J Glob Optim (2018) 71:691–716
https://doi.org/10.1007/s10898-018-0612-7

Global optimization of MIQCPs with dynamic piecewise relaxations

Pedro A. Castillo Castillo[1] · **Pedro M. Castro**[2] · **Vladimir Mahalec**[1]

Received: 2 June 2017 / Accepted: 23 January 2018 / Published online: 14 February 2018

Abstract We propose a new deterministic global optimization algorithm for solving mixed-integer bilinear programs. It relies on a two-stage decomposition strategy featuring mixed-integer linear programming relaxations to compute estimates of the global optimum, and constrained non-linear versions of the original non-convex mixed-integer nonlinear program to find feasible solutions. As an alternative to spatial branch-and-bound with bilinear envelopes, we use extensively piecewise relaxations for computing estimates and reducing variable domain through optimality-based bound tightening. The novelty is that the number of partitions, a critical tuning parameter affecting the quality of the relaxation and computational time, increases and decreases dynamically based on the computational requirements of the previous iteration. Specifically, the algorithm alternates between piecewise McCormick and normalized multiparametric disaggregation. When solving ten benchmark problems from the literature, we obtain the same or better optimality gaps than two commercial global optimization solvers.

Keywords Mixed-integer nonlinear programming · Global optimization of quadratic programs with bilinear terms · Piecewise linear relaxations · Optimality-based bound tightening

1 Introduction

We aim to solve a special class of nonconvex mixed-integer nonlinear programming (MINLP) problems to ε-global optimality, where ε is a non-zero tolerance. Problem **P** is a mixed-integer quadratically constrained problem (MIQCP) where nonlinearities are due to bilinear terms

✉ Pedro M. Castro
pmcastro@fc.ul.pt

[1] Department of Chemical Engineering, McMaster University, Hamilton, ON L8S 4A7, Canada

[2] Centro de Matemática Aplicações Fundamentais e Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisbon, Portugal

$x_i x_j$ of continuous variables x with finite lower x^L and upper x^U bounds, and binary variables y appear linearly in the constraints:

$$\begin{array}{lll} f^*_{\mathbf{P}} = \min f_0(x, y) & & \\ \text{s.t.} \quad f_q(x, y) \leq 0 & & \forall q \in \mathbf{Q}/\{0\} \\ f_q(x, y) = \sum_{(i,j)\in \mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y + d_q & & \forall q \in \mathbf{Q} \\ w_{ij} = x_i x_j & & \forall (i, j) \in \mathbf{BL} \\ 0 \leq x^L \leq x \leq x^U & & \\ x \in \mathbb{R}^{lx}, \quad y \in \{0, 1\}^{ly}, \quad w \in \mathbb{R}^{|\mathbf{BL}|} & & \end{array} \tag{P}$$

BL is an (i, j)-index set defining all bilinear terms, **Q** represents the set of all functions appearing in the constraints and objective function ($q = 0$), which excludes auxiliary equations defining new sets of bilinear variables w. The total number of original continuous variables is lx, while the number of original binary variables is ly. We assume that **P** is feasible with global optimal solution $f^*_{\mathbf{P}}$.

Many relevant engineering problems can be formulated as **P**. Others, closely resemble **P**, the difference being the presence of constraints with exponential terms for estimating the capital cost. Examples can be found in pooling problems [1–4], synthesis of general multicomponent process networks [5,6], design of water networks [7–11], short-term planning of oil refineries [12,13], scheduling of crude-oil blending operations [14–16] and hydro energy systems [17].

Nonconvex optimization problem **P** can present multiple local and global optima. Gradient-based methods cannot guarantee finding a global solution and they do not tell, at termination, how far the best feasible solution is from the best possible solution (i.e., the best estimate of the global optimum) [18]. Deterministic global optimization algorithms are required for such purposes, with their development being a very active research area.

Deterministic global optimization algorithms rely on a relaxation of **P** to compute estimates of the global solution, on various techniques to iteratively improve such estimates, and on methods to compute feasible solutions. They aim to reduce the relative difference between the best feasible and best possible solutions below ε. The quality of the best possible solution depends on the tightness of the relaxation, which in turn depends on the size of the domain of the variables (i.e., $x^U - x^L$). The smaller the domain, the closer the relaxation is to the original nonconvex function.

Spatial branch-and-bound (SBB) is the most common method to systematically reduce the domain of the variables. In SBB, branching is applied on discrete variables, as well as on continuous variables involved in nonlinear terms [19,20]. Branching occurs one variable at a time and it generates two child nodes, each with a smaller domain than the parent node, leading to potentially tighter relaxations. Whenever the best possible solution at a node becomes worse than the best feasible solution, the node is fathomed (pruned). Cutting planes and bound-tightening techniques have been incorporated in SBB algorithms to improve the relaxation and reduce the number of nodes to explore. Although SBB guarantees convergence to an ε-global solution, computational time can grow exponentially with problem size.

The tightest linear relaxation for a bilinear term is given by McCormick envelopes [21]. They are generated by four inequalities that have a low computational cost. Many deterministic global optimization algorithms employ McCormick envelopes as their only relaxation technique for bilinear terms [5,12,22,23]. However, McCormick envelopes usually provide a weak relaxation when at least one of the variables involved in a bilinear term has a significantly large domain. This led to the development of piecewise linear relaxation techniques

that improve the quality of the relaxation by partitioning the variables domain (the larger the number of partitions, the tighter the relaxation). However, since piecewise linear relaxations introduce additional binary and continuous variables, there exists a trade-off between tightness and the computational effort required to solve the MILP to optimality.

The piecewise McCormick relaxation [24–27] partitions the domain of one of the variables involved in a bilinear term and constructs McCormick envelopes for each partition. The major drawback of piecewise McCormick is that the number of additional binary variables increases linearly with the number of partitions. This important issue fostered the development of relaxation techniques where the number of binary variables increases logarithmically with respect to the number of partitions [2,28,29], with one example being normalized multiparametric disaggregation. Piecewise linear relaxations have been used in SBB algorithms [2,30,31], but they can also be deployed as an alternative to the SBB framework [13,28,32–34], as well as in decomposition methods [26].

The semidefinite programming (SDP) relaxation of MIQCPs has also been extensively studied. Problem (**P**) is a lifted reformulation, with extra variables w_{ij} and non-convex constraints $w_{ij} = x_i x_j$. It can be relaxed as a pair of inequalities, $W - xx^T \succcurlyeq 0$ (convex) and $xx^T - W \succcurlyeq 0$ (non-convex). In order to produce strong convex relaxations, Saxena et al. [35] use the convex SDP inequality to derive convex quadratic cuts and exploit the non-convex inequality to derive disjunctive cuts. Their cutting plane algorithm also relies on McCormick envelopes to strengthen the initial relaxation of the MIQCP, later removing all non-binding (at the solution of the convex relaxation) RLT inequalities when generating disjunctive cuts. In the companion paper [36], Saxena et al. study methods that capture the strength of such extended SDP relaxations but are defined only in the space of the x variables. By replacing the RLT convexification of (**P**) with an alternative that splits matrix A, defining bilinear terms $x^T Ax$, as a difference of positive semidefinite and symmetric matrices, they show how to project the extended RLT formulation in the original space by solving linear programs (LPs). A similar procedure is performed when adding the convex inequality $W - xx^T \succcurlyeq 0$ to the extended RLT formulation, leading to the solution of SDPs rather than LPs. For the GLOBALLib instances, relaxations from projected formulations are almost as strong as those from [35], with the advantage of being solved two orders of magnitude faster.

In this work, we present a deterministic global optimization algorithm to solve MINLP problems of type **P**. The main novelty is the use of a dynamic partitioning scheme for piecewise relaxations, not only to compute the lower bound, but also for performing optimality-based bound tightening (OBBT) for all variables appearing in bilinear terms [33]. The extensive use of OBBT contrasts with commercial global optimization solvers [37–39], which apply some restricted version of it, always featuring the simplest bilinear envelopes [40]. Dynamic partitioning refers to changing the number of partitions between iterations. Although the same term has been applied in [34], there are major differences between the two algorithms, as can be seen in Fig. 1.

Nagarajan et al. [34] assume that a local solution (x^*, y^*, w^*) to (**P**) is given and divide their global optimization algorithm in two parts. In part one, a sequence of OBBT iterations is performed to reduce the domain of all x variables. The procedure stops when bound improvement in consecutive iterations falls below a specified tolerance. Part two involves the solution of relaxation problems (**PR**) derived from piecewise McCormick envelopes. The domain of bilinearly appearing variables x_i and x_j is partitioned in a non-uniform way. Partitions are dynamically added around the current solution (local solution x^* in the first iteration and optimal solution from the relaxation problem x^R in subsequent iterations) until the normalized improvement on the lower bound LB from (**PR**) is less than a given tolerance,

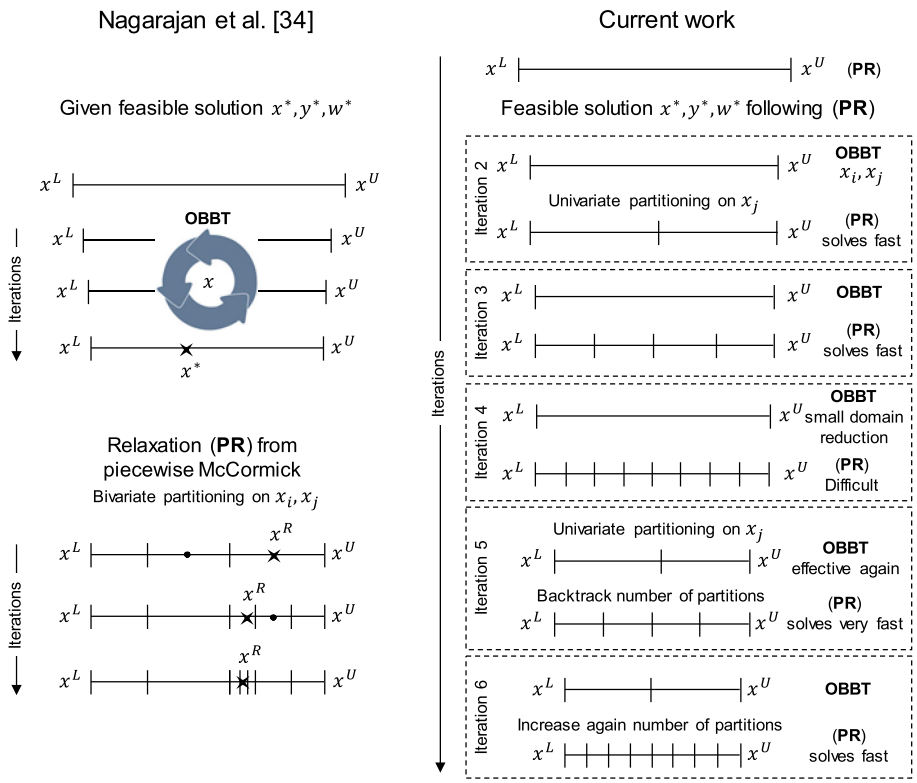

Fig. 1 Comparison of dynamic partitioning schemes

x^R variables remain in the same partitions and the size of such partitions is already very small, or the computation hits a time limit.

