J Glob Optim (2013) 57:1039–1063
DOI 10.1007/s10898-012-0022-1

Global optimization of bilinear programs with a multiparametric disaggregation technique

Scott Kolodziej · Pedro M. Castro · Ignacio E. Grossmann

Received: 18 May 2012 / Accepted: 4 December 2012 / Published online: 3 January 2013

Abstract In this paper, we present the derivation of the multiparametric disaggregation technique (MDT) by Teles et al. (J. Glob. Optim., 2011) for solving nonconvex bilinear programs. Both upper and lower bounding formulations corresponding to mixed-integer linear programs are derived using disjunctive programming and exact linearizations, and incorporated into two global optimization algorithms that are used to solve bilinear programming problems. The relaxation derived using the MDT is shown to scale much more favorably than the relaxation that relies on piecewise McCormick envelopes, yielding smaller mixed-integer problems and faster solution times for similar optimality gaps. The proposed relaxation also compares well with general global optimization solvers on large problems.

Keywords Global optimization · Mixed integer linear programming · Mixed integer nonlinear programming · Quadratic optimization · Disjunctive programming

1 Introduction

Bilinear programs, for the purpose of this paper, can be written as the following nonconvex nonlinear programming problem:

$$\text{Min } z = f_0 = \sum_{(i,j)\in BL_0} a_{ij0} x_i x_j + h_0(x)$$

S. Kolodziej · I. E. Grossmann (✉)
Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA15213-3890, USA
e-mail: grossmann@cmu.edu

P. M. Castro
Laboratório Nacional de Energia e Geologia, Unidade de Modelaçãoe Otimização de Sistemas Energéticos, 1649-038 Lisbon, Portugal

Subject to

$$f_q = \sum_{(i,j)\in BL_q} a_{ijq} x_i x_j + h_q(x) \leq 0 \quad q \in Q\backslash\{0\} \qquad \textbf{(P)}$$

$$x \in S \cap \Omega \subset \mathbb{R}^n$$

where $h_q(x)$ is convex and twice differentiable, a_{ijq} is a scalar with $i \in I$, $j \in J$, and $q \in Q$ represents the set of all functions f_q, including the objective function f_0 and all constraints. BL_q is an (i, j)-index set which defines the bilinear terms present in the problem. Although $i \neq j$ for strictly bilinear problems, $i = j$ can be allowed to accommodate quadratic problems. The set $S \subset \mathbb{R}^n$ is convex, and $\Omega \subset \mathbb{R}^n$ is an n-dimensional hyperrectangle defined in terms of the initial variable bounds x^L and x^U:

$$\Omega = \left\{ x \in \mathbb{R}^n : 0 \leq x^L \leq x \leq x^U \right\}$$

The global optimization of bilinear programs of the form of (P) is important in such areas as water networks and petroleum blending operations [1–4]. Nonconvex, bilinear constraints are required to model the mixing of various streams in these systems, and are in some cases the only nonlinearities in the models. The pooling problem, stemming from the original Haverly paper [5], contains these bilinear constraints and has received much attention in the literature [1,2,6–9]. Recently, Misener and Floudas have demonstrated a novel logarithmic relaxation for modeling bilinear terms with piecewise McCormick envelopes while addressing various classes of pooling problems [1].

Water network optimization problems containing bilinear terms have also received much attention in the literature [3,4,10–12]. The same blending constraints present in the pooling problem are present in water network problems, and thus numerous advances in solving bilinear programs have been made addressing these problems.

The global optimization of general nonconvex bilinear programs has received significant attention in the literature [13–27]. The convex McCormick envelopes [1] coupled with spatial branch and bound search frameworks have been the basis for many of these global optimization techniques, with piecewise McCormick envelopes being a more recent development. Variations of this approach have been suggested, generalizing the convex envelopes to piecewise over- and under-estimators [1,17]. Novel ways of representing bilinear terms through reformulation have been another approach reported in the literature [16]. Misener et al. [1], building on the work of Vielma and Nemhauser [28,29], have shown that a relaxation of the bilinear terms can be achieved with a logarithmic number of binary variables. Teles et al. [30] have introduced a technique to approximate polynomial constraints that exhibits a similar property, where the main element is the discretization of a subset of the variables.

In this paper, we show that the mixed-integer constraints of the multiparametric disaggregation technique (MDT) presented in [30], when applied to the bilinear terms in program (P), can be derived from disjunctive programming and convex hull reformulation. This approximation technique can be used under some conditions to obtain an upper bound. As the main novelty, we propose a rigorous lower bounding formulation derived from the upper bounding constraints. After proving that the solution from the upper and lower bounding formulations converge to that of the original nonlinear formulation in the limit of an infinite number of discretization intervals, two global optimization algorithms are proposed based on such bounds. Finally, we conclude with a comparison of this new relaxation approach and the one based on piecewise McCormick envelopes, and report computational results on small and large problems.

2 Discretization with multiparametric disaggregation

Given a nonconvex bilinear term $w_{ij} = x_i \cdot x_j$, the MDT described by Teles et al. [30] can be used to obtain an upper bound on problem (**P**). The complete formulation can be derived from a generalized disjunctive programming (GDP) model [31] followed by a convex hull reformulation [33] and exact linearization [32]. For simplicity in the notation, we first rewrite the bilinear product $w_{ij} = x_i \cdot x_j$ as a single bilinear term $w = u \cdot v$. This product can be represented exactly with the following constraints and disjunction:

$$w = u \cdot v \tag{1}$$

$$v = \sum_{\ell \in \mathbb{Z}} \lambda_\ell \tag{2}$$

$$\bigvee_{k=0}^{9} \left[\lambda_\ell = 10^\ell \cdot k \right] \quad \forall\, \ell \in \mathbb{Z} \tag{3}$$

where v is discretized through the disjunction in (3) that selects one digit $k \in K = \{0, 1, \ldots, 9\}$ for each power $\ell \in \mathbb{Z}$. Here we assume a basis of 10, although other bases can be selected [11]. Note that since (3) is defined over the domain of all the integer numbers, this implies an infinite number of disjunctions. Furthermore, v can represent any positive real number.

First, we consider the convex hull reformulation [33] of the disjunction in (3) after which we introduce the disaggregated variables,

$$\lambda_\ell = \sum_{k=0}^{9} \hat{\lambda}_{k,\ell} \quad \forall \ell \in \mathbb{Z} \tag{4}$$

$$\hat{\lambda}_{k,\ell} = 10^\ell \cdot k \cdot z_{k,\ell} \quad \forall\, \ell \in \mathbb{Z}, k \in K \tag{5}$$

$$\sum_{k=0}^{9} z_{k,\ell} = 1 \quad \forall\, \ell \in \mathbb{Z} \tag{6}$$

$$z_{k,\ell} \in \{0, 1\} \quad \forall\, \ell \in \mathbb{Z}, k \in K \tag{7}$$

Substituting (5) into (4) and then into (2) leads to the fully discretized (but still exact representation) of v:

$$v = \sum_{\ell \in \mathbb{Z}} \sum_{k=0}^{9} 10^\ell \cdot k \cdot z_{k,\ell} \tag{8}$$

Considering the product $w = u \cdot v$ by substituting (8) into (1) leads to (9), which involves nonlinear terms $u \cdot z_{k,\ell}$.

$$w = u \cdot \left[\sum_{\ell \in \mathbb{Z}} \sum_{k=0}^{9} 10^\ell \cdot k \cdot z_{k,\ell} \right] \tag{9}$$

Performing an exact linearization [32], we introduce new continuous variables $\hat{u}_{k,\ell} = u \cdot z_{k,\ell}$ so that:

$$w = \sum_{\ell \in \mathbb{Z}} \sum_{k=0}^{9} 10^\ell \cdot k \cdot \hat{u}_{k,\ell} \tag{10}$$

Since $u \cdot z_{k,\ell} = \begin{cases} 0 \text{ if } z_{k,\ell} = 0 \\ u \text{ if } z_{k,\ell} = 1 \end{cases}$ and $\hat{u}_{k,\ell}$ is non-negative, we introduce the following lower and upper bounding constraints, where u^U and u^L are the non-negative upper and lower bounds on u.

$$u^L \cdot z_{k,\ell} \leq \hat{u}_{k,\ell} \leq u^U \cdot z_{k,\ell} \quad \forall \ell \in \mathbb{Z}, k \in K \tag{11}$$

Finally, multiplying equation (6) by u and replacing the bilinear terms by the new continuous variables, results in (12). The full set of mixed integer linear constraints for the exact representation of bilinear product $w = u \cdot v$ is thus given by Eqs. (6–8) and (10–12).

$$u = \sum_{k=0}^{9} \hat{u}_{k,\ell} \quad \forall \ell \in \mathbb{Z} \tag{12}$$

We should note that the same model formulation can be obtained by the special structured RLT reformulation as described in Sherali et al. [48]. Constraints (6–8) are directly linked to the choice of using a base-10 representation for variable v. Then, the bounds of variable u are known: $u^L \leq u \leq u^U$. Multiplying the lower and upper bounding constraints $u - u^L \geq 0$ and $u^U - u \geq 0$ by $z_{k,\ell} \geq 0$ and replacing the bilinear term $u \cdot z_{k,\ell}$ by $\hat{u}_{k,\ell}$, leads to (11). Performing the same variable replacement in the constraint defining variable w, results in (10) while (12) can be obtained as previously described.

