, and also retaining (14e) corresponding to only these same indices $\forall i, j \in J_N$ with $i \leq j$, and where we have additionally included the implied bounding constraints $l_i l_j \leq w_{ij} \leq u_i u_j, \forall i, j \in \mathcal{N}$ with $i \leq j$ in (15f) for tightening the underlying LP relaxation obtained upon deleting (15g).

P2 : Minimize $c_0^T x + g_0^T w$ (15a)

subject to

$$c_i^T x + g_i^T w \le h_i, \ \forall i = 1, \ldots, Q \tag{15b}$$

$$Ax = b \tag{15c}$$

$$Bw_{(Bj)} + Nw_{(Nj)} = bx_j, \quad \forall j \in \mathcal{N} \tag{15d}$$

$$\left[(l_i \le x_i \le u_i) * (l_j \le x_j \le u_j)\right]_L \ge 0, \quad \forall i, j \in J_N \ \text{ with } \ i \le j \tag{15e}$$

$$l \le x \le u, \ l_i l_j \le w_{ij} \le u_i u_j, \quad \forall i, j \in \mathcal{N} \ \text{ with } \ i \le j \tag{15f}$$

$$w_{ij} = x_i x_j, \quad \forall i, j \in J_N \text{ with } i \le j. \tag{15g}$$

Remark 2 Note that the equality constraints of type $Ax = b$ can also be used to directly eliminate the basic variables x_B for any given basis B via the substitution $x_B = B^{-1}(b - Nx_N)$, and we can then correspondingly apply the regular RLT process described in [19] to the resulting problem in the space of the $n - m$ nonbasic variables. In this context, the RLT procedure is invariant under such affine transformations as shown in [20]. However, a potential drawback of implementing this transformation is that the sparse nonlinear constraints (as well as nonlinear objective terms) may possibly become dense and the resulting linear programming relaxations might become relatively more difficult to solve. In our computations, we compare the relative efforts for solving problems using the proposed reduced RLT procedure as well as by applying RLT to the reduced nonbasic variable space problem representation.

Let RLT(P1) denote the RLT-based branch-and-bound algorithm described in [19,20] as applied to Problem P1, where lower bounds are computed via the LP relaxation (14a–14f), and where partitioning is performed on the hyperrectangle defined by (14f). In addition, denote by RLT_{SDP}(P1) the foregoing solution process in which we also incorporate semidefinite programming (SDP)-based cuts as in [16,17]. Likewise, define RLT(P2) and RLT_{SDP}(P2) with respect to Problem P2, where in particular, the branching is now performed only on the sub-hyperrectangle defined by $l_j \le x_j \le u_j, \forall j \in J_N$. By Proposition 1 and the theory expounded in [19], we are assured (infinite) convergence to a global optimum by this partial partitioning process.

Remark 3 Note that the linear programming relaxation $\overline{\text{P2}}$ of Problem P2 can be tightened by including (14e) in lieu of the constraints (15e) and the implied bound restrictions in (15f). This would be similar to solving P1 itself via an RLT approach, but while partitioning on just the nonbasic variable intervals. We refer to this strategy as RLT^{NB}(P1) and $\text{RLT}^{\text{NB}}_{\text{SDP}}$(P1), respectively implemented without and with SDP cuts. Moreover, as alluded earlier, we mention (but do not adopt in our implementation) that we can further strengthen $\overline{\text{P2}}$ by incorporating RLT bound-factor product constraints of order $\delta' > \delta$; in this context, note that the constraints in (15e) are implied by bound-factor constraints generated via all $(J_1 \cup J_2) \subseteq J_N^{\delta'}$ as proven in [19]. However, this can significantly increase the size of the resulting relaxations and needs a careful design of additional filtering mechanisms. The identities $x_B = B^{-1}b - B^{-1}Nx_N$ can be used to further tighten the bounds on the basic variables, x_B, and hence the implied bounds on the w-variables in (15f), by consecutively minimizing and maximizing $B_i^{-1}b - B_i^{-1}Nx_N$ subject to (15b–15f), $\forall i = 1, \ldots, m$, where B_i^{-1} is the ith row of B^{-1}. Hence, whenever a node is partitioned, we can perform such a *range reduction* for the basic variables (see also [3,12,20,22]). However, we do not implement this additional strategy in order to assess the independent effect of the approach proposed herein, but we recommend its consideration for future research.

4 Extension to polynomial programming problems

In this section, we generalize the theory and strategies of Sect. 3 to address nonconvex polynomial programming problems. For ease in reading, we restate the polynomial program **PP** of order δ below:

$$\begin{aligned}
\textbf{PP}: \text{Minimize } & \phi_0(x) && \text{(16a)}\\
\text{subject to}\\
& \phi_r(x) \geq \beta_r, \ \forall r = 1, \ldots, R_1 && \text{(16b)}\\
& \phi_r(x) = \beta_r, \ \forall r = R_1 + 1, \ldots, R && \text{(16c)}\\
& Ax = b && \text{(16d)}\\
& x \in \Omega \equiv \{0 \leq l_j \leq x_j \leq u_j < \infty, \forall j \in \mathcal{N}\}. && \text{(16e)}
\end{aligned}$$

As part of the RLT reformulation process, we multiply the linear equality constraints $Ax = b$ in (16d) with distinct monomials $\prod_{j\in J} x_j$ of order $d = 1, \ldots, \delta - 1$. Hence, introducing the RLT-variables

$$w_J = \prod_{j\in J} x_j, \quad \forall J \subseteq \mathcal{N}^d, d = 2, \ldots, \delta, \tag{17}$$

where we also designate $w_j \equiv x_j$ when $J = \{j\} \subseteq \mathcal{N}$, we impose the linearized RLT constraints:

$$\left[(Ax = b) \times \prod_{j\in J} x_j\right]_L, \text{ yielding } Aw_{(.J)} = bw_J, \quad \forall J \subseteq \mathcal{N}^d, d = 1, \ldots, \delta - 1,$$

where $w_{(.J)} \equiv (w_{(1,\,J)}, w_{(2,\,J)}, \ldots, w_{(n,\,J)})$, and where (j, J) is the multi-set that combines j with the indices in J in nondecreasing order. Partitioning $Ax = b$ according to $Bx_B + Nx_N = b$ as before, where B is a basis of A, and using notation similar to that in (7), the foregoing set of equations can be written in the corresponding partitioned form as follows:

$$Bw_{(BJ)} + Nw_{(NJ)} = bw_J, \ \ \forall J \subseteq \mathcal{N}^d, d = 1, \ldots, \delta - 1. \tag{18}$$

Furthermore, notationally, given a multi-set $J \subseteq \mathcal{N}^d$ for any $d \in \{2, \ldots, \delta\}$, define $J_B \overline{\cap} J$ as the (multi-) subset of J that contains only the basic variable indices.

Proposition 3 *Let the equality system $Ax = b$ in (16d) be partitioned as $Bx_B + Nx_N = b$ for any basis B of A, and define*

$$Z = \left\{(x, w) : \text{(16d)}, \text{(18)}, \quad \textit{and} \quad w_J = \prod_{j\in J} x_j, \quad \forall J \subseteq J_N^d, \quad \textit{for } d = 2, \ldots, \delta\right\}. \tag{19}$$

Then, we have (17) holding true for any $(x, w) \in Z$.

