Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations — Source link

Ruth Misener, Christodoulos A. Floudas

Institutions: Princeton University

Published on: 24 May 2012 - Mathematical Programming (Springer-Verlag)

Topics: Global optimization, Quadratic programming, Quadratic growth, Relaxation (approximation) and Packing problems

Noname manuscript No.
(will be inserted by the editor)

Global Optimization of Mixed-Integer Quadratically-Constrained Quadratic Programs (MIQCQP) through Piecewise-Linear and Edge-Concave Relaxations

Ruth Misener · Christodoulos A. Floudas

Received: November 14, 2011

Abstract We propose a deterministic global optimization approach, whose novel contributions are rooted in the edge-concave and piecewise-linear underestimators, to address nonconvex mixed-integer quadratically-constrained quadratic programs (MIQCQP) to ε-global optimality. The facets of low-dimensional $(n \leq 3)$ edge-concave aggregations dominating the termwise relaxation of MIQCQP are introduced at every node of a branch-and-bound tree. Concave multivariable terms and sparsely distributed bilinear terms that do not participate in connected edge-concave aggregations are addressed through piecewise-linear relaxations. Extensive computational studies are presented for point packing problems, standard and generalized pooling problems, and examples from GLOBALLib [55].

1 Introduction

Nonconvex quadratically-constrained quadratic programs (nonconvex QCQP) and those admitting integer variables (MIQCQP), are ubiquitous in process systems applications including heat integration networks, separation systems, reactor networks, reactor-separator-recycle systems, and batch processes (*e.g.*, [2, 5, 18, 24, 26, 28, 29, 31, 33, 42, 44, 45, 47, 48, 50, 66, 67, 70, 71, 88, 91]).

Our recent work has focused on the pooling problem, an optimization challenge of maximizing profit subject to feedstock availability, intermediate storage capacity, demand, and product specification constraints [40, 61, 63, 64]. The pooling problem, which is a MIQCQP under the assumption of linearly blending qualities, has important practical applications to many process systems engineering domains, includ-

The authors gratefully acknowledge support from the National Science Foundation (CBET – 0827907). R.M. is further thankful for her NSF Graduate Research Fellowship (DGE-0646086).

R. Misener · C.A. Floudas
Department of Chemical and Biological Engineering, Princeton University, Princeton, NJ 08544-5263
Tel.: 609-258-4595, Fax: 609-258-0211
E-mail: floudas@titan.princeton.edu

ing petroleum refining, water systems, supply-chain operations, and communications [13, 44, 60, 88].

Large-scale pooling problems were addressed to ε-global optimality using a disjunctive relaxation formulation that activates appropriate under- and over-estimators in specific domain segments and integrates this relaxation scheme into a branch-and-bound global optimization algorithm. Similar underestimators have been exploited in an array of process network applications [18, 42, 45, 59, 66, 71, 91].

Complementing the piecewise-linear relaxations, we investigate cuts generated through edge-concave aggregations. Edge-concave functions admit a vertex polyhedral envelope and therefore have a convex hull consisting entirely of linear facets [58, 79, 80, 81]. We detect possible edge-concave aggregations of two or three variables in MIQCQP, determine the convex hull of the aggregated terms, and integrate facets that strictly dominate the termwise relaxation into our underestimation scheme. The facets that do not strictly dominate the termwise relaxation are still useful for interval arithmetic-based bounds reduction within the context of branch-and-bound global optimization.

The proposed global optimization algorithm, whose novel contributions are rooted in the underestimators, determines an appropriate relaxation scheme employing piecewise-linear underestimators and edge-concave relaxations and integrates the resulting mixed-integer linear program (MILP) into a branch-and-bound global optimization algorithm. We begin in Section 2 by defining MIQCQP and situating our investigations within the context of other work. Section 3 describes the algorithmic decisions with respect to relaxation formulation (§3.1), bounds reduction (§3.2), and other choices (§3.3). Section 4 computationally investigates the performance of the piecewise-linear underestimators and edge-concave relaxations and seeks to elucidate the conditions under which either is advantageous. Section 5 concludes the paper.

2 Problem Introduction and Literature Review

We consider Mixed-Integer Quadratically-Constrained Quadratic Programs:

$$\begin{aligned} \min\ & x^T \cdot Q_0 \cdot x + a_0 \cdot x + c_0 \cdot y \\ \text{s.t.}\ & x^T \cdot Q_m \cdot x + a_m \cdot x + c_m \cdot y \le b_m \quad \forall\, m \in \{1, \ldots, M\} \\ & x \in \Re^C;\ y \in \{0, 1\}^B \end{aligned} \tag{MIQCQP}$$

where C, B, and M represent the number of continuous variables, binary variables, and constraints, respectively. We assume that it is possible to infer finite bounds $\left[x_i^L, x_i^U\right]$ on the variables participating in nonlinear terms, that matrices $Q_m\ \forall\, m \in \{0, \ldots, M\}$ are upper triangular, and that the continuous component may be nonconvex (*i.e.*, $\exists\, m \in \{0, \ldots, M\} : Q_m \not\succeq 0$). We alternatively denote quadratic products as:

$$x^T \cdot Q_m \cdot x = \sum_{i=0}^{C} \sum_{j=i}^{C} Q_{m,i,j} \cdot x_i \cdot x_j \quad \forall\, m \in \{0, \ldots, M\}$$

Our approach is related to the work of Al-Khayyal and Falk [6] that globally optimized bilinear programs by replacing each nonconvex term $(x_i \cdot x_j)$ in MIQCQP with an auxiliary variable $(z_{i,j})$ representing its convex hull [54]:

$$z_{i,j} \geq \max\left\{x_i \cdot x_j^L + x_i^L \cdot x_j - x_i^L \cdot x_j^L;\; x_i \cdot x_j^U + x_i^U \cdot x_j - x_i^U \cdot x_j^U\right\} \tag{1}$$

$$z_{i,j} \leq \min\left\{x_i \cdot x_j^L + x_i^U \cdot x_j - x_i^U \cdot x_j^L;\; x_i \cdot x_j^U + x_i^L \cdot x_j - x_i^L \cdot x_j^U\right\}. \tag{2}$$

and integrated the resulting linear relaxation into a branch-and-bound global optimization algorithm [27]. For the sake of brevity, we do not discuss basic branch-and-bound global optimization in this paper, but the reader is referred to an array of excellent books, research papers, and review articles [3, 4, 16, 27, 30, 32, 38, 75, 82].

This paper's primary contribution is rooted in tightening the linear relaxation of MIQCQP, so we limit our discussion of MIQCQP to previous successful efforts towards generating tight relaxations. We mention several methodologies. First, there have been advances that reduce MIQCQP to bilinear programs with the fewest number of complicating variables [19, 41]. This technique transforms MIQCQP into a form that can be exploited by primal-dual global optimization algorithms [5, 28, 36, 34, 35, 36, 89, 90].

Efforts towards reformulating MIQCQP have also taken the form of reducing the number of nonconvex bilinear terms [12, 17, 49]. For example, Ben-Tal et al. [17] showed that the dual of MIQCQP is sometimes smaller than the primal, Audet et al. [12] eliminated bilinear terms in the pooling problem through mass balances at the intermediate nodes, and Liberti and Pantelides [49] generalized the contribution of Audet et al. [12] to automatically eliminate unnecessary bilinear terms in MIQCQP.

A number of methods add redundant constraints to MIQCQP that tighten the MILP relaxation [9, 11, 22, 45, 67, 70, 73, 75, 76, 77, 78, 82]. We distinguish between techniques that automatically generate cuts for MIQCQP [9, 11, 22, 73, 75, 76, 77, 78] and redundant equations that are designed through close analysis of specific models [45, 67, 70, 82]. The generic approaches include those based on the Reformulation-Linearization Technique (RLT) [11, 75, 76, 77, 78] and efforts to integrate semidefinite programming (SDP) relaxations (or linear projections of SDP relaxations) into the underestimation scheme [9, 14, 22, 73, 72]. The model-specific approaches are based on careful analysis of optimization problem classes [9, 45, 67, 70, 82]

Rather than adding relaxations of redundant nonlinear constraints to the MILP relaxation of MIQCQP, an alternative set of techniques adds cuts to strengthen the relaxation of specific equations through eigenvector projections [23, 65, 69, 72], polyhedral facets [10, 15, 21], or the KKT necessary optimality conditions [83, 84]. The use of polyhedral facets is motivated by the following observation: although Equations (1) – (2) represent the convex hull of a single bilinear term, the sum of these termwise convex hulls in the MILP relaxation of objective or constraint $m \in \{0, \ldots, M\}$ does not necessarily generate the convex hull of m itself. Therefore, there has been work towards uncovering the vertex polyhedral properties of a bilinear equation to generate a family of valid cuts that characterize the convex hull [10, 15, 21, 56, 57, 58, 68].

These polyhedral facets can be alternatively determined through edge-concave relaxations. Edge-concave functions admit a vertex polyhedral envelope and therefore have a convex hull consisting entirely of linear facets [58, 79, 80, 81]. Although we only investigate low-dimensional edge-concave aggregations of MIQCQP in this paper, the edge-concave relaxation paradigm is relevant to a much broader class of expressions because it generates the convex hull of aggregated terms that follow a few simple rules [63, 79]. The derivation of explicit facets of the convex hull for trilinear monomials by Meyer and Floudas [56, 57] uses the same triangulation principles.

A final set of methods which tighten the linear relaxation of MIQCQP construct alternative relaxations for $z_{i,j} = x_i \cdot x_j$. Linderoth [51] generated termwise convex envelopes over triangular regions rather than solely rectangles. It is also possible to generate an alternative relaxation of $z_{i,j} = x_i \cdot x_j$ which is tighter than the convex hull of an individual bilinear term through the *ab initio* piecewise relaxation of non-convex bilinear terms as first developed by Meyer and Floudas [59] and Karuppiah and Grossmann [45]. Recognizing the importance of formulating these piecewise-linear relaxations in the most computationally effective manner possible, Wicaksono and Karimi [91] introduced fifteen mathematically-equivalent alternative formulations and compared the relaxation performance on several test cases. We recently proposed five additional piecewise-linear formulations and conducted a comprehensive comparative study on the computational performance of these formulations over a collection of benchmark pooling problems [40]. Hasan and Karimi [42] studied the possibility of bivariate partitioning, that is, segmenting both variables participating in each bilinear term. Other groups who have used piecewise-linear underestimators include: Bergamini et al. [18] in their Outer Approximation for Global Optimization Algorithm; Saif et al. [71] in a reverse osmosis network case study; and Pham et al. [66] in a fast-solving algorithm that generates near-optimal solutions.

Each of the previously-mentioned partitioning schemes requires a number of binary variables that scales linearly with the number of disjunctive segments in the relaxation. Vielma and Nemhauser [85] and Vielma et al. [86] recently proposed modeling piecewise functions with a number of binary switches that scales logarithmically with the number of partitions. Motivated by their work, we recently introduced a novel formulation for the logarithmically-sized piecewise relaxation of MIQCQP and tested the performance of this new formulation [64].

The primary novelty of the methodology described in this paper is rooted in the integration of edge-concave and piecewise-linear relaxations that tightly underestimate MIQCQP. The edge-concave relaxations consider low-dimensional $(n \leq 3)$ term aggregations in an effort to reduce the complexity of deriving cutting planes. Other than our own work with respect to relaxing the polynomial non-exhaust benzene emissions function, this contribution represents, to the best of our knowledge, the first effort towards integrating the edge-concave based relaxation methodology described by Meyer and Floudas [58] into a branch-and-bound global optimization algorithm. For the piecewise-linear relaxations [40, 61, 63, 64], we propose a new preprocessing step to choose the best variables for partitioning.

3 Theoretical and Algorithmic Development

This section describes the algorithmic decisions with respect to relaxation formulation (§3.1) for generating the edge-concave facets (§3.1.1 – 3.1.3), eigenvector projections (§3.1.4), piecewise-linear underestimators (§3.1.5 – 3.1.8), bounds reduction (§3.2) through RLT (§3.2.1) and the edge-concave paradigm (§3.2.2), and other accessory choices (§3.3).

3.1 Tight Relaxation Generation

3.1.1 Properties of Edge-Concave Facets

Our discussion of edge-concave facets is based on work of Tardella [79, 80, 81] and Meyer and Floudas [58].

Definition 3.1.1.1 [80]: Let $D = \{d_1, \ldots, d_k\}$ be a set of vectors such that for each edge E of a polyhedron P, D contains a vector parallel to E. Function $f(x_1, \ldots, x_n)$ is *edge-concave* on P if and only if it is concave on all segments in P that are parallel to an edge of P.