The algorithm proposed in this work does not assume a feasible solution is given. In the first iteration, it solves a simple relaxation problem (McCormick envelopes) to try to find one very quickly. This process is repeated in subsequent iterations to improve the upper bound UB, and consequently the bounds from OBBT (step omitted from Fig. 1 since the focus is on comparing the lower bounding procedure). The integration of OBBT and (**PR**) steps is the first major difference compared to [34]. The second difference is that our algorithm relies on univariate and uniform partitioning. Uniform partitioning, by giving the same importance to all regions of the domain, may protect us against frequent x^R movements from narrower to wider partitions, meaning potentially fewer iterations at the expense of more partitions (larger problems) per iteration. The next partitioning level is decided based on (**PR**)'s computational requirements. Figure 1 assumes (**PR**) solves fast before reaching $N = 8$ partitions in iteration 4, coinciding with OBBT becoming ineffective. To further improve domain reduction, it is thus worth to try piecewise relaxation strategies for OBBT, not considered in [34], by selecting $N = 2$. Iteration 5 also backtracks to $N = 4$ for (**PR**), illustrating that dynamic partitioning can go in both directions. Finally, and to benefit from the better scaling of problem size with the number of partitions when reaching $N = 10$, the algorithm will change the relaxation technique from piecewise McCormick to normalized multiparametric disaggregation.

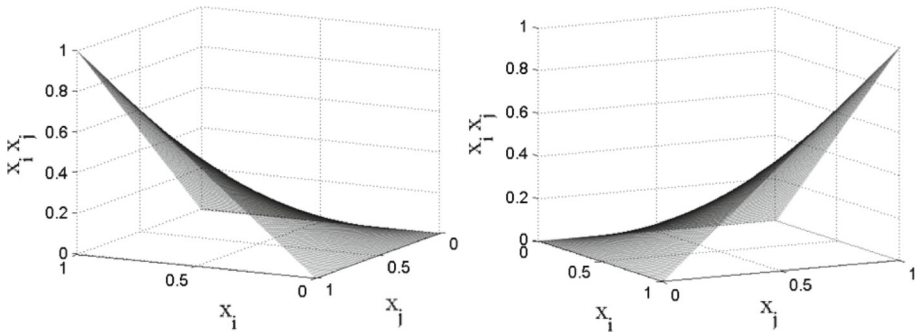

Fig. 2 Bilinear function $x_i x_j$ in $[0, 1]^2$

2 Computing lower bounds

If y variables remain binary and all original constraints $q \in \mathbf{Q}/\{0\}$ are kept, the simplest relaxation of $\mathbf{P}$ is obtained by removing equations $w_{ij} = x_i x_j$. However, it is also the weakest. Narrowing the domain of variables w_{ij} to regions $\mathbf{WR_{ij}}$ will potentially lead to a tighter relaxation. Since $\mathbf{P}$ is feasible, so is its relaxation $\mathbf{PR}$. If $f^R_{\mathbf{PR}}$ is the global optimal solution of $\mathbf{PR}$, then $f^R_{\mathbf{PR}} \leq f^*_{\mathbf{P}}$, representing a lower bound on the global optimal solution of $\mathbf{P}$.

$$
\begin{array}{lll}
f^R_{\mathbf{PR}} = \min f_0(x, y) & & \\
\text{s.t.} \quad f_q(x, y) \leq 0 & & \forall q \in \mathbf{Q}/\{0\} \\
f_q(x, y) = \sum_{(i,j)\in\mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y + d_q & & \forall q \in \mathbf{Q} \\
w_{ij} \in \mathbf{WR_{ij}} & & \forall (i, j) \in \mathbf{BL} \\
0 \leq x^L \leq x \leq x^U & & \\
x \in \mathbb{R}^{lx}, \quad y \in \{0, 1\}^{ly}, \quad w \in \mathbb{R}^{|\mathbf{BL}|} & &
\end{array}
\tag{PR}
$$

Three alternative ways of defining regions $\mathbf{WR_{ij}}$ for relaxation problem $\mathbf{PR}$ will be discussed next.

2.1 McCormick relaxation (SMCR)

The standard McCormick relaxation for bilinear function $w_{ij} = x_i x_j$ represented in Fig. 2, is given by Eqs. (1–4). These equations define regions $\mathbf{WR_{ij}}$ that form the convex hull for $x_i x_j$, see Fig. 3.

$$w_{ij} \geq x_i x_j^L + x_j x_i^L - x_i^L x_j^L \quad \forall (i, j) \in \mathbf{BL} \tag{1}$$

$$w_{ij} \geq x_i x_j^U + x_j x_i^U - x_i^U x_j^U \quad \forall (i, j) \in \mathbf{BL} \tag{2}$$

$$w_{ij} \leq x_i x_j^L + x_j x_i^U - x_i^U x_j^L \quad \forall (i, j) \in \mathbf{BL} \tag{3}$$

$$w_{ij} \leq x_i x_j^U + x_j x_i^L - x_i^L x_j^U \quad \forall (i, j) \in \mathbf{BL} \tag{4}$$

2.2 Piecewise linear relaxations

Piecewise McCormick and normalized multiparametric disaggregation are well-known examples of piecewise relaxation techniques that typically use the same number of parti-

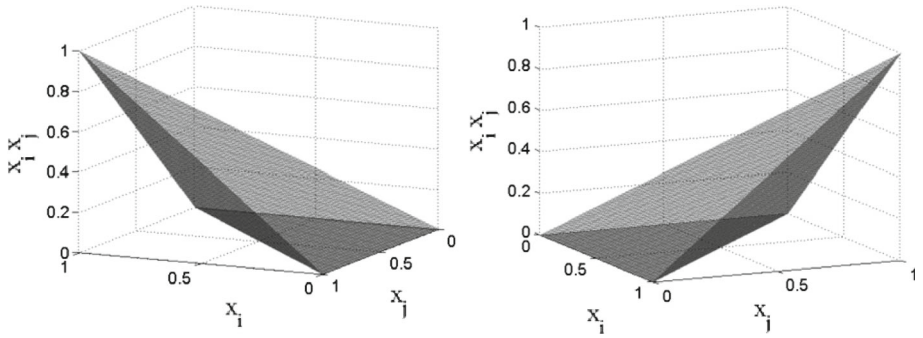

Fig. 3 Feasible region from McCormick envelopes for bilinear function $x_i x_j$ in $[0, 1]^2$

tions N for every partitioned variable x_j. Both introduce additional binary variables into the problem, creating non-convex regions $\mathbf{WR_{ij}}$.

2.2.1 Piecewise McCormick relaxation (PMCR)

Piecewise McCormick uses binary variable z_{jn} to identify the active (n) partition for variable x_j. The McCormick envelopes in Eqs. (1–4) can then benefit from tighter bounds $x_j^L \leq x_{jn}^L$ and $x_{jn}^U \leq x_j^U$, computed by Eqs. (5–6). The mixed-integer linear relaxation can be formulated as a disjunction and convex-hull reformulated [41] into Eqs. (7–15). Notice the new continuous disaggregated variables $\hat{x}_{jn}$ and $\hat{x}_{ijn}$. The feasible region associated to PMCR using 4 partitions is illustrated in Fig. 4. Notice that it is closer to the original bilinear function (Fig. 2) than SMCR (Fig. 3).

$$x_{jn}^L = x_j^L + \frac{\left(x_j^U - x_j^L\right)(n-1)}{N} \quad \forall j : (i,j) \in \mathbf{BL}, n \in \{1, \ldots, N\} \tag{5}$$

$$x_{jn}^U = x_j^L + \frac{\left(x_j^U - x_j^L\right)(n)}{N} \quad \forall j : (i,j) \in \mathbf{BL}, n \in \{1, \ldots, N\} \tag{6}$$

$$w_{ij} \geq \sum_{n=1}^{N} \left(\hat{x}_{ijn} x_{jn}^L + \hat{x}_{jn} x_i^L - z_{jn} x_i^L x_{jn}^L\right) \quad \forall (i,j) \in \mathbf{BL} \tag{7}$$

$$w_{ij} \geq \sum_{n=1}^{N} \left(\hat{x}_{ijn} x_{jn}^U + \hat{x}_{jn} x_i^U - z_{jn} x_i^U x_{jn}^U\right) \quad \forall (i,j) \in \mathbf{BL} \tag{8}$$

$$w_{ij} \leq \sum_{n=1}^{N} \left(\hat{x}_{ijn} x_{jn}^L + \hat{x}_{jn} x_i^U - z_{jn} x_i^U x_{jn}^L\right) \quad \forall (i,j) \in \mathbf{BL} \tag{9}$$

$$w_{ij} \leq \sum_{n=1}^{N} \left(\hat{x}_{ijn} x_{jn}^U + \hat{x}_{jn} x_i^L - z_{jn} x_i^L x_{jn}^U\right) \quad \forall (i,j) \in \mathbf{BL} \tag{10}$$

$$x_i = \sum_{n=1}^{N} \hat{x}_{ijn} \quad \forall (i,j) \in \mathbf{BL} \tag{11}$$

