A New Approach for Global Optimization of a Class of MINLP Problems with Applications to Water Management and Pooling Problems

Débora C. Faria and Miguel J. Bagajewicz
University of Oklahoma, 100E. Boyd, Norman, OK 73019

DOI 10.1002/aic.12754
Published online November 11, 2011 in Wiley Online Library (wileyonlinelibrary.com).

One of the biggest challenges in solving optimization engineering problems is rooted in the nonlinearities and nonconvexities, which arise from bilinear terms corresponding to component material balances and/or concave functions used to estimate capital cost of equipments. The procedure proposed uses an MILP lower bound constructed using partitioning of certain variables, similar to the one used by other approaches. The core of the method is to bound contract a set of variables that are not necessarily the ones being partitioned. The procedure for bound contraction consists of a novel interval elimination procedure that has several variants. Once bound contraction is exhausted the method increases the number of intervals or resorts to a branch and bound strategy where bound contraction takes place at each node. The procedure is illustrated with examples of water management and pooling problems. © 2011 American Institute of Chemical Engineers *AIChE J,* 58: 2320–2335, 2012
Keywords: mathematical programming, optimization, global optimization

Introduction

Several methods have been proposed for global optimization, many of which became popular in the chemical engineering community, some having reached commercial status, like BARON,[1] COCOS, GlobSol, ICOS, LGO, LINGO, OQNLP, Premium Solver, or others that are well-known like the αBB.[2] A more comprehensive understanding of evolution and advances of global optimization can be found in several books[3–7] and recent articles.[8–11]

The problems we address are those that contain bilinear terms where a flow rate and a concentration participate in bilinear terms, as part of component balances in processes. In addition, we consider the presence of univariate concave terms that are part of the objective function, typically representing equipment costs. Water management problems, as well as pooling problems are such problems. We do not rule out extensions to other bilinear problems, which are part of future work.

There are a few different approaches for global optimization. Because this is not a review and because we focus on a special type of bilinear problems, we only cite recent academic efforts that are close to the methodology we present and the specific optimization problems we target. Briefly, Lagrangean-based approaches[12–15] and disjunctive programming-based methods[16] have been used. In turn, several global optimization procedures to handle bilinear and concave terms that are based on interval analysis have been presented: Intervals analysis arithmetic was first presented by Moore[17] and then several subsequent articles tried to develop and improve global optimization approaches using interval analysis. Later, Vaidyanathan and El-Halwagi[18] proposed an algorithm based on deleting infeasible subspaces using different tools to accelerate existing interval-based methods.[19–21]

Another important class of methods are those based on branch and bound on key variables. Zamora and Grossmann[22] presented a method for problems containing bilinear and fractional terms as well as univariate concave functions where they use such a branch and bound method, but also performing variable bounds contractions at each node. For their lower bound model, they make use of a lower bound problem constructed using McCormick-type underestimators for the bilinear and fractional terms, and linear underestimators for the concave functions. The set of complicating variables on which the branching is performed is chosen as the one that shows the largest deviation from the corresponding bilinear term, fractional term or the concave function. The upper bound solution, together with a special LP (or convex NLP) contraction subproblem helps finding what bounds can be contracted. These new bounds are found by maximizing/minimizing each of their values subject to the linearized objective being smaller than the current UB. They also use new bounding inequalities generated by the Lagrangean multipliers of the contraction subproblem to improve the LBs in the branch and bound procedure.

To address bilinear terms in generalized pooling problems, Meyer and Floudas[23] used a piecewise reformulation-linearization technique (RLT), similar to the one proposed originally by Sherali and Alamedinne.[24] Reformulation consists of obtaining new redundant nonlinear constraints that are

Correspondence concerning this article should be addressed to M. J. Bagajewicz at bagajewicz@ou.edu.