2.1 Upper bounding formulation

Since in practice it is infeasible to compute the infinite sums over all integers, we represent v to a finite level of precision, v', leading to a continuous but approximate representation of the bilinear term, w'. The constraints in (8) and (10) are modified in (13–14) to allow for a maximum power of 10 (P) and a minimum power of 10 (p). For the remaining constraints, (6–7) and (11–12), it suffices to replace $\ell \in \mathbb{Z}$ with $\ell \in L = \{p, p+1, \ldots, P\}$. These sets of constraints correspond to the equations proposed by Teles et. al. [30]:

$$w' = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot \hat{u}_{k,\ell} \tag{13}$$

$$v' = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot z_{k,\ell} \tag{14}$$

When we incorporate them into problem (**P**) by redefining $w_{ij} = x_i \cdot x_j$, and selecting x_j as the variable on which discretization is performed, the resulting problem (**P′**) will represent a mixed-integer approximation to the original problem:

$$\begin{aligned} &\text{Min } z' = f_0 = \sum_{(i,j) \in BL_0} a_{ij0} w_{ij} + h_0(x) \\ &\text{Subject to} \\ &f_q(x) = \sum_{(i,j) \in BL_q} a_{ijq} w_{ij} + h_q(x) \leq 0 \quad q \in Q \backslash \{0\} \end{aligned} \tag{P′}$$

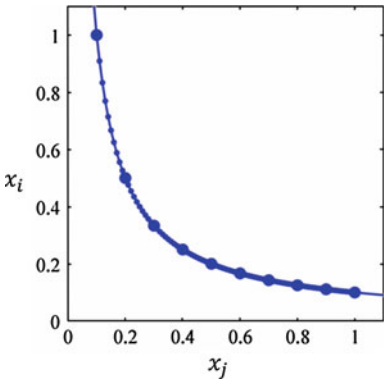

Fig. 1 Feasible region for a bilinear curve $x_i \cdot x_j = 0.1$, $p = -2$, $P = -1$. The *solid curve* represents the exact bilinear curve, while the *dots* represent the approximated (and incomplete) feasible region resulting from the upper bounding formulation

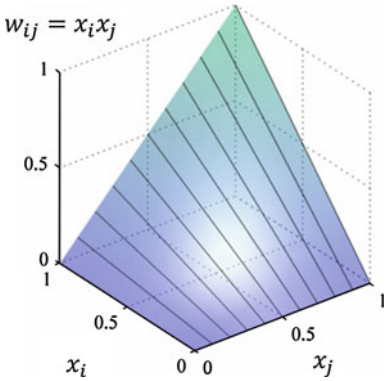

Fig. 2 Feasible region for a bilinear surface. The surface is the exact bilinear curve, while the *solid lines* represent the reduced feasible region resulting from the upper bounding formulation for $p = P = -1$

$$w_{ij} = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot \hat{x}_{ijk\ell} \quad \forall (i,j) \in BL_q, q \in Q$$

$$x_j = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot z_{jk\ell} \quad \forall j \in \{j | (i,j) \in BL_q, q \in Q\}$$

$$x_i = \sum_{k=0}^{9} \hat{x}_{ijk\ell} \quad \forall (i,j) \in BL_q, q \in Q, \ell \in L$$

$$x_i^L \cdot z_{jk\ell} \leq \hat{x}_{ijk\ell} \leq x_i^U \cdot z_{jk\ell} \quad \forall (i,j) \in BL_q, q \in Q, k \in K, \ell \in L$$

$$\sum_{k=0}^{9} z_{jk\ell} = 1 \quad \forall j \in \{j | (i,j) \in BL_q, q \in Q\}, \ell \in L$$

$$z_{jk\ell} \in \{0,1\} \quad \forall j \in \{j | (i,j) \in BL_q, q \in Q\}, k \in K, \ell \in L$$

$$x \in S \cap \Omega \subset \mathbb{R}^n$$

where x_j and w_{ij} represent discrete and continuous approximations to the variables, respectively. Note that if the convex functions $h_q(x)$ are linear, problem ($\mathbf{P}'$) represents a mixed integer linear program (MILP). Otherwise it corresponds to an MINLP.

Further, note that problem ($\mathbf{P}'$) is a restricted version of problem ($\mathbf{P}$), or equivalently problem ($\mathbf{P}$) is a relaxation of problem ($\mathbf{P}'$). It then follows that the solution of problem ($\mathbf{P}'$) either yields an upper bound on problem ($\mathbf{P}$) such that $z' = z^U \geq z$, or else problem ($\mathbf{P}'$) is infeasible. This restricted feasible region can be seen in Figs. 1 and 2.

2.2 Infeasibilities in the discretized problem

The mixed-integer approximation problem ($\mathbf{P}'$) can be infeasible even if the original problem ($\mathbf{P}$) is feasible. For example, if bounds such as $10^{p-1} \leq x_j \leq 2 \times 10^{p-1}$ are enforced, ($\mathbf{P}'$) will be infeasible, while such a constraint is completely valid and will not necessarily result in an infeasible problem ($\mathbf{P}$). Thus, the parameters p and P must be chosen appropriately to avoid such infeasibilities.

Some general guidelines for ensuring precision-based feasibility can be established. For example, the largest power of 10 (P) must be large enough such that 10^P is of the same order of magnitude as the upper bound on x_j. More precisely: $P = \lfloor \log_{10} x_j^U \rfloor$.

Additionally, p must be small enough to ensure that at least one (and preferably many) discretization points lie between the lower and upper bounds for x_j. Thus, $p \leq P$ is the absolute minimum requirement, but feasibility is more likely as p is decreased. Note that these guidelines do not guarantee feasibility of ($\mathbf{P}'$) in all cases, but represent the minimum level of precision needed given reasonable bounds on x_j.

3 Lower bounding with multiparametric disaggregation

To obtain a lower bounding problem using multiparametric disaggregation, we first note that in the discretized problem, there always exists a gap between discretization points for a finite p. Thus, we can introduce a slack variable Δx_j such that $x_j^R = x'_j + \Delta x_j$, where x'_j is the discretized representation of x_j from Sect. 2, x_j^R is the continuous representation of x_j and the slack variable Δx_j is bounded between 0 and 10^p.

Again switching to the notation $w = u \cdot v$ for the bilinear term, we have for the continuous representation of v, denoted as v^R:

$$v^R = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^\ell \cdot k \cdot z_{k,\ell} + \Delta v \tag{15}$$

$$0 \leq \Delta v \leq 10^p \tag{16}$$

For the continuous representation of the bilinear term, w^R, note that:

$$w^R = u \cdot v^R = u \cdot \left(v' + \Delta v\right) = w' + u \cdot \Delta v = w' + \Delta w \tag{17}$$

where w' and v' are given by (13–14). The slack variable Δw replaces the bilinear term $u \cdot \Delta v$ that can be relaxed using the McCormick envelope, (18–19), which in this case coincides with the RLT bound factor products $u^L \leq u \leq u^U$ and $0 \leq \Delta v \leq 10^p$:

$$u^L \cdot \Delta v \leq \Delta w \leq u^U \cdot \Delta v \tag{18}$$

$$\left(u - u^U\right) \cdot 10^p + u^U \cdot \Delta v \leq \Delta w \leq \left(u - u^L\right) \cdot 10^p + u^L \cdot \Delta v \tag{19}$$

Introducing these constraints into Problem (**P**), and expressing the variables in terms of the original variables $w_{ij} = x_i \cdot x_j$, we obtain the new optimization problem, (**PR**):

$$\text{Min } z^R = f_0 = \sum_{(i,j)\in BL_0} a_{ij0} w_{ij} + h_0(x)$$

subject to

$$\begin{aligned}
& f_q(x) = \sum_{(i,j)\in BL_q} a_{ijq} w_{ij} + h_q(x) \le 0 \quad q \in Q\backslash\{0\} \\
& w_{ij} = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot \hat{x}_{ijk\ell} + \Delta w_{ij} \quad \forall (i,j) \in BL_q, q \in Q \\
& x_j = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot z_{jk\ell} + \Delta x_j \quad \forall j \in \{j|(i,j) \in BL_q, q \in Q\} \quad \textbf{(PR)} \\
& x_i = \sum_{k=0}^{9} \hat{x}_{ijk\ell} \quad \forall (i,j) \in BL_Q, q \in Q, \ell \in L \\
& x_i^L \cdot z_{jk\ell} \le \hat{x}_{ijk\ell} \le x_i^U \cdot z_{jk\ell} \quad \forall (i,j) \in BL_q, q \in Q, \ell \in L, k \in K \\
& \sum_{k=0}^{9} z_{jk\ell} = 1 \quad \forall j \in \{j|(i,j) \in BL_q, q \in Q\}, \quad \ell \in L \\
& \left.\begin{aligned}
& x_i^L \cdot \Delta x_j \le \Delta w_{ij} \le x_i^U \cdot \Delta x_j \\
& \Delta w_{ij} \le \left(x_i - x_i^L\right) \cdot 10^p + x_i^L \cdot \Delta x_j \\
& \Delta w_{ij} \ge \left(x_i - x_i^U\right) \cdot 10^p + x_i^U \cdot \Delta x_j
\end{aligned}\right\} \forall (i,j) \in BL_q, q \in Q \\
& 0 \le \Delta x_j \le 10^p \quad \forall j \in \{j|(i,j) \in BL_q, q \in Q\} \\
& z_{jk\ell} \in \{0,1\} \quad \forall j \in \{j|(i,j) \in BL_q, q \in Q\}, k \in K, \ell \in L \\
& x \in S \cap \Omega \subset \mathbb{R}^n
\end{aligned}$$