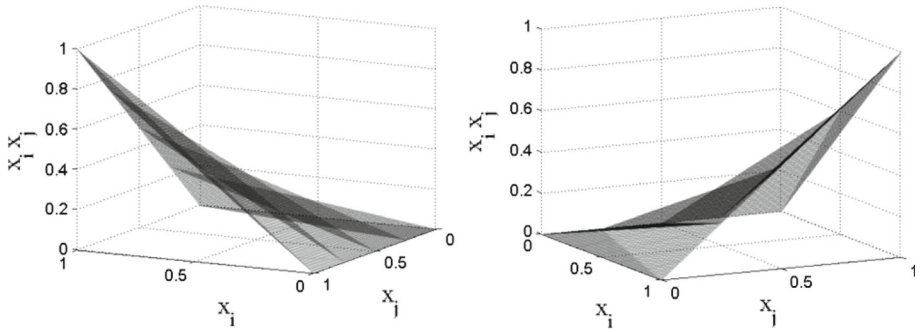

Fig. 4 Feasible region from piecewise McCormick relaxation with 4 partitions for bilinear term $x_i x_j$ in $[0, 1]^2$

$$x_j = \sum_{n=1}^{N} \hat{x}_{jn} \quad \forall j : (i, j) \in \mathbf{BL} \tag{12}$$

$$\sum_{n=1}^{N} z_{jn} = 1 \forall j : (i, j) \in \mathbf{BL} \tag{13}$$

$$x_i^L z_{jn} \leq \hat{x}_{ijn} \leq x_i^U z_{jn} \quad \forall (i, j) \in \mathbf{BL}, n \in \{1, \ldots, N\} \tag{14}$$

$$x_{jn}^L z_{jn} \leq \hat{x}_{jn} \leq x_{jn}^U z_{jn} \quad \forall j : (i, j) \in \mathbf{BL}, n \in \{1, \ldots, N\} \tag{15}$$

2.2.2 Normalized multiparametric disaggregation (NMDT)

Normalized multiparametric disaggregation provides an equivalent relaxation to PMCR but can be orders of magnitude more efficient computationally. However, the number of partitions is restricted to powers of ten, i.e. $N = 10^{-p}$, with $p \in \mathbb{Z}^-$ being the accuracy parameter chosen by the user. The normalized [0, 1] domain of variable x_j is discretized considering all digits $k \in \{0, \ldots, 9\}$ of the decimal representation system and positions $l \in \{p, \ldots, -1\}$. It is then linked to the real domain of x_j through continuous variable λ_j and global bounds x_j^L and x_j^U. Note that continuous variable $\Delta\lambda_j$ allows λ_j to take continuous values between discrete points. The active partition for x_j is identified by the non-zero values of $(-p)$ binary variables z_{jkl}, one per position l. The number of binary variables per variable is thus $10 \log_{10} N$ versus N when using PMCR. Equations (16–26) provide the NMDT relaxation that also requires continuous variables v_{ij}, Δv_{ij}, and $\hat{x}_{ijkl}$. It is illustrated in Fig. 5 for $p = -1$ ($N = 10$), which is already very similar to Fig. 2.

$$w_{ij} = x_i x_j^L + v_{ij} \left(x_j^U - x_j^L \right) \quad \forall (i, j) \in \mathbf{BL} \tag{16}$$

$$x_j = x_j^L + \lambda_j \left(x_j^U - x_j^L \right) \quad \forall j : (i, j) \in \mathbf{BL} \tag{17}$$

$$\lambda_j = \Delta\lambda_j + \sum_{l=p}^{-1} \sum_{k=0}^{9} 10^l \cdot k \cdot z_{jkl} \quad \forall j : (i, j) \in \mathbf{BL} \tag{18}$$

$$0 \leq \Delta\lambda_j \leq 10^p \quad \forall j : (i, j) \in \mathbf{BL} \tag{19}$$

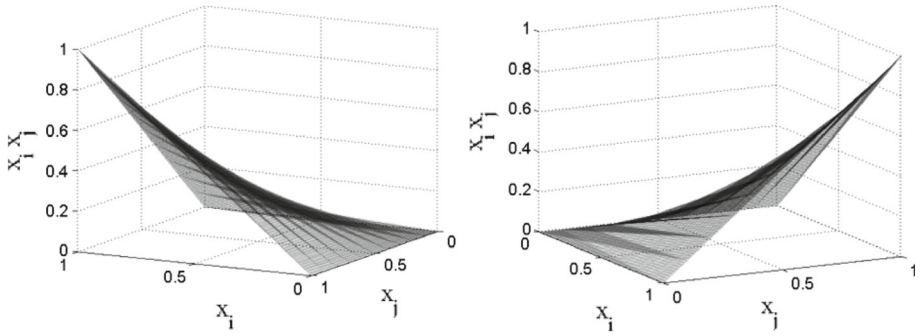

Fig. 5 Feasible region from normalized multiparametric disaggregation with $p = -1$ for bilinear term $x_i x_j$ in $[0, 1]^2$

$$v_{ij} = \sum_{l=p}^{-1} \sum_{k=0}^{9} 10^l \cdot k \cdot \hat{x}_{ijkl} + \Delta v_{ij} \quad \forall (i,j) \in \mathbf{BL} \tag{20}$$

$$x_i^L \Delta\lambda_j \leq \Delta v_{ij} \leq x_i^U \Delta\lambda_j \quad \forall (i,j) \in \mathbf{BL} \tag{21}$$

$$\Delta v_{ij} \leq 10^p \left(x_i - x_i^L\right) + x_i^L \Delta\lambda_j \quad \forall (i,j) \in \mathbf{BL} \tag{22}$$

$$\Delta v_{ij} \geq 10^p \left(x_i - x_i^U\right) + x_i^U \Delta\lambda_j \quad \forall (i,j) \in \mathbf{BL} \tag{23}$$

$$x_i = \sum_{k=0}^{9} \hat{x}_{ijkl} \quad \forall (i,j) \in \mathbf{BL}, l \in \{p, \ldots, -1\} \tag{24}$$

$$\sum_{k=0}^{9} z_{jkl} = 1 \quad \forall j : (i,j) \in \mathbf{BL}, l \in \{p, \ldots, -1\} \tag{25}$$

$$x_i^L z_{jkl} \leq \hat{x}_{ijkl} \leq x_i^U z_{jkl} \quad \forall (i,j) \in \mathbf{BL}, l \in \{p, \ldots, -1\}, k \in \{0, \ldots, 9\} \tag{26}$$

3 Optimality-based bound tightening (OBBT)

For all three relaxation techniques described in Sect. 2, the volume of region $\mathbf{WR_{ij}}$ depends on bounds x_i^L, x_i^U, x_j^L and x_j^U. It is thus desirable to strengthen such bounds (raise x_i^L and x_j^L, and decrease x_i^U and x_j^U) to obtain a tighter relaxation (higher $f_{\mathbf{PR}}^R$). One way to do it, is through optimality-based bound tightening (OBBT). For each variable $h \in \mathbf{BLV} = \{h | (i,j) \in \mathbf{BL} \wedge (h = i \vee h = j)\}$ involved in a bilinear term, lower and upper bounds are computed by solving one minimization and one maximization problem, respectively. These problems, denoted as **PRB**, are like relaxation problem **PR** but with a different objective function (now the variable to minimize/maximize) and an additional constraint, which imposes the value of the objective function in **P**, $f_0(x, y)$, to be less or equal than the current upper bound UB.

$$
\begin{aligned}
& x_h^L = \min x_h \left(x_h^U = \max x_h\right) \\
& \text{s.t.} \quad f_0(x, y) \leq UB \\
& f_q(x, y) \leq 0 && \forall q \in \mathbf{Q}/\{0\} \\
& f_q(x, y) = \sum_{(i,j)\in \mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y + d_q && \forall q \in \mathbf{Q} \\
& w_{ij} \in \mathbf{WR_{ij}} && \forall (i, j) \in \mathbf{BL} \\
& 0 \leq x^L \leq x \leq x^U \\
& x \in \mathbb{R}^{lx}, y \in \mathcal{Y}, w \in \mathbb{R}^{|\mathbf{BL}|}
\end{aligned}
\tag{PRB}
$$

Remark 1 Given that many problems may need to be solved, the complexity of problems **PRB** should be manageable. Region $\mathbf{WR_{ij}}$ will be generated from either the standard or piecewise McCormick envelopes with a low number of partitions ($N < 10$). With the former, binary variables are further relaxed, $\mathcal{Y} \in [0, 1]^{ly}$, to work with linear problems (LPs) instead of MILPs ($\mathcal{Y} \in \{0, 1\}^{ly}$).

Other types of probing methods can be found in the literature that also solve bounded relaxations of the problem to extract further information on the variables, e.g. to identify conflicts between binary variables y [42]. They are not part of this work.

4 Generating upper bounds

Previous work has shown that an effective way to compute a good feasible solution to non-convex MINLP problem **P**, is to rely on a two-stage MILP/NLP strategy. Any feasible solution to MILP problem **PR** can be used to extract the values x^R, y^R and w^R of variables x, y and w. Parameters y^R will then replace binary variables y in **P**, reducing it to NLP problem **PF**. **PF** will be solved by a local NLP solver, after initializing variables x and w with parameters x^R and w^R, to facilitate convergence. Note that **PF** is a restricted version of **P**, and so it is not necessarily feasible. If feasible, the optimal solution (x^*, y^*, w^*) of **PF** is an upper bound UB on the global solution of **P**, i.e. $f^*_{\mathbf{PF}} \geq f^*_{\mathbf{P}}$.

$$
\begin{aligned}
& f^*_{\mathbf{PF}} = \min f_0(x) \\
& \text{s.t.} \quad f_q(x) \leq 0 && \forall q \in \mathbf{Q}/\{0\} \\
& f_q(x) = \sum_{(i,j)\in \mathbf{BL}} a_{ijq} w_{ij} + B_q x + C_q y^R + d_q && \forall q \in \mathbf{Q} \\
& w_{ij} = x_i x_j && \forall (i, j) \in \mathbf{BL} \\
& 0 \leq x^L \leq x \leq x^U \\
& x \in \mathbb{R}^{lx}, \quad w \in \mathbb{R}^{|\mathbf{BL}|}
\end{aligned}
\tag{PF}
$$

5 Global optimization algorithm

We now propose a global optimization algorithm for the solution of any mixed-integer non-linear program that can be written as problem **P**. It is summarized in Fig. 6 and detailed in Tables 1 and 2.