obtained by multiplying groups of valid constraints from the original problem. These new constraints are, of course, redundant in the original problem, but may be nonredundant in a convex relaxation. They partition the continuous space of one of the variables participating in a bilinear term in several intervals to generate a MINLP, allowing them to linearize the model to be able to generate lower bounds. Thus, linearization involves substituting bilinearities by a new variable and adding new constraints obtained by multiplying the bound inequalities (RLT). They suggest that a LB/UB scheme through which a gap can be reduced can be based in three alternatives: branching on the continuous variable, branching on the integers corresponding to the partitions and a third one that they adopt: augmenting the LB problem with a set of binary variables to model a partition of the continuous space, and then reformulating and linearizing the problem as an MILP. The method does not involve a search for upper bounds and as they state, ''several attempts and reformulations before a solution can be validated'' are needed. Thus, the method is used just to verify the gap relative to the best known optimum solution and no procedure is presented to reduce the gap between lower and upper bounds. Different numbers of partitions of the continuous variables are considered to obtain the best lower bound. The method is able to generate very tight lower bounds at a cost of significant computational efforts due to the increase in numbers of binary variables. Later, Misener and Floudas[25] compared the performance of different models for pooling problems as well as different techniques, including reformulation and the use of piecewise affine underestimators. Later, Gounaris et al.[26] compared the performance of different linear relaxations and Misener and Floudas[27] introduced a quadratically constrained MINLP model that reduces the number of bilinear terms for pooling problems, used Gounaris et al.[26] and Wicaksono and Karimi[28] piecewise underestimations and implemented a branch and bound strategy to solve efficiently several case studies. Finally, Misener et al.[29] extended the pooling problem to add EPA constraints and proposed MILP relaxations of bilinear and concave terms, solving several test problems.

Karuppiah and Grossmann[30] use a deterministic spatial branch and contract algorithm. To obtain a lower bound for the original NLP model, the bilinear terms are relaxed using the convex and concave envelopes,[31] and the concave terms of the objective function are replaced by underestimators generated by the secant of the concave term. To improve the tightness of the lower bound, piecewise underestimators generated from partitioning of the flow variables are used to construct tighter envelopes and concave underestimators. The model is solved using disjunctions. They also choose the variable of the bilinear term that participates in the larger number of constraints to reduce the number of disjunctions. Logical cuts are also included to aid in the convergence (for example, if two flows are to be identical, then they should fall within the same interval). Their algorithm then follows a bound contraction procedure which is a simplified version of the one used by Zamora and Grossmann.[22] As in Meyer and Floudas[23] the number of partitions can make the lower bound tighter, but extra computational effort is needed. In a second article, Karuppiah and Grossmann[32] extended the previous method to solve the multiscenario case of the integrated water systems. In both cases, the relaxed model, which renders a lower bound, is used in a LB/UB framework. In the first case[30] a spatial branch and bound procedure is used. For the latter multiscenario case[32] a spatial branch and cut algorithm is applied. The cuts are generated using a decomposition based on Lagrangean relaxation.

Bergamini et al.[33] proposed an outer approximation method (OA) for global optimization; improvements followed later.[34] The major modifications are related to a new formulation of the underestimators (which replace the concave and bilinear terms) using the delta-method of piecewise functions (see Padberd[35]); and, the replacement of the most expensive step (global solution of the bounding problem) by a strategy based on the mathematical structure of the problem, which searches for better feasible solutions of fixed network structures. The improved outer approximation method relies in three subproblems that need to be solved to feasibility instead to optimality. In turn, the model always look for solutions that are strictly lower (using a tolerance) than the current optimum solution.

Wicaksono and Karimi[28] presented a piecewise underestimation technique that leads to MILP underestimating problems. Later Hasan and Karimi[36] explored the numerical efficiency of different variants. Pham et al.[37] presented a similar technique to obtain piecewise underestimators and propose a branch and bound method, as well as refinement techniques of the partitioned grid to solve pooling problems globally. Following the ideas of these articles, we present a variable partitioning methodology to obtain lower bounds and a new bound contraction procedure. Our lower bound model uses some modified versions of well-known over and underestimators (some of which is used in the aforementioned literature review), to obtain MILP models. Our procedure differs from most of the previous approaches based on LB/UB schemes in that it does not use a branch and bound methodology as the core of the method. Instead, we first partition certain variables into several intervals and then use a bound contraction procedure directly using an interval elimination strategy. Conceptually, the technique can work if a sufficiently high number of intervals are used, but if that becomes computationally too expensive, we allow a spatial branch and bound to be used.

Although introducing logical cuts and/or performing bound contraction using interval arithmetic, either as a preliminary step or after each contraction, the method does not necessarily require introducing these logical cuts, reformulations or interval arithmetic-based contraction.