Proof We establish this result by induction on $d = 2, \ldots, \delta$. For the case $d = 2$, we have (17) holding true for any $(x, w) \in Z$ by Proposition 1, using (16d), (18) written for $d = 1$, and $w_J = \prod_{j\in J} x_j$ for $J \subseteq J_N^2$ from (19).

Hence, assume that the result is true for $\{2, \ldots, d\}$, where $2 \le d \le \delta - 1$, i.e.,

$$w_J = \prod_{j \in J} x_j, \forall J \subseteq \mathcal{N}^{\Delta}, \ \Delta = 2, \ldots, d, \tag{20}$$

and examine the case of $d + 1$.

Consider the RLT product constraints (18) for $J \subseteq J_N^d$. Since $w_J = \prod_{j \in J} x_j$ and $w_{(NJ)} = x_N \prod_{j \in J} x_j = x_N w_J, \forall J \subseteq J_N^d$ by (19), we get from (18) that

$$w_{(BJ)} = B^{-1}(b - Nx_N)w_J = x_B w_J = x_B \prod_{j \in J} x_j, \quad \forall J \subseteq J_N^d.$$

This identity can be restated as follows:

$$w_J = \prod_{j \in J} x_j, \quad \forall J \subseteq \mathcal{N}^{d+1} : |J_B \overline{\cap} J| = 1. \tag{21}$$

Next, consider (18) for $J \subseteq \mathcal{N}^d$ such that $|J_B \overline{\cap} J| = 1$. Noting that $w_J = \prod_{j \in J} x_j$, and $w_{(NJ)} = x_N \prod_{j \in J} x_j = x_N w_J, \forall J \subseteq \mathcal{N}^d$ with $|J_B \overline{\cap} J| = 1$ by (20) and (21), we get from (18) that

$$w_{(BJ)} = B^{-1}(b - Nx_N)w_J = x_B w_J = x_B \prod_{j \in J} x_j, \forall J \subseteq \mathcal{N}^d : |J_B \overline{\cap} J| = 1.$$

The foregoing identity essentially states that

$$w_J = \prod_{j \in J} x_j, \forall J \subseteq \mathcal{N}^{d+1} : |J_B \overline{\cap} J| = 2. \tag{22}$$

Continuing in this fashion, by considering (18) for $J \subseteq \mathcal{N}^d$ such that $|J_B \overline{\cap} J| = 2$, we get similar to (21) and (22) that (17) holds true for all $J \subseteq \mathcal{N}^{d+1}$ such that $|J_B \overline{\cap} J| = 3$. Finally, by considering (18) for $J \subseteq \mathcal{N}^d$ such that $|J_B \overline{\cap} J| = d$, we get that (17) holds true for all $J \subseteq \mathcal{N}^{d+1}$ such that $|J_B \overline{\cap} J| = d + 1$, i.e., for the case where all indices are basic. Consequently, (17) holds true for the case $d + 1$. □

Upon including the RLT restrictions (18) along with the bound-factor RLT constraints as in (23e) and (23f) below, and substituting w_J for the monomial $\prod_{j \in J} x_j, \forall J \subseteq \mathcal{N}^d, d = 2, \ldots, \delta$, we reformulate Problem PP as follows, where (23a), (23b), and (23c) are linearizations of (16a), (16b), and (16c), respectively:

PP1 : Minimize $[\phi_0(x)]_L$ (23a)

subject to

$$[\phi_r(x)]_L \ge \beta_r, \ \forall r = 1, \ldots, R_1 \tag{23b}$$

$$[\phi_r(x)]_L = \beta_r, \ \forall r = R_1 + 1, \ldots, R \tag{23c}$$

$$Ax = b \tag{23d}$$

$$Bw_{(BJ)} + Nw_{(NJ)} = bw_J, \ \forall J \subseteq \mathcal{N}^d, d = 1, \ldots, \delta - 1 \tag{23e}$$

$$\Big[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j)\Big]_L \ge 0, \ \forall (J_1 \cup J_2) \subseteq \mathcal{N}^{\delta} \tag{23f}$$

$$l \le x \le u \tag{23g}$$

$$w_J = \prod_{j \in J} x_j, \forall J \subseteq \mathcal{N}^d, d = 2, \ldots, \delta. \tag{23h}$$

Utilizing Proposition 3, we reduce the number of defining identities in (23h) by imposing them only for $J \subseteq J_N^d, \forall d = 2, \ldots, \delta$, within (24h) below. Likewise, since we are only required to enforce (24h) instead of (23h), we retain the RLT bound-factor product constraints in (23f) derived only for $(J_1 \cup J_2) \subseteq J_N^\delta$. Furthermore, similar to Problem P2, we explicitly include the implied interval bounding constraints for the w_J-variables as in (24g) for strengthening the underlying linear programming relaxation obtained upon deleting (24h).

PP2 : Minimize $[\phi_0(x)]_L$ (24a)

subject to

$$[\phi_r(x)]_L \geq \beta_r, \ \forall r = 1, \ldots, R_1 \tag{24b}$$

$$[\phi_r(x)]_L = \beta_r, \ \forall r = R_1 + 1, \ldots, R \tag{24c}$$

$$Ax = b \tag{24d}$$

$$Bw_{(BJ)} + Nw_{(NJ)} = bw_J, \ \forall J \subseteq \mathcal{N}^d, d = 1, \ldots, \delta - 1 \tag{24e}$$

$$\left[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \right]_L \geq 0, \ \forall (J_1 \cup J_2) \subseteq J_N^\delta \tag{24f}$$

$$l \leq x \leq u, \quad \text{and} \quad \prod_{j \in J} l_j \leq w_J \leq \prod_{j \in J} u_j, \ \forall J \subseteq \mathcal{N}^d, \ d = 2, \ldots, \delta, \ |J_B \overline{\cap} J| \geq 1 \tag{24g}$$

$$w_J = \prod_{j \in J} x_j, \ \forall J \subseteq J_N^d, d = 2, \ldots, \delta. \tag{24h}$$

To enhance the foregoing reduced RLT-based relaxation, we append within (24f) in PP2 those additional constraints from (23f) for which the LP relaxation $\overline{\text{PP1}}$ of PP1 yields positive associated dual variables. Hence, in the branch-and-bound process for solving the reduced form PP2, we initially solve the LP relaxation $\overline{\text{PP1}}$ at the root node, and identify the subset of constraints from (23f) given by

$$J_{+duals} \equiv \{(J_1, J_2) : (J_1 \cup J_2) \subseteq \mathcal{N}^\delta \setminus J_N^\delta \quad \text{and} \quad u_{(J_1, J_2)} > 0\},$$

where $u_{(J_1, J_2)}$ denotes the value of the dual variable associated with the corresponding constraint in (23f) at an optimal solution for $\overline{\text{PP1}}$. We then include the particular constraints from (23f) for $(J_1, J_2) \in J_{+duals}$ within the reduced relaxation PP2 for each node subproblem, appropriately updated based on the variable bounding intervals for the current node. Let us refer to this set of additional constraints at any node as C_{+duals}. Note that whenever we branch on some variable x_k, say, by splitting its current interval $l_k \leq x_k \leq u_k$ at the value $\bar{x}_k \in (l_l, u_k)$ as discussed in Sect. 1 to obtain the respective bounding restrictions $l_k \leq x_k \leq \bar{x}_k$ and $\bar{x}_k \leq x_k \leq u_k$ for the two child-nodes, we may not obtain a monotonic increase in the lower bound from the parent node to the child-nodes. The reason for this is that, because we have included only a selected subset of constraints (C_{+duals}) corresponding to $(J_1, J_2) \in J_{+duals}$ from (23f) within (24f) (after further revising the bound on x_k), the imposed bound-factor constraints do not necessarily imply all corresponding lower order bound-factor constraints as per the result in [19]. Hence, in order to preserve monotonicity in the lower bounds, we include certain additional sets of constraints from the parent node subproblem within the current node relaxation as explained next.