Tardella [79] proved that edge-concave functions admit a vertex polyhedral envelope (*i.e.*, that the facets of the convex hull can be determined solely from the vertices of P). Although edge-concavity represents a broad class of functions, we limit ourselves to several special cases to simplify the detection and exploitation of the vertex polyhedral envelope. Tardella [80] observed that for the special case of twice-continuously differentiable function f defined on a box P, the edge-concave definition is equivalent to $\frac{\partial^2 f}{\partial x_i^2} \leq 0 \; \forall \; i = 1, \ldots, n$ [80]:
Applying this property, observe that the following functions are edge-concave on a box: $f(x_i) = \alpha \cdot x_i$; $f(x_i) = -1 \cdot |\alpha| \cdot x_i^2$; and $f(x_i, x_j) = \alpha \cdot x_i \cdot x_j$ for $x_i, x_j \in \Re$ and scalar α. Because the sum of edge-concave functions is itself edge-concave [68, 80], the objective and constraints in MIQCQP can each be decomposed into the sum of (1) an edge-concave function, (2) a convex function, and (3) an integer linear function.

While performing such a decomposition and generating the vertex polyhedral envelope of the edge-concave portion of each equation in MIQCQP would generate a tight relaxation, the brute force method of checking each of the simplices defined by the vertices of polyhedron P for facet-defining hyperplanes is combinatorially complex [10, 15, 80]. Therefore, like Meyer and Floudas [58] and Anstreicher and Burer [10], we limit ourselves to low-dimensional cases ($n \leq 3$) so that each edge-concave function has six or fewer distinct facet-defining hyperplanes. Specifically, we decompose the edge-concave portion of each $m \in \{0, \ldots, M\}$ into aggregated functions of the form:

$$\begin{aligned} f(x_i, x_j, x_k) = & \alpha_1 \cdot x_i \cdot x_i + \alpha_2 \cdot x_i \cdot x_j + \alpha_3 \cdot x_i \cdot x_k + \alpha_4 \cdot x_i + \alpha_5 \cdot x_j \cdot x_j + \\ & \alpha_6 \cdot x_j \cdot x_k + \alpha_7 \cdot x_j + \alpha_8 \cdot x_k \cdot x_k + \alpha_9 \cdot x_k \end{aligned} \quad \text{(EC-AGG)}$$

where $\alpha_1, \ldots, \alpha_9$ are scalars and α_1, α_5, α_8 are non-positive scalars. The twin goals of generating aggregations of form EC-AGG are (1) integrating dominant cuts into the MILP relaxation of MIQCQP and (2) adding equations that tighten a bounding scheme based on interval arithmetic (see discussion in §3.2.2). Aggregated functions of the form EC-AGG achieving both goals are most desirable, so we elucidate the conditions under which the facets of EC-AGG dominate the termwise relaxation of

MIQCQP. After detecting low-dimensional aggregations with facets that may dominate the termwise relaxation, we find aggregations augmenting the bounding scheme.

When the convex envelope of a sum is equivalent to the sum of the convex envelope, no dominant polyhedral cut can be introduced to tighten the sum of functions. Using the notation of Tardella [80, 81] and Meyer and Floudas [58]:

Definition 3.1.1.2 [81]: When the convex envelope of a sum of functions coincides with the sum of the convex envelopes of the functions, we say that the sum of functions is *sum decomposable*.

Therefore, only low-dimensional aggregations that are *not* sum decomposable may have a LP relaxation that dominates the termwise relaxation and we do not search for dominant cuts within sum decomposable EC-AGG. Separable functions are clearly sum decomposable. Function $h = f + g$ is sum decomposable when g is affine [80]. Meyer and Floudas [58] exploited pairwise compatibility as a sufficient test to determine if *almost separable* functions are sum decomposable. To further characterize sum decomposibility, we define $\mathrm{conv}_S(f)$ on $\mathrm{conv}(S)$ as the convex envelope of f on the convex hull of $S \subset \Re^n$ and state the following result from Tardella [81], which generalizes work of Meyer and Floudas [58], without proof:

Theorem 3.1.1.3 [81]: Let V_p be the set of vertices on a polytope P, define edge-concave functions $f, g \mapsto \Re$ with facet representations $\{f_i : i \in I\}$ and $\{g_j : j \in J\}$ defining the convex hull of f and g, respectively, and let $F_i = \{x \in P : \mathrm{conv}_{V_P}(f)(x) = f_i(x)\}$, $i \in I$ and $G_j = \left\{x \in P : \mathrm{conv}_{V_P}(g)(x) = g_j(x)\right\}$, $j \in J$ denote the linearity domains of $\mathrm{conv}_{V_P}(f)$ and $\mathrm{conv}_{V_P}(g)$ (*i.e.*, the sets F_i and G_i are polyhedra composed of facet-defining hyperplanes f_i and g_j). The following are equivalent:

1 $\mathrm{conv}_{V_P}(f) + \mathrm{conv}_{V_P}(g)$ is vertex polyhedral;
2 $\mathrm{conv}_{V_P}(f) + \mathrm{conv}_{V_P}(g) = \mathrm{conv}_{V_P}(f+g)$;
3 $F_i \cap G_j$ has all vertices in $V_P\ \forall\, i \in I,\ j \in J$.

For the specific case of *almost separable* function $h(x,y,z) = f(x,y,z) + g(x,y,z) = \hat{f}(x,y) + \hat{g}(x,z)$ defined on $V = X \times Y \times Z$ where X, Y, Z are the vertex sets of polytopes, then the three conditions listed above are further equivalent to:

4 $F_i^X \cap G_j^X$ has all vertices in X for all linearity domains F_i of $\mathrm{conv}_{V_P}(f)$ and G_i of $\mathrm{conv}_{V_P}(g)$.

Based on the preceding, we make the following observations specific to MIQCQP:

Observation 3.1.1.4: Including or excluding the affine sum $\alpha_4 \cdot x_i + \alpha_7 \cdot x_j + \alpha_9 \cdot x_k$ in EC-AGG does not make a difference in tightening the relaxation of MIQCQP [80].

Observation 3.1.1.5: If $\alpha_3 = \alpha_6 = 0$, the function $f(x_i, x_j, x_k) = f(x_i, x_j) + f(x_k)$ is separable and there is no advantage to aggregating $f(x_i, x_j)$ with $f(x_k)$. Symmetric cases hold for $\alpha_2 = \alpha_3 = 0$ and $\alpha_2 = \alpha_6 = 0$.

Observation 3.1.1.6: The function $f(x_i, x_j) = \alpha_1 \cdot x_i \cdot x_i + \alpha_2 \cdot x_i \cdot x_j + \alpha_5 \cdot x_j \cdot x_j$ is sum decomposable. To see this, note that convex envelope $[\mathrm{conv}(\alpha_2 \cdot x_i \cdot x_j)]$ is vertex polyhedral and both the convex envelopes $[\mathrm{conv}(\alpha_1 \cdot x_i \cdot x_i)]$ and $[\mathrm{conv}(\alpha_5 \cdot x_j \cdot x_j)]$ are affine functions (namely, $\alpha_1 \cdot \left[x_i \cdot (x_i^U + x_i^L) - x_i^U \cdot x_i^L\right]$ and $\alpha_5 \cdot \left[x_j \cdot (x_j^U + x_j^L) - x_j^U \cdot x_j^L\right]$).

Because $[\mathrm{conv}(\alpha_1 \cdot x_i \cdot x_i) + \mathrm{conv}(\alpha_2 \cdot x_i \cdot x_j) + \mathrm{conv}(\alpha_5 \cdot x_j \cdot x_j)]$ is the sum of a vertex polyhedral function and two affine functions, it is itself vertex polyhedral. By Condition 1 of Theorem 3.1.1.3, the facets of aggregate function $f(x_i, x_j)$ introduce no dominant cuts to MIQCQP.

Observation 3.1.1.7: Function $f(x_i, x_j, x_k) = \alpha_2 \cdot x_i \cdot x_j + \alpha_6 \cdot x_j \cdot x_k$ is almost separable and meets the fourth condition in Theorem 3.1.1.3, so it is sum decomposable and its convex envelope is equivalent to the sum of the termwise convex envelopes.

Observation 3.1.1.8: Combining Observations 3.1.1.4 – 3.1.1.7, note that functions of form EC-AGG in MIQCQP satisfying $\alpha_2 \neq 0$, $\alpha_3 \neq 0$, $\alpha_6 \neq 0$ (called EC-AGG$^{\mathrm{TRIP}}$ hereafter) are necessary to generate polyhedral cuts dominating the termwise relaxation of MIQCQP.

Observation 3.1.1.9: Even for aggregations with $\alpha_2 \neq 0$, $\alpha_3 \neq 0$, $\alpha_6 \neq 0$, not all the facets of EC-AGG are necessarily tighter than the termwise relaxation of EC-AGG. Therefore, we seek facets that introduce dominant cuts to MIQCQP. We compare each of the unique facets of EC-AGG$^{\mathrm{TRIP}}$ with each of the eight possible termwise underestimators. A termwise underestimator cannot strictly dominate one of the EC-AGG$^{\mathrm{TRIP}}$ facets, so any facet that is not equal to one of the eight termwise relaxations must itself be a dominant cut.

The preceding necessary (Observation 3.1.1.8) and sufficient (Observation 3.1.1.9) conditions allow us to augment the MILP relaxation of MIQCQP with the facets of EC-AGG$^{\mathrm{TRIP}}$ that strictly dominate the termwise relaxation of EC-AGG$^{\mathrm{TRIP}}$. It is important to note that, for the secondary goal of bounds reduction, we detect nonseparable but sum decomposable aggregations and include linear terms in EC-AGG (see §3.2.2 for a discussion of variable bounding).

3.1.2 Implementation for Generating EC-AGG Aggregations and Determining Facet-Defining Hyperplanes

The procedure for (1) generating aggregations of the form EC-AGG within MIQCQP and (2) determining the facet-defining hyperplanes of EC-AGG that dominate the termwise relaxation of MIQCQP is presented in this section. To generate aggregations EC-AGG, we begin by searching each equation $m \in \{0, \ldots, M\}$ for nonzero triplets $\alpha_2 \cdot x_i x_j = Q_{m,i,j} \cdot x_i x_j$, $\alpha_3 \cdot x_i x_k = Q_{m,i,k} \cdot x_i x_k$, $\alpha_6 \cdot x_j x_k = Q_{m,j,k} \cdot x_j x_k$ (recalling that Q_m is upper triangular, note that $i < j < k$). To avoid introducing too many extra cuts, we place each term in equation m in at most one aggregation (*i.e.*, if $Q_{m,i,j} \cdot x_i x_j \in$ EC-AGG$_{m,i,j,k}$, then $Q_{m,i,j} \cdot x_i x_j \notin$ EC-AGG$_{m,i,j,\ell}$ for $k \neq \ell$). After aggregating triplets, we also aggregate nonseparable, sum decomposable edge-concave terms for the purpose of bounds reduction (see §3.2.2). Our aggregation scheme is generated in a preprocessing step and does not change over the course of the branch-and-bound global optimization algorithm.

Meyer and Floudas [58] designed a method to generate the facets of the convex envelope of any edge-concave function with dimension three or fewer. For three dimensional aggregations, Meyer and Floudas [58] proposed (1) determining the dominance pattern on each minimal affine dependency of the cube, (2) matching the dominance pattern to a reorientation of one of the six triangulation types of the 3-cube, and (3) calculating the facets of the convex envelope. Because EC-AGG is low dimensional, the computational load of re-calculating the appropriate triangulations at each node of the branch-and-bound tree is minimal. We do store the relaxation of each aggregation so that the convex and concave hulls are not re-computed if variable bounds remain the same between two calls to the facet-generating routine.

Illustration 3.1.2.1: As an example of determining the facets of the convex hull, consider function $f(x_i, x_j, x_k) = \alpha_2 \cdot x_i \cdot x_j + \alpha_3 \cdot x_i \cdot x_k$ such that $\alpha_2 > 0$ and $\alpha_3 < 0$. By Observation 3.1.1.7, this function is sum decomposable, but it is still aggregated for the purpose of bounds tightening. These dominant and non-dominated circuits are compared to the six equivalence classes of the 3-cube and a re-orientation of what Figure 4.4 in Meyer and Floudas [58] denotes triangulation type A is found to be both a superset of the strictly dominant affine dependencies and a subset of the non-dominated affine dependencies. The facets of the convex envelope for $f(x_i, x_j, x_k)$ are determined using triangulation type A.