This article is organized as follows: We present the solution strategy first, followed by a description of the partitioning procedure model and the lower bound MILP models. Then, we discuss the bound contraction procedure, as well as the auxiliary branch and bound procedures. Finally, examples are presented and discussed.

Solution strategy

After partitioning one of the variables in the bilinear terms, our method consists of a bound contraction step that uses a procedure for eliminating intervals. Once the bound contraction is exhausted, the method relies on increasing the number of intervals, or on a branch and bound strategy where the interval elimination takes place at each node. The partitioning methodology (outlined later), generates linear models that guarantee to be lower bounds of the problem. Upper bounds are needed for the bound contraction procedure. These upper bounds are usually obtained using the

original MINLP model often initialized by the results of the lower bound model. In many cases, one can use some information of the solution of the lower bound linear problem to obtain feasible solutions to the original MINLP problem, thus, obtaining an upper bound.

Before we outline the strategy, we define variables:

- *Partitioning variables*: These are the variables that are partitioned into intervals and used to construct linear relaxations of the bilinear and concave terms. The resulting models are MILP.

- *Bound contracted variables*: These variables are partitioned into intervals, but only for the purpose of performing their bound contraction. The lower bound model will simply identify the interval in which the variable to be bound contracted lies and use this information in the elimination procedure. Clearly, these variables need not be the same as the partitioned variables.

- *Branch and bound variables*: These are the variables for which a branch and bound procedure is tried. It need not be the same set as the other two variables.

For example, in water management problems the bilinear terms are composed of the product of flow rates and concentrations. Thus, one can have a problem in which the partitioning variables are all or part of the concentrations, the bound contracted variables, be the flow rates and the B&B variables the flow rates as well. As we discuss later, the B&B is more efficient when the variables used are different from the partitioning variables when using McCormick's envelopes, which has information of the nonpartitioned variable. Alternatively, one can use concentrations for both the partitioning and BC variables, with flow rates for B&B, or the partitioning variables could be both flow rates and concentrations (in which case the LB model is more efficient), the BC variables as well as the B&B variables the flow rates or the concentrations or both, and so on.

We also note that although the bound contract variable and branch and bound variable do not need to be the same as the partitioned one, it is normal to have them being bound contracted or branched, as opposed to picking other variables. In some cases, picking the variable to bound contract different form the one to partition renders tighter lower bounds as bound contraction takes place. However, we point out that the feasible region of the lower bound model can only become close to the feasible region of the original problem when the partitioned variables have discrete bounds within an ϵ tolerance, and this can only be done through bound contraction and/or using branch and bound.

The global optimization strategy is now summarized as follows:

- Construct a lower bound model partitioning variables in bilinear and quadratic terms, thus, relaxing the bilinear terms as well as adding piecewise linear underestimators of concave terms of the objective function. If the concave terms are not part of the objective function, then overestimators can be used, but this is not included in this article.

- The lower bound model is run identifying certain intervals as containing the solution for specific variables that are to be bound contracted. These variables need not be the same variables as the ones using to construct the lower bound.

- Based on this information the value of the upper bound found by running the original MINLP using the information obtained by solving the lower bound model to obtain a good starting point. Other *ad hoc* upper bounds can be constructed. For example, in water management problems, one can leave the same flows predicted by the lower bound and calculate the outlet concentrations of the units, which is most of the time feasible.

- A strategy based on the successive running of lower bounds where certain intervals are temporarily forbidden is used to eliminate regions of the feasible space. This is the bound contraction part.

- The process is repeated with new bounds until convergence or until the bounds cannot be contracted anymore.

- If the bound contraction is exhausted, there are two possibilities to guarantee global optimality:

  ○ Increase the partitioning of the variables to a level in which the sizes of the intervals are small enough to generate a lower bound within a given acceptable tolerance to the upper bound; or,

  ○ Recursively split the problem in two or more subproblems using a strategy such as the ones based on branch and bound procedure.

The first option of increased partitioning will not lead to further improvement in bound contraction if degenerate solutions (or very close to the global solutions) exist for different partitions of the partitioned variables. In other words, when degenerate solutions are present bound contraction will stop progressing, even if the gap between lower and upper bound is small.