Assumed given are the selection of partitioned variables x_j in every bilinear term, variable bounds x^L and x^U, and a variety of tuning parameters. Problem-specific settings include the pre-specified values that the number of partitions can take when solving **PR** (N_{PR}) and **PRB** (N_{PRB}), maximum computational time and relative optimality tolerance (e.g. $TIME^{max}_{PR}$, ε_{PR}). The other parameters will be named while describing the algorithm.

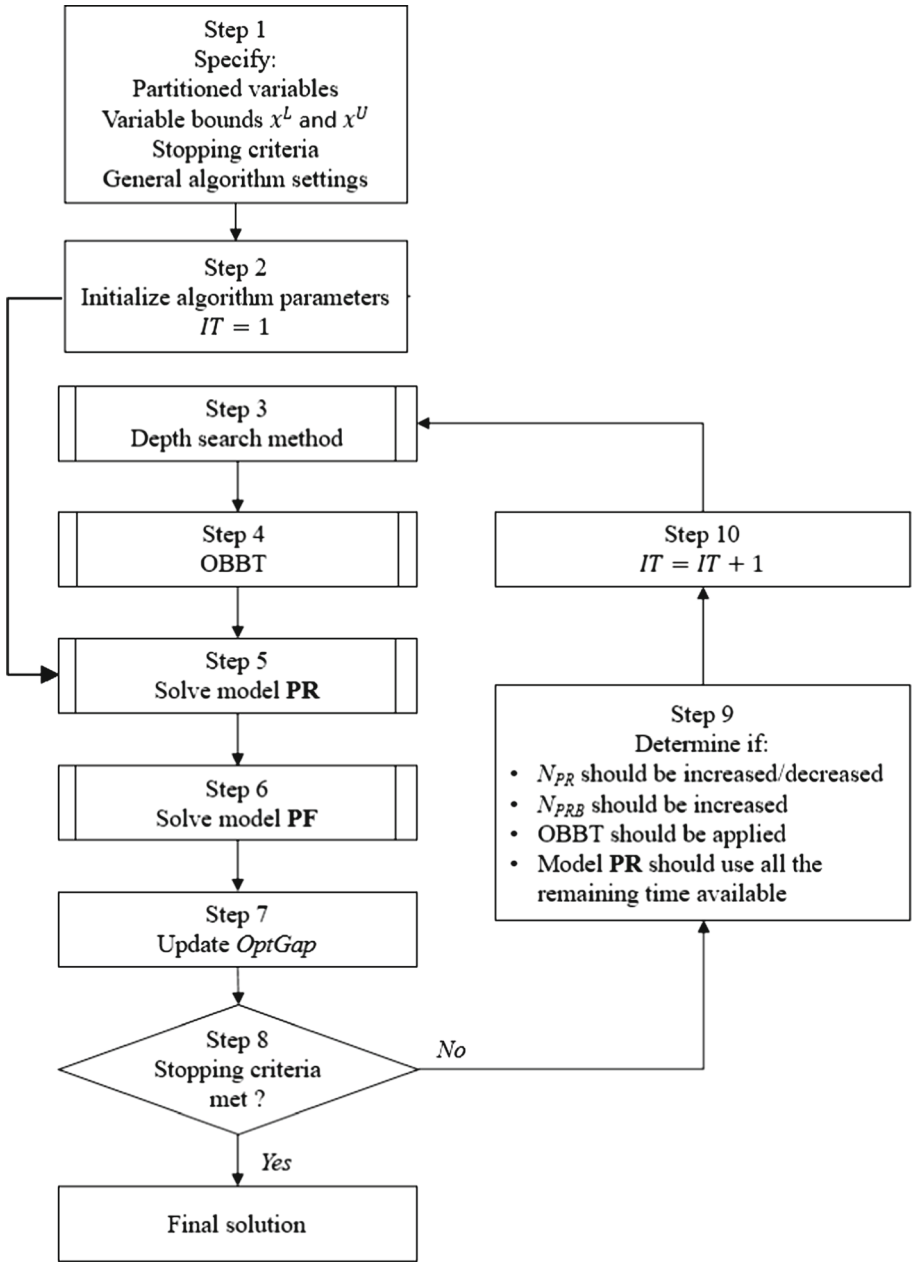

Fig. 6 Flowchart of the proposed global optimization algorithm

Following the initialization step, the algorithm computes the lower bound LB using the simplest McCormick relaxation. In subsequent iterations, step 5 will typically involve piecewise linear relaxations. Note that once OBBT loses efficiency (flag $LAST_{PR} = 1$), the maximum computational time $TIME_{PR}^{max}$ will be reset to the remaining time to run the algorithm. Step 5 solves one MILP problem of type **PR**, gathering a maximum of $npool$ solutions in a pool. If the optimal solution $f_{\mathbf{PR}}^{R}$ is higher than the lower bound, the latter is updated.

Remark 2 Region $\mathbf{WR}_{ij}$ in problem **PR** is computed using piecewise McCormick envelopes whenever $N_{PR} \in \{1, 2, \ldots, 9\}$. Normalized multiparametric disaggregation is used instead for $N_{PR} \in \{10, 100, 1000, \ldots\}$ ($p \in \{\ldots, -3, -2, -1\}$).

Remark 3 The lower bound is updated using the best possible solution at termination for problem **PR** and not the best-found feasible solution. The same is true for problem **PRB**, when it is an MILP.

In step 6, we use the values $\left(x^R, y^R, w^R\right)$ of the variables in the previous solutions to help computing upper bounds. A total of $npool$ problems of type **PF** are solved in parallel using $npar$ threads. Amongst those that are feasible, the one with the lowest objective function $f_{\mathbf{PF}}^{*}$ can set the upper bound UB. Note that **PF** is solved by a local NLP solver and so this step is much faster than steps 4–5. It is the reason why no execution-time constraints are enforced.

With the lower and upper bound, step 7 computes the relative optimality gap $OptGap$. Step 8 stops the algorithm if the termination criteria is met, either a relative tolerance below ε or a computational time ($TIME$) above maximum value $TIME^{max}$. Decisions related to the dynamic partitioning scheme are taken in step 9.

The details of step 9 can be found in Table 2. Two flags are used: $NN_{PR}^{nec} = 1$ indicates that we have the necessary conditions for increasing the number of partitions in problem **PR**; $NN_{PR} = 1$ gives the sufficient condition for selecting the next setting in $\left\{N_{PR,first}, \ldots, N_{PR,last}\right\}$, see 9c. These are the initial values for the first entry in 9a, which checks the time spent solving **PR** ($TIME_{PR}$).

If greater or equal to $TIME_{PR}^{max}$, it means that we should try to backtrack and reduce the number of partitions in the next iteration to reduce the complexity of **PR**, unless we are already in the coarsest setting $N_{PR,first}$ or have previously backtracked to N_{PR}; either way, we will definitely not increase N_{PR}, i.e. $NN_{PR} = 0$. The same is true if $TIME_{PR}$ is within $TIME_{PR}^{max}$ and the maximum time ratio tr_{PR}^{max}, and $LAST_{PR} = 0$. We also set $NN_{PR}^{nec} = 0$ to later decide how to improve the lower bound.

It the number of partitions did not increase in the previous iteration ($NN_{PR} = 0$) and **PR** was solved rather fast (below minimum time ratio tr_{PR}^{min}), we will try to generate a better lower bound by rising N_{PR} in the next iteration. This concludes step 9a.

Step 9b takes measures when the average domain reduction in OBBT is below the minimum target of ADR^{min}. This is not an issue if **PR** problems can be solved rather fast ($NN_{PR} = 1$), we simply avoid spending time in the next iteration with an inefficient OBBT by making $DO_{OBBT} = 0$. On the other hand, if we do not meet the necessary condition to increase N_{PR} ($NN_{PR}^{nec} = 0$), we may need to move towards termination of the algorithm.

If the next possible value of N_{PRB} is lower than N_{PR}, then we might still be able to get a good domain reduction by increasing N_{PRB}. Counters of LP (C_{PRB}^{LP}) and MILP (C_{PRB}^{MILP}) problems solved are then reset. Else, we increase the appropriate counter by one. We then proceed to the last if-then-else. If we have already solved at least one MILP in OBBT and found that $ADR < ADR^{min}$, then the most reasonable thing to do is to give all remaining time to **PR** by making $LAST_{PR} = 1$. If the domain reduction was low but we have been solving

LPs in OBBT, then we also move towards the end while allowing one more OBBT run, now solving MILPs, after selecting the next value of N_{PRB}.

We then proceed to the next iteration in step 10. Steps 3 and 4 are the first procedures of iteration IT but do not occur in the first iteration to quickly compute an optimality gap.

Step 3 involves a depth search and is detailed in Sect. 5.1.