Illustration 3.1.2.2: As a specific example of detecting dominant cuts by applying Observations 3.1.1.8 and 3.1.1.9, consider EC-AGG$^{\text{TRIP}}$:

$$\begin{aligned} f(x_i, x_j, x_k) =& 0.5 \cdot x_i x_j - 0.9 \cdot x_i x_k - x_j x_k \\ & x_i \in [-10, 0]\,;\; x_j \in [4, 10]\,;\; x_k \in [7, 10] \end{aligned}$$

with McCormick relaxation:

$$f(x_i, x_j, x_k) \geq \begin{cases} -7 \cdot x_i - 15 \cdot x_j + 5 \cdot x_k - 30 \\ -7 \cdot x_i - 12 \cdot x_j - 1 \cdot x_k \\ -4.3 \cdot x_i - 15 \cdot x_j - 4 \cdot x_k + 60 \\ -4.3 \cdot x_i - 12 \cdot x_j - 10 \cdot x_k + 90 \\ -4 \cdot x_i - 10 \cdot x_j + 5 \cdot x_k - 50 \\ -4 \cdot x_i - 7 \cdot x_j - 1 \cdot x_k - 20 \\ -1.3 \cdot x_i - 10 \cdot x_j - 4 \cdot x_k + 40 \\ -1.3 \cdot x_i - 7 \cdot x_j - 10 \cdot x_k + 70 \end{cases} \qquad \text{(TW-RLX)}$$

and facets of the convex envelope determined through the Meyer and Floudas [58] algorithm:

$$f(x_i, x_j, x_k) \geq \begin{cases} -7 \cdot x_i - 15 \cdot x_j + 5 \cdot x_k - 30 \\ -1.3 \cdot x_i - 7 \cdot x_j - 10 \cdot x_k + 70 \\ -5.2 \cdot x_i - 12 \cdot x_j - 1 \cdot x_k + 18 \\ -3.1 \cdot x_i - 10 \cdot x_j - 4 \cdot x_k + 40 \\ -4 \cdot x_i - 10 \cdot x_j - 1 \cdot x_k + 10 \\ -4.3 \cdot x_i - 12 \cdot x_j - 4 \cdot x_k + 48 \end{cases} \qquad \text{(EC-RLX)}$$

The first two constraints of EC-RLX are equivalent to the first and last cuts of TW-RLX, so we do not add them to the MILP relaxation of MIQCQP. However, we do augment the MILP relaxation of MIQCQP with the final four cuts of EC-RLX.

We conclude this section with a summary of our strategy. As a preprocessing step, we decompose each of the equations in MIQCQP into aggregations with the form EC-AGG. Then, for the MILP relaxation of every branch-and-bound tree node, we recompute the facets of the edge-concave terms with $\alpha_2 \neq 0$, $\alpha_3 \neq 0$, $\alpha_6 \neq 0$ and add the cuts that dominate the termwise relaxation of MIQCQP.

3.1.3 Edge-Concave Aggregation for Specific Classes of MIQCQP

Stepping back from this analysis of generic MIQCQP, we make observations specific to process network problems and the density of the quadratic matrices Q_m.

Property 3.1.3.1: Process networks equations are generally sum decomposable.

Nonlinearities in process network constraints typically track node quality:

$$\sum_{i \in I} p_{i,n}^k \cdot f_{i,n} - p_n^k \cdot \sum_{j \in J} f_{j,n} \quad \forall\, n \in N;\ k \in K \tag{3}$$

where $f_{i,n}$ is the flow of stream i into node n, $p_{i,n}^k$ is quality k of stream i entering node n, p_n^k is quality k of node n itself, and $f_{j,n}$ is the flow of stream j leaving node n. Expression 3 excludes nonlinear blending and costing rules also present in process networks problems, but Expression 3 (or a simplification thereof) can be found in problems related to pooling, wastewater systems, data reconciliation, heat exchanger networks, distillation sequences, etc. [2, 18, 26, 28, 29, 31, 42, 44, 45, 60, 61, 63, 64, 66, 67, 70, 71, 88, 91]. For example, the product quality bounds in a standard pooling problem with a single layer of intermediate nodes are represented as:

$$\sum_{l:(l,j)} p_{l,k} \cdot y_{l,j} + \sum_{i:(i,j)} C_{i,k} \cdot z_{i,j} \leq \sum_{l:(l,j)} P_{j,k}^U \cdot y_{l,j} + \sum_{i:(i,j)} P_{j,k}^U \cdot z_{i,j} \quad \forall\, j,k \tag{4}$$

where the variables in the Equation 4 are $p_{l,k}, y_{l,j}, z_{i,j}$ and the nonlinear bilinear terms (in the first left-hand-side summation) are equivalent to the first summation in Expression 3 [60].

To see that Expression 3 is sum decomposable, we begin by noting that the two summations are separable from one another (and therefore sum decomposable). The terms in the first summation are also separable because each of the variables $p_{i,n}^k$ and $f_{i,n}$ appear only once in Expression 3. The second summation $p_n^k \cdot \sum_{j \in j} f_{j,n}$ is almost separable and the participating terms satisfy Condition 4 of Theorem 3.1.1.3.

Because the termwise relaxation of Expression 3 is equivalent to its convex hull, there are no additional cuts that can be introduced on an equation-by-equation basis to tighten process network MIQCQP. Thus, for the specific case of process network problems, neither aggregating edge-concave functions as presented in this paper nor generating multiterm relaxations as suggested by Bao et al. [15] tightens the relaxation of MIQCQP. This observation that the termwise convex relaxation of MIQCQP is equivalent to its convex hull motivates the advantage of using generic RLT techniques [75, 76, 77, 78], specially designed cuts [45, 67, 70, 82], or piecewise-linear relaxations [18, 40, 42, 45, 59, 61, 63, 64, 66, 71, 91] to tighten sum decomposable process networks problems.

As a second observation regarding MIQCQP, note from the computational results of Bao et al. [15] that their multiterm cuts are especially effective for dense matrices (*i.e.*, for problems where there will be many aggregations EC-AGG with $\alpha_2 \neq 0$, $\alpha_3 \neq 0$, $\alpha_6 \neq 0$). By the MIQCQP-specific observations we made in response to Theorem

1 [81], these are the same circumstances under which we expect a tighter relaxation from edge-concave aggregations. The low-dimensional aggregations proposed here are quickly-generated (and therefore re-computed at every node of the branch-and-bound tree). The cuts of Bao et al. [15] are applicable to dimensions higher than three but are more costly to generate and therefore they could be used effectively only at the root node. The computational studies presented in Section 4 use only low-dimensional aggregations, but a hybrid algorithm could use the Bao et al. [15] multiterm relaxation at the root node and the edge-concave aggregations we discuss in subsequent nodes.

3.1.4 Eigenvector Projections

Eigenvector projections are a common relaxation strategy for nonconvex quadratic programming problems [23, 65, 69] that also have been used for underestimating MIQCQP [72]. The major difference between the seminal strategy proposed first by Rosen and Pardalos [69] and the more recent effort of Saxena et al. [72] is that Saxena et al. [72] do not transform the variables participating in connected, nonconvex quadratic terms into separable terms but rather augment the relaxation of each quadratic expression with a convex relaxation of the eigenvectors.

The Saxena et al. [72] treatment is suited for MIQCQP because nonlinearities may appear in multiple equations within MIQCQP and variable transformation is therefore undesirable. Our approach is similar to that of Saxena et al. [72] except that (1) we segment each quadratic matrix $x^T \cdot Q_m \cdot x$ into separable, multivariable terms $\sum_i x_i^T \cdot Q_{m,i} \cdot x_i$ before finding the eigenvectors and eigenvalues of each separable multiterm $Q_{m,i}$ and (2) we only augment the MILP relaxation of MIQCQP with eigenvector projections of $x_i^T \cdot Q_{m,i} \cdot x_i$ that are *not* sum decomposable (see Definition 3.1.1.2).

3.1.5 Definition of the Piecewise-Linear Underestimators

We have previously discussed our use of piecewise-linear underestimators for the pooling problem [40, 61, 63, 64]. Like other process networks problems, the pooling problem is sum decomposable and therefore the edge-concave strategies described in the previous section do not tighten the relaxation of MIQCQP. However, by using appropriately-designed RLT-style cuts and constructing piecewise-linear underestimators that are tighter than the convex hull of each term, we were able to effectively close the optimality gap for large-scale problems [61, 63, 67, 82].

Suppose we wish to generate an underestimator for bilinear term $z = x \cdot y$ that is tighter than its convex hull. The envelope in Equations (1) – (2) is dependent on the size of the domain, so we partition variable x into N_P segments of length $a = (x^U - x^L)/N_P$.

We consider three MILP reformulation schemes scaling either linearly or logarithmically with the number of partitions that are outlined in Table 1. The first reformulation, which is presented in detail in Misener et al. [64], uses a number of binary variables that scales linearly with the number of partitions [40, 91].

The second reformulation is based on the work of Vielma and Nemhauser [85] and Vielma et al. [86] that recently proposed modeling piecewise functions with

Table 1: Additional variables and constraints for the relaxation of a bilinear term [64].

	Contin Vars	Binry Vars	Constraints
McC Hull	1	–	4
PW Linear Scheme	$N_P\ +1$	N_P	$N_P\ +8$
PW Log Scheme 1	$2\cdot N_P\ +1$	$\lceil \log_2 N_P \rceil$	$N_P+2\cdot\lceil \log_2 N_P \rceil+8$
PW Log Scheme 2†	$2\cdot\lceil \log_2 N_P \rceil+1$	$\lceil \log_2 N_P \rceil$	$3\cdot\lceil \log_2 N_P \rceil+6$

† Applicable to powers of two (*i.e.*, $\log_2 N_P = \lceil \log_2 N_P \rceil$)

a number of binary switches that scales logarithmically with the number of partitions (*i.e.*, $N_L = \lceil \log_2 N_P \rceil$). Although the following formulation maps from the partition containing x to λ using a base-2 representation, note that any injective function $B : \{1,\ldots,N_P\} \mapsto \{0,1\}^{\lceil \log_2 N_P \rceil}$ could formulate the SOS1-like constraints for the activation of exactly one of the N_P segments [85]. This logarithmic formulation uses binary switch $\lambda \in \{0,1\}^{N_L}$ and continuous switches $\Delta y \in \left[0, y^U - y^L\right]^{N_P}$ and $\hat{\lambda} \in [0,1]^{N_P}$:

Logarithmic Partitioning Scheme 1:

$$x^L + \sum_{n_L=1}^{N_L} 2^{N_L-n_L}\cdot a\cdot\lambda(n_L) \le x \le x^L + a + \sum_{n_L=1}^{N_L} 2^{N_L-n_L}\cdot a\cdot\lambda(n_L) \tag{5a}$$

$$\sum_{n_P=1}^{N_P} \hat{\lambda}(n_P) = 1 \tag{5b}$$

$$\sum_{n_P:\lfloor \frac{n_P-1}{2^{N_L-n_L}} \rfloor \pmod 2 = 0} \hat{\lambda}(n_P) \le (1-\lambda(n_L)) \quad \forall\, n_L \in \{1,\ldots,N_L\} \tag{5c}$$

$$\sum_{n_P:\lfloor \frac{n_P-1}{2^{N_L-n_L}} \rfloor \pmod 2 = 1} \hat{\lambda}(n_P) \le \lambda(n_L) \quad \forall\, n_L \in \{1,\ldots,N_L\} \tag{5d}$$

$$\Delta y(n_P) \le (y^U - y^L)\cdot\hat{\lambda}(n_P) \quad \forall\, n_P \in \{1,\ldots,N_P\} \tag{5e}$$

$$y = y^L + \sum_{n_P=1}^{N_P} \Delta y(n_P) \tag{5f}$$

$$z \ge x\cdot y^L + \sum_{n_P=1}^{N_P} \left[x^L + a\cdot(n_P-1)\right]\cdot\ \Delta y(n_P) \tag{5g}$$

$$z \ge x\cdot y^U + \sum_{n_P=1}^{N_P} \left[x^L + a\cdot\ n_P\right]\cdot\left[\Delta y(n_P) - (y^U - y^L)\cdot\hat{\lambda}(n_P)\right] \tag{5h}$$

$$z \le x\cdot y^L + \sum_{n_P=1}^{N_P} \left[x^L + a\cdot\ n_P\right]\cdot\ \Delta y(n_P) \tag{5i}$$

$$z \le x\cdot y^U + \sum_{n_P=1}^{N_P} \left[x^L + a\cdot(n_P-1)\right]\cdot\left[\Delta y(n_P) - (y^U - y^L)\cdot\hat{\lambda}(n_P)\right] \tag{5j}$$

$$x^L \le x \le x^U; \quad y^L \le y \le y^U \tag{5k}$$

When the number of partitions N_P is a power of two (*i.e.*, $N_L = \log_2 N_P = \lceil \log_2 N_P \rceil$), we use a second logarithmic formulation with binary switch $\lambda \in \{0, 1\}^{N_L}$ and continuous switches $\Delta y \in \left[0, y^U - y^L\right]^{N_L}$ and $s \in \left[0, y^U - y^L\right]^{N_L}$ [64].