While (**PR**) does not exactly represent the product $w_{ij} = x_i \cdot x_j$ and is feasible for values of w_{ij}, x_i, and x_j that do not satisfy $w_{ij} = x_i \cdot x_j$, the bilinear term is feasible in (**PR**). Thus, (**PR**) is a relaxation of (**P**). The relaxed feasible region resulting from (**PR**) can be seen in Figs. 3 and 4.

The following property can be readily established from the above discussion:

Property 1 The solution of problem (**PR**) yields a lower bound for problem (**P**), i.e. $z^R \le z$.

4 Discussion of global optimization algorithms

The upper and lower bounding schemes described can be combined into a global optimization algorithm. First, the following property can be established:

Property 2 As p approaches $-\infty$, z' approaches z^R.

Proof As p approaches $-\infty$ in (**PR**)

$$\lim_{p\to-\infty} \Delta x_j = \lim_{p\to-\infty} 10^p = 0$$

$$\lim_{p\to-\infty} \Delta w_{ij} = x_i^U \cdot \lim_{p\to-\infty} \Delta x_j = \left(x_i - x_i^L\right) \cdot \lim_{p\to-\infty} 10^p + x_i^L \cdot \lim_{p\to-\infty} \Delta x_j = 0$$

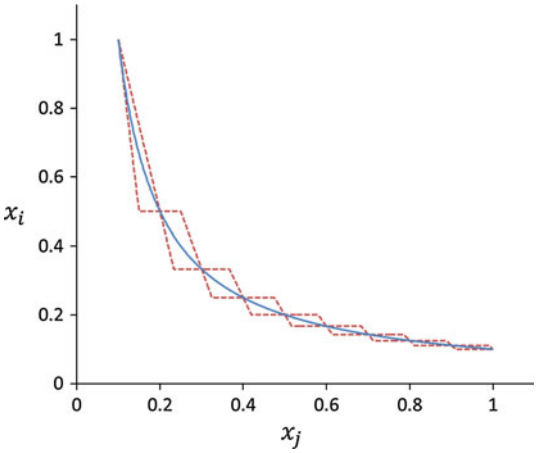

Fig. 3 Plot of the lower bounding feasible region for $w_{ij} = 0.1$ and $p = P = -1$. The *solid line* is the true curve, while the *dotted lines* represent the boundaries of the relaxed feasible region of (PR). Note the similarities to piecewise McCormick envelopes

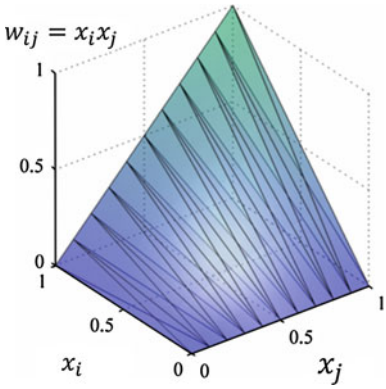

Fig. 4 Feasible region of the relaxed problem (PR). The surface is the exact bilinear curve, while the *envelopes* represent the relaxed feasible region resulting from the lower bounding formulation for $p = P = -1$

Thus, since the variables Δx_j and Δw_{ij} are eliminated, this yields problem (**P**′), and hence z' approaches z^R. □

From Property 2, we can establish that as precision is increased (i.e. p approaches $-\infty$), both (**P**′) and (**PR**) converge to the same value. Assuming P is large enough such that $10^P \geq x_j^U$, we can further state that (**P**′) and (**PR**) converge such that $z' = z^R = z$.

4.1 Algorithm 1

The first global optimization algorithm that can be established from the aforementioned upper and lower bounding is as follows. First, we start at some coarse level of discretization such that $P \geq p$, and solve both (**PR**) and (**P**′). If the difference in solutions to the upper and lower bounding problems is sufficiently small, then the algorithm terminates; otherwise, precision is increased and the problems are resolved. The algorithm is then as follows:

Algorithm 1

Step 0. Choose $p = P = \lfloor \log_{10} x_j^U \rfloor$
Step 1. Solve (**PR**) to obtain the lower bound z^R.
Step 2. Solve (**P**′) to obtain the upper bound z'. If (**P**′) is infeasible, let $z' = +\infty$.

Step 3. If $(z'-z^R)/z^R \leq \varepsilon$, STOP, the solution is globally optimal. Otherwise, set $p = p-1$, and return to step 1.

4.2 Algorithm 2

While Algorithm 1 follows most naturally from the problems (**P**′) and (**PR**), it has several shortcomings. Notably, because (**P**′) and (**PR**) are fairly similar and increase similarly in problem size as precision is added, solving (**P**′) and (**PR**) repeatedly becomes increasingly expensive. Thus, an alternative method for obtaining an upper bound is to use a local NLP algorithm in place of solving the problem (**P**′).

Algorithm 2

Step 0. Choose $p = P = \lfloor \log_{10} x_j^U \rfloor$
Step 1. Solve (**PR**) to obtain the lower bound z^R.
Step 2. Solve (**P**) using a local NLP algorithm to obtain some upper bound z using the solution to (**PR**) as a starting point.
Step 3. If $(z-z^R)/z^R \leq \varepsilon$, STOP, the solution is globally optimal. Otherwise, set $p = p-1$ and return to step 1.

Algorithm 2 is generally more computationally efficient than Algorithm 1, as the solution of (**P**) using a local NLP algorithm is obtained much faster than the increasingly large MILP that (**P**′) becomes as $P - p$ grows.

4.3 Extensions to MINLP

The algorithms in Sects. 4.1 and 4.2 can be readily extended for solving MINLPs with the following general form:

$$\text{Min } z = f_0(x, y)$$

subject to

$$\begin{aligned} & f_q(x, y) \leq 0 \quad q \in Q \backslash \{0\} \\ & f_q(x, y) = \sum_{(i,j) \in BL_q} a_{ijq} x_i x_j + h_q(x, y) \quad q \in Q \qquad \textbf{(P-MINLP)} \\ & x \in S \cap \Omega \subset \mathbb{R}^n \\ & y \in \{0, 1\} \end{aligned}$$

where $h_q(x, y)$ is jointly convex in x and y. For most cases in practice the variables y appear in linear form in these terms [34].

Analogous problems (**P**′-**MINLP**) and (**PR-MINLP**) can be derived, and Algorithm 1 can be used without modification. However, Algorithm 2 requires some modification, as (**P-MINLP**) cannot be solved using a local NLP algorithm because it is an MINLP. A heuristic can be used to compute the upper bound as in Algorithm 2:

Algorithm 3

Step 0. Choose $p = P = \lfloor \log_{10} x_j^U \rfloor$
Step 1. Solve (**PR-MINLP**) to obtain z^R.

Step 2. Fix the binary variables y in (**P-MINLP**) to the values found by the solution of (**PR-MINLP**) in Step 1, reducing it to an NLP. Solve (**P-MINLP**) with these fixed binary variables using a local NLP algorithm to obtain some z using the solution to (**PR-MINLP**) as a starting point.

Step 3. If $(z-z^R)/z^R \leq \varepsilon$, STOP, the solution is globally optimal. Otherwise, set $p = p-1$ and return to step 1.

A disadvantage of this algorithm is that it could take, in the worst case, an infinite number of iterations to converge. By fixing the binary variables to that of the solution to (**PR-MINLP**), (**P-MINLP**) can be rendered infeasible, and algorithm would continue until (**PR-MINLP**) has enough discretization points to exactly represent (**P-MINLP**). If this occurs, heuristics or other approaches may be utilized to obtain an upper bound in place of solving (**P-MINLP**) with fixed binary variables. However, in practice, this is unlikely to occur.