Whenever we branch on some variable x_k, then in addition to incorporating the constraints C_{+duals} within PP2 for the child-node subproblem, we also include those bound-factor constraints in C_{+duals} from the parent node formulation (in the same form therein) that involve

the particular bound-factor that is associated with the branching restriction. An exception to this rule is when some latter type of constraint contains the corresponding branching restriction bound-factor only once. In this case, Proposition 4 below advocates the inclusion of a tighter constraint instead (i.e., Constraint (27) below in lieu of (25)).

Proposition 4 *For any parent node problem, consider the following associated bound-factor RLT constraint in C_{+duals} for some $(J_1, J_2) \in J_{+duals}$:*

$$\left[\prod_{j\in J_1}(x_j - l_j)\prod_{j\in J_2}(u_j - x_j)\right]_L \geq 0. \tag{25}$$

Suppose that we branch on a variable x_k, and consider the child-node for which we impose the additional restriction $x_k \geq \bar{x}_k > l_k$ (the case for the child-node corresponding to the restriction $x_k \leq \bar{x}_k < u_k$ is similar). Suppose further that we have $k \in J_1$ and $k \notin (J_1 \cup J_2) - \{k\}$, so that the child-node constraint for this particular $(J_1, J_2) \in J_{+duals}$ is given by

$$\left[(x_k - \bar{x}_k)\prod_{j\in J_1-k}(x_j - l_j)\prod_{j\in J_2}(u_j - x_j)\right]_L \geq 0. \tag{26}$$

Let $F_{\delta-1}(J_1-k, J_2) \equiv \prod_{j\in J_1-k}(x_j-l_j)\prod_{j\in J_2}(u_j-x_j)$, and consider the corresponding bound-factor constraint:

$$[F_{\delta-1}(J_1 - k, J_2)]_L \geq 0. \tag{27}$$

Then, (26) and (27) imply (25).

Proof Let (x, w) be feasible to (26) and (27). Then, the left-hand side of (25) evaluated at such a solution is given by:

$$\begin{aligned}
[(x_k - l_k)F_{\delta-1}(J_1 - k, J_2)]_L &= [x_k F_{\delta-1}(J_1 - k, J_2)]_L - l_k[F_{\delta-1}(J_1 - k, J_2)]_L \\
&\geq [x_k F_{\delta-1}(J_1 - k, J_2)]_L - \bar{x}_k[F_{\delta-1}(J_1 - k, J_2)]_L \\
&= [(x_k - \bar{x}_k)F_{\delta-1}(J_1 - k, J_2)]_L \\
&\geq 0.
\end{aligned}$$

□

As alluded above, in the case described in Proposition 4 (and similarly for the child-node corresponding to the additional restriction $x_k \leq \bar{x}_k$), we include (26) and (27) for the particular $(J_1, J_2) \in J_{+duals}$ in lieu of (26) and (25) in order to derive a tighter child-node relaxation.

Furthermore, along any branch within the branch-and-bound tree, constraints of the type (25) or (27) (as explained above) that are generated at the upstream nodes are progressively inherited by the downstream nodes. Although this inheritance procedure guarantees tighter child-node relaxations, in order to preserve monotonicity in the lower bounds while keeping the size of the relaxations from growing excessively, we do the following. Consider any node (other than the root node) where, in addition to (24f), we have the constraints C_{+duals} plus other restrictions inherited from upstream nodes. Denote $C_{inherit}$ as the latter set of constraints. Now, suppose that we branch on some variable x_k and consider the child-node obtained for the branching restriction $x_k \geq \bar{x}_k$ (the case of $x_k \leq \bar{x}_k$ is similar). Then, to construct $C_{inherit}$ for this child-node's relaxation, denoted $C^{child}_{inherit}$, let C_{active} denote

the constraints in the parent's node relaxation that have associated positive dual variables, let $C_{inherit}^{parent}$ denote the inherited constraints present in the parent node relaxation, and let C_k^{parent} denote those bound-factor restrictions within C_{+duals} for the parent node relaxation that involve the bound-factor $(x_k - l_k)$. Accordingly, we set

$$C_{inherit}^{child} = \{C_k^{parent} \cap C_{active}\} \cup \{C_{inherit}^{parent} \cap C_{active}\}, \tag{28}$$

where for any restriction of the type (25) in $C_k^{parent} \cap C_{active}$ that satisfies the conditions of Proposition 4, we utilize the corresponding tighter relaxations (27). Let LB_{parent} and LB_{child} be the lower bounds thus obtained for the parent and child-node, respectively. Then, we have the following result:

Proposition 5 $LB_{child} \geq LB_{parent}$.

Proof Let PP2^{parent} and PP2^{child} respectively denote the parent and child-node LP relaxations. Consider Problem PP2$_{active}^{parent}$ that is constructed from the parent node relaxation PP2^{parent} by removing those bound-factor restrictions that have associated zero dual values at optimality from among the constraints in $C_{+duals} \cup C_{inherit}^{parent}$. Hence, $v(\text{PP2}_{active}^{parent}) = v(\text{PP2}^{parent}) = LB_{parent}$, where $v(\text{P})$ denotes the optimal objective function value for any given problem P. But PP2^{child} contains stronger restrictions within the set of constraints (24b–24g) than PP2^{parent} (or PP2$_{active}^{parent}$) based on the updated bound on the branching variable x_k (by [14]), plus it contains all the remaining constraints (other than (24b–24g) from PP2$_{active}^{parent}$ by (28) (possibly tightened via Proposition 4), in addition to other constraints in C_{+duals}. Hence, $LB_{child} = v(\text{PP2}^{child}) \geq v(\text{PP2}_{active}^{parent}) = LB_{parent}$. □

In our computational results, we found that this enhancement of PP2 significantly tightens the underlying relaxations, while suitably filtering the RLT constraints to control its size. Henceforth, we shall therefore assume that PP2 has been thus enhanced.