3.1.6 Sharpness Properties for the Piecewise-Linear Underestimators

For these piecewise-linear relaxations to work effectively in the context of a MILP branch-and-bound solver, they must be sharp (*i.e.*, the linear programming relaxation of the linear and logarithmic MILP formulations must be equivalent to the convex hull of the MILP model). We state without proof the result of Wicaksono and Karimi [91] that the Linear Partitioning Scheme is sharp. Appendix A proves that the linear programming relaxation of Logarithmic Partitioning Scheme 1 is nondominated by the convex hull in Equations (1) – (2) and that the smaller Logarithmic Partitioning Scheme 2 is sharp in the case where the number of partitions N_P is a power of two.

Property 3.1.6.1 [91]: The linear programming relaxation of the Linear Partitioning Scheme is nondominated by the convex hull.

Property 3.1.6.2: The linear programming relaxation of Logarithmic Partitioning Scheme in 1 Equations (5g) - (5j) is nondominated by the convex hull.
Proof: See Appendix A.

Property 3.1.6.3: The linear programming relaxation of Logarithmic Partitioning Scheme 2 in Misener et al. [64] is nondominated by the convex hull when the number of partitions is a power of two (*i.e.*, $N_L = \log_2 N_P = \lceil \log_2 N_P \rceil$).
Proof: See Appendix A.

3.1.7 Piecewise-Linear Underestimators for Concave Quadratic Terms

When the preprocessing scheme chooses to partition a variable participating in a concave quadratic term, our treatment of the partitioned variable remains the same as presented in Section 3.1.5. However, the nonlinear image of the domain is represented according to the convex hull representation that Sherali [62, 74] defined for piecewise functions rather than an equivalent of the formulations in Section 3.1.5. This formulation is both sharp and locally ideal [46].

3.1.8 Variable Partitioning for the Piecewise-Linear Underestimators

The piecewise underestimating schemes will always be at least as good as the simple McCormick underestimating scheme [54]. However, despite the additional tightness advantage afforded by the piecewise-linear scheme, we want to avoid introducing too much extra machinery to the MILP relaxation of MIQCQP.

For concave multivariable terms identified via the preprocessing strategy outlined in Section 3.1.4, we create auxiliary variables for the inner product of an eigenvector and the participating variables $v_{m,n}^T x$ and partition along the auxiliary variables corresponding to the most negative eigenvalue. This is similar to the method of Rosen and Pardalos [69] except that we may not choose to partition along every eigenvector of every concave term and therefore prioritize the most negative eigenvalues. We partition as many as ten auxiliary variables corresponding to the eigenvectors.

For generic nonconvex terms that do not participate in concave expressions, we are motivated by past work that transformed MIQCQP into bilinear programs with the fewest number of complicating variables for input into a primal-dual based global optimization algorithm [5, 19, 28, 36, 34, 35, 36, 41, 89, 90]. Although this paper does not use a primal-dual based method, the following contributions assist our approach: (1) establishing a co-occurrence graph for bilinear terms and (2) identifying complicating variables.

We begin our selection of which variables to partition (and thereby which nonlinear terms to piecewise-linearly underestimate) by establishing the equivalent of the co-occurence graph proposed by Hansen and Jaumard [41] where the nodes represent the variables participating nonlinearly in MIQCQP and the edges represent the nonlinear terms $\{x_i \cdot x_j : i, j = 1, \ldots, n; i < j; \exists m \in \{0, \ldots, M\} : Q_{m,i,j} \neq 0 \wedge Q_{m,i,j} \notin \text{EC-AGG}_m^{\text{TRIP}}\}$ in MIQCQP.

We exclude edges representing bilinear terms that always participate in active edge-concave triplets from the co-occurence graph because we are already adding edge-concave-based cuts related to those nonlinear terms (§3.1.1 – 3.1.2). By excluding the variables that are always aggregated into a EC-AGG$^{\text{TRIP}}$ term, we effectively integrate the two complementary underestimation strategies of (1) low-dimensional polyhedral facets for equations that are not sum decomposable and (2) tight piecewise relaxations for the more sparsely-distributed remaining terms.

However, unlike a generalized Benders-based scheme where at least one variable in each bilinear term must be a complicating variable, we do not have to partition at least one variable in each bilinear term. Therefore, if our preprocessing scheme selects a variable i to partition, it excludes the possibility of partitioning any j where $\exists m \in \{0, \ldots, M\} : Q_{m,i,j} \neq 0$. In other words, there is at most one partitioned variable in each bilinear term. We begin by selecting the variable i participating in the greatest number of nonlinear terms (*i.e.*, the node i with the greatest number of associated edges). After excluding node i and its associated edges, we find the node $j : \nexists m \in \{0, \ldots, M\} : Q_{m,i,j} \neq 0$ with the greatest number of associated edges. For large problems where, even after this preprocessing scheme is employed, there are a large number of variables tagged as needing partitioning, we limit the number of variables to partition to 30.

3.2 Bounds Tightening

Feasibility-based bounds tightening (FBBT) is a commonly-used technique using interval arithmetic to infer variable bounds [3, 4, 16]. Because it is computationally inexpensive, we follow Androulakis et al. [8], Sherali and Tuncbilek [77], and Audet et al. [11] in augmenting a FBBT scheme with several additional sets of constraints. Namely, we add additional equations to the LP relaxation of MIQCQP based on the Reformulation Linearization Technique (RLT) and the edge-concave aggregations. We cycle through this augmented LP relaxation, inferring tighter bounds though interval arithmetic, until the volume of the hyperrectangle representing the bounds of the nonlinearly participating variables fails to fall by at least a factor of 0.95.

3.2.1 Reformulation Linearization Technique

The additional RLT constraints we use in the context of bounds reduction are derived from the work of Sherali and co-workers [75, 76, 77, 78]. For each (i, j) pair such that $\exists m \in \{0,\ldots,M\} : Q_{m,i,j} \neq 0$, *i.e.*, term $x_i \cdot x_j$ participates in MIQCQP, we add the termwise convex hull to the FBBT scheme.

For each equation $m \in \{0,\ldots,M\} : Q_m = 0 \cap c_m = 0$, *i.e.*, linear equations with solely continuous variables, we consider product of m with the bounds on each continuous variable j. To reduce the complexity, we only augment the FBBT scheme with products whose bilinear terms already actively participate in MIQCQP. Finally, for each pair of equations $m, n \in \{0,\ldots,M\} : Q_m = Q_n = 0 \cap c_m = c_n = 0$, we add products to FBBT when the products do not introduce any additional bilinear terms.

3.2.2 Edge-Concave Paradigm

The non-dominant facets of the edge-concave aggregations developed in Section 3.1.1 are very useful for bounds reduction. As a trivial example, consider function $f(x_i) = -x_i^2 + x_i \leq -1$ on domain $[0.5, 2]$. Following Observation 3.1.1.4, the preprocessing step of our implementation identifies this as a sum decomposable but still nonseparable term and aggregates the sum. A FBBT bounds tightening scheme without the edge-concave paradigm would have allowed these bounds to persist, but performing standard interval arithmetic on the edge-concave derived facet $\left[-x_i^L - x_i^U + 1\right] \cdot x + \left[2 \cdot (x_i^L + x_i^U) - 4\right] \leq -1$ tightens the variable bounds from $[0.5, 2]$ to $[4/3, 2]$ to $[11/7, 2]$ and would eventually converge to $[(1+\sqrt{5})/2, 2]$ if we allowed the process to continue beyond the volume improvement parameter 0.95.

3.3 Other Global Optimization Considerations

We have employed a number of typical algorithmic choices to implement our branch-and-bound algorithm [3, 4, 6, 8, 16, 27, 82]. In addition to using FBBT at each node of the branch-and-bound tree, we tighten the root node relaxation of MIQCQP using optimality-based bounds tightening (OBBT) with the same volume improvement parameter 0.95 used for the FBBT scheme [53]. After the root node, OBBT continues as long as it significantly tightens nonlinearly participating variables. A parent node deactivates OBBT in its children nodes once the volume representing the hyperrectangle of the variable bounds fails to fall by a factor of at least 0.95 [64]. OBBT operates on the LP relaxation of MIQCQP rather than addressing any binary terms in MIQCQP or using the tight piecewise-linear relaxations [6, 54].

We select an appropriate branching variable via reliability branching, a technique that combines dynamic strong branching with a pseudocost heuristic to predict the best branching variable [1, 16]. We do not branch on integer variables because our MILP relaxations of MIQCQP are addressed directly using a MILP solver. In the notation of Achterberg et al. [1], we use reliability parameter $\eta_{\mathrm{REL}} = 8$ and maximum number of simplex iterations $\gamma_{\mathrm{ITER}} = 1000$. We scale the pseudocosts with the infeasibility of a variable (**rb-inf** in the analysis of Belotti et al. [16]) and continue to update the pseudocosts even after they are deemed reliable. Although we generally find the advantage of exploring fewer branch-and-bound tree nodes worth the computational expense of generating reliable pseudocosts, we reduce the reliability parameter η_{REL} to $\lfloor \frac{\mathrm{CPU_{MAX}}}{100 \cdot \mathrm{CPU_{ROOT}}} \rfloor$ if the solution time for the root node of the branch-and-bound tree takes a significant fraction of the total allotted CPU limit. When using the piecewise-linear relaxation schemes, we use a plain, maximum error-based branching scheme that branches on the variable with the greatest difference between the linear programming relaxation and the nonlinear representation (*i.e.*, branch on the variable with $\arg\max_i \sum_j |z_{i,j} - x_i \cdot x_j|$). We branch at a convex combination of the variable midpoint $(\lambda = 0.15)$ and the solution to the MILP relaxation $(1 - \lambda = 0.85)$ but require that the branching point be a minimum distance away from the variable bounds $(x_i^L + 0.10 \cdot (x_i^U - x_i^L) \le x_i^{\mathrm{BRANCH}} \le x_i^U - 0.10 \cdot (x_i^U - x_i^L))$.

We initialize our search for good upper bounds using feasible solutions to the MILP relaxation of MIQCQP at every branch-and-bound tree node.

4 Computational Studies

4.1 Experimental Implementation

We performed our computational studies on a Linux workstation containing one Intel Core 2 Quad processor with four 2.83 GHz cores. Our code base is written in C++ and interfaces CPLEX 11.1 [43] for the MILP relaxations, SNOPT 5.3 [39] for the local NLP solves, and LAPACK [7] for determining the facets of the edge-concave functions and the eigenvectors and eigenvalues of each connected multiterm. Our code is itself built as a library and linked to GAMS 23.6 [20] using the GAMS Modeling Object (GMO) interface. The code linking our solver to GAMS is adapted from the COIN-OR/GAMSLinks project [52, 87].

We ran each of the 43 computational experiments in GAMS 23.6 [20] under two termination criteria: (1) an optimality gap $\varepsilon = \frac{UB-LB}{|LB|} \le 1 \times 10^{-6} = 1 \times 10^{-4}\%$ and (2) a 7200 CPU s time limit. We consider point packing problems [9], process network problems [37, 62, 64], and other examples from GLOBALLib [37, 55]. Comparisons between the algorithmic components are based on performance profiles first defined by Dolan and Moré [25]:

$$\begin{aligned} t_{p,s} &\equiv \text{Performance metric (\textit{e.g.}, CPU s) for problem } p \text{ by technique } s \in S \\ r_{p,s} &\equiv \frac{t_{p,s}}{\min\left\{t_{p,s'} \;:\; s' \in S\right\} \;\forall p \in P};\; s \in S \\ \rho_s(\tau) &= \frac{1}{n_p} \operatorname{size}\left\{p \in P \;:\; r_{p,s} \le \tau\right\} \end{aligned}$$

The plots in this paper, which diagram $\rho_s(\tau)$ as a function of τ, were generated using the MATLAB performance profile implementation from `http://www.mcs.anl.gov/~more/cops/perf.m`.