5 Comparison with piecewise McCormick envelopes

A common approach to solving bilinear programs of the form (**P**) is to bound the bilinear terms using McCormick convex envelopes [35]. This formulation results in a lower bounding LP if the only nonlinearities are bilinear, or a lower bounding convex NLP if there are remaining convex nonlinearities. For each bilinearity $w_{ij} = x_i \cdot x_j$, we introduce instead the following constraints:

$$\left.\begin{array}{l} w_{ij} \geq x_i \cdot x_j^L + x_i^L \cdot x_j - x_i^L \cdot x_j^L \\ w_{ij} \geq x_i \cdot x_j^U + x_i^U \cdot x_j - x_i^U \cdot x_j^U \\ w_{ij} \leq x_i \cdot x_j^L + x_i^U \cdot x_j - x_i^U \cdot x_j^L \\ w_{ij} \leq x_i \cdot x_j^U + x_i^L \cdot x_j - x_i^L \cdot x_j^U \end{array}\right\} \quad \forall (i,j) \in BL_q, q \in Q \tag{20}$$

This relaxation yields a lower bound on problem (**P**). However, this lower bound can be weak depending on the bounds on x_i and x_j. To improve the quality of the lower bound, these convex envelopes can be used on discretized portions of the variable range. Thus, piecewise McCormick envelopes can be introduced in (**P**) to obtain a tighter lower bound at the cost of becoming an MILP or convex MINLP [10,17,36]. The envelopes, defined over a set of N points on the domain of variable x_j, can be represented by the following disjunctive constraints:

$$\begin{array}{l} \vee_{n=1}^{N} \left[\begin{array}{c} y_{jn} \\ \left.\begin{array}{ll} w_{ij} \geq x_i \cdot x_{jn}^L + x_i^L \cdot x_j - x_i^L \cdot x_{jn}^L & \forall \{i | (i,j) \in BL_q, q \in Q\} \\ w_{ij} \geq x_i \cdot x_{jn}^U + x_i^U \cdot x_j - x_i^U \cdot x_{jn}^U & \forall \{i | (i,j) \in BL_q, q \in Q\} \\ w_{ij} \geq x_i \cdot x_{jn}^L + x_i^U \cdot x_j - x_i^U \cdot x_{jn}^L & \forall \left\{i | (i,j) \in BL_q, q \in Q\right\} \\ w_{ij} \geq x_i \cdot x_{jn}^U + x_i^L \cdot x_j - x_i^L \cdot x_{jn}^U & \forall \{i | (i,j) \in BL_q, q \in Q\} \end{array}\right\} \\ x_{jn}^L \leq x_j \leq x_{jn}^U \end{array}\right] \quad \forall \{j | (i,j) \in BL_q, q \in Q\} \\ x_i^L \leq x_i \leq x_i^U \quad \forall \{i | (i,j) \in BL_q, q \in Q\} \\ \left.\begin{array}{l} x_{jn}^L = x_j^L + \left(x_j^U - x_j^L\right) \cdot (n-1)/N \\ x_{jn}^U = x_j^L + \left(x_j^U - x_j^L\right) \cdot n/N \end{array}\right\} \quad \forall \left\{j | (i,j) \in BL_q, q \in Q\right\}, \quad n \in \{1, \ldots, N\} \end{array} \tag{21}$$

Applying the convex hull reformulation [33] for the above disjunctive constraints yields

$$\left.\begin{array}{l} w_{ij} \geq \sum_{n=1}^{N} \left(\hat{x}_{ijn} \cdot x_{jn}^{L} + x_i^{L} \cdot x_{jn} - x_i^{L} \cdot x_{jn}^{L} \cdot y_{jn}\right) \\ w_{ij} \geq \sum_{n=1}^{N} \left(\hat{x}_{ijn} \cdot x_{jn}^{U} + x_i^{U} \cdot \hat{x}_{jn} - x_i^{U} \cdot x_{jn}^{U} \cdot y_{jn}\right) \\ w_{ij} \leq \sum_{n=1}^{N} \left(\hat{x}_{ijn} \cdot x_{jn}^{L} + x_i^{U} \cdot \hat{x}_{jn} - x_i^{U} \cdot x_{jn}^{L} \cdot y_{jn}\right) \\ w_{ij} \leq \sum_{n=1}^{N} \left(\hat{x}_{ijn} \cdot x_{jn}^{U} + x_i^{L} \cdot \hat{x}_{jn} - x_i^{L} \cdot x_{jn}^{U} \cdot y_{jn}\right) \\ x_i = \sum_{n=1}^{N} \hat{x}_{ijn} \end{array}\right\} \forall (i,j) \in BL_q, q \in Q$$

$$\left.\begin{array}{l} x_j = \sum_{n=1}^{N} \hat{x}_{jn} \\ \sum_{n=1}^{N} y_{jn} = 1 \end{array}\right\} \forall \left\{ j \,|\, (i,j) \in BL_q, q \in Q \right\}$$

$$x_i^{L} \cdot y_{jn} \leq \hat{x}_{ijn} \leq x_i^{U} \cdot y_{jn} \forall (i,j) \in BL_q, \quad q \in Q, n \in \{1, \ldots, N\}$$

$$\left.\begin{array}{l} x_{jn}^{L} \cdot y_{jn} \leq \hat{x}_{jn} \leq x_{jn}^{U} \cdot y_{jn} \\ y_{jn} \in \{0,1\} \end{array}\right\} \forall \left\{ j \,|\, (i,j) \in BL_q, q \in Q \right\}, \quad n \in \{1, \ldots, N\} \qquad (22)$$

By adding these piecewise McCormick envelope constraints (22) to problem (**P**), we can define a new relaxed MILP or convex MINLP, (**PR-PCM**). Furthermore, we can compare the performance of (**PR-PCM**) to the lower bounding problem derived from multiparametric disaggregation, (**PR**).

6 Illustrative example (P1)

We first consider as an example the bilinear program by Quesada and Grossmann [37] and originally reported by Al-Khayyal and Falk [18]:

$$\text{Min } f = -x_1 + x_1 x_2 - x_2$$

subject to

$$\begin{aligned} -6x_1 + 8x_2 &\leq 3 \\ 3x_1 - x_2 &\leq 3 \\ 0 \leq x_1, x_2 &\leq 1.5 \end{aligned} \qquad (\textbf{P1})$$

The global optimum of this bilinear program is $f = -1.08333$ at (1.165864, 0.497580). Two other local minima correspond to: $f = -1.0$ at (1, 1), and $f = -1.005$ at (0.917, 1.062).

6.1 Lower bounding problems

In order to compare the lower bounds predicted by multiparametric disaggregation and piecewise McCormick envelopes, we solve the relaxation problems (**PR**) and (**PR-PCM**) resulting from multiparametric disaggregation and piecewise McCormick, respectively. For the specific example Problem (**P1**), we can derive analogous relaxed problems (**P1R**) and (**P1R-PCM**) using the MDT and piecewise McCormick envelopes, respectively. Note that x_1 is the variable being discretized.

Problem size and computational results are shown in Table 1 for the lower bounds predicted by (**P1R**) at $P = 0$ and $p = \{0, -1, \ldots, -6\}$ and (**P1R-PCM**) at $N = \{1, 15, 150, 1{,}500, 15{,}000\}$. Solving (**P1R-PCM**) at various levels of discretization, it becomes clear that problem size increases approximately exponentially with each order of

$\text{Min} f = -x_1 + w_{12} - x_2$
subject to
$-6x_1 + 8x_2 \leq 3$
$3x_1 - x_2 \leq 3$
$0 \leq x_1, x_2 \leq 1.5$

$w_{12} = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot \hat{x}_{k,\ell} + \Delta w_{12}$
$x_1 = \sum_{\ell=p}^{P} \sum_{k=0}^{9} 10^{\ell} \cdot k \cdot z_{k,\ell} + \Delta x_1$
$x_2 = \sum_{k=0}^{9} \hat{x}_{k,\ell} \quad \forall \ell \in L$
$\hat{x}_{k,\ell} \leq 1.5 \cdot z_{k,\ell} \forall \ell \in L, k \in K$
$\sum_{k=0}^{9} z_{k,\ell} = 1 \quad \forall \ell \in L$
$0 \leq \Delta w_{12} \leq 1.5 \Delta x_1$
$(x_2 - 1.5) \cdot 10^p + 1.5\Delta x_1 \leq \Delta w_{12} \leq x_2 \cdot 10^p$
$0 \leq \Delta x_1 \leq 1.5 \cdot 10^p$
$z_{k,\ell} \in \{0, 1\} \forall \ell \in L, k \in K$
$0 \leq \hat{x}_{k,\ell} \leq 1.5 \forall \ell \in L, k \in K$

(**P1R**)