We refer to the likewise generalized RLT-based branch-and-bound algorithms described in Sect. 3 as applied to Problems PP1 and (the enhanced) PP2, by RLT(PP1) and RLT(PP2), respectively. Similarly, the respective algorithms that incorporate SDP cuts are denoted by RLT$_{\text{SDP}}$(PP1) and RLT$_{\text{SDP}}$(PP2). Furthermore, note that the discussions in Remarks 2 and 3 hold true for polynomial programming problems as well. In particular, the RLT-based branch-and-bound algorithms RLT$^{\text{NB}}$(PP1) and RLT$_{\text{SDP}}^{\text{NB}}$(PP1) as applied to Problem PP1 involve partitioning on just nonbasic variables, and are implemented without and with SDP cuts, respectively. For the variants RLT(PP2), RLT$_{\text{SDP}}$(PP2), RLT$^{\text{NB}}$(PP1), and RLT$_{\text{SDP}}^{\text{NB}}$(PP1) that branch only on the nonbasic variables, we select a branching variable index as

$$j^* \in \arg\max_{j \in J_N} \theta_j,$$

where θ_j is computed as follows:

$$\theta_j \equiv \sum_{d=1}^{\delta-1} \sum_{J \in J_N^d} \left| \bar{w}_{J \cup j} - \bar{w}_J \bar{x}_j \right|, \quad \forall j \in J_N.$$

The remainder of the branch-and-bound algorithm remains the same as that described in Sect. 1.

5 Static and dynamic basis selection techniques

In this section, we propose techniques for selecting a basis B for implementing the RLT reduction process for polynomial programming problems, which is then accordingly employed in generating Problem PP2 and optimizing it via RLT(PP2) and RLT_{SDP}(PP2), or utilized within the algorithmic procedures RLT^{NB}(PP1) and $\text{RLT}^{\text{NB}}_{\text{SDP}}$(PP1). We refer to this as a *static approach*. We also describe a *dynamic approach* in which the basis B is suitably revised during the algorithmic process in order to restructure Problem PP2 in a desirable fashion. Several alternative mechanisms are devised to obtain different variants of these static and dynamic approaches.

5.1 Static approach

In this approach, we begin by solving the LP relaxation $\overline{\text{PP1}}$ of Problem PP1, which is derived by deleting (23h). Let $(\bar{x}, \bar{w})$ be the optimal solution obtained for Problem $\overline{\text{PP1}}$, and let $\lambda_J, \forall J \subseteq \mathcal{N}^\delta$, be the maximum of the resulting dual multipliers associated with the bound-factor constraints in (23f) that contain the variable w_J. The following alternative procedures determine an ordered set L of indices $j \in \mathcal{N}$, for each of which the corresponding basis B is then determined by utilizing the first m linearly independent columns of A that are associated with the variables having indices selected in the order in which they appear within L. The motivation for these methods is to explicitly enforce the product relationships (17) for the variables that are less likely to take on values at their range bounds, thereby tending to accelerate the convergence of the branch-and-bound process.

Method S.1: For the first method, we examine $\lambda_J, \forall J \subseteq \mathcal{N}^\delta$, in nonincreasing order and select the index set $J = J_1^*$ having the largest λ_J-value. Accordingly, we incorporate the distinct indices $j \in J_1^*$ in arbitrary (say, increasing) order within the list L, and next consider the ordered list of $\lambda_J, \forall J \subseteq \mathcal{N}^\delta - J_1^*$, to select the second index set J_2^* to insert within L. Continuing in this fashion, we construct the ordered list L of indices $j = 1, \ldots, n$, and then we reverse the order of indices in this list for subsequently selecting the corresponding basis B.

Method S.2: As an alternative to Method S.1, we construct the list L by arranging the indices $j \in \mathcal{N}$ in nondecreasing order of $\sum_{J \subseteq \mathcal{N}^{\delta-1}} (1 + k_{jJ}) \lambda_{(\{j\} \cup J)}$, where k_{jJ} is the number of times that j appears within J.

Method S.3: For the third method, we compute the number of bound-factor constraints, Q, to be generated in (24f) for the reduced RLT strategy. After sorting the constraints in (23f) in nonincreasing order of their dual values, we pick the first Q (possibly fewer) constraints associated with nonzero dual values to construct the set $\mathcal{N}_Q^\delta$ of the corresponding multi-sets $J \in (J_1 \cup J_2)$. For each variable index $j \in \mathcal{N}$, we compute $\sum_{J \subseteq \mathcal{N}_Q^\delta} k_{jJ}$ and sort the variables in nondecreasing order of this measure to form the list L.

Method S.4: This method constructs the desired list L based on an aggregate violation measure with respect to the RLT variable relationships. More specifically, for each $j \in \mathcal{N}$, we compute the total violation with respect to (17) via $\sum_{d=1}^{\delta-1} \sum_{J \subseteq \mathcal{N}^d} |\bar{x}_j \bar{w}_J - \bar{w}_{(\{j\} \cup J)}|$, and sort the variables $j = 1, \ldots, n$ in nondecreasing order of this measure within the list L.

5.2 Dynamic approach

In this approach, for Method D.k, k = 1, ..., 4, we respectively implement the method S.k, k = 1, ..., 4, at the root node of the branch-and-bound tree to commence the branch-and-bound algorithm. Then, based on a specified *depth-parameter* Δ, for any active node selected at depth Δ in the branch-and-bound tree, we compute an updated vector of (dual) multipliers associated with (23f) as given by taking an element-wise maximum of the corresponding dual multipliers associated with (23f) that were computed for $\overline{\text{PP1}}$ at the root node, and the corresponding dual multipliers for the current node (where we take undefined elements as zero in the latter case). Accordingly, we invoke the particular Method S.k, k $\in \{1, \ldots, 4\}$, at the current node using this resulting updated (dual) vector to recompute the basis B of A. The recomputed basis is then subsequently used to formulate (the enhanced) Problem PP2 for the present node and for all its descendents. Note that different bases can therefore be possibly specified for reformulating Problem PP2 at the different active nodes at depth Δ, and the particular reduced RLT algorithmic procedure is applied accordingly to each node's reformulated problem.

Remark 4 As a variant of Methods D.k for k = 1, ..., 4, we could specify multiple depth parameters $\Delta_1 < \Delta_2 < \cdots < \Delta_{\mathfrak{D}}$ for some $\mathfrak{D} \geq 2$, and invoke a similar reformulation of Problem PP2 based on the determination of a revised basis for nodes at each of the levels $\Delta_1, \ldots, \Delta_{\mathfrak{D}}$ of the branch-and-bound tree. We advocate this investigation for future research.