Step 4 executes optimality-based bound tightening to reduce the variables domain. It is triggered by $DO_{OBBT} = 1$ and involves solving two **PRB** problems per variable, after setting the number of partitions N to N_{PRB}. Since the number of variables involved in bilinear terms can be significantly large, it is much more efficient to solve the multiple instances of problem **PRB** in parallel rather than sequentially (see results in Sect. 7.6). This procedure is repeated until OBBT has been applied on all x_h variables involved in bilinear terms. We then compute the average domain reduction ADR (%) using Eq. (27).

$$ADR = \frac{1}{|\mathbf{BLV}|} \sum_{h \in \mathbf{BLV}} \left[\frac{\left(x_h^{U,previous} - x_h^{L,previous}\right) - \left(x_h^{U,updated} - x_h^{L,updated}\right)}{\left(x_h^{U,previous} - x_h^{L,previous}\right)} \times 100 \right] \tag{27}$$

Remark 4 $N_{PRB} = 0$ triggers the computation of relaxed region $\mathbf{WR_{ij}}$ of bilinear function $w_{ij} = x_i x_j$ from the McCormick envelopes with binary variables relaxed (recall Remark 1).

5.1 Depth search method

MIQCPs have two sources of complexity: (1) a combinatorial source from binary variables; (2) a non-convexity source from bilinear terms. A stronger combinatorial component is associated to a higher difficulty finding the global optimal solution and can be addressed by generating a larger number of feasible solutions for **P**. This should be done preferably in the earlier stages of the global optimization algorithm, since a better upper bound (UB) helps to improve the bounds computed by problem **PRB**. It is activated in the second iteration ($IT = 2$) or when $ADR < ADR^{min}$, if general setting $DO_{DS} = 1$.

The depth search method works by dynamically increasing the number of partitions in **PR** from the current N_{PR} value. Note that it is not needed to solve **PR** to optimality since the focus here is not on the lower bound. Because the MILP solver normally finds multiple feasible solutions in the early nodes of the search tree, we stop at time $TIME_{PR}^{max}$. Solutions obtained after solving **PR** with more partitions are potentially better (higher $f_{\mathbf{PR}}^{R}$), leading to values of the model variables that are closer to the feasible region of **P**. As explained in Sect. 4, these values are then used to initialize **PF**, potentially leading to a better UB. The depth search method stops after IT_{DS}^{max} increments in the number of partitions, resetting N_{PR} to its original value.

Overall, depth search is very similar to the search performed by the main algorithm. However, it does not use OBBT and it always increases the number of partitions from one iteration to the next.

6 Benchmark problems

Two different sets of MIQCP benchmark problems from the literature are used to evaluate the performance of the proposed global optimization algorithm.

Table 1 Global optimization algorithm

1. Given

Selection of non-partitioned x_i and partitioned variables x_j for every bilinear term in **P**

Variable bounds: x^L and x^U

Settings for solving **PR**: $N_{PR} \in \{N_{PR,first}, \ldots, N_{PR,last}\}$, $TIME_{PR}^{max}$, ε_{PR}, $npool$

Settings for solving **PRB**: $N_{PRB} \in \{N_{PRB,first}, \ldots, N_{PRB,last}\}$, $TIME_{PRB}^{max}$, ε_{PRB}, $npar$

General settings: $TIME^{max}$, ε, ADR^{min}, DO_{DS}, tr_{PR}^{min}, tr_{PR}^{max}, IT_{DS}^{max}

2. Initialization

$LB = -\infty, UB = +\infty, IT = 1, IT_{DS} = 1, TIME = 0,\ ADR = ADR^{min} + 0.1,\ DO_{OBBT} = 1,$
$LAST_{PR} = 0, N_{PR} = N_{PR,first}, N_{PRB} = N_{PRB,first},\ NN_{PR}^{nec} = 1, NN_{PR} = 1, C_{PRB}^{LP} = 0, C_{PRB}^{MILP} = 0.$

3. Depth search method (see Section 5.1)

If $DO_{DS} = 1$ and $IT > 1$,

If $IT = 2$ or $ADR < ADR^{min}$,

Generate up to IT_{DS}^{max} feasible solutions; if best is better than UB, update UB; $update(TIME)$

4. Optimality-based bound tightening (OBBT)

If $IT > 1$ and $DO_{OBBT} = 1$,

$N = N_{PRB}$

For every bilinear appearing variable x_h and using $npar$ threads

Solve **PRB** minimizing x_h up to $TIME_{PRB}^{max}$ to obtain $\underline{x}_h$; $x_h^L = \max(x_h^L, \underline{x}_h)$

Solve **PRB** maximizing x_h, up to $TIME_{PRB}^{max}$ to obtain $\overline{x}_h$; $x_h^U = \min(x_h^U, \overline{x}_h)$

Compute average domain reduction ADR; $update(TIME)$

5. Lower bound computation

If $LAST_{PR} = 1$,

$TIME_{PR}^{max} = TIME^{max} - TIME$

$N = N_{PR}$

If $N_{PR} \geq 10$,

Solve problem **PR** with $\mathbf{WR_{ij}}$ from NMDT up to $TIME_{PR}^{max}$, storing $npool$ feasible solutions

Else,

Solve problem **PR** with $\mathbf{WR_{ij}}$ from PMCR up to $TIME_{PR}^{max}$, storing $npool$ feasible solutions

$LB = \max(LB, f_{\mathbf{PR}}^R)$; $update(TIME_{PR})$; $update(TIME)$.

6. Upper bound computation

For every (x^R, y^R, w^R) solution in $npool$ and using $npar$ threads,

Initialize (x, w) with (x^R, w^R).

Solve problem **PF**.

If feasible and $f_{\mathbf{PF}}^* < UB$, $UB = f_{\mathbf{PF}}^*$, update optimal solution (x^*, y^*, w^*)

$update(TIME)$

7. Optimality gap computation

$OptGap = [\frac{UB-LB}{UB}] \times 100.$

8. Check termination criteria

Stop if $OptGap \leq \varepsilon$ or $TIME \geq TIME^{max}$

9. Modify number of partitions

See details in Table 2

10. Continue to next iteration

$IT = IT + 1$; go to Step 3

Table 2 Global optimization algorithm—dynamic partitioning scheme

9. Modify number of partitions
$NN_{PR} = NN_{PR}^{nec}$
a. Check time spent solving problem PR
If $NN_{PR} = 1$,
If $TIME_{PR} \geq TIME_{PR}^{max}$,
If $N_{PR} \neq N_{PR,first}$ and $N_{PR} \notin N_{PR}^{back}$,
$N_{PR}^{back} = N_{PR}^{back} \cup N_{PR}$
$N_{PR} = previous(N_{PR})$
$NN_{PR}^{nec} = 0$; $NN_{PR} = 0$.
ElseIf $TIME_{PR} \geq tr_{PR}^{max} \cdot TIME_{PR}^{max}$,
If $LAST_{PR} = 0$,
$NN_{PR}^{nec} = 0$; $NN_{PR} = 0$.
ElseIf $NN_{PR} = 0$,
If $TIME_{PR} \leq tr_{PR}^{min} \cdot TIME_{PR}^{max}$,
$NN_{PR}^{nec} = 1$; $NN_{PR} = 1$.
b. Check average domain reduction
If $NN_{PR} = 1$ and $ADR < ADR^{min}$,
$DO_{OBBT} = 0$
If $NN_{PR}^{nec} = 0$,
If $next(N_{PRB}) < N_{PR}$,
$N_{PRB} = next(N_{PRB})$
$C_{PRB}^{LP} = 0$; $C_{PRB}^{MILP} = 0$.
Else,
If $N_{PRB} = 0$,
$C_{PRB}^{LP} = C_{PRB}^{LP} + 1$
Else,
$C_{PRB}^{MILP} = C_{PRB}^{MILP} + 1$
If $N_{PRB} \geq 1$,
If $C_{PRB}^{MILP} \geq 1$,
If $ADR < ADR^{min}$,
$NN_{PR}^{nec} = 1$, $NN_{PR} = 0$, $LAST_{PR} = 1$, and $DO_{OBBT} = 0$
Else,
If $C_{PRB}^{LP} \geq 2$,
If $ADR < ADR^{min}$,
$NN_{PR}^{nec} = 1$, $NN_{PR} = 0$, $LAST_{PR} = 1$, and $DO_{OBBT} = 1$
$N_{PRB} = next(N_{PRB})$
c. Partitions of problem PR
If $NN_{PR} = 1$,
$N_{PR} = next(N_{PR})$

The first set deals with the short-term scheduling of a hydroelectric system [17], where the aim is to maximize the daily profit considering hourly changing electricity prices and start-up costs for the power plants. Power generation is modelled as a bilinear function of discharge flowrate and head change, with binary variables identifying if a plant is producing energy on a given hour (needed to enforce lower and upper bounds on power production and discharge flowrate) and startups. Like in our previous global optimization studies [31,33], we consider the original problem with 7 reservoirs (HYD7) and simpler versions with 2 (HYD2) and 4 reservoirs (HYD4).

Table 3 MIQCP model statistics

Benchmark problem	HYD2	HYD4	HYD7	SC1TP1–SC3TP1	SC1TP3–SC4TP3
Equations	573	1145	2003	1504	4526
Binary variables	96	192	336	12	36
Total variables	433	865	1513	1234	3716
Variables in bilinear terms	118	260	473	342	1132
Bilinear terms	192	384	672	476	1608

The second set consists of planning problems from a petroleum refinery [13]. The objective is to minimize the total operating cost of the system that includes processing units with alternative operating modes and storage tanks. Binary variables identify active modes and products being blended. Bilinear terms appear as the product of volumetric flows and quality properties in the material balances. We solve seven problems with different crude-oil supply and product demand data. Three involve a single period of operation (SC1TP1-SC3TP1), while in the others, the weekly time horizon is divided in three periods of fixed length (SC1TP3-SC4TP3).

The model statistics in Table 3 show that the ratio between the number of binary variables and bilinear terms varies significantly between the two sets of problems (1:2 vs. 1:50). For the hydro problems, a stronger combinatorial component is associated to a higher difficulty finding the global optimal solution and can be addressed by generating a larger number of feasible solutions of **P**. We thus activate the depth search method ($DO_{DS} = 1$), with a maximum of five increments in the number of partitions ($IT_{DS}^{max} = 5$). The refinery problems do not benefit from the time-consuming depth search method and so $DO_{DS} = 0$.