$w_{12} \leq \sum_{n=1}^{N} 1.5\hat{x}_{1n} + \hat{x}_{2n} x_{1n}^{L} - 1.5 x_{1n}^{L} y_n$
$w_{12} \leq \sum_{n=1}^{N} \hat{x}_{2n} x_{1n}^{U}$
$w_{12} \geq \sum_{n=1}^{N} \hat{x}_{2n} x_{1n}^{L}$
$w_{12} \geq \sum_{n=1}^{N} 1.5\hat{x}_{1n} + \hat{x}_{2n} x_{1n}^{U} - 1.5 x_{1n}^{U} y_n$
$x_1 = \sum_{n=1}^{N} \hat{x}_{1n}$
$x_2 = \sum_{n=1}^{N} \hat{x}_{2n}$
$y_n x_{1n}^{L} \leq \hat{x}_{1n} \leq y_n x_{1n}^{U} \forall n = 1, 2 \ldots N$
$0 \leq \hat{x}_{2n} \leq 1.5 y_n \forall n = 1, 2 \ldots N$
$\sum_{n=1}^{N} y_n = 1$
$x_{1n}^{L} = x_1^{L} + x_n^{U} - x_n^{L}/N \cdot (n-1) \forall n = 1, 2 \ldots N$
$x_{1n}^{U} = x_1^{L} + x_n^{U} - x_n^{L}/N \cdot n \forall n = 1, 2 \ldots N$
$y_n \in \{0, 1\} \forall n = 1, 2 \ldots N$

(**P1R-PCM**)

magnitude decrease in the optimality gap. However, this is not the case when solving (**P1R**) since as precision is added, i.e. p is decreased, a linear relationship holds. Note that the discretization level of MDT at $p = -1$ matches that of PCM at $N = 15 (p = -2$ is equivalent to $N = 150$ and so on) and that the lower bounds are exactly the same. Upper bounds are also reported by using as a starting point the solution of the lower bounding relaxation problem as in Algorithm 2 in Sect. 4.2. Thus, the MDT columns are equivalent to the iterations of the algorithm with the reported CPU time being the total time required for the lower and upper bounding problems. These times would need to be accumulated to compare Algorithm 2 directly to BARON, unless the discretization level (p) is chosen a priori. Results for solving the original NLP (**P1**) with the global optimization solver BARON are reported in the second column.

As seen in Table 1, the relaxed problems (**P1R**) and (**P1R-PCM**) are considerably larger than the original NLP (**P1**) since the addition of both continuous and binary variables increases problem size. However, note that the multiparametric disaggregation problem (**P1R**) requires fewer additional continuous variables and fewer constraints than when utilizing piecewise McCormick envelopes in (**P1R-PCM**). For a single McCormick envelope, a relatively weak lower bound of -1.5 is obtained leading to an upper bound that is in fact a suboptimal solution. Multiparametric disaggregation with $p = -4$ approaches an optimality gap of 0.003 %, while piecewise McCormick envelopes, even with 1,500 partitions, only approach an optimality gap of 0.034 %. Further discretization and refinement of the solution is quickly solved using multiparametric disaggregation, as the $p = -5$ and $p = -6$ problems are solved in <1.2 s and fewer than 130 variables. In contrast, increasing the number of partitions to 15,000 in (**P1R-PCM**), leads to a MILP that cannot be solved in 1-h of computational time.

7 Numerical experiments

The performance of the underestimating problems from multiparametric disaggregation (**PR**) and piecewise McCormick (**PR-PCM**) is further evaluated through the solution of three

Table 1 Comparison between multiparametric disaggregation (MDT) and piecewise McCormick (PCM), problem (**P1**). For MDT, $P = 0$ is used

Method	BARON[a]	MDT	MDT	MDT	MDT	MDT	MDT	MDT	PCM	PCM	PCM	PCM	PCM
p or N	–	0	−1	−2	−3	−4	−5	−6	1	15	150	1,500	15,000
Binary variables	0	2	12	22	32	42	52	62	1	15	150	1,500	15,000
Total variables	3	10	30	50	70	90	110	130	7	49	454	4,504	45,004
Equations	3	15	37	59	81	103	125	147	14	70	610	6,010	60,010
Lower bound (CPLEX)	−1.08333	−1.33333	−1.11667	−1.08667	−1.08367	−1.08337	−1.08334	−1.08333	−1.5	−1.11667	−1.08667	−1.08367	−1.38848
Upper bound (CONOPT)	−1.08333	−1.08333	−1.08333	−1.08333	−1.08333	−1.08333	−1.08333	−1.08333	−1.00521	−1.08333	−1.08333	−1.08333	−1.08333
Optimality gap (%)	0.0000	18.8	2.98	0.31	0.034	0.003	0.0003	0.0000	33.0	2.98	0.31	0.034	22.0
CPU time (s)	0.34	0.24	0.24	0.23	0.25	0.25	0.58	1.19	0.22	0.25	0.68	7.70	*3,600*

[a] Applied directly to (**P1**)

Italic values indicate that the maximum computational time was reached

Table 2 Comparison between multiparametric disaggregation (MDT) and piecewise McCormick (PCM), problem (**P2**)

Method	BARON[a]	MDT	MDT	MDT	MDT	MDT	PCM	PCM	PCM
p or N	–	(0,0,0)	$(-1, -1, -1)$	$(-2, -2, -2)$	$(-3, -3, -3)$	$(-4, -4, -4)$	(24,12,18)	(240,120,180)	(2,400,1,200,1,800)
Binary variables	0	47	77	107	137	167	54	540	5,400
Total variables	9	238	378	518	658	798	296	2,780	27,620
Equations	10	413	647	881	1,115	1,349	513	4,509	44,469
Lower bound (CPLEX)	10122.48	10121.33	10121.33	10122.21	10122.46	10122.49	10121.84	10121.84	10122.29
Upper bound (CONOPT)	10122.49	10122.49	10122.49	10122.49	10122.49	10122.49	10122.49	10122.49	10122.49
Optimality gap (%)	0.0001	0.011	0.011	0.003	0.0003	0.0000	0.006	0.006	0.002
CPU Time (s)	0.35	0.43	0.35	0.56	0.66	1.11	0.44	1.42	133

[a] Applied directly to (**P2**)
For MDT, $P = (2, 1, 1)$ is used

Table 3 Comparison between multiparametric disaggregation (MDT) and piecewise McCormick (PCM), problem (**P3**)

	BARON[a]	MDT	MDT	MDT	MDT	PCM	PCM
$P = (4, 4, 4)$ **(P3a)**							
p or N		(2,2,2)	(1,1,1)	(0,0,0)	$(-1, -1, -1)$	(99,90,90)	(990,900,900)
Lower bound (CPLEX)	7049.241	6378.038	6978.526	7042.766	7048.580	6378.038	5544.592
Upper bound (CONOPT)	7049.248	7049.248	7049.248	7049.248	7049.248	7049.248	7049.248
Optimality gap (%)	0.0001	10.5	1.01	0.09	0.009	10.5	27.1
CPU time (s)	0.44	1.28	18.1	278	2,673	7.32	*3,600*
$P = (4, 3, 3, 3)$ **(P3b)**							
p or N		(1,1,1,1)	(0,0,0,0)	$(-1, -1, -1, -1)$	$(-2, -2, -2, -2)$	(900,99,99,99)	(9000,990,990,990)
Lower bound (CPLEX)		6610.973	7002.303	7044.518	7048.775	6610.973	2732.460
Upper bound (CONOPT)		7049.248	7049.248	7049.248	7049.248	7049.248	7049.248
Optimality gap (%)		6.63	0.67	0.067	0.0067	6.63	158
CPU time (s)		2.26	31.8	184	1,898	436	*3,600*
$P = (3, 3, 3, 3, 3)$**(P3c)**							
p or N		(1,1,1,1,1)	(0,0,0,0,0)	$(-1, -1, -1, -1, -1)$	$(-2, -2, -2, -2, -2)$	(99,99,99,99,99)	(990,990,990,990,990)
Lower bound (CPLEX)		6591.393	6999.466	7044.224	7048.745	6591.393	2909.401
Upper bound (CONOPT)		7049.248	7049.248	7049.248	7049.248	7049.248	7049.248
Optimality gap (%)		6.95	0.71	0.07	0.007	6.95	142
CPU time (s)		1.18	6.24	45.7	520	8.34	*3,600*

[a] Applied directly to (**P3**)

Italic values indicate that the maximum computational time was reached

additional small test problems from the literature for different accuracy levels. Since in all problems the functions $h_q(x)$ in (**P**) are linear, the resulting bounding MILP problems were solved in GAMS 23.8.2 [38] using CPLEX 12.4 (1 thread) [39]. Default options were used except for the relative optimality tolerance, equal to 10^{-6} and a maximum computational effort equal to 3,600 CPU seconds. Similarly as in problem (**P1**), the original nonlinear programs were solved by CONOPT 3 [40], following initialization with the values from the MILP, and by BARON 10.2 [41]. We also report results for larger problems in the areas of water networks and multiperiod blending. Such problems were also solved by GloMIQO 1.0.0 [49]. The computational experiments were performed on an Intel i7 950 processor running at 3.07 GHz, with 8 GB of RAM, running Windows 7.