4.2 Test Cases

Point Packing: Using the notation of Anstreicher [9], we consider the point packing problem of maximizing the separation distance between n points in the unit square:

$$\begin{array}{lll} \min & -\theta & \\ \text{s.t.} & -(x_i - x_j)^2 - (y_i - y_j)^2 \le -\theta, & 1 \le i < j \le n \\ & x_{i+1} \le x_i & 1 \le i \le n-1 \\ & 0 \le x_i \le \frac{1}{2} & 1 \le i \le \lceil \frac{n}{2} \rceil \\ & x \in [0,\, 1]^n;\; y \in [0,\, 1]^n & \end{array} \tag{PP}$$

By Observation 3.1.1.6, all of the equations in (PP) are sum decomposable. Therefore, any computational distinction between using an edge-concave strategy or not is based entirely on augmenting the FBBT scheme with the redundant equations as discussed in Section 3.2.2. Additionally, all of the connected multivariable terms in (PP) are concave, so any partitioning scheme will be along the most negative eigenvector (in this case, $v_{i,j}^T = \left[1/\sqrt{2},\, -1/\sqrt{2}\right]$).

Note that, because our MIQCQP methods do not automatically infer problem symmetry, PP does not include the RLT-based cuts that Anstreicher [9] conjectures to produce an LP relaxation as tight as the SDP relaxation. For PP(n), there are $2 \cdot n$ continuous variables, $\frac{n \cdot (n+1)}{2}$ equations, $2 \cdot n$ concave quadratic terms, and $n \cdot (n-1)$ nonconvex bilinear terms. We tested 14 instances ($2 \le n \le 15$) and checked the correctness our computational results (shown in Tables 3 and 4) with the website `packomania.com/`. Figures 1 and 3 diagram the results in Tables 3 and 4.

Process Networks: The test suite of twenty pooling problems comprises the standard and generalized instances from our previous work [64]. We additionally consider three process networks examples from the GLOBALLib test library [37, 55]. By Property 3.1.3.1, the process network problems are sum decomposable and therefore using edge-concave strategies affects only the bounding scheme. Because the connected multivariable terms in process networks problems are neither convex nor concave, our algorithm selects variables to partition based on the co-occurrence graph discussed in Section 3.1.8. Figure 2 diagrams the results in Tables 5 and 6.

Other GLOBALLib Problems: We consider six additional examples (`camshape100`, `camshape200`, `camshape400`, `dispatch`, `st_iqpbk1`, and `st_iqpbk1`) from GLOBALLib [55] with quadratic equations that are *not* sum decomposable. These examples, which directly demonstrate the advantage of the edge-concave underestimators, are characterized in Table 2. Table 2 also demonstrates the duality gap closed at the root node by the edge-concave underestimators.

Each of the 37 process network and point packing test problems was run four times with the following algorithmic choices:

McC Only: Equations (1) – (2) were used to relax MIQCQP. No edge-concave aggregations or piecewise-linear relaxations were introduced (see Tables 3 and 5).

McC + EC: In addition to Equations (1) – (2), the dominant facets of the low-dimensional aggregations EC-AGG$^{\text{TRIP}}$ identified via Observations 3.1.1.4 – 3.1.1.9 augmented each relaxation of MIQCQP. As described in Section 3.2.2, nondominant facets of edge-concave aggregations were integrated into the FBBT scheme (see Tables 3 and 5).

McC + EC + PW Lin $N = 4$**:** edge-concave aggregations were still used to generate tighter relaxations and reduce variable bounds. We additionally employed piecewise relaxations using the linear underestimation scheme rather than Equations (1) – (2) (see Tables 4 and 6).

McC + EC + PW Log $N = 4$**:** These runs used the second logarithmic underestimation scheme to piecewise relax the variables not participating in EC-AGG$^{\text{TRIP}}$ (see Tables 4 and 6).

The piecewise strategies for the point packing problems were additionally considered for $N = 2, 8$. We graph the results of $N = 2, 8$ in Figures 1 and 3 to illustrate the performance differences but do not tabulate the results for $N = 2, 8$ because they are fairly similar to $N = 4$. The $N = 4$ parameter for process networks problems is based on our previous studies [64].

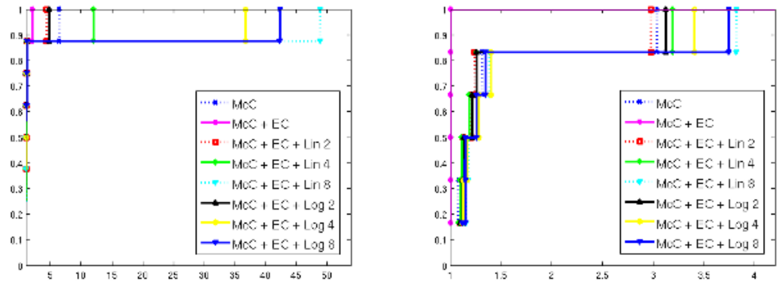
Table 2: GLOBALLib Test Instances [55]

Problem Name	# Cnt Vars	# Eqns	# Bln Terms	Root Node Rlxn		Glob Opt.	Gap Clsd
				McC Only	McC + EC		
camshape100	200	201	198	-4.8321	-4.6583	-4.2842	0.32
camshape200	400	401	398	-4.9213	-4.8475	-4.2785	0.11
camshape400	800	801	798	-5.0645	-5.0243	-4.2757	0.06
dispatch	5	3	6	3153.30	3155.29	3155.29	1.00
st_iqpbk1	9	8	36	-1298.96	-1204.63	-621.49	0.14
st_iqpbk1	9	8	36	-2601.98	-2413.95	-1195.23	0.13

4.3 Discussion

To discuss the trade-offs of the edge-concave and piecewise-linear strategies, we have integrated each of the strategies into a global optimization algorithm and compare the strategies with the context of the entire global optimization process. This holistic view makes the comparisons fairly subtle but also allows us to draw better-informed conclusions.

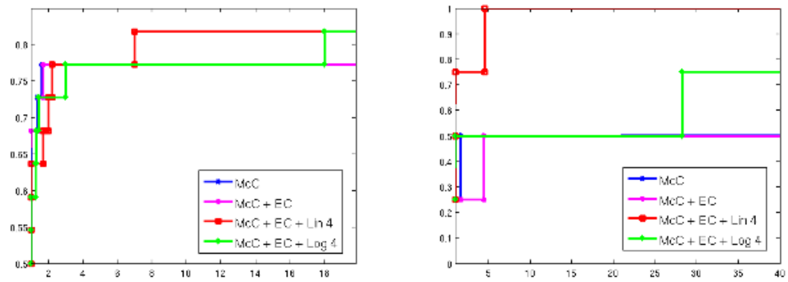
The six instances from GLOBALLib [55] with non-sum decomposable multivariable terms are the most straightforward of the test cases to analyze. Table 2 compares the gap closed at the root node relaxation of the **McC** and **McC + EC** strategies (observe that these root node relaxations do not correspond to naive application of the underestimators described in this paper because of our extensive bounds tightening strategies). Defining the gap closed as:



(a) Performance Profile: CPU s ($2 \leq n \leq 9$)

(b) Performance Profile: Optimality Gap at 7200 CPU s ($10 \leq n \leq 15$)

Fig. 1: Performance Profiles of Algorithmic Strategies for Point Packing [25].



(a) Performance Profile: CPU time (s) for the 23 process network problems

(b) Performance Profile: Optimality Gap at 7200 CPU s for `meyer10`, `meyer15`, `ex5_2_5`, `ex5_3_3`

Fig. 2: Performance Profiles of Algorithmic Strategies for Process Networks [25].

$$\text{Gap Closed} \equiv \frac{\text{Rlx}_{McC+EC} - \text{Rlx}_{McC}}{\text{Global Opt} - \text{Rlx}_{McC}},$$

notice in Table 2 that the edge-concave underestimators have added dominant cuts into the LP relaxation. Observe from the results in Tables 3 and 5 that in the nine test cases with at least a ten percent difference between **McC** and **McC + EC**, seven give the advantage to using both McCormick and edge-concave strategies. Further, note in Figure 1 that there is a significant performance distinction between **McC** and **McC + EC** for the point packing problems. Because the equations in (PP) are sum decomposable, this advantage must be exlusively based on the bounds tightening strategy that appends cuts redundant in the LP relaxation to the FBBT scheme. Generating the facets of the low-dimensional aggregations and integrating them into the FBBT scheme takes time (observe in Figure 3 that on average **McC + EC** explores

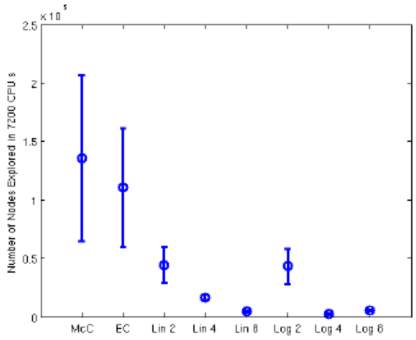


Fig. 3: Nodes Explored (in 7200 CPU s) for Point Packing Problems ($10 \leq n \leq 15$). The centered point for each choice diagrams the average number of nodes explored and the error bars above and below the point each represent a standard deviation.

20% fewer nodes than **McC**), but Figure 1 demonstrates that additional complexity at each node pays dividends. There is no particular advantage to the edge-concave aggregations for the process networks problems, but there is no particular disadvantage, either (see Table 5 and Figure 2). These results suggest that there is a consistent advantage to using low-dimensional edge-concave strategies.

The analysis of the piecewise-linear relaxation schemes is more complex. A piecewise-linear relaxation that partitions 30 variables into 4 segments is equivalent to exploring level 90 of a global optimization tree with a pre-determined branching scheme that has no associated bounding. The initial nodes of the branch-and-bound tree are therefore significantly tighter than in a polyhedral underestimation scheme, but there is a tradeoff between the solution time at each node and the number of nodes explored. This tradeoff is illustrated in Figure 3 where the average number of nodes explored in the 7200 CPU s time limit is graphed for each of the relaxation schemes for (PP). For the case of the point packing problems, taking this tradeoff is not advantageous (see Figure 2), but using the piecewise-linear relaxation schemes increases the probability that (1) the process networks problems will solve to global optimality and (2) for the process networks that do not solve to global optimality, that the gap remaining at the time limit will be significantly reduced (see Figure 2). When **McC** or **McC + EC** are sufficient to solve the process network problem, they tend to do so faster than the piecewise schemes, but the best strategies for closing the gap on the most difficult problems are the piecewise methods.

Observe in Table 6 that our pooling problem-specific solver APOGEE [64] regurally outperforms the more generic work presented here. APOGEE makes branching and bounding decisions based on topological reasoning. APOGEE reduces the big-M multipliers controlling the activation/deactivation of storage tanks and inter-node connections in later nodes of the branch-and-bound global optimization tree as variable bounds shrink. This generic MIQCQP solver does not have embedded knowledge as to the special structure of the pooling problem.

McC + EC + PW Lin $N = 4$ and **McC + EC + PW Log** $N = 4$ are more similar to one another than **McC** and **McC + EC**, but of the 10 test instances where there is at least a ten percent difference between the behavior of the two, eight give the advantage to the linear scheme. These results corroborate the computational studies of and Vielma et al. [85, 86] and our own previous work [64].

In summary, while we find strong evidence for the value of low-dimensional edge-concave aggregations in any MIQCQP, the piecewise-linear underestimation scheme is best applied to specific classes of MIQCQP (*e.g.*, process networks problems with many nonlinearly-participating variables).

5 Conclusion

We have presented an underestimation scheme for MIQCQP that is easily integrated into a branch-and-bound global optimization algorithm. The facets of low-dimensional $(n \leq 3)$ edge-concave aggregations dominating the termwise relaxation of MIQCQP are introduced at every node of a branch-and-bound tree. These edge-concave aggregations significantly improved the computational performance of our global optimization algorithm. The piecewise-linear relaxations, which were used to address both concave multivariable terms and the more sparsely distributed quadratic and bilinear terms that do not participate in edge-concave aggregations, solved the large-scale process networks problems more reliably than the traditional relaxation and closed a more significant fraction of the gap. Many of the MIQCQP instances addressed in this paper represent large-scale test cases.

Acknowledgements We are sincerely grateful to Alex Meeraus and Michael Bussieck of the GAMS Development Corporation who freely provided us a GAMS 23.6 license and the GAMS Modeling Object API. We also thank the COIN-OR GAMSLinks Developer Stefan Vigerske for suggesting routines in the GAMSLinks code base that helped us link our MIQCQP solver to GAMS.

References

1. T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Oper. Res. Lett.*, 33(1):42–54, 2005.
2. N. Adhya, M. Tawarmalani, and N. V. Sahinidis. A Lagrangian approach to the pooling problem. *Ind. Eng. Chem. Res.*, 38(5):1965 – 1972, 1999.
3. C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice differentiable NLPs-I. Theoretical advances. *Comput. Chem. Eng.*, 22:1137 – 1158, 1998a.
4. C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, αBB, for general twice differentiable NLPs-II. Implementation and computional results. *Comput. Chem. Eng.*, 22:1159 – 1179, 1998b.
5. A. Aggarwal and C. A. Floudas. Synthesis of general distillation sequences - nonsharp separations. *Comput. Chem. Eng.*, 14(6):631–653, 1990.
6. F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Math. Oper. Res.*, 8(2):273 – 286, 1983.

7. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, third edition, 1999.
8. I. P. Androulakis, C. D. Maranas, and C. A. Floudas. αBB: A global optimization method for general constrained nonconvex problems. *J. Global Optim.*, 7:337 – 363, 1995.
9. K. M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *J. Global Optim.*, 43(2-3):471 – 484, 2009.
10. K. M. Anstreicher and S. Burer. Computable representations for convex hulls of low-dimensional quadratic forms. *Math. Program.*, 124(1-2):33–43, 2010.
11. C. Audet, P. Hansen, B. Jaumard, and G. Savard. A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Math. Program.*, 87(1):131 – 152, 2000.
12. C. Audet, J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenovic. Pooling problem: Alternate formulations and solution methods. *Manage. Sci.*, 50(6):761 – 776, 2004.
13. M. Bagajewicz. A review of recent design procedures for water networks in refineries and process plants. *Comput. Chem. Eng.*, 24:2093 – 2113, 2000.
14. X. Bao, N. V. Sahinidis, and M. Tawarmalani. Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. *Math. Program.* 10.1007/s10107-011-0462-2.
15. X. Bao, N. V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs. *Optimization Methods and Software*, 24(4-5):485 – 504, 2009.
16. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.
17. A. Ben-Tal, G. Eiger, and V. Gershovitz. Global minimization by reducing the duality gap. *Math. Program.*, 63:193 – 212, 1994.
18. M. L. Bergamini, I. Grossmann, N. Scenna, and P. Aguirre. An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. *Comput. Chem. Eng.*, 32(3):477 – 493, 2008.
19. J. Brimberg, P. Hansen, and N. Mladenovic. A note on reduction of quadratic and bilinear programs with equality constraints. *J. Glob. Optim.*, 22(1-4):39–47, 2002.
20. A. Brooke, D. Kendrick, and A. Meeraus. General algebraic modeling language (GAMS). `http://www.gams.com/`, 2011. Version 23.6.
21. S. Burer and A. N. Letchford. On nonconvex quadratic programming with box constraints. *SIAM J. Optim.*, 20(2):1073–1089, 2009.
22. S. Burer and D. Vandenbussche. A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Math. Program.*, 113(2):259–282, 2008.

23. R. Cambini and C. Sodini. Decomposition methods for solving nonconvex quadratic programs via branch and bound. *J. Glob. Optim.*, 33:313–336, 2005.
24. A. R. Ciric and C. A. Floudas. A retrofit approach for heat exchanger networks. *Comput. Chem. Eng.*, 13(6):703 – 715, 1989.
25. E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91:201–213, 2002.
26. D. C. Faria and M. J. Bagajewicz. On the appropriate modeling of process plant water systems. *AIChE J.*, 56(3):668 – 689, 2010.
27. C. A. Floudas. *Deterministic Global Optimization : Theory, Methods and Applications*. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Dordrecht, Netherlands, 2000.
28. C. A. Floudas and A. Aggarwal. A decomposition strategy for global optimum search in the pooling problem. *ORSA J. Comput.*, 2:225 – 235, 1990.
29. C. A. Floudas and S. H. Anastasiadis. Synthesis of distillation sequences with several multicomponent feed and product streams. *Chem. Eng. Sci.*, 43(9):2407–2419, 1988.
30. C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *J. Global Optim.*, 45(1):3 – 38, 2009.
31. C. A. Floudas and I. E. Grossmann. Synthesis of flexible heat-exchanger networks with uncertain flowrates and temperatures. *Comput. Chem. Eng.*, 11(4): 319–336, 1987.
32. C. A. Floudas and P. M. Pardalos. State-of-the-art in global optimization - computational methods and applications - preface. *J. Glob. Optim.*, 7(2):113, 1995.
33. C. A. Floudas and G. E. Paules. A mixed-integer nonlinear programming formulation for the synthesis of heat-integrated distillation sequences. *Comput. Chem. Eng.*, 12(6):531 – 546, 1988.
34. C. A. Floudas and V. Visweswaran. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. *Comput. Chem. Eng.*, 14(12):1397 – 1417, 1990.
35. C. A. Floudas and V. Visweswaran. Primal-relaxed dual global optimization approach. *J. Optim. Theory Appl.*, 78(2):187 – 225, 1993.
36. C. A. Floudas, A. Aggarwal, and A. R. Ciric. Global optimum search for nonconvex NLP and MINLP problems. *Comput. Chem. Eng.*, 13(10):1117 – 1132, 1989.
37. C. A. Floudas, P. M. Pardalos, C. S. Adjiman, W. R. Esposito, Z. H. Gms, S. T. Harding, J. L. Klepeis, C. A. Meyer, and C. A. Schweiger. *Handbook of Test Problems in Local and Global Optimization*. Kluwer Academic Publishers, 1999.
38. C. A. Floudas, I. G. Akrotirianakis, S. Caratzoulas, C. A. Meyer, and J. Kallrath. Global optimization in the 21st century: Advances and challenges. *Comput. Chem. Eng.*, 29:1185 – 1202, 2005.
39. P. E. Gill, W. Murray, and M. A. Saunders. SNOPT. `http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm`, 1999. Version 5.3.
40. C. E. Gounaris, R. Misener, and C. A. Floudas. Computational comparison of piecewise-linear relaxations for pooling problems. *Ind. Eng. Chem. Res.*, 48(12):

5742 – 5766, 2009.
41. P. Hansen and B. Jaumard. Reduction of indefinite quadratic programs to bilinear programs. *J. Glob. Optim.*, 2:41–60, 1992.
42. M. M. F. Hasan and I. A. Karimi. Piecewise linear relaxation of bilinear programs using bivariate partitioning. *AIChE J.*, 56(7):1880 – 1893, 2010.
43. ILOG. CPLEX. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, 2009. Version 11.1.
44. J. Jeżowski. Review of water network design methods with literature annotations. *Ind. Eng. Chem. Res.*, 49(10):4475 – 4516, 2010.
45. R. Karuppiah and I. E. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Comput. Chem. Eng.*, 30:650 – 673, 2006.
46. A. B. Keha, I. R. de Farias, and G. L. Nemhauser. Models for representing piecewise linear cost functions. *Oper. Res. Lett.*, 32(1):44 – 48, 2004.
47. A. C. Kokossis and C. A. Floudas. Synthesis of isothermal reactor–separator–recycle systems. *Chem. Eng. Sci.*, 46(5 - 6):1361 – 1383, 1991.
48. A. C. Kokossis and C. A. Floudas. Optimization of complex reactor networks–II. nonisothermal operation. *Chem. Eng. Sci.*, 49(7):1037 – 1051, 1994.
49. L. Liberti and C. C. Pantelides. An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *J. Glob. Optim.*, 36(2):161–189, 2006.
50. X. Lin and C. A. Floudas. Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. *Comput. Chem. Eng.*, 25(4 - 6):665 – 674, 2001.
51. J. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Math. Program.*, 103(2):251 – 282, 2005.
52. R. Lougee-Heimer. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47(1):57 –66, 2003.
53. C. D. Maranas and C. A. Floudas. Finding all solutions of nonlinearly constrained systems of equations. *J. Glob. Optim.*, 7(2):143–182, 1995.
54. G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part 1-convex underestimating problems. *Math. Program.*, 10(1):147 – 175, 1976.
55. A. Meeraus. Globallib. `http://www.gamsworld.org/global/globallib.htm`.
56. C. A. Meyer and C. A. Floudas. Trilinear monomials with positive or negative domains: Facets of the convex and concave envelopes. In C. A. Floudas and P. M. Pardalos, editors, *Frontiers in Global Optimization*, pages 327–352. Kluwer Academic Publishers, 2003.
57. C. A. Meyer and C. A. Floudas. Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes. *J. Glob. Optim.*, 29(2):125–155, 2004.
58. C. A. Meyer and C. A. Floudas. Convex envelopes for edge-concave functions. *Math. Program.*, 103(2):207–224, 2005.

59. C. A. Meyer and C. A. Floudas. Global optimization of a combinatorially complex generalized pooling problem. *AIChE J.*, 52(3):1027 – 1037, 2006.
60. R. Misener and C. A. Floudas. Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, 8(1):3 – 22, 2009.
61. R. Misener and C. A. Floudas. Global optimization of large-scale pooling problems: Quadratically constrained MINLP models. *Ind. Eng. Chem. Res.*, 49(11): 5424 – 5438, 2010.
62. R. Misener, C. E. Gounaris, and C. A. Floudas. Global optimization of gas lifting operations: A comparative study of piecewise linear formulations. *Ind. Eng. Chem. Res.*, 48(13):6098 – 6104, 2009.
63. R. Misener, C. E. Gounaris, and C. A. Floudas. Mathematical modeling and global optimization of large-scale extended pooling problems with the (EPA) complex emissions constraints. *Comput. Chem. Eng.*, 34(9):1432 – 1456, 2010.
64. R. Misener, J. P. Thompson, and C. A. Floudas. APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Comput. Chem. Eng.*, 35(5):876–892, 2011.
65. P. M. Pardalos. Global optimization algorithms for linearly constrained indefinite quadratic problems. *Comput. Math. Appl.*, 21(6-7):87 – 97, 1991.
66. V. Pham, C. Laird, and M. El-Halwagi. Convex hull discretization approach to the global optimization of pooling problems. *Ind. Eng. Chem. Res.*, 48:1973 – 1979, 2009.
67. I. Quesada and I. E. Grossmann. Global optimization of bilinear process networks with multicomponent flows. *Comput. Chem. Eng.*, 19:1219 – 1242, 1995.
68. A. D. Rikun. A convex envelope formula for multilinear functions. *J. Glob. Optim.*, 10:425 – 437, 1997.
69. J. B. Rosen and P. M. Pardalos. Global minimization of large-scale constrained concave quadratic problems by separable programming. *Math. Program.*, 34(2): 163–174, 1986.
70. J. P. Ruiz and I. E. Grossmann. Exploiting vector space properties to strengthen the relaxation of bilinear programs arising in the global optimization of process networks. *Optimization Letters*, 5:1–11, 2011.
71. Y. Saif, A. Elkamel, and M. Pritzker. Global optimization of reverse osmosis network for wastewater treatment and minimization. *Ind. Eng. Chem. Res.*, 47 (9):3060 – 3070, 2008.
72. A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations. *Math. Program.* 10.1007/s10107-010-0340-3.
73. A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: Extended formulations. *Math. Program.*, 124(1-2):383–411, 2010.
74. H. D. Sherali. On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. *Oper. Res. Letters*, 28(4):155 – 160, 2001.
75. H. D. Sherali and W. P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Nonconvex Optimiza-

tion and Its Applications. Kluwer Academic Publishers, Dordrecht, Netherlands, 1999.

76. H. D. Sherali and A. Alameddine. A new Reformulation-Linearization Technique for bilinear programming problems. *J. Global Optim.*, 2:379 – 410, 1992.
77. H. D. Sherali and C. H. Tuncbilek. A reformulation-convexification approach for solving nonconvex quadratic-programming problems. *J. Glob. Optim.*, 1995 (7):1, 1-31.
78. H. D. Sherali and C. H. Tuncbilek. New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Oper. Res. Letters*, 21(1):1 – 9, 1997.
79. F. Tardella. On a class of functions attaining their maximum at the vertices of a polyhedron. *Discret. Appl. Math.*, 22:191–195, 1988/89.
80. F. Tardella. On the existence of polyhedral convex envelopes. In C. A. Floudas and P. M. Pardalos, editors, *Frontiers in Global Optimization*, pages 563–573. Kluwer Academic Publishers, 2003.
81. F. Tardella. Existence and sum decomposition of vertex polyhedral convex envelopes. *Optim. Lett.*, 2:363–375, 2008.
82. M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Applications, Software, and Applications*. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Norwell, MA, USA, 2002.
83. D. Vandenbussche and G. L. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Math. Program.*, 102(3):559–575, 2005.
84. D. Vandenbussche and G. L. Nemhauser. A polyhedral study of nonconvex quadratic programs with box constraints. *Math. Program.*, 102(3):531–557, 2005.
85. J. P. Vielma and G. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Math. Program.*, 2010. In Press (DOI: 10.1007/s10107-009-0295-4).
86. J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Oper. Res.*, 58(2):303 – 315, 2010.
87. S. Vigerske. COIN-OR/GAMSLinks. `https://projects.coin-or.org/GAMSlinks/`, 2011. Trunk Revision 1026.
88. V. Visweswaran. MINLP: Applications in blending and pooling. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2114 – 2121. Springer Science, 2 edition, 2009.
89. V. Visweswaran and C. A. Floudas. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: II. application of theory and test problems. *Comput. Chem. Eng.*, 14(12):1419 – 1434, 1990.
90. V. Visweswaran and C. A. Floudas. New properties and computational improvement of the GOP algorithm for problems with quadratic objective functions and constraints. *J. Global Optim.*, 3:439 – 462, 1993.
91. D. S. Wicaksono and I. A. Karimi. Piecewise MILP under-and overestimators for global optimization of bilinear programs. *AIChE J.*, 54(4):991 – 1008, 2008.