6.1 Tuning parameters

The optimization algorithm presented in Sect. 5 has a few parameters affecting its performance. Most of the values selected were independent of problem type, while one was tuned to adjust to instance size. It is beyond the scope of this paper to present a thorough computational study involving such parameters.

MILP problems **PR** were solved for a number of partitions $N_{PR} \in \{1, 2, 4, 8, 10, 100, 1000\}$. The termination criteria were either a relative optimality tolerance $\varepsilon_{PR} = 0.0001\%$ or a maximum time $TIME_{PR}^{max}$ equal to: 400 s while OBBT is effective; or the remaining time available, otherwise. The number of partitions N_{PR} will increase in the next iteration if the time solving PR divided by $TIME_{PR}^{max}$ is less or equal than $tr_{PR}^{min} = 0.05$. On the other hand, if the time ratio is greater of equal than $tr_{PR}^{max} = 0.75$, N_{PR} will not change. The solution pool option of the MILP solver was active, with a pool capacity of $npool = 60$, thus leading to a maximum of 60 instances of **PF** solved in parallel per iteration.

The OBBT step involves solving LPs, $N_{PRB} = 0$, and MILP problems, $N_{PRB} \in \{2, 3, 4, 5, 6, 7\}$ (recall Remark 4). In the latter case, the relative tolerance for problems **PRB** is $\varepsilon_{PRB} = 0.0001\%$, while the maximum time $TIME_{PRB}^{max}$ is instance dependent: 130, 135, 145, 45 and 70 s for problems HYD2, HYD4, HYD7, SC#TP1 and SC#TP3, respectively. A maximum of $npar = 80$ instances were solved in parallel and the minimum average domain reduction to consider OBBT effective was $ADR^{min} = 5\%$.

For the hydro problems, the algorithm terminates when the optimality gap $\varepsilon \leq 0.0001\%$ or upon reaching a wall time $TIME^{max} = 18{,}000$ s. For the refinery problems, the values

are 0.01%, and 3600/10,800 s when dealing with one/three periods. The partitioned variables in the hydro problems are the discharge flowrates. In the refinery problems, the partitioned variables are the stream flowrates, the inventory levels in the storage tanks, and the quality variables associated with the specific gravity.

7 Numerical results

All mathematical models and the global optimization algorithm were implemented in GAMS 24.7.3, taking advantage of its parallel computing grid facility. The MILP problems were solved by CPLEX 12.6.3, running in parallel deterministic mode and using up to 8 threads. CONOPT 3.17A solved the NLP problems. The MINLP benchmark problems were also solved by commercial global optimization solvers BARON 16.5 [37] and ANTIGONE 1.1 [39] using the same termination criteria. The former is centered around spatial branch-and-bound, while the latter focuses more on solving piecewise linear relaxations, applying bound tightening techniques, and generating different types of cutting planes. The hardware consisted of a server with an AMD Opteron^{TM} Processor 6386 SE (2.79 GHz), 32 available cores, 64 GB RAM, and running Windows Server 2008 R2 Enterprise.

7.1 Comparison to our previous algorithms

The global optimization algorithms in our previous work have used piecewise relaxations in a different manner, see details in Table 4. They are responsible for the literature results in Table 5.

Results in [31] for the hydro problems came from a spatial branch-and-bound algorithm using the NMDT relaxation with $N_{PR} = 10$ partitions. OBBT was called in every node of the tree, prior to solving the relaxation problem (as in the current work), and involved solving a sequence of LPs ($N_{PRB} = 0$).

The algorithm solving the refinery problems in [13] used dynamic partitioning in the relaxation step as a replacement to spatial B&B, similarly to the one proposed in this work. The difference is that the number of partitions only increased, until reaching the computational time limit. Now, we enforce timing constraints per iteration to use the available time more efficiently, backtracking on the number of partitions whenever the relaxation problem cannot be solved to optimality. The two algorithms also share the parallel solution strategy for the bound contracting problems. However, the current algorithm calls OBBT more often, once per iteration and while domain reduction remains effective, instead of following the finding of a better solution. More importantly, our new algorithm adjusts to problem complexity by dynamically switching between McCormick and piecewise McCormick relaxations. In the former case, binary variables y are relaxed, leading to LPs instead of the MILPs ($N_{PRB} = 1$) in [13].

7.2 Performance overview

Table 5 shows the optimality gap and computational time required by the different algorithms, and results from the literature. The highlight is that the new algorithm always returns the lowest optimality gap. It can solve four problems to the given tolerance, compared to three problems by ANTIGONE and one by BARON. Our previous attempts with algorithms featuring piecewise relaxation methods and optimality-based bound tightening solved none of these benchmark problems to optimality. An ability to find the global optimal solution is

Table 4 Features of current and previous algorithms

Reference/feature	Spatial B&B	Partitioning relaxation step	OBBT calls	OBBT strategy	OBBT type of problems	Partitioning OBBT step
Hydro problems [31]	Yes	Static ($N = 10$)	In every node of the tree	Sequential	LPs	N.A.
Refinery problems [13]	No	Dynamic (up)	Upon finding better solution	Parallel	MILPs (McCormick)	N.A.
Current work	No	Dynamic (up/down)	While domain reduction is good	Parallel	LPs or MILPs (piecewise McCormick)	Dynamic (up, $N \leq 7$)

Table 5 Comparison between proposed algorithm, commercial global optimization solvers and literature results

Problem	Optimum	Optimality gap (%)				Wall time (s)			
		Literature[a]	BARON	ANTIGONE	Algorithm	Literature[a]	BARON	ANTIGONE	Algorithm
HYD2	209,721.0	0.023	GO	0.536	GO	WTL	17,417	WTL	5430
HYD4	371,811.8	0.585[b]	1.359[b]	1.323[b]	0.465	WTL	WTL	WTL	WTL
HYD7	744,963.7	1.369[b]	2.285[b]	1.816[b]	1.260[b]	WTL	WTL	WTL	WTL
SC1TP1	55,568.1[c]	0.12	0.42	GO	GO	WTL	WTL	614	2973
SC2TP1	49,799.3[c]	0.06	0.21	GO	GO	WTL	WTL	169	2062
SC3TP1	53,798.8[c]	0.18	0.43	GO	GO	WTL	WTL	376	2900
SC1TP3	55,561.8[c]	0.79	1.96	0.93	0.60[b]	WTL	WTL	WTL	WTL
SC2TP3	49,878.5[c]	0.27	1.36[b]	0.30	0.26[b]	WTL	WTL	WTL	WTL
SC3TP3	53,785.8[c]	0.97	2.35	1.11[b]	0.80[b]	WTL	WTL	WTL	WTL
SC4TP3	55,323.3[c]	0.66	2.39	0.79	0.54	WTL	WTL	WTL	WTL

WTL wall time limit ($TIME^{max}$), *GO* global optimal solution (optimality gap $\leq \varepsilon$)

[a]Results for algorithms described in Sect. 7.1

[b]Best-found solution is suboptimal

[c]Slightly different values compared to [13] (costs within 0.04% except for SC2TP1: 0.17%) due to scaling up of sulfur content values to avoid numerical issues after four iterations with OBBT

Table 6 Detailed information about the performance of the new algorithm

Problem	HYD2	HYD4	HYD7	SC1TP1	SC2TP1	SC3TP1	SC1TP3	SC2TP3	SC3TP3	SC4TP3
Iterations	14	8	6	5	4	5	7	7	6	7
Time solving **PR** problems (s)	1140	12,907	12,386	571	424	553	940	864	900	938
Time solving **PF** problems (s)	155	86	69	63	46	58	143	108	107	130
Time in OBBT stage (s)	3902	3706	4418	2289	1554	2243	9665	9776	9750	9687
Time in depth search (s)	213	1301	1126	–	–	–	–	–	–	–
Instances of **PF** solved	203	111	134	160	107	116	256	166	218	273
Instances of **PRB** solved	2698	3610	3748	2630	1988	2630	13,104	13,094	10,936	13,104
ADR 1st iteration OBBT (%)	33.6	33.5	27.0	84.0	86.0	83.5	66.2	73.0	67.3	70.3
ADR final versus initial bounds (%)	99.5	67.0	32.9	97.8	95.2	97.4	80.0	85.9	78.1	81.8
Final N_{PR}	1000	4	4	4	8	4	4	4	4	4
Final N_{PRB}	7	3	2	2	0	2	2	2	2	2

ADR average domain reduction

also an important performance metric. The commercial global optimization solvers are doing better in this respect, returning suboptimal solutions in three problems compared to our new algorithms' four. It is an indication that there is still room for improving the upper bounding procedure.

BARON solves HYD2 three times slower, returning considerable larger gaps for the other problems. The poorer performance in the refinery problems might be due to the large number of variables involved in bilinear terms (see Table 3), and thus the potentially large number of nodes to explore in spatial branch-and-bound. ANTIGONE is an overall better performer than BARON and is considerable faster in the single period refinery problems. One possible explanation for the latter behavior is that cutting planes, or other techniques, are more efficient at reducing the domain of model variables than OBBT, when the problem size is small.

7.3 More detailed performance information

To understand how the algorithm is solving the benchmark problems, we show in Table 6 information related to: the total number of iterations; total time spent solving problems **PR** (step 5) and **PF** (step 6); executing OBBT (step 4) and depth search procedures (step 3); number of **PF** and **PRB** instances solved; average domain reduction in first and last OBBT call, with respect to the initial bounds; and number of partitions used in **PR** and **PRB** (final setting).

Piecewise relaxations are explored further in HYD2, with the algorithm reaching the maximum defined number of partitions for **PR** ($N_{PR} = 1000$) and **PRB** ($N_{PRB} = 7$). This is not surprising, considering that HYD2 has the fewest bilinear terms and variables appearing in bilinear terms (recall Table 3). As a consequence, we obtain the largest domain reduction (99.5%). Notice that HYD2 is the only problem taking advantage of the relaxation from multiparametric disaggregation.