7.1 Small test problems

7.1.1 P2

Problem (**P2**) is originally from the compilation of test problems by Rijckaert and Martens [42] but has been converted to a bilinear program following simple transformations and addition of new variables. The global optimal solution is $f = 10122.4932$ at (78, 33, 29.995740, 45, 36.775327, 0.030303, 27.264982, 0.033338). Variables x_1, x_2 and x_3 are selected for parameterization/partitioning.

$$
\begin{aligned}
\min\ & f = 5.3578x_3^2 + 0.8357x_1x_5 + 37.2392x_1 \\
s.t.\ & 0.00002584x_3x_5 - 0.00006663x_2x_5 - 0.0000734x_1x_4 \le 1 \\
& 0.000853007x_2x_5 + 0.00009395x_1x_4 - 0.00033085x_3x_5 \le 1 \\
& 1330.3294x_6 - 0.42x_1 - 0.30586x_7 \le x_5 \\
& 0.00024186x_2x_5 + 0.00010159x_1x_2 + 0.00007379x_3^2 \le 1 \\
& 2275.1327x_8 - 0.2668x_1 - 0.40584x_4 \le x_5 \\
& 0.00029955x_3x_5 + 0.00007992x_1x_3 + 0.00012157x_3x_4 \le 1 \\
& 1 = x_2x_6 \\
& x_7x_2 = x_3^2 \\
& 1 = x_8x_3 \\
& 78 \le x_1 \le 102, \quad 33 \le x_2 \le 45, \quad 27 \le x_3 \le 45, \quad 27 \le x_4 \le 45, \quad 27 \le x_5 \le 45 \\
& \frac{1}{45} \le x_6 \le \frac{1}{33}, \quad \frac{27^2}{45} \le x_7 \le \frac{45^2}{33}, \quad \frac{1}{45} \le x_8 \le \frac{1}{27}.
\end{aligned}
$$

7.1.2 P3

Test problem (**P3**) corresponds to Problem 106 in Hock and Schittowski [43]. While the objective function and the first three constraints are linear, there are three bilinear inequalities. To study how the choice of parameterized/partitioning variables affects computational performance, three versions are considered: (a) 3 parameterized/partitioned variables, x_1, x_2 and x_3; (b) 4 variables, x_2, x_5, x_6 and x_8; (c) 5 variables, x_4, x_5, x_6, x_7 and x_8. The global optimal solution is $f = 7049.2479$ at (579.573535, 1359.977872, 5109.696534, 182.039993, 295.612139, 217.960007, 286.427855, 395.612139).

$$
\begin{aligned}
\min \quad & f = x_1 + x_2 + x_3 \\
s.t. \quad & 0.0025(x_4 + x_6) - 1 \leq 0 \\
& 0.0025(-x_4 + x_5 + x_7) - 1 \leq 0 \\
& 0.01(-x_5 + x_8) - 1 \leq 0 \\
& 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \leq 0 \\
& x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \leq 0 \\
& x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \leq 0 \\
& 100 \leq x_1 \leq 10000,\ 1000 \leq x_2, x_3 \leq 10000,\ 10 \leq x_4, x_5, x_6, x_7, x_8 \leq 100
\end{aligned}
$$

7.1.3 P4

Test problem (**P4**) is taken from Shen and Zhang [44] and after a few transformations the following bilinear problem results. It features a global optimal solution $f = 460212.25$ at (43.165468, 45, 70, 1.228374, 27.749229, 0.814083). Variables x_2, x_5 and x_6 are chosen as the parameterized variables.

$$
\begin{aligned}
\min \quad & f = 168x_1x_2 + 3651.2x_5 + 40000x_6 \\
s.t. \quad & 1.0425x_1 - x_2 \leq 0 \\
& 0.00035x_1x_2 - 1 \leq 0 \\
& 1.25x_4 - x_1 + 41.63 \leq 0 \\
& x_1x_2 - x_3x_5 = 0 \\
& x_4x_6 - 1 = 0 \\
& 40 \leq x_1 \leq 44,\ 40 \leq x_2 \leq 45,\ 60 \leq x_3 \leq 70,\ 0.1 \leq x_4 \leq 1.4 \\
& 22.85714 \leq x_5 \leq 33,\ 0.714286 \leq x_6 \leq 10
\end{aligned}
$$

7.2 Computational statistics

Tables 1, 2, 3 and 4 give the computational results for problems (**P1**–**P4**) as a function of the discretization level, and the columns correspond to the iterations of Algorithm 2, where the relaxation problem is either generated by multiparametric disaggregation (**PR**), MDT columns, or by the piecewise McCormick envelopes (**PR-PCM**), PCM columns. We provide the problem size, lower bound from the relaxation problem, and upper bound from a local NLP solver (the values of the latter remain the same independent of the accuracy level). The optimality gap and total computational effort (CPLEX plus CONOPT, the latter being almost negligible) are also reported. Note that the triplets in Table 2 are related to the number of discrete points for the three variables selected. The n-tuples in Tables 3 and 4 have a similar meaning. In problems (**P2**–**P3**), the discretization points of PCM match exactly those being used by MDT (e.g. second column of PCM should be compared with second column of MDT).

An interesting observation is that for the same accuracy level, the relaxation from piecewise McCormick (PCM) is tighter than MDT in (**P2**) but identical in (**P1**) and (**P3**). If one looks at the original NLP problems, the latter problems have strictly bilinear terms whereas (**P2**) features a quadratic term in x_3, which is one of the variables being discretized. Applying the piecewise relaxation to a quadratic term $w_{ii} = x_i \cdot x_i$ makes it possible to use tighter bounds for both parts of x_i in each partition n, see (23), whereas with MDT just one part is discretized, meaning the use of the overall bounds for the other part. This provides the

Table 4 Comparison between multiparametric disaggregation (MDT) and piecewise McCormick (PCM), problem (**P4**)

Method p or N	BARON[a]	MDT (0,0,0)	MDT (−1, −1, −1)	MDT (−2, −2, −2)	MDT (−3, −3, −3)	MDT (−4, −4, −4)	PCM (5,10,10)	PCM (50,100,100)	PCM (500,1000,1000)	PCM (5000,10000,10000)
Binary variables	0	35	65	95	125	155	25	250	2,500	25,000
Total variables	7	86	146	206	266	326	85	760	7,510	75,010
Equations	6	106	172	238	304	370	127	1,027	10,027	100,027
Lower bound (CPLEX)	460211.8	451223.8	459917.9	460177.9	460209.7	460211.8	457162.4	459976.0	460171.7	460209.0
Upper bound (CONOPT)	460212.3	460212.3	460212.3	460212.3	460212.3	460212.3	460212.3	460212.3	460212.3	460212.3
Optimality gap (%)	0.0001	1.99	0.06	0.007	0.0006	0.0001	0.67	0.05	0.009	0.0007
CPU time (s)	0.24	0.23	0.25	0.36	0.47	0.45	0.33	0.56	7.85	787

For MDT, $P = (1, 1, 1)$ is used

[a] Applied directly to (**P4**)

Table 5 Comparison for water-using networks design problems [11]

Problem	Original NLP		BARON		GloMIQO		MDT					
	Variables	Equations	CPU Time (s)	Gap (%)	CPU time (s)	Gap (%)	Binary variables	Total variables	Equations	p	CPU time (s)	Gap (%)
Ex14	123	81	*3,600*	2.01	1,980	0.01	445	4,560	2,683	0	*3,600*	0.39
Ex15	136	87	*3,600*	1.52	1,916	0.01	576	5,860	3,384	0	*3,600*	0.48
Ex17	74	38	*3,600*	2.44	*3,600*	3.14	104	970	887	1	3.61	0.01
Ex18	60	37	*3,600*	0.98	*3,600*	0.82	178	1,272	887	0	2.11	0.01
Ex20	171	84	*3,600*	3.75	*3,600*	2.48	451	5,572	2,833	0	*3,600*	0.39

Results for MDT obtained using algorithm 2 (data for last iteration)
Italic values indicate that the maximum computational time was reached

explanation for multiparametric disaggregation being less tight than piecewise McCormick, which can be observed in Fig. 5.

$$\vee_{n=1}^{N} \begin{bmatrix} y_{in} \\ w_{ii} \geq x_i \cdot x_{in}^L + x_{i,n}^L \cdot x_i - x_{in}^L \cdot x_{in}^L \\ w_{ii} \geq x_i \cdot x_{in}^U + x_{i,n}^U \cdot x_i - x_{in}^U \cdot x_{in}^U \\ w_{ii} \leq x_i \cdot x_{in}^L + x_{i,n}^U \cdot x_i - x_{in}^U \cdot x_{in}^L \\ x_{in}^L \leq x_i \leq x_{in}^U \end{bmatrix} \tag{23}$$
$$x_{in}^L = x_i^L + \left(x_i^U - x_i^L\right) \cdot (n-1)/N$$
$$x_{in}^U = x_i^L + \left(x_i^U - x_i^L\right) \cdot n/N$$

For (**P4**), the lower bounds of x_5 and x_6 are not integer values and so, while the number of discrete points is roughly the same, their location is not, making it possible for the relaxation from MDT to be better than the one from PCM. For instance, $p = (-2, -2, -2)$ leads to a slightly better lower bound than $N = (500, 1,000, 1,000)$. Due to the better performance of the new method, there are more columns for MDT than for PCM, meaning that higher accuracy levels, i.e. lower optimality gaps, can be achieved by the former for a given computational time.