6 Computational results

In this section, we evaluate the relative effectiveness of utilizing the proposed reduced size RLT formulations as compared to the original RLT formulation within a branch-and-bound algorithm. We begin by studying the performance of the different delineated static basis selection methods for implementing RLT(PP2), both with and without v-semidefinite cuts (Algorithms RLT(PP2) and RLT_{SDP}(PP2)). Next, utilizing the best performing (static) basis selection method as identified in the foregoing experiment, we compare the performance of RLT implemented on Problems PP1 and PP2, where for the former, we also investigate partitioning the bounding intervals only for the nonbasic variables (Algorithms RLT(PP1), RLT^{NB}(PP1), and RLT(PP2), respectively). In addition, we also examine the performance of these procedures with SDP cuts (RLT_{SDP}(PP1), $\text{RLT}^{\text{NB}}_{\text{SDP}}$(PP1), and RLT_{SDP}(PP2), respectively). All algorithms were implemented on a workstation having a 2.33 GHz Intel Xeon processor with 3.25 GB of RAM and running Windows XP. Furthermore, we also present corresponding results using the commercial software BARON (Version 9.0.6) [13,24], which was implemented on a workstation having two 2.13 Ghz Intel Xeon processors with 4GB of RAM and running Windows 7. Finally, the polynomial test problems were also optimized using the mixed-integer nonlinear programming software Couenne [1] on a workstation having 2.4 Ghz Intel Xeon processor with 8 GB of RAM and running Linux. (The last two sets of runs were made on different workstations since BARON, Couenne, and CPLEX 11.1 were installed on different computers in our laboratory; however, as seen in the sequel, the performance results are significantly different, which makes the relative comparisons obvious.) The RLT algorithms employ the dual optimizer with the steepest descent pricing rule implemented within CPLEX (Version 11.1) while using default settings for solving the underlying LP relaxations, and utilize SNOPT Version 7 [7] as the nonlinear programming

Table 1 Performances of different static basis selection methods with RLT(PP2) and RLT_{SDP}(PP2)

δ (# of instances)	Average CPU time				$n_{\text{premature}}$				Average % opt. gap*			
	S.1	S.2	S.3	S.4	S.1	S.2	S.3	S.4	S.1	S.2	S.3	S.4
RLT(PP2)												
4 (12)	135	110	115	160	2	1	1	3	4.6	2.2	2.2	2.3
5 (12)	199	150	143	197	4	3	3	4	24.9	6.8	6.7	10.9
6 (8)	204	256	204	221	3	4	3	3	6.4	3.1	2.6	2.8
Overall	176	162	148	189	9	8	7	10	14.2	4.3	4.3	5.9
RLT_{SDP}(PP2)												
4 (12)	42	35.6	34.6	84.8	0	0	0	1	–	–	–	0.1
5 (12)	35.1	26.6	25.1	25.7	0	0	0	0	–	–	–	–
6 (8)	22	21.2	23	38.7	0	0	0	0	–	–	–	–
Overall	34.4	28.6	28.1	51.1	0	0	0	1	–	–	–	0.1

* The average % optimality gap computed over the premature termination cases

(local search) solver for computing upper bounds by initializing this search process from the respective solutions obtained for the corresponding LP relaxations. Furthermore, we used Matlab® (Version R2008a [11]) for determining the rank of the candidate matrices during the basis selection process.

To facilitate an appropriate testing environment, we randomly generated polynomial programs for given values of the degree δ, number of variables (n), number of constraints (m and R), and densities of the objective function and constraints. The percentage of nonzero objective and constraint coefficients were specified as 100 and 25%, of which 50% were taken as positive, and the degree of the ith constraint was set to $[(i-1) \mod \delta]+1, \forall i = 1, \ldots, R_1$. Each of the constraint functions was evaluated at l_j and $u_j, \forall j \in \mathcal{N}$, and the right-hand side was set equal to the average of these two values. The generated instances were checked for feasibility prior to computational analysis and the right-hand sides of the inequality constraints for infeasible instances were suitably decreased to attain feasibility. We did not include any nonlinear equality constraints; hence $R = R_1$. By varying the degree of the program, the number of variables and inequality constraints, and the size of the equality system as controllable problem parameters, we tried to assess the effectiveness of the proposed algorithms over a wide range of polynomial programming test problems. Utilizing 32 relatively moderately-sized problems in a primary analysis as described below, we first select the best performing basis selection method for formulating Problem PP2, and provide comparisons against applying RLT to Problem PP1, both with and without SDP cuts. Following this, we further evaluate the most competitive algorithmic variants using a second set of 10 more challenging larger-sized polynomial programming test problems. (see [4]) For our computational experiments, we used $\varepsilon = 0.001$ as the optimality gap tolerance, and set a run-time limit of 500 and 3,600 CPU seconds, respectively, for the relatively moderate and large-sized problem instances.

We begin with an explorative analysis to assess the relative performances of the four different static basis selection methods S.1, . . ., S.4 discussed in Sect. 5.1, as well as the effect of v-semidefinite cuts on computational effort. Table 1 provides a summary of the results obtained by reporting the average CPU times over the stated number of instances for each δ-value, along with the number of premature termination cases ($n_{\text{premature}}$), and

the associated average percentage optimality gaps over such cases. As evident from Table 1, the performance differences between the tested basis selection methods underscores the importance of selecting an appropriate basis. The basis selection method S.3 outperformed the other methods when used within Algorithm RLT(PP2). Upon enhancing this procedure with the v-semidefinite cuts described in [16] (Algorithm RLT_{SDP}(PP2)), the performance improved significantly, also alleviating the premature termination cases. The average CPU time improvements for RLT_{SDP}(PP2) over RLT(PP2) using the respective basis selection methods S.1, S.2, S.3, and S.4 were 81, 82, 81, and 73%. Moreover, the reductions in the overall average CPU times for RLT_{SDP}(PP2) using S.3 over the alternatives S.1, S.2, and S.4 were 18.3, 1.9, and 45%, respectively. Based on these results, we designated S.3 as the basis selection method for all subsequent runs.

It is worth mentioning here that some of the early termination cases were due to numerical issues encountered while solving the underlying LP relaxations. Decreasing the feasibility tolerance of CPLEX to 10^{-9} from its default value of 10^{-6} helped resolve some of these premature termination cases, but naturally increased the effort required for other instances. Hence, we retained the default settings for the feasibility tolerance in our computational analysis. Another relevant note on implementation issues is related to the inheritance of SDP cuts. Although the SDP cuts generated for any node subproblem are valid for all descendent nodes, we let each current node relaxation inherit only those SDP cuts that were generated at upstream nodes up to four levels in our implementation. Note that this limited SDP cut inheritance scheme can possibly (though not likely) affect the monotonicity of the lower bounds as established by Proposition 5.

Table 2 presents the detailed results obtained for the best performing static basis selection method S.3 when utilized within Algorithms RLT(PP2) and RLT_{SDP}(PP2), and also includes comparative results for Algorithms RLT(PP1) and RLT^{NB}(PP1), along with their SDP cut-enhanced variants RLT_{SDP}(PP1) and $\text{RLT}^{\text{NB}}_{\text{SDP}}$(PP1). Five common instances out of the 32 test problems terminated prematurely using Algorithms RLT(PP2) and RLT(PP1) with an average optimality gap of 5.9 and 8.0%, respectively. Additionally, RLT(PP1) failed to solve the instance $(\delta, n, m, R) = (4, 10, 3, 4)$ and terminated with a 0.14% optimality gap, whereas RLT(PP2) optimized this instance within 397 CPU seconds. Two other instances with $(\delta, n, m, R) = (5, 8, 4, 6)$ and $(6, 5, 3, 6)$ terminated prematurely using RLT(PP2) with a 0.3% average optimality gap, whereas RLT(PP1) optimized these instances within 34 CPU seconds, on average. However, when implemented with SDP cuts, all problems were optimized by both procedures within the set time limit, where the average CPU times were 40.4 s for RLT_{SDP}(PP1) versus 28.1 s for the proposed algorithm RLT_{SDP}(PP2), yielding a 30.4% savings in effort. We also implemented RLT^{NB}(PP1) and $\text{RLT}^{\text{NB}}_{\text{SDP}}$(PP1), as delineated in Remark 3. Although RLT^{NB}(PP1) decreased the average computational effort as compared to RLT(PP1), it turned out that when these methods were enhanced with SDP cuts, $\text{RLT}^{\text{NB}}_{\text{SDP}}$(PP1) consumed more effort than both RLT_{SDP}(PP1) and RLT_{SDP}(PP2). Hence, RLT_{SDP}(PP1) and RLT_{SDP}(PP2) were selected for further comparison.