Table 3: Computational Results on the Point Packing Test Instances: McCormick vs. Edge-Concave (No Partitioning)

Problem Name	McC Only					McC + EC				
	Time	Gap $\left(\frac{UB-LB}{\lvert LB \rvert}\right)$	LB	UB	# Nodes	Time	Gap $\left(\frac{UB-LB}{\lvert LB \rvert}\right)$	LB	UB	# Nodes
PP 2	0	1e-06	-2.000e+00	-2.000e+00	1	0	1e-06	-2.000e+00	-2.000e+00	1
PP 3	0	1e-06	-1.072e+00	-1.072e+00	5	0	1e-06	-1.072e+00	-1.072e+00	5
PP 4	0	1e-06	-1.000e+00	-1.000e+00	7	0	1e-06	-1.000e+00	-1.000e+00	7
PP 5	1	1e-06	-5.000e-01	-5.000e-01	93	2	1e-06	-5.000e-01	-5.000e-01	37
PP 6	6	1e-06	-3.611e-01	-3.611e-01	382	6	1e-06	-3.611e-01	-3.611e-01	382
PP 7	210	1e-06	-2.872e-01	-2.872e-01	21181	**33**	**1e-06**	**-2.87187e-01**	**-2.872e-01**	2474
PP 8	149	1e-06	-2.680e-01	-2.679e-01	8594	151	1e-06	-2.680e-01	-2.679e-01	8594
PP 9	256	1e-06	-2.500e-01	-2.500e-01	11381	256	1e-06	-2.500e-01	-2.500e-01	11381
PP 10	–	3.49e-01	-2.725e-01	-1.775e-01	266266	**–**	**1.12e-01**	**-2.005e-01**	**-1.775e-01**	205671
PP 11	–	4.92e-01	-3.125e-01	-1.586e-01	155422	**–**	**3.76e-01**	**-2.540e-01**	**-1.586e-01**	122809
PP 12	–	5.49e-01	-3.352e-01	-1.511e-01	129465	**–**	**4.50e-01**	**-2.746e-01**	**-1.511e-01**	101429
PP 13	–	6.29e-01	-3.611e-01	-1.340e-01	107738	**–**	**5.43e-01**	**-2.932e-01**	**-1.340e-01**	92503
PP 14	–	6.60e-01	-3.584e-01	-1.217e-01	86141	**–**	**6.09e-01**	**-3.117e-01**	**-1.217e-01**	78137
PP 15	–	6.84e-01	-3.632e-01	-1.146e-01	69315	**–**	**6.31e-01**	**-3.106e-01**	**-1.146e-01**	62716

Table 4: Computational Results on the Point Packing Test Instances: Edge-Concave Aggregations; Linear vs. Logarithmic Partitioning $N = 4$

Problem Name	McC + EC + PW Lin $N = 4$					McC + EC + PW Log $N = 4$				
	Time	Gap $\left(\frac{UB-LB}{\lvert LB\rvert}\right)$	LB	UB	# Nodes	Time	Gap $\left(\frac{UB-LB}{\lvert LB\rvert}\right)$	LB	UB	# Nodes
PP 2	0	1e-06	-2.000e+00	-2.000e+00	1	0	1e-06	-2.000e+00	-2.000e+00	1
PP 3	0	1e-06	-1.072e+00	-1.072e+00	5	0	1e-06	-1.072e+00	-1.072e+00	5
PP 4	1	1e-06	-1.000e+00	-1.000e+00	7	0	1e-06	-1.000e+00	-1.000e+00	7
PP 5	1	1e-06	-5.000e-01	-5.000e-01	4	1	1e-06	-5.000e-01	-5.000e-01	1
PP 6	7	1e-06	-3.611e-01	-3.611e-01	382	7	1e-06	-3.611e-01	-3.611e-01	382
PP 7	**396**	**1e-06**	**-2.872e-01**	**-2.872e-01**	2556	1212	1e-06	-2.872e-01	-2.872e-01	1885
PP 8	149	1e-06	-2.680e-01	-2.679e-01	8594	150	1e-06	-2.680e-01	-2.679e-01	8594
PP 9	255	1e-06	-2.500e-01	-2.500e-01	11381	260	1e-06	-2.500e-01	-2.500e-01	11381
PP 10	**–**	**3.67e-01**	**-2.803e-01**	**-1.775e-01**	20989	–	3.92e-01	-2.917e-01	-1.775e-01	3429
PP 11	**–**	**4.73e-01**	**-3.012e-01**	**-1.586e-01**	17148	–	5.27e-01	-3.355e-01	-1.586e-01	3060
PP 12	**–**	**5.33e-01**	**-3.235e-01**	**-1.511e-01**	16873	–	5.76e-01	-3.563e-01	-1.511e-01	2604
PP 13	–	6.04e-01	-3.380e-01	-1.340e-01	15760	–	6.33e-01	-3.648e-01	-1.340e-01	2219
PP 14	–	6.62e-01	-3.607e-01	-1.217e-01	15318	–	6.88e-01	-3.899e-01	-1.217e-01	2156
PP 15	–	6.94e-01	-3.666e-01	-1.121e-01	14280	–	7.10e-01	-3.915e-01	-1.136e-01	2068

Table 5: Computational Results on the Pooling Test Instances: McCormick vs. Edge-Concave (No Partitioning)

Problem Name	McC Only					McC + EC				
	Time	Gap $\left(\frac{UB-LB}{\lvert LB\rvert}\right)$	LB	UB	# Nodes	Time	Gap $\left(\frac{UB-LB}{\lvert LB\rvert}\right)$	LB	UB	# Nodes
Adhya 1	0	1e-06	-5.498e+02	-5.498e+02	29	0	1e-06	-5.498e+02	-5.498e+02	29
Adhya 2	0	1e-06	-5.498e+02	-5.498e+02	21	0	1e-06	-5.498e+02	-5.498e+02	21
Adhya 3	1	1e-06	-5.610e+02	-5.610e+02	15	0	1e-06	-5.610e+02	-5.610e+02	15
Adhya 4	0	1e-06	-8.776e+02	-8.776e+02	5	0	1e-06	-8.776e+02	-8.776e+02	5
BenTal 4	0	1e-06	-4.500e+02	-4.500e+02	1	0	1e-06	-4.500e+02	-4.500e+02	1
BenTal 5	0	1e-06	-3.500e+03	-3.500e+03	1	0	1e-06	-3.500e+03	-3.500e+03	1
Foulds 2	0	1e-06	-1.100e+03	-1.100e+03	1	0	1e-06	-1.100e+03	-1.100e+03	1
Foulds 3	1	1e-06	-8.000e+00	-8.000e+00	1	1	1e-06	-8.000e+00	-8.000e+00	1
Foulds 4	2	1e-06	-8.000e+00	-8.000e+00	1	2	1e-06	-8.000e+00	-8.000e+00	1
Foulds 5	1	1e-06	-8.000e+00	-8.000e+00	1	1	1e-06	-8.000e+00	-8.000e+00	1
Haverly 1	0	1e-06	-4.000e+02	-4.000e+02	1	0	1e-06	-4.000e+02	-4.000e+02	1
Haverly 2	0	1e-06	-6.000e+02	-6.000e+02	3	0	1e-06	-6.000e+02	-6.000e+02	3
Haverly 3	0	1e-06	-7.500e+02	-7.500e+02	3	0	1e-06	-7.500e+02	-7.500e+02	3
RT 2	0	1e-06	-4.392e+03	-4.392e+03	13	0	1e-06	-4.392e+03	-4.392e+03	13
Lee 1	45	1e-06	-4.640e+03	-4.640e+03	733	48	1e-06	-4.640e+03	-4.640e+03	819
Lee 2	89	1e-06	-3.849e+03	-3.849e+03	1319	90	1e-06	-3.849e+03	-3.849e+03	1305
Meyer 4	**–**	**1.62e-02**	**1.082e+06**	**1.099e+06**	38654	–	8.21e-01	1.081e+06	1.090e+06	39640
Meyer 10	**–**	**4.58e-03**	**1.086e+06**	**1.091e+06**	587	–	2.02e-02	1.086e+06	1.108e+06	574
Meyer 15	–	6.39e-02	9.054e+05	9.633e+05	74	–	6.82e-02	9.018e+05	9.633e+05	68
Ex 5.2.5	–	5.93e-01	-8.595e+03	-3.500e+03	422013	–	5.94e-01	-8.623e+03	-3.500e+03	369851
Ex 5.3.3	–	–	1.669e+00	–	2040	–	–	1.669e+00	–	104724
Ex 8.4.1	105	1e-06	6.186e-01	6.186e-01	2241	113	1e-06	6.186e-01	6.186e-01	2241

Table 6: Computational Results on the Pooling Test Instances: Edge-Concave Aggregations; Linear vs. Logarithmic Partitioning $N = 4$

Problem Name	McC + EC + PW Lin $N=4$					McC + EC + PW Log $N=4$				
	Time	Gap $\left(\frac{UB-LB}{\|LB\|}\right)$	LB	UB	# Nodes	Time	Gap $\left(\frac{UB-LB}{\|LB\|}\right)$	LB	UB	# Nodes
Adhya 1	1	1e-06	-5.498e+02	-5.498e+02	19	0	1e-06	-5.498e+02	-5.498e+02	15
Adhya 2	0	1e-06	-5.498e+02	-5.498e+02	13	0	1e-06	-5.498e+02	-5.498e+02	13
Adhya 3	0	1e-06	-5.610e+02	-5.610e+02	11	0	1e-06	-5.610e+02	-5.610e+02	11
Adhya 4	1	1e-06	-8.776e+02	-8.776e+02	5	0	1e-06	-8.776e+02	-8.776e+02	5
BenTal 4	0	1e-06	-4.500e+02	-4.500e+02	1	0	1e-06	-4.500e+02	-4.500e+02	1
BenTal 5	0	1e-06	-3.500e+03	-3.500e+03	1	0	1e-06	-3.500e+03	-3.500e+03	1
Foulds 2	0	1e-06	-1.100e+03	-1.100e+03	1	0	1e-06	-1.100e+03	-1.100e+03	1
Foulds 3	2	1e-06	-8.000e+00	-8.000e+00	1	3	1e-06	-8.000e+00	-8.000e+00	1
Foulds 4	2	1e-06	-8.000e+00	-8.000e+00	1	2	1e-06	-8.000e+00	-8.000e+00	1
Foulds 5	0	1e-06	-8.000e+00	-8.000e+00	1	1	1e-06	-8.000e+00	-8.000e+00	1
Haverly 1	0	1e-06	-4.000e+02	-4.000e+02	1	0	1e-06	-4.000e+02	-4.000e+02	1
Haverly 2	0	1e-06	-6.000e+02	-6.000e+02	1	0	1e-06	-6.000e+02	-6.000e+02	1
Haverly 3	0	1e-06	-7.500e+02	-7.500e+02	1	0	1e-06	-7.500e+02	-7.500e+02	1
RT 2	**7**	**1e-06**	**-4.392e+03**	**-4.392e+03**	1131	18	1e-06	-4.392e+03	-4.392e+03	11
Lee 1	63	1e-06	-4.640e+03	-4.640e+03	235	**28**	**1e-06**	**-4.640e+03**	**-4.640e+03**	443
Lee 2	151	1e-06	-3.849e+03	-3.849e+03	127	115	1e-06	-3.849e+03	-3.849e+03	333
Meyer 4	257	1e-06	1.086e+06	1.086e+06	118	372	1e-06	1.086e+06	1.086e+06	612
Meyer 10	**–**	**2.09e-02**	**1.064e+06**	**1.086e+06**	31	–	4.20e-02	1.042e+06	1.086e+06	297
Meyer 15	**–**	**1.08e-04**	**9.436e+05**	**9.437e+05**	11	–	8.98e-02	9.350e+05	1.019e+06	5
Ex 5.2.5	–	3.59e-01	-5.464e+03	-3.500e+03	915	–	3.47e-01	-5.359e+03	-3.500e+03	1195
Ex 5.3.3	–	3.45e-01	2.404e+00	3.234e+00	864	**–**	**2.74e-01**	**2.539e+00**	**3.234e+00**	1605
Ex 8.4.1	**77**	**1e-06**	**6.186e-01**	**6.186e-01**	741	103	1e-06	6.186e-01	6.186e-01	745