While MDT is not as tight as PCM, the latter generates considerably larger MILP problems for the same accuracy level. More specifically, with PCM we get exactly an exponential increase in the number of binary variables with the change of power and roughly the same behavior with respect to the number of total variables and constraints. In other words, the number of 0–1 variables grows linearly with the number of partitions. In contrast, the number of 0–1 variables for MDT grows logarithmically with the number of discrete points and linearly with the change of power, keeping problems tractable for a wider accuracy range. Notice that with MDT all problems can be solved with an optimality gap of less than 0.01 % in less than one hour, while PCM closes only to a gap of 6.63 % for (**P3**).

Since problem size is related to the number of parameterized/partitioned variables, one might be tempted to keep this number as low as possible. However, the results for (**P3**) give opposite results, since the worst performance is obtained for 3 parameterized variables (gap = 0.009 %), followed by 4 (gap = 0.0067 % in 1898 CPUs) and then 5 (gap = 0.0071 %

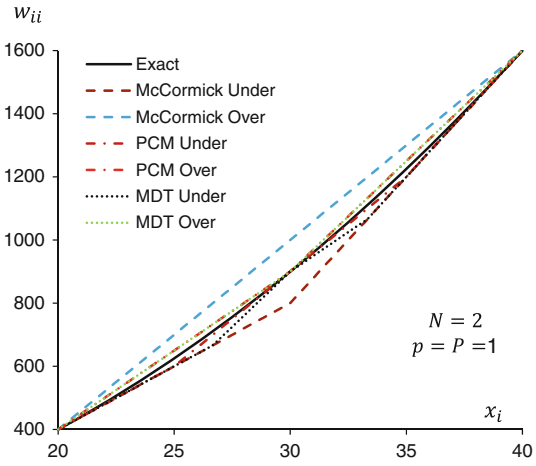

Fig. 5 Feasible region from conventional McCormick, piecewise McCormick (PCM) and multiparametric disaggregation relaxation (MDT) for a quadratic term over a specific region. Notice that the underestimation from PCM is strictly tighter than MDT in the region $x_i \in]25, 35[\setminus \{30\}$

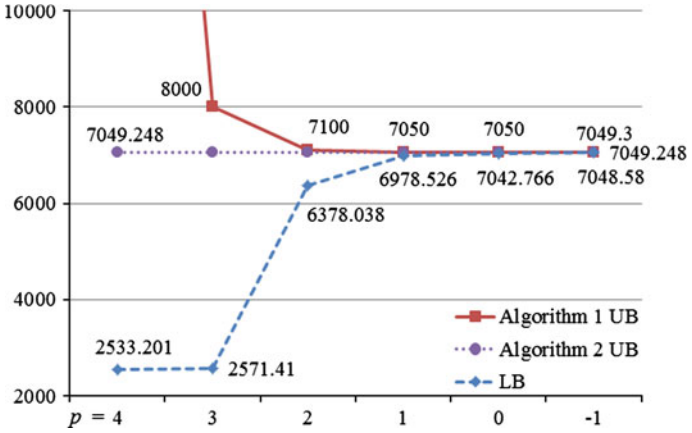

Fig. 6 Comparison of Algorithms 1 and 2 for problem (P3a). The upper bound from Algorithm 1 for $p = 4$ is equal to 30,000 (off scale)

in 520 CPUs, which improves to just 0.0020 % for $p = (-3, \ldots, -3)$, calculated using the best possible solution at time of termination).

7.2.1 Evaluation of Algorithm 1

Compared to Algorithm 2 in Tables 2, 3 and 4, Algorithm 1 generally leads to larger optimality gaps, particularly in the first iterations at coarse discretization levels, and it also requires more CPU time since two MILPs are solved at each iteration. Comparatively, solving (**P**) with a local solver can be done almost instantaneously. The lower bounding problem is the same as in Algorithm 2, while the upper bounding problem (**P**′) yields considerably worse bounds than the ones returned from the local NLP solver (recall that these were always global optimal solutions). As is illustrated in Fig. 6 for test problem (**P3a**), the solutions from both (**P**′) and

(**PR**) typically become closer to the optimum with an increase in accuracy. However, it is clear that using a local NLP solver, as in Algorithm 2, is a superior approach. While it is possible for the solution of (**P**′) to be closer to the global optimum than the solution of (**P**) using a local NLP algorithm, in practice, this generally does not occur, especially when using the solution of (**PR**) as a starting point.

7.3 Results for larger problems

As seen in Tables 1, 2, 3 and 4, when compared to the commercial solver BARON, multiparametric disaggregation is competitive in (**P1**), (**P2**) and (**P4**) but is orders of magnitude slower in (**P3**). This is to be expected given the very small size of problems and the reduced number of bilinear terms, which facilitates the spatial branch-and-bound procedure in BARON. In order to evaluate how both methods scale with problem size, we solved the five most difficult (higher optimality gap at termination when solved with BARON) water-using network design problems in Teles et al. [11], which also correspond to bilinear programs. In such problems, the discretized variables are concentrations featuring $P = 3$ in Ex17 and $P = 2$ in the other examples. The relative optimality tolerance was set to 0.01 % and the total computational time to 1-h.

Table 5 lists the results for global optimization solvers BARON and GloMIQO together with the implementation of algorithm 2 for MDT. While BARON fails to prove optimality in all five cases, GloMIQO and MDT are successful in two, which interestingly are not the same. GloMIQO takes roughly half an hour to tackle Ex14 and Ex15, while it takes just a few seconds for MDT to solve EX17 and Ex18. In the full set of problems from [11], MDT performs better than BARON 84 % of the time and slightly better than GloMIQO (53 %) when considering the CPUs performance metric. The average optimality gaps are however considerably lower, 0.088 % for MDT, 0.466 % for GloMIQO and 0.711 % for BARON.

In Table 6, several blending problems [45,46] are solved using BARON, GloMIQO, and Algorithm 3 at a single level of discretization. These are multiperiod blending problems with varying numbers of tanks, time periods, and product qualities. Each problem is identified such that 6T-3P-2Q-029 is a 6 tank, 3 time period, 2 quality problem, with a unique 3-digit identifier to distinguish it from other problems of the same size. The results for MDT are reported for $P = 0$ and $p = -3$ and were solved using Gurobi [47] using 12 threads. As these problems are originally MINLPs, larger than those in Table 5, and 12 threads are utilized, the computational difference is much more significant. GloMIQO and BARON both use MILP solvers in their algorithms which can utilize multiple threads, but this effect is not significant as the subproblems being solved are generally very small. Because of the use of multiple threads and multiple CPUs in this comparison, wall times are reported for all three methods. While GloMIQO and BARON are unable to converge to an optimality gap of 0.1 % within the time limit of 2 h, the MDT is able to close the gap in all but two cases. In these two cases an additional level of discretization would close the gap, but these results show that if a reasonably fine discretization is chosen a priori, the optimality gaps can be significantly less than those of commercial global optimization solvers. For the one problem that BARON and GloMIQO were able to close the gap within the time limit, MDT outperformed them by approximately 17 and 1,400 times, respectively, although the problem is small enough and is solved fast enough that meaningful conclusions should not be drawn from this result alone.

Table 6 Comparison for multiperiod blending problems [45,46]

Problem	Original MINLP			BARON		GloMIQO		MDT (p = -3)				
	Binary variables	Total variables	Equations	Wall time (s)	Gap (%)	Wall time (s)	Gap (%)	Binary variables	Total variables	Equations	Wall time (s)	Gap (%)
6T-3P-2Q-029	36	103	214	21.12	0.10	1,771	0.10	420	1,819	1,990	1.25	0.00
8T-3P-2Q-146	87	223	624	*7,200*	18.0	*7,200*	1.96	855	5,743	7,441	870.88	0.16
8T-3P-2Q-718	87	223	607	*7,200*	62.8	*7,200*	76.2	855	5,569	7,151	97.69	0.00
8T-3P-2Q-721	87	223	628	*7,200*	8.82	*7,200*	1.04	855	5,743	7,444	11.78	0.00
8T-4P-2Q-480	124	313	885	*7,200*	134	*7,200*	0.39	1,148	8,233	10,093	741.28	0.41
8T-4P-2Q-531	104	273	737	*7,200*	12.1	*	*	1,128	7,933	9,577	31.84	0.00
8T-4P-2Q-852	120	305	861	*7,200*	11.3	*7,200*	0.26	1,144	8,225	10,069	22.10	0.00

For MDT, $P = 1$ is used
*Solver failure
Italic values indicate that the maximum computational time was reached

8 Conclusions

This paper has presented a derivation based on disjunctive programming and exact linearization of the multiparametric disaggregation technique (MDT) for solving bilinear programming problems. The derivation, which was shown to be equivalent to applying the reformulation linearization technique (RLT), gives rise to an MILP approximation that yields an upper bound if feasible with respect to the original problem. A lower bounding MILP relaxation problem has also been derived, which can then be used as basis for rigorous global optimization algorithms. As has been shown with the smaller test problems, the relaxation from the MDT was shown to be as tight as the relaxation based on piecewise McCormick envelopes (PCM) for strictly bilinear terms but weaker for quadratic terms. The real advantage of MDT comes from the more favorable scaling of problem size as discretization is increased, which is translated into an ability to reach considerably lower optimality gaps and a clearly better computational performance. For large problems it was shown that multiparametric disaggregation can outperform global solvers BARON and GloMIQO.