The final OBBT domain reduction is strongly dependent on problem size, decreasing to 67 and 32.9% when the number of reservoirs in the hydro problems increases from 2 to 4 and 7, and from above 95% to below 86% when switching from the single to the three-period refinery problems.

The time spent performing optimality-based bound tightening typically far exceeds the time spent solving relaxation problems. The two exceptions are HYD4 and HYD7, for which domain reduction became ineffective for $N_{PRB} = 3$ and 2 (while reaching the $TIME^{max}_{PRB}$ limit), and all remaining time was allocated to the final **PR** problems with $N_{PR} = 4$ partitions. Refinery problems SC#TP3 exhibited a similar behavior, with the larger number of **PRB** instances solved explaining the longer OBBT time. Options to improve the algorithm performance for such problems involve extending the time limit and reducing the number of **PRB** instances to be solved in parallel.

7.4 Closing the gap

The extensive use of time-consuming yet very efficient piecewise relaxation techniques by our algorithm, is clearly visible when plotting the optimality gap as a function of wall time, see Figs. 7 and 8. Recall from Fig. 6 that the optimality gap is only updated after a sequence of procedures: OBBT (tightens the variable bounds); solving problem **PR** (computes the lower bound, which may only improve with respect to the *LB* incumbent in the last moments of solving the MILP to optimality); solving NLP problems **PF** (compute the upper bound). The consequence is a stepwise profile with major drops in optimality gap compared to a smoother profile from the commercial solvers. Notice that there is still some progress towards the end

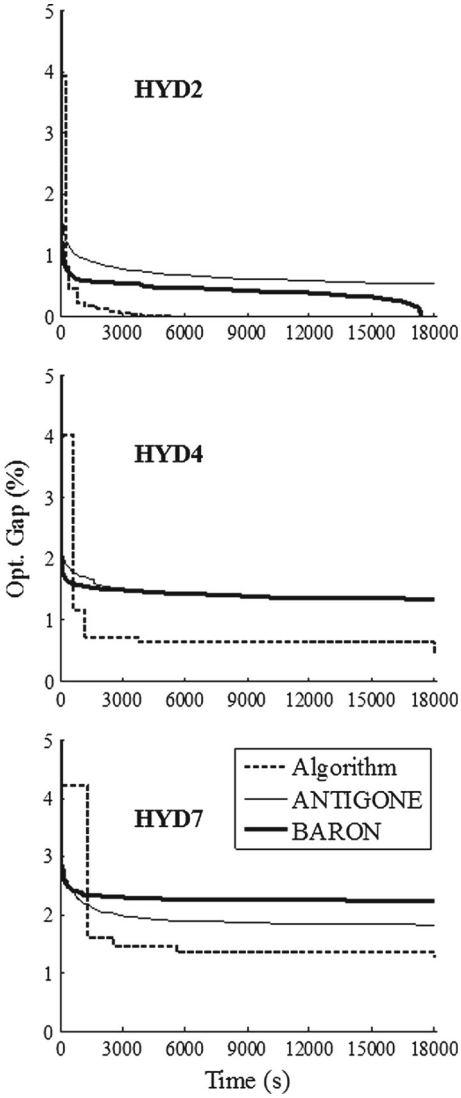

Fig. 7 Optimality gap versus wall time for hydro problems

of the search (SC#TP3 problems in Fig. 8), when the solvers have already plateaued. One disadvantage is that it may take a few hundred seconds to go below the gaps of ANTIGONE and BARON.

7.5 Removing the effect of OBBT

The four problems that were solved by the proposed algorithm to global optimality have in common the reduction of the domain of the variables involved in bilinear terms to less than 95% of the initial ranges, on average. We now test the performance of the commercial solvers after setting the variables domain to the final range obtained by our algorithm. The influence of the upper bound is also removed by initializing with the optimum.

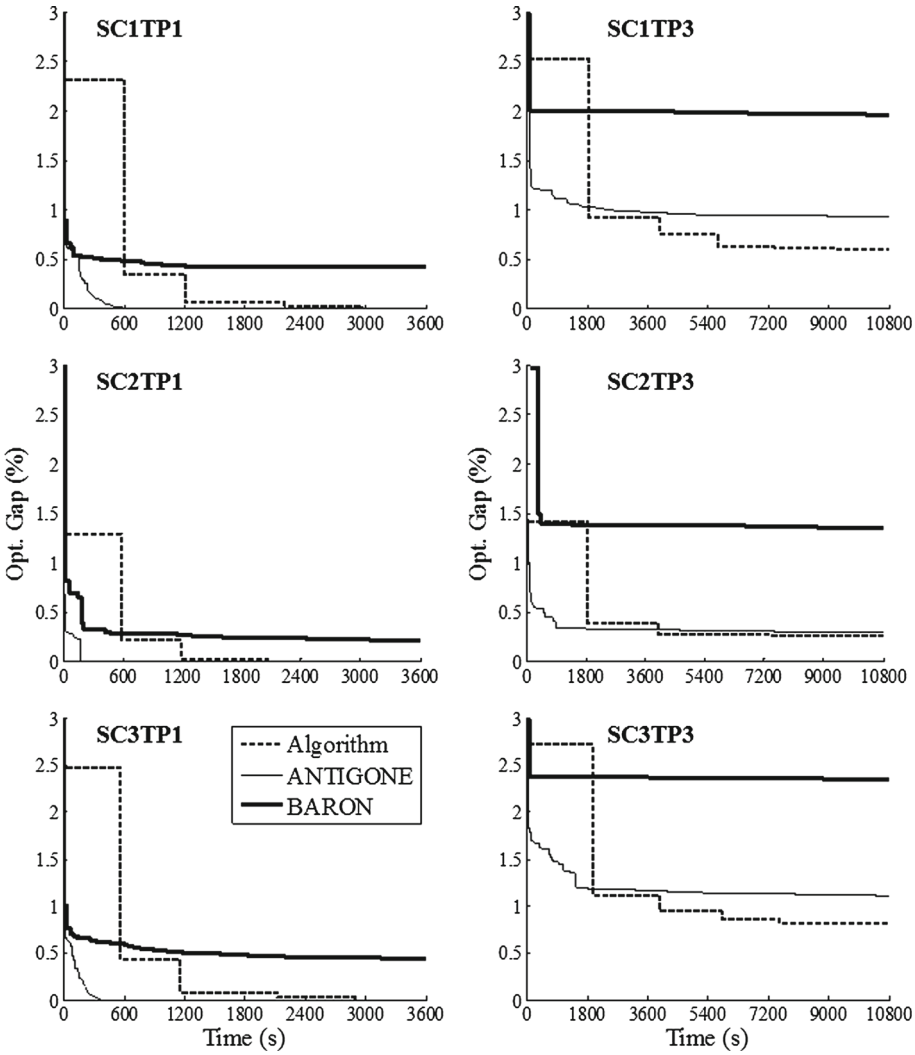

Fig. 8 Optimality gap versus wall time for refinery problems

The results in Table 7 show that the warm start helps ANTIGONE and BARON to solve such four problems in less than a minute. Improvements for HYD4, HYD7 and the refinery problems (with ANTIGONE) are minor. For the latter, BARON reduces the gap to less than half the values in Table 5. Neither solver can reach optimality gaps as low as the proposed algorithm, highlighting the importance of piecewise relaxations.

7.6 Sequential versus parallel OBBT

It remains to explain our choice for a parallel rather than a sequential implementation of optimality-based bound tightening. Figure 9 shows the optimality gap versus time profiles for refinery problems SC1TP1 and SC1TP3 and a fixed number of partitions in problems

Table 7 Performance of commercial solvers after warm start

Problem	Optimality gap (%)		Wall time (s)	
	ANTIGONE	BARON	ANTIGONE	BARON
HYD2	GO	GO	49	3
HYD4	1.241	1.157	WTL	WTL
HYD7	1.805	2.092	WTL	WTL
SC1TP1	GO	GO	3	3
SC2TP1	GO	GO	3	4
SC3TP1	GO	GO	3	8
SC1TP3	0.84	0.85	WTL	WTL
SC2TP3	0.28	0.28	WTL	WTL
SC3TP3	1.03	1.10	WTL	WTL
SC4TP3	0.64	0.72	WTL	WTL

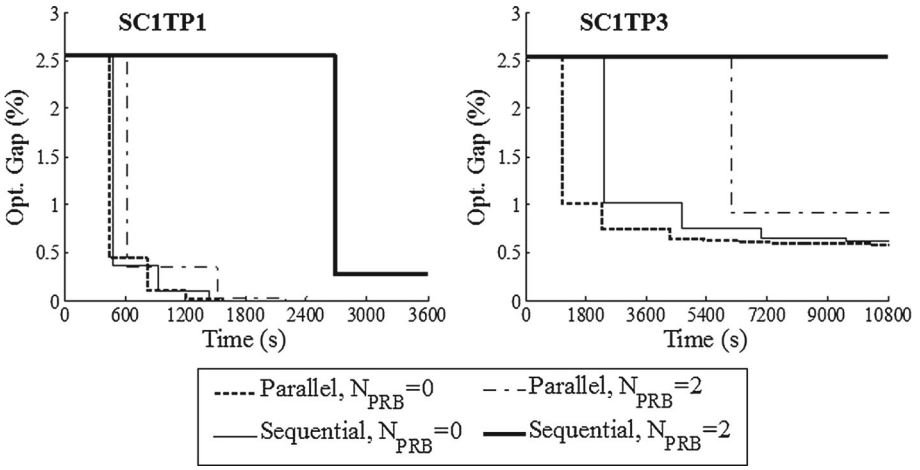

Fig. 9 Optimality gap profiles for sequential versus parallel OBBT

PR ($N_{PR} = 2$) and **PRB**, leading to the solution of LP ($N_{PRB} = 0$) or MILP problems ($N_{PRB} = 2$).