A Sharpness Proofs for Piecewise-Linear Underestimators with a Logarithmic Number of Binary Variables

Property 3.1.6.2: The linear programming relaxation of Logarithmic Partitioning Scheme in 1 Equations (5g) - (5j) is nondominated by the convex hull in Equations (1) – (2).
Proof: To prove that Equations (5g) - (5j) are nondominated by Equations (1) – (2), we relax $\lambda \in \{0,1\}^{N_L}$ to $\lambda \in [0,1]^{N_L}$ and observe:

$$\begin{aligned} z &\geq x \cdot y^L + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot (n_P - 1)\right] \cdot \Delta y(n_P) \\ &\overset{(1)}{=} x \cdot y^L + x^L \cdot y - x^L \cdot y^L + \sum_{n_P=1}^{N_P} a \cdot (n_P - 1) \cdot \Delta y(n_P) \overset{(2)}{\geq} x \cdot y^L + x^L \cdot y - x^L \cdot y^L \end{aligned} \tag{6a}$$

Equality (6a.1) holds because Equation (5f) implies that $\sum_{n_P=1}^{N_P} x^L \cdot \Delta y(n_P) = x^L \cdot y - x^L \cdot y^L$. Inequality (6a.2) follows from $a \cdot (n_P - 1) \cdot \Delta y(n_P) \geq 0 \; \forall \, n_P \in \{1, \ldots, N_P\}$.

$$\begin{aligned} z &\geq x \cdot y^U + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot \; n_P \qquad\right] \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \hat{\lambda}(n_P)\right] \\ &\overset{(1)}{=} x \cdot y^U + x^U \cdot y - x^U \cdot y^U + \sum_{n_P=1}^{N_P} a \cdot (n_P - N_P) \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \hat{\lambda}(n_P)\right] \\ &\overset{(2)}{\geq} x \cdot y^U + x^U \cdot y - x^U \cdot y^U \end{aligned} \tag{6b}$$

Equality (6b.1) follows from the definition of a and Equations (5b) and (5f) because $x^L + a \cdot n_P = x^U + a \cdot (n_P - N_P)$ and $\sum_{n_P=1}^{N_P} x^U \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \hat{\lambda}(n_P)\right] = x^U \cdot y - x^U \cdot y^U$. Inequality (6b.2) follows from inequality (5e) because $a \cdot (n_P - N_P) \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \hat{\lambda}(n_P)\right] \geq 0 \; \forall \, n_P \in \{1, \ldots, N_P\}$.

$$\begin{aligned} z &\leq x \cdot y^L + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot \; n_P \qquad\right] \cdot \Delta y(n_P) \\ &\overset{(1)}{=} x \cdot y^L + x^U \cdot y - x^U \cdot y^L + \sum_{n_P=1}^{N_P} a \cdot (n_P - N_P) \cdot \Delta y(n_P) \overset{(2)}{\leq} x \cdot y^L + x^U \cdot y - x^U \cdot y^L \end{aligned} \tag{6c}$$

Equality (6c.1) holds by the definition of a and Equation (5f). Inequality (6c.2) follows from $a \cdot (n_P - N_P) \cdot \Delta y(n_P) \leq 0 \; \forall \, n_P \in \{1, \ldots, N_P\}$.

$$\begin{aligned} z &\leq x \cdot y^U + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot (n_P - 1)\right] \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \hat{\lambda}(n_P)\right] \\ &\overset{(1)}{=} x \cdot y^U + x^L \cdot y - x^L \cdot y^U + \sum_{n_P=1}^{N_P} a \cdot (n_P - 1) \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \hat{\lambda}(n_P)\right] \\ &\overset{(2)}{\leq} x \cdot y^U + x^L \cdot y - x^L \cdot y^U \end{aligned} \tag{6d}$$

Equality (6d.1) is a result of Equations (5b) and (5f). Inequality (6d.2) holds by Equation (5e). □
Property 3.1.6.3: The linear programming relaxation of Logarithmic Partitioning Scheme 2 in Equations (7e) - (7h) is nondominated by the convex hull in Equations (1) – (2) when the number of partitions is a power of two (*i.e.*, $N_L = \log_2 N_P = \lceil \log_2 N_P \rceil$).
Logarithmic Partitioning Scheme 2 [64]:

$$x^L + \sum_{n_L=1}^{N_L} 2^{N_L - n_L} \cdot a \cdot \lambda(n_L) \leq x \leq x^L + a + \sum_{n_L=1}^{N_L} 2^{N_L - n_L} \cdot a \cdot \lambda(n_L) \tag{7a}$$

$$\Delta y(n_L) \leq (y^U - y^L) \cdot \lambda(n_L) \quad \forall \, n_L \in \{1, \ldots, N_L\} \tag{7b}$$

$$\Delta y(n_L) = (y - y^L) - s(n_L) \quad \forall \, n_L \in \{1, \ldots, N_L\} \tag{7c}$$

$$s(n_L) \le (y^U - y^L) \cdot (1 - \lambda(n_L)) \quad \forall\, n_L \in \{1, \ldots, N_L\} \tag{7d}$$

$$z \ge x \cdot y^L + \; x^L \qquad \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \Delta y(n_L) \right] \tag{7e}$$

$$z \ge x \cdot y^U + (x^L + a) \cdot (y - y^U) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \left(\Delta y(n_L) - \lambda(n_L) \cdot (y^U - y^L) \right) \right] \tag{7f}$$

$$z \le x \cdot y^L + (x^L + a) \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \Delta y(n_L) \right] \tag{7g}$$

$$z \le x \cdot y^U + \; x^L \qquad \cdot (y - y^U) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \left(\Delta y(n_L) - \lambda(n_L) \cdot (y^U - y^L) \right) \right] \tag{7h}$$

$$x^L \le x \le x^U; \quad y^L \le y \le y^U \tag{7i}$$

Proof: To prove that Equations (7e) - (7h) are nondominated by Equations (1) – (2) for powers of two, we relax $\lambda \in \{0, 1\}^{N_L}$ to $\lambda \in [0, 1]^{N_L}$ and observe:

$$z \ge x \cdot y^L + x^L \cdot y - x^L \cdot y^L + \left[a \cdot 2^{N_L - n_L} \cdot \sum_{n_L=1}^{N_L} \Delta y(n_L) \right] \overset{(1)}{\ge} x \cdot y^L + x^L \cdot y - x^L \cdot y^L \tag{8a}$$

Inequality (8a) results from the bounds of Δy.

$$\begin{aligned}
z &\ge x \cdot y^U + (x^L + a) \cdot (y - y^U) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \left(\Delta y(n_L) - \lambda(n_L) \cdot (y^U - y^L) \right) \right] \\
&\overset{(1)}{=} x \cdot y^U + x^U \cdot y - x^U \cdot y^U + (x^U - x^L - a) \cdot (y^U - y) + \\
&\qquad \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \left(y - y^L - s(n_L) - \lambda(n_L) \cdot (y^U - y^L) \right) \right] \\
&\overset{(2)}{\ge} x \cdot y^U + x^U \cdot y - x^U \cdot y^U + (x^U - x^L - a) \cdot (y^U - y) + \\
&\qquad \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \left(y - y^L - (1 - \lambda(n_L)) \cdot (y^U - y^L) - \lambda(n_L) \cdot (y^U - y^L) \right) \right] \\
&\overset{(3)}{=} x \cdot y^U + x^U \cdot y - x^U \cdot y^U + (x^U - x^L - a) \cdot (y^U - y) + \\
&\qquad \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \left(y - y^U \right) \right] \\
&\overset{(4)}{=} x \cdot y^U + x^U \cdot y - x^U \cdot y^U
\end{aligned} \tag{8b}$$

Equality (8b.1) results from addition and subtraction of $x^U \cdot (y - y^L)$ and the definition of $s(n_L)$. Inequality (8b.2) follows from Equation (7d) and Equality (8b.3) simplifies the expression. Assuming N_P is a power of two, Equality (8b.4) holds because $x^U - x^L - a = \sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L}$.

$$\begin{aligned}
z &\le x \cdot y^L + (x^L + a) \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \Delta y(n_L) \right] \\
&\overset{(1)}{=} x \cdot y^L + x^U \cdot y - x^U \cdot y^L - (x^U - x^L - a) \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \Delta y(n_L) \right] \\
&\overset{(2)}{\le} x \cdot y^L + x^U \cdot y - x^U \cdot y^L - (x^U - x^L - a) \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot (y - y^L) \right] \\
&\overset{(3)}{=} x \cdot y^L + x^U \cdot y - x^U \cdot y^L
\end{aligned} \tag{8c}$$

Equality (8c.1) results from addition and subtraction of $x^U \cdot (y - y^L)$. Inequality (8c.2) follows from Equation (7c). Equality (8c.3) holds because N_P is a power of two.

$$\begin{aligned} z &\leq x \cdot y^U + x^L \cdot y - x^L \cdot y^U + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{N_L - n_L} \cdot \left(\Delta y(n_L) - \lambda(n_L) \cdot (y^U - y^L) \right) \right] \\ &\overset{(1)}{\leq} x \cdot y^U + x^L \cdot y - x^L \cdot y^U \end{aligned} \tag{8d}$$

Equality (8d.1) follows from Equation (7b). □

Although we have proven sharpness of Logarithmic Scheme 2 for powers of two, observe that Equalities (8b.4) and (8c.3) are violated when $\log_2 N_P \neq \lceil \log_2 N_P \rceil$. Therefore, we select Logarithmic Relaxation Scheme 1 when the number of partitions is not a power of two. However, observe in Table 1 that Logarithmic Partitioning Scheme 2 is smaller than Logarithmic Partitioning Scheme 1, so we choose to use the second scheme exclusively when the number of partitions is a power of two. Based on our previous computational study on a test suite of pooling problems [64], the advantage of piecewise-linear relaxations is fairly robust in the range $N_P = 3, 4, 5$ and we therefore choose $N_P = 4$ for the computations in this work because of the complexity advantage in using the smaller Logarithmic Partitioning Scheme 2.

B Complexity of the Piecewise-Linear Underestimators

Observe in Table 7 that the additional complexity introduced by the piecewise-linear relaxations scales with both the number of partitioned variables and the number of piecewise-relaxed bilinear terms. The additional variables and constraints scaling with the number of piecewise-relaxed bilinear terms are unavoidable, but we reduce the complexity of the piecewise-linear underestimators by minimizing the number of partitioned variables. Because the piecewise-linear relaxation formulations introduce binary variables and linear constraints for each partitioned variable in MIQCQP, our goal of partitioning as few variables as possible is similar to the objective of minimizing the number of complicating variables for a primal-dual context [5, 19, 28, 34, 35, 36, 41, 89, 90].

Table 7: Additional variables and constraints for the relaxation of MIQCQP with N_{VAR} partitioned variables participating in N_{BIL} bilinear terms

	Contin Vars	Bnry Vars	Constraints
McC Hull	N_{BIL}	–	$4 \cdot N_{\mathrm{BIL}}$
Lin Rlxn	$N_{\mathrm{BIL}}(N_P + 1)$	$N_{\mathrm{VAR}} \cdot N_P$	$N_{\mathrm{BIL}}(N_P + 5) + N_{\mathrm{VAR}} \cdot 3$
Log Rlxn 1	$N_{\mathrm{BIL}}(2 \cdot N_P + 1)$	$N_{\mathrm{VAR}} \cdot N_L$	$N_{\mathrm{BIL}}(N_P + 5) + N_{\mathrm{VAR}} \cdot (2 \cdot N_L + 3)$
Log Rlxn 2†	$N_{\mathrm{BIL}}(2 \cdot N_L + 1)$	$N_{\mathrm{VAR}} \cdot N_L$	$N_{\mathrm{BIL}}(3 \cdot N_L + 4) + N_{\mathrm{VAR}} \cdot 2$

† Applicable to powers of two (*i.e.*, $\log_2 N_P = \lceil \log_2 N_P \rceil$)

$N_L = \lceil \log_2 N_P \rceil$; $N_{\mathrm{BIL}} \equiv$ number of bilinear terms; $N_{\mathrm{VAR}} \equiv$ number of partitioned variables