Using the basis selection method S.3, we also applied Remark 2 to eliminate the basic variables x_B via the substitution $x_B = B^{-1}(b - Nx_N)$, and accordingly reformulated the problems equivalently in the space of the $n - m$ nonbasic variables. Among the 32 test problems, we obtained three early termination cases using RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$, where the overall average CPU effort for this procedure was 52.7 s, as compared with 28.1 CPU seconds (or a savings of 46.7%) for RLT_{SDP}(PP2). However, for individual problems, the relative effectiveness of RLT_{SDP}(PP2) versus RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$ mainly depends on the size of the LP relaxations and the quality of the lower bounds at the root node. RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$ typically produces worse lower bounds at the root node than RLT_{SDP}(PP2), but the

Table 2 CPU times for RLT(PP2), RLT(PP1), and RLT^{NB}(PP1) with and without v-semidefinite cuts

(δ, n, m, R)	RLT(PP2)	RLT(PP1)	RLT^{NB}(PP1)	RLT_{SDP}(PP2)	RLT_{SDP}(PP1)	$\text{RLT}^{\text{NB}}_{\text{SDP}}$(PP1)
(4, 8, 3, 4)	4.3	10.8	10.9	2.6	5	5.1
(4, 8, 3, 6)	20.1	70.2	53.6	18.4	41.6	32.3
(4, 8, 4, 4)	7.1	19.4	25.6	4.1	7.6	7.6
(4, 8, 4, 6)	500.3	500.5	225.1	41.5	84.9	93.8
(4, 9, 3, 4)	21.6	36.6	37.6	12.7	19	19.3
(4, 9, 3, 6)	5.3	6	6.5	6.2	5.1	5.4
(4, 9, 4, 4)	13.7	13.9	14.1	14.1	17.3	13
(4, 9, 4, 6)	30.1	40.9	35	25.5	34.9	32
(4, 10, 3, 4)	397.1	505.9	504.7	156.2	156.1	238.3
(4, 10, 3, 6)	95.3	92.8	93.4	50.3	37.9	38.6
(4, 10, 4, 4)	9.8	9.7	10.2	11.9	7.3	7.8
(4, 10, 4, 6)	272.3	148.5	158	71.1	73.8	74
Average	114.7	121.2	97.9	34.6	40.9	47.3
(5, 6, 3, 4)	17.4	48.5	52.7	4.6	7	6.6
(5, 6, 3, 6)	11.1	25.4	24.2	3	6.3	6.3
(5, 6, 4, 4)	1	1.2	1.2	1.1	1.2	1.1
(5, 6, 4, 6)	1.3	2.2	2.2	1.2	1.2	1.2
(5, 7, 3, 4)	500.8	500.1	152.5	25.3	45.8	45.1
(5, 7, 3, 6)	9.1	8	8.2	8.2	26.9	27.1
(5, 7, 4, 4)	500.6	501.2	500.7	25.5	46.9	50.5
(5, 7, 4, 6)	19.7	44.3	34.7	19.6	36.7	28.7
(5, 8, 3, 4)	85.3	106.9	124.3	76.8	126.6	127.6
(5, 8, 3, 6)	63.1	66.6	96.3	32.6	78.9	80.1
(5, 8, 4, 4)	6.1	5.3	5.3	5.5	5	5.1
(5, 8, 4, 6)	502.1	53.2	96.7	97.3	42.2	163.7
Average	143.1	113.6	91.6	25.1	35.4	45.3
(6, 5, 2, 4)	500.7	500.7	500.6	11.6	22.5	22.5
(6, 5, 2, 6)	3.8	8.2	7.5	4.3	6.9	6.8
(6, 5, 3, 4)	2.5	2.5	2.4	3	2.5	2.6
(6, 5, 3, 6)	500.4	14.9	14.5	3	5.3	4.6
(6, 6, 2, 4)	503.5	501.4	500.6	37.9	93.3	93.8
(6, 6, 2, 6)	92.8	206.2	227.8	101.5	210.7	258.7
(6, 6, 3, 4)	15.5	20.8	21	11.7	24.4	25
(6, 6, 3, 6)	15.9	30.3	34	11.2	12.7	12.4
Average	204.4	160.6	163.5	23	47.3	53.3
Overall avg.	147.8	128.2	111.9	28.1	40.4	48

size of the relaxations for the former approach is smaller than those for the latter approach, which results in faster LP solution times for the former algorithm. We therefore designed a hybrid algorithmic approach, denoted $\mathbf{RLT}_{\text{SDP}}$**(Hybrid)**, which is based on a parameter μ computed as the product of the ratios $\frac{\text{GAP}_1}{\text{GAP}_2}$ and $\frac{\text{N}_1}{\text{N}_2}$, where GAP_1 and GAP_2 are the respective root node optimality gaps obtained using Algorithms RLT_{SDP}(PP2) and RLT_{SDP}(PP1)

Table 3 Performances of different dynamic basis selection methods with RLT(PP2) and RLT_{SDP}(PP2)

δ (# of instances)	Average CPU time				$n_{\text{premature}}$				Average % opt. gap*			
	D.1	D.2	D.3	D.4	D.1	D.2	D.3	D.4	D.1	D.2	D.3	D.4
RLT(PP2)												
4 (12)	127	110	115	130	2	1	1	2	0.7	2.2	2.2	29.4
5 (12)	166	151	144	156	3	3	3	3	32.5	6.3	6.7	10
6 (8)	208	259	210	183	3	4	3	2	6.3	3.7	3.3	1.4
Overall	162	163	149	153	8	8	7	7	14.7	4.5	4.6	13
RLT_{SDP}(PP2)												
4 (12)	44.8	35.6	34.9	113.4	0	0	0	1	–	–	–	28.7
5 (12)	74.6	26.7	25.4	65.9	1	0	0	1	1.5	–	–	2
6 (8)	22.5	21.7	24	27.9	0	0	0	0	–	–	–	–
Overall	50.4	28.8	28.6	74.2	1	0	0	2	1.5	–	–	15.4

* The average % optimality gap computed over the premature termination cases

in $\mathbb{R}^{n-m}$ (prior to generating SDP cuts), and where N_1 and N_2 are the respective products of the number of variables and constraints in the LP relaxations for these two approaches. Hence, RLT_{SDP}(Hybrid) computes μ at the root node, and whenever $\mu < 1$, it proceeds by implementing RLT_{SDP}(PP2); otherwise, it implements RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$. This hybrid algorithm suitably compromises between the robustness of RLT_{SDP}(PP2) and the swiftness of RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$, and it optimized all the 32 test problems within 19.4 CPU seconds on average, versus 28.1 and 52.7 CPU seconds, respectively, for RLT_{SDP}(PP2) and RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$, resulting in respective savings in effort of 31 and 63.2%.