Results for the easiest SC1TP1 problem show that there are no major differences between the sequential and parallel implementations when solving LP problems. The optimality gaps are better when optimizing the bounds for one variable after the other (as expected) and not much time is lost compared to the parallel approach, for which the overhead of exchanging information between the threads is high. After the second iteration, the gaps become very similar and the lower computational time starts to be noticeable. Switching to MILP problems improves the relaxation quality and makes the parallel implementation far more competitive, with three iterations of OBBT taking less time and returning significantly smaller gaps than one iteration with the sequential approach.

Sequential OBBT with a piecewise relaxation ($N_{PRB} = 2$) is no longer an option for the larger SC1TP3, i.e. three hours are not enough to complete one iteration. We can still tackle one iteration with the parallel approach, but it is far more efficient to rely on the standard McCormick relaxation. Overall, the benefits from a parallel implementation of OBBT become

increasingly more important with the increase in the number of variables in bilinear terms and the number of partitions in **PRB**.

8 Conclusions

This paper has presented a new global optimization algorithm for mixed-integer quadratically constrained problems that does not employ spatial branch-and-bound. The novel aspect is the use of dynamic partitioning in piecewise relaxations, not only to compute lower bounds for the problem being minimized, but also to reduce the domain of the variables involved in bilinear terms. Relaxations range from the simplest bilinear envelopes at the start, to univariate piecewise McCormick, up to normalized multiparametric disaggregation, which is computationally more efficient for 10 partitions and beyond. The first type provides a quick lower bound, with the algorithm then switching to piecewise relaxations to refine such estimate. The number of partitions keeps increasing while the relaxation problem remains solvable to the given tolerance within the specified time. In case of severe increase in complexity, the algorithm backtracks to the previous setting, focusing more on optimality-based bound tightening (OBBT). Heuristic rules are used to decide when to increase/decrease the number of partitions in OBBT.

The algorithm has been designed to take advantage of parallel computing when doing OBBT and computing upper bounds. Rather than reducing one at a time the domain of the many variables that appear in bilinear terms, which leads to the tightest bounds, multiple variables are handled simultaneously to reduce the computational wall time. A solution pool is activated when solving the MILP relaxation problems, to generate alternative initialization points for solving restricted NLPs of the original non-convex problem that, if feasible, provide upper bounds. These are also solved in parallel.

The algorithm has been tested on ten industrially relevant benchmark problems, three hydroelectric scheduling problems with more discrete decisions and petroleum refinery planning problems with a larger number of bilinear terms. The computational results have shown that more problems can be solved to ε-global optimality. For the other six problems, the final optimality gaps were better than the values reported in the literature and lower than the ones from state-of-the-art commercial global optimization solvers ANTIGONE and BARON. The latter remained above our algorithm even when starting from a reduced variable range (from our last OBBT iteration). It shows that as problem size increases, piecewise relaxations with just 2 and 4 partitions can already provide tighter lower bounds than spatial branch-and-bound. Commercial solvers should thus use them to a greater extent.

Acknowledgements Support by Ontario Research Foundation, McMaster Advanced Control Consortium, and Fundação para a Ciência e Tecnologia (Projects IF/00781/2013 and UID/MAT/04561/2013), is gratefully appreciated.

References

1. Meyer, C.A., Floudas, C.A.: Global optimization of a combinatorially complex generalized pooling problem. AIChE J. **52**, 1027–1037 (2006)
2. 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**, 876–892 (2011)

3. Castro, P.M.: New MINLP formulation for the multiperiod pooling problem. AIChE J. **61**, 3728–3738 (2015)
4. Lotero, I., Trespalacios, F., Grossmann, I.E., Papageorgiou, D.J., Cheon, M.-S.: An MILP-MINLP decomposition method for the global optimization of a source based model of the multiperiod blending problem. Comput. Chem. Eng. **87**, 13–35 (2016)
5. Quesada, I., Grossmann, I.E.: Global optimization of bilinear process networks with multicomponent flows. Comput. Chem. Eng. **19**, 1219–1242 (1995)
6. Lee, S., Grossmann, I.E.: Global optimization of nonlinear generalized disjunctive programming with bilinear equality constraints: applications to process networks. Comput. Chem. Eng. **27**, 1557–1575 (2003)
7. Faria, D.C., Bagajewicz, M.J.: Novel bound contraction procedure for global optimization of bilinear MINLP problems with applications to water management problems. Comput. Chem. Eng. **35**, 446–455 (2011)
8. Rubio-Castro, E., Ponce-Ortega, J.M., Serna-González, M., El-Halwagi, M.M., Pham, V.: Global optimization in property-based interplant water integration. AIChE J. **59**, 813–833 (2013)
9. Alnouri, S., Linke, P., El-Halwagi, M.M.: Spatially constrained interplant water network synthesis with water treatment options. In: Eden, M.R., Siirola, J.D.S., Towler, G.P. (eds.) Proceedings of the 8th International Conference on Foundations of Computer-Aided Process Design, pp. 237–242. Elsevier, Amsterdam (2014)
10. 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)
11. Koleva, M.N., Styan, C.A., Papageorgiou, L.G.: Optimisation approaches for the synthesis of water treatment plants. Comput. Chem. Eng. (2017)
12. Andrade, T., Ribas, G., Oliveira, F.: A strategy based on convex relaxation for solving the oil refinery operations planning problem. Ind. Eng. Chem. Res. **55**, 144–155 (2016)
13. Castillo Castillo, P., Castro, P.M., Mahalec, V.: Global optimization algorithm for large-scale refinery planning models with bilinear terms. Ind. Eng. Chem. Res. **56**, 530–548 (2017)
14. Castro, P.M., Grossmann, I.E.: Global optimal scheduling of crude oil blending operations with RTN continuous-time and multiparametric disaggregation. Ind. Eng. Chem. Res. **53**, 15127–15145 (2014)
15. Cerdá, J., Pautasso, P.C., Cafaro, D.C.: Efficient approach for scheduling crude oil operations in marine-access refineries. Ind. Eng. Chem. Res. **54**, 8219–8238 (2015)
16. Zhao, Y., Wu, N., Li, Z., Qu, T.: A novel solution approach to a priority-slot-based continuous-time mixed integer nonlinear programming formulation for a crude-oil scheduling problem. Ind. Eng. Chem. Res. **55**, 10955–10967 (2016)
17. Catalão, J.P.S., Pousinho, H.M.I., Mendes, V.M.F.: Hydro energy systems management in Portugal: profit-based evaluation of a mixed-integer nonlinear approach. Energy **36**, 500–507 (2011)
18. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin (2013)
19. Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. J. Glob. Optim. **8**, 107–138 (1996)
20. Smith, E.M.B., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. Comput. Chem. Eng. **21**, S791–S796 (1997)
21. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. Math. Program. **10**, 147–175 (1976)
22. Karuppiah, R., Grossmann, I.E.: Global optimization for the synthesis of integrated water systems in chemical processes. Comput. Chem. Eng. **30**, 650–673 (2006)
23. Alfaki, M., Haugland, D.: A multi-commodity flow formulation for the generalized pooling problem. J. Glob. Optim. **56**, 917–937 (2013)
24. Bergamini, M.L., Aguirre, P., Grossmann, I.: Logic-based outer approximation for globally optimal synthesis of process networks. Comput. Chem. Eng. **29**, 1914–1933 (2005)
25. Wicaksono, D.S., Karimi, I.A.: Piecewise MILP under- and overestimators for global optimization of bilinear programs. AIChE J. **54**, 991–1008 (2008)
26. Li, X., Chen, Y., Barton, P.I.: Nonconvex generalized benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems. Ind. Eng. Chem. Res. **51**, 7287–7299 (2012)
27. Castro, P.M.: Tightening piecewise McCormick relaxations for bilinear problems. Comput. Chem. Eng. **72**, 300–311 (2015)
28. Kolodziej, S., Castro, P.M., Grossmann, I.E.: Global optimization of bilinear programs with a multiparametric disaggregation technique. J. Glob. Optim. **57**, 1039–1063 (2013)
29. Castro, P.M.: Normalized multiparametric disaggregation: an efficient relaxation for mixed-integer bilinear problems. J. Glob. Optim. **64**, 765–784 (2016)

30. Faria, D.C., Bagajewicz, M.J.: A new approach for global optimization of a class of MINLP problems with applications to water management and pooling problems. AIChE J. **58**, 2320–2335 (2012)
31. Castro, P.M.: Spatial branch-and-bound algorithm for MIQCPs featuring multiparametric disaggregation. Optim. Methods Softw. **32**, 719–737 (2017)
32. Castro, P.M., Teles, J.P.: Comparison of global optimization algorithms for the design of water-using networks. Comput. Chem. Eng. **52**, 249–261 (2013)
33. Castro, P.M., Grossmann, I.E.: Optimality-based bound contraction with multiparametric disaggregation for the global optimization of mixed-integer bilinear problems. J. Glob. Optim. **59**, 277–306 (2014)
34. Nagarajan, H., Lu, M., Yamangil, E., Bent, R.: Tightening McCormick relaxations for nonlinear programs via dynamic multivariate partitioning. In: Rueher, M. (ed.) Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5–9, 2016, Proceedings, pp. 369–387. Springer, Cham (2016)
35. Saxena, A., Bonami, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. Math. Program. **124**, 383–411 (2010)
36. Saxena, A., Bonami, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. Math. Program. **130**, 359–413 (2011)
37. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**, 225–249 (2005)
38. Misener, R., Floudas, C.A.: GloMIQO: global mixed-integer quadratic optimizer. J. Glob. Optim. **57**, 3–50 (2013)
39. Misener, R., Floudas, C.A.: ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. J. Glob. Optim. **59**, 503–526 (2014)
40. Gleixner, A.M., Berthold, T., Müller, B., Weltge, S.: Three enhancements for optimization-based bound tightening. J. Glob. Optim. **67**, 731–757 (2017)
41. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J. Algebr. Discrete Methods **6**, 466–486 (1985)
42. Atamturk, A., Nemhauser, G.L., Savelsbergh, M.W.P.: Conflict graphs in solving integer programming problems. Eur. J. Oper. Res. **121**, 40–55 (2000)