Acknowledgments Ignacio Grossmann and Scott Kolodziej acknowledge financial support from the National Science Foundation under Grant OCI-0750826. Pedro Castro gratefully acknowledges financial support from the Luso-American Foundation, under the 2011 Portugal-U.S. Research Networks Program.

References

1. Misener, R., Thompson, J.P., Floudas, C.A.: APOGEE: global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. Comput. Chem. Eng. **35**(5), 876–892 (2011)
2. Misener, R., Floudas, C.A.: Advances for the pooling problem: modeling, global optimization, and computational studies. Appl. Comput. Math. **8**(1), 3–22 (2009)
3. Bagajewicz, M.: A review of recent design procedures for water networks in refineries and process plants. Comput. Chem. Eng. **24**(9–10), 2093–2113 (2000)
4. Jeżowski, J.: Review of water network design methods with literature annotations. Ind. Eng. Chem. Res. **49**(10), 4475–4516 (2010)
5. Haverly, C.A.: Studies of the behavior of recursion for the pooling problem. SIGMAP Bull. **25**, 19–28 (1978)
6. Quesada, I., Grossmann, I.E.: Global optimization of bilinear process networks with multicomponent flows. Comput. Chem. Eng. **19**(12), 1219–1242 (1995)
7. Tawarmalani, M., Sahinidis, N. V.: Convexification and global optimization in continuous and mixed-integer nonlinear programming. Kluwer, Dordrecht, pp. 254–284 (2002)
8. Meyer, C.A., Floudas, C.A.: Global optimization of a combinatorially complex generalized pooling problem. AIChE J. **52**(3), 1027–1037 (2006)
9. Misener, R., Floudas, C.A.: Global optimization of large-scale generalized pooling problems: quadratically constrained MINLP models. Ind. Eng. Chem. Res. **49**(11), 5424–5438 (2010)
10. Karuppiah, R., Grossmann, I.E.: Global optimization for the synthesis of integrated water systems in chemical processes. Comput. Chem. Eng. **30**, 650–673 (2006)
11. Teles, J.P., Castro, P.M., Matos, H.A.: Global optimization of water networks design using multiparametric disaggregation. Comput. Chem. Eng. **40**, 132–147 (2012)
12. Ahmetović, E., Grossmann, I.E.: Global superstructure optimization for the design of integrated process water networks. AIChE J. **57**(2), 434–457 (2010)
13. Sherali, H.D., Alameddine, A.: A new reformulation linearization technique for bilinear programming problems. J. Glob. Optim. **2**, 379–410 (1992)
14. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: a computational approach. In: Abraham, A., Hassanien, A., Siarry, P., Engelbrecht, A. (eds.) Foundations of Computational Intelligence, vol. 3, pp. 153–234. Springer, Heidelberg (2009)
15. Liberti, L., Pantelides, C.C.: An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. J. Glob. Optim. **36**, 161 (2006)

16. Ruiz, J.P., Grossmann, I.E.: Exploiting vector space properties to strengthen the relaxation of bilinear programs arising in the global optimization of process networks. Optim. Lett. **5**, 1 (2011)
17. Wicaksono, D.S., Karimi, I.A.: Piecewise MILP under- and overestimators for global optimization of bilinear programs. AIChE J. **54**(4), 991–1008 (2008)
18. Al-Khayyal, F.A., Falk, J.E.: Jointly constrained biconvex programming. Math. Oper. Res. **8**(2), 273–286 (1983)
19. Smith, E.M.B., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. Comput. Chem. Eng. **21**, S791–S796 (1997)
20. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin (1996)
21. Floudas, C.A., Visweswaran, V.: Quadratic optimization. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization. Kluwer, Dordrecht (1995)
22. Shor, N.: Dual quadratic estimates in polynomial and Boolean programming. Ann. Oper. Res. **25**(1), 163–168 (1990)
23. Xu, H.K.: An iterative approach to quadratic optimization. J. Optim. Theory Appl. **116**(3), 659–678 (2003)
24. Yu, N.: Semidefinite relaxation and nonconvex quadratic optimization. Optim. Methods Softw. **9**(1–3), 141–160 (1998)
25. Ye, Y.: Approximating quadratic programming with bound and quadratic constraints. Math. Program. **84**(2), 219–226 (1999)
26. Adhya, N., Tawarmalani, M., Sahinidis, N.V.: A Lagrangian approach to the pooling problem. Ind. Eng. Chem. Res. **38**(5), 1956–1972 (1999)
27. Bergamini, M.L., Aguirre, P., Grossmann, I.E.: Logic-based outer approximation for globally optimal synthesis of process networks. Comput. Chem. Eng. **29**, 1914–1933 (2005)
28. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. Oper. Res. **58**, 303–315 (2010)
29. Vielma, J.P., Nemhauser, G.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Math. Program. (in press 2010). doi:10.1007/s10107-009-0295-4
30. Teles, J.P., Castro, P.M., Matos, H.A.: Multiparametric disaggregation technique for global optimization of polynomial programming problems. J. Glob. Optim (2011). doi:10.1007/s10898-011-9809-8
31. Grossmann, I.E., Ruiz, J.P.: Generalized disjunctive programming: a framework for formulation and alternative algorithms for MINLP optimization. In: Lee, J., Leyffer, S. (eds.) IMA Volume 154, Mixed Integer Nonlinear Programming (2011)
32. Oral, M., Kettani, O.: A linearization procedure for quadratic and cubic mixed-integer problems. Oper. Res. **40**(Suppl 1): S109–S116 (1992) (Optimization)
33. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J. Algebraic Discret. Math. **6**, 466–486 (1985)
34. Grossmann, I.E.: Review of nonlinear mixed-integer and disjunctive programming techniques. Optim. Eng. **3**(3), 227–252 (2002)
35. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs. Part I. Convex underestimating problems. Math. Program. **10**, 146 (1976)
36. Gounaris, C.E., Misener, R., Floudas, C.A.: Computational comparison of piecewise-linear relaxations for pooling problems. Ind. Eng. Chem. Res. **48**(12), 5742–5766 (2009)
37. Quesada, I., Grossmann, I.E.: A global optimization algorithm for linear fractional and bilinear programs. J. Glob. Optim. **6**(1), 39–76 (1995)
38. Brook, A., Kendrick, D., Meeraus, A.: GAMS, a user's guide. ACM SIGNUM Newslett. **23**(3–4) (1988)
39. IBM. IBM ILOG CPLEX V12.1—User's Manual for CPLEX, IBM (2009)
40. Drud, A.S.: CONOPT–A Large-Scale GRG Code. INFORMS J. Comput. **6**(2), 207–216 (1994)
41. Sahinidis, N.: BARON: a general purpose global optimization software package. J. Glob. Optim. **8**, 201–205 (1996)
42. Rijckaert, M., Martens, X.: Comparison of generalized geometric programming algoritms. J. Optim. Theory Appl. **26**, 205 (1978)
43. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Vol. 187 of Lecture Notes in Economics and Mathematical Systems. Springer, Berlin (1981)
44. Shen, P., Zhang, K.: Global optimization of signomial geometric programming using linear relaxation. Appl. Math. Comput. **150**(1), 99–114 (2004)
45. Kolodziej, S.P.: Global Optimization of the Multiperiod Blend Problem. Master's Thesis, Carnegie Mellon University, Pittsburgh (2012)
46. Kolodziej, S.P., Grossmann, I.E., Furman, K.C., Sawaya, N.W.: A novel global optimization approach to the multiperiod blending problem (Submitted, 2012)

47. Gurobi Optimizer Reference Manual Version 4.5. Gurobi Optimization. http://www.gurobi.com/doc/45/refman/ (2011)
48. Sherali, H.D., Adams, W.P., Driscoll, P.J.: Exploiting special structures in constructing a hierarchy of relaxations for 0–1 mixed integer problems. Oper. Res. **46**(3), 396–405 (1998)
49. Misener R., Floudas, C.A.: Global mixed-integer quadratic optimizer. J. Glob. Optim (in press, 2012). DOI:10.1007/s10898-012-9874-7