Next, we considered the dynamic basis selection methods. Table 3 presents the results obtained, displaying similar information as in Table 1 for the case of the static methods. When implemented within RLT(PP2), the dynamic methods D.1 and D.4 with $\Delta = 1$ outperformed the corresponding static methods S.1 and S.4, while decreasing the effort by 8.4 and 19.2%, respectively. However, when utilized within RLT_{SDP}(PP2), the methods D.1 and D.4 increased the average CPU time as compared with S.1 and S.4 by 46.6 and 45.3%, respectively. On the other hand, the static and dynamic variants of the other two basis selection methods performed similarly. This experiment reveals that dynamically updating the basis can have different effects, depending on the basis selection method used and other accompanying algorithmic strategies implemented such as SDP cuts. On a positive note, our best performing static basis selection method S.3 remained robustly superior under both variations of using SDP cuts and dynamically updating the basis. Hence, we continue with the static basis selection method S.3 for the remaining runs.

Next, we compare the performances of RLT_{SDP}(PP2), RLT_{SDP}(PP1), and RLT_{SDP}(Hybrid) using the set of 10 larger-sized polynomial programming instances, where S.3 is utilized for selecting the basis within RLT_{SDP}(PP2) and RLT_{SDP}(Hybrid). Table 4 presents the results obtained, including those for the implementation of RLT_{SDP}(PP1) in the reduced nonbasic variable space $\mathbb{R}^{n-m}$. The proposed algorithm RLT_{SDP}(PP2) reduced the average computational times in comparison with RLT_{SDP}(PP1) by 37.6%. RLT_{SDP}(PP1) terminated prematurely for three test cases with an average optimality gap of 10.9%, whereas all three instances were optimized by RLT_{SDP}(PP2) within 2,846 CPU seconds on average. As seen in Table 4, using the substitution process of Remark 2 to reformulate the original problem itself in $\mathbb{R}^{n-m}$, reduced the computational effort required by RLT_{SDP}(PP1), in comparison

Table 4 Performances of RLT_{SDP}(PP2), RLT_{SDP}(PP1), and RLT_{SDP}(Hybrid) for solving the 10 larger-sized problems

(δ, n, m, R)	Original problem in $\mathbb{R}^n$			Original problem in $\mathbb{R}^{n-m}$	RLT_{SDP}(Hybrid)
	RLT_{SDP}(PP2)	RLT_{SDP}(PP1)		RLT_{SDP}(PP1)	
	CPU time	CPU time	% opt. gap	CPU time	CPU time
(4, 11, 1, 6)	1,557	2,330		1,329	1,561
(4, 11, 2, 6)	354	587		948	355
(4, 12, 1, 6)	310	453		2,495	316
(4, 12, 2, 6)	1,912	3,642	0.5	1,801	1,915
(5, 9, 1, 6)	320	726		752	329
(5, 9, 2, 6)	3,622	3,691	23	408	498
(6, 7, 1, 6)	1,290	3,208		511	1,301
(6, 7, 2, 6)	314	1,325		54	318
(7, 6, 1, 6)	3,003	3,635	9	1,532	1,601
(7, 6, 2, 6)	433	1,436		22	359
Average	1,312	2,103	11	985	855

with its respective implementations in $\mathbb{R}^n$ by 53.2%. Note that the particular basis used for the substitution process greatly influences the performance of RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$. For instance, the respective average CPU times were 3,315, 1,838, 985, and 2,936 s when the methods S.1, S.2, S.3, and S.4 were utilized for selecting the basis within RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$. Continuing with S.3, we observe that although this reduced implementation of RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$ also decreased the average computational effort in comparison with RLT_{SDP}(PP2) by 24.9%, the latter performed better on three of the 10 instances. Indeed, as seen from Table 4, the proposed algorithm RLT_{SDP}(Hybrid) that automatically composes these two procedures optimized all instances within 855 CPU seconds on average, reducing the computational effort required by RLT_{SDP}(PP1) and RLT_{SDP}(PP2) in $\mathbb{R}^n$, and by RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$, by 59.3, 34.8, and 13.2%, respectively. Hence, overall, we advocate RLT_{SDP}(Hybrid) for implementation.

Finally, we compare the performances of the software packages Couenne and BARON with RLT_{SDP}(Hybrid), where the corresponding results are presented in Table 5. In nine out of ten instances, Couenne and BARON terminated prematurely with respective average optimality gaps of 437 and 260%. The remaining final instance in Table 5 was optimized in 1906 CPU seconds using BARON, whereas Couenne could only decrease the optimality gap of this instance to 150% within the time limit of 3,600 CPU seconds. The implementation of BARON in the reduced nonbasic variable space decreased the required effort by 10% in comparison with its implementation in $\mathbb{R}^n$, but was yet unable to solve eight of the ten problem instances, having an unresolved average optimality gap of 246%. Overall, our proposed algorithm RLT_{SDP}(Hybrid) reduced the average computational times consumed by Couenne, BARON in $\mathbb{R}^n$, and BARON in $\mathbb{R}^{n-m}$ by 77, 75, and 72%, respectively.

7 Summary and conclusions

This paper addresses reduced size Reformulation-Linearization Technique (RLT)-based formulations for polynomial programming problems having linear equality subsystems. It is

Table 5 Performances of Couenne, BARON, and RLT_{SDP}(Hybrid) for solving the 10 larger-sized problems

(δ, n, m, R)	Original problem in $\mathbb{R}^n$				Original problem in $\mathbb{R}^{n-m}$		RLT_{SDP}(Hybrid)
	Couenne		BARON		BARON		
	CPU time	% opt. gap	CPU time	% opt. gap	CPU time	% opt. gap	CPU time
(4, 11, 1, 6)	3,613	420	3,600	289	3,600	290	1,561
(4, 11, 2, 6)	3,630	524	3,600	279	3,600	328	355
(4, 12, 1, 6)	3,646	361	3,600	192	3,600	461	316
(4, 12, 2, 6)	3,617	746	3,600	504	3,600	547	1,915
(5, 9, 1, 6)	3,692	276	3,600	143	3,600	145	329
(5, 9, 2, 6)	3,653	602	3,600	336	3,600	108	498
(6, 7, 1, 6)	3,688	294	3,600	179	3,600	76	1,301
(6, 7, 2, 6)	3,628	149	3,600	119	3,600	16	318
(7, 6, 1, 6)	3,611	559	3,600	296	1,917		1,601
(7, 6, 2, 6)	3,647	150	1,906		37		359
Average	3,643	408	3,431	260	3,075	246	855

shown that a strict subset of the RLT defining identities, selected via a basis partitioning scheme, implies the remaining defining identities and enables the removal of a major portion of the RLT constraints, while maintaining an equivalent representation.

When both the original and reduced relaxation procedures were enhanced with v-semidefinite (or SDP) cuts, the static basis selection methods S.1, S.2, and S.3 implemented within RLT_{SDP}(PP2) outperformed RLT_{SDP}(PP1), decreasing the effort by 14.9, 29.1, and 30.4%, respectively, on a test-bed of 32 moderately sized instances, whereas S.4 increased the average CPU time for RLT_{SDP}(PP2) by 26.4%. Using the best performing method S.3, which tends to retain the most contributing bound-factor relationships from the original RLT formulation within the reduced representation, we further solved 10 relatively larger-sized polynomial programming problems. (As mentioned earlier, all these test problems are available from the authors upon request - see [4].) The results obtained achieved a reduction in CPU times for the proposed algorithm RLT_{SDP}(PP2) over RLT_{SDP}(PP1) and BARON (Version 9.0.6) by 37.6 and 61.8%, respectively. Using the basis selection method S.3, we also designed a hybrid algorithm, denoted RLT_{SDP}(Hybrid), which performs a more extensive node zero analysis to automatically compose a solution strategy that either implements RLT_{SDP}(PP2) or adopts the original RLT_{SDP}(PP1) procedure in a reduced nonbasic variable space in $\mathbb{R}^{n-m}$. This composite procedure, RLT_{SDP}(Hybrid), more robustly optimized all instances, reducing the average CPU time required by RLT_{SDP}(PP1) and RLT_{SDP}(PP2) in $\mathbb{R}^n$, and by RLT_{SDP}(PP1) in $\mathbb{R}^{n-m}$, by 52, 31, and 63.2%, respectively, for the 32 moderately sized problems, and by 59.3, 34.8, and 13.2%, respectively, for the 10 larger-sized problems. Furthermore, the respective average CPU times for the 10 larger-sized problems were 3,643, 3,431, and 3,075 CPU seconds for Couenne and BARON in $\mathbb{R}^n$, and for BARON in $\mathbb{R}^{n-m}$, with respective average optimality gaps of 408, 260, and 246%, whereas RLT_{SDP}(Hybrid) solved all these problems to global optimality within an average of 855 CPU seconds (yielding a savings in effort of 76.5, 75.1, and 72.2%, respectively, besides providing optimal solutions). Hence, we recommend RLT_{SDP}(Hybrid) as a method of choice among the tested procedures for solving polynomial programming problems.

This research can be extended in several directions. For instance, additional constraint filtering techniques (see [5,18,22], for example) can be employed to further reduce the size of the RLT relaxations. Also, similar to the use of SDP cuts, the underlying RLT-based linear programming relaxation can be tightened via grid-factor product restrictions ([15]) as well as by using suitable higher order RLT constraints. Similarly, we could tighten the variable bounds by implementing range reduction strategies as recommended in Remark 3 (see also [3,12,20,22] for a general discussion on range reduction strategies), while specifically exploiting the linear equality system. Finally, as suggested in Remark 4, certain multiple-depth variants of the dynamic basis selection strategy can be investigated, which utilize information from the root node and the current node to update the basis dynamically in a repetitive fashion.

Acknowledgments This research has been supported by the *National Science Foundation* under Grant No. CMMI-0969169. One of the authors (Leo Liberti) was partially supported by grants: ANR 07-JCJC-0151 "ARS", Digiteo Chair 2009-14D "RMNCCO", Digiteo Emergence 2009-55D "ARM". The authors gratefully acknowledge Philip E. Gill, Walter Murray, and Michael A. Saunders for permitting the use of the SNOPT solver. We also thank the guest editor, Nguyen V. Thoai, and two anonymous referees for their constructive comments that helped improve the presentation in this paper.

References

1. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optim. Methods Softw. **24**(4–5), 597–634 (2009)
2. Cafieri, S., Hansen, P., Liberti, L., Letocart, L., Messine, F.: Tight and compact convex relaxations for polynomial programming problems. Manuscript, LIX, École Polytechnique, F-91128 Palaiseau, France
3. Caprara, A., Locatelli, M.: Global optimization problems and domain reduction strategies. Math. Program. **125**(1), 123–137 (2010)
4. Dalkiran, E.: http://filebox.vt.edu/users/dalkiran/website/#ProblemRRLT (2011)
5. Dalkiran, E., Sherali, H. D.: Theoretical filtering of RLT bound-factor constraints for solving polynomial programming problems to global optimality. Manuscript, Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA (2011)
6. Floudas, C.A., Visweswaran, V.: Quadratic optimization. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization, pp. 217–270. Kluwer, Boston, MA (1995)
7. Gill, P.E., Murray, W., Saunders, M.A.: An SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005)
8. Liberti, L.: Linearity embedded in nonconvex programs. J. Glob. Optim. **33**, 157–196 (2005)
9. Liberti, L.: Effective RLT tightening in continuous bilinear programs. Internal Report 2003.18, Politecnico di Milano, 20133 Milano, Italy
10. Liberti, L., Pantelides, C.C.: An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. J. Glob. Optim. **36**, 161–189 (2006)
11. MATLAB: version 7.6.0 (R2008a). The MathWorks Inc., Natick, MA (2008)
12. Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. J. Glob. Optim. **8**(2), 107–138 (1996)
13. Sahinidis, N.V., Tawarmalani, M.: BARON 9.0.6: Global optimization of mixed-integer nonlinear programs. User's Manual (2010)
14. Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer, Boston, MA (1999)
15. Sherali, H.D., Dalkiran, E.: Combined bound-grid-factor constraints for enhancing RLT relaxations for polynomial programs. J. Glob. Optim. (2011). doi:10.1007/s10898-010-9639-0
16. Sherali, H.D., Dalkiran, E., Desai, J.: Enhancing RLT-based relaxations for polynomial programming problems via a new class of v-semidefinite cuts. Comput. Optim. Appl. (accepted for publication)
17. Sherali, H.D., Fraticelli, B.M.P.: Enhancing RLT relaxations via a new class of semidefinite cuts. J. Glob. Optim. **22**, 233–261 (2002)
18. Sherali, H.D., Smith, J.C., Adams, W.P.: Reduced first-level representations via the reformulation-linearization technique: results, counterexamples, and computations. Discrete Appl. Math. **101**, 247–267 (2000)
19. Sherali, H.D., Tuncbilek, C.H.: A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. J. Glob. Optim. **2**(1), 101–112 (1992)

20. Sherali, H.D., Tuncbilek, C.H.: A reformulation-convexification approach for solving nonconvex quadratic programming problems. J. Glob. Optim. **7**, 1–31 (1995)
21. Sherali, H.D., Tuncbilek, C.H.: Comparison of two reformulation-linearization technique based linear programming relaxations for polynomial programming problems. J. Glob. Optim. **10**, 381–390 (1997)
22. 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)
23. Shor, N.Z.: Dual quadratic estimates in polynomial and Boolen programming. Ann. Oper. Res. **25**, 163–168 (1990)
24. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**, 225–249 (2005)