We discuss all these steps in the next few sections.

Partitioning methodology

We show here two different partitioning strategies. The proposed approach consists of partitioning one of the variables of the bilinear terms, but one could also partition both.

Bilinear and Quadratic Terms. There are different ways to linearize the bilinear terms using discrete points of one (or both) given variable(s). We use:

- *Direct partitioning* (see notation). Some details of this technique were presented earlier.[38]

- *Convex envelopes*[31] as used by Karuppiah and Grossmann.[30]

To deal with the product of continuous variables and binary variables, we consider three variants of each procedure.

We now explain the basics of these techniques using the following generic case. Consider z to be the product of two continuous variables x and y

$$z = xy \tag{1}$$

where both x and y subject to certain bounds

$$x^L \le x \le x^U \tag{2}$$

$$y^L \le y \le y^U \tag{3}$$

Assume now that variable y is partitioned using D-1 intervals. The starting point of each interval is given by

$$\hat{y}_d = y^L + (d-1)\frac{(y^U - y^L)}{D-1} \quad \forall d = 1..D \quad y^L \le y \le y^U \tag{4}$$

In the case of the *direct partitioning*, we simply substitute the variable y in the product $x = y$ by its discrete bounds,

thus, allowing z to be inside of one of the intervals, that is, between two successive discrete values. Binary variables (v_d) are used to assure that only one interval is picked

$$\sum_{d=1}^{D-1} \hat{y}_d v_d \leq y \leq \sum_{d=1}^{D-1} \hat{y}_{d+1} v_d \tag{5}$$

$$\sum_{d=1}^{D-1} v_d = 1 \tag{6}$$

$$z \leq x \sum_{d=1}^{D-1} \hat{y}_{d+1} v_d \tag{7}$$

$$z \geq x \sum_{d=1}^{D-1} \hat{y}_d v_d \tag{8}$$

Equation 5 states that y falls within the interval corresponding to the binary variable v_d, of which only one is equal to one (Eq. 6 enforces this). This is done for the partitioning variables, but if x (or a subset of it) is the BC variable, then a similar partitioning as the one in Eqs. 5 and 6 is included.

In turn, Eqs. 7 and 8 bound the value of z to correspond to a value of y in the given interval.

In the case of using *McCormick's envelopes* for each interval, the equations are

$$z \geq x^L y + \sum_{d=1}^{D-1} \left(x \hat{y}_d v_d - x^L \hat{y}_d v_d \right) \tag{9}$$

$$z \geq x^U y + \sum_{d=1}^{D-1} \left(x \hat{y}_{d+1} v_d - x^U \hat{y}_{d+1} v_d \right) \tag{10}$$

$$z \leq x^L y + \sum_{d=1}^{D-1} \left(x \hat{y}_{d+1} v_d - x^L \hat{y}_{d+1} v_d \right) \tag{11}$$

$$z \leq x^U y + \sum_{d=1}^{D-1} \left(x \hat{y}_d v_d - x^U \hat{y}_d v_d \right) \tag{12}$$

which are used in conjunction with Eqs. 5 and 6.

When x (or a subset of it) is the BC variable, then we only add Eqs. 5 and 6 for these variables, but do not incorporate the bounds of each interval in the aforementioned Eqs. 9–12.

Note that even if the bilinearity generated by the multiplication of y and x was eliminated, we still have variable x being multiplied by the binary variable v_d in both cases. Once again there are different ways to linearize the product of a continuous and binary variable. These methods, in various forms, are very well known and we present next our implementation.

In the case of quadratic terms, we rewrite Eq. 1 as $z = xx$ and we proceed to partition one of the x variables. We only illustrate the bilinear case in this article.

- *Direct Partitioning Variants.* When using the *direct partitioning* we linearize the product of x and the binary variable v_d using three different procedures:

Direct partitioning procedure 1, (DPP1); Let w_d be a positive variable ($w_d \geq 0$), such that $w_d = x\ v_d$. Then, Eqs. 7 and 8 are substituted by

$$z \leq \sum_{d=1}^{D-1} \hat{y}_{d+1}\ w_d \tag{13}$$

$$z \geq \sum_{d=1}^{D-1} \hat{y}_d\ w_d \tag{14}$$

and w_d is now obtained from the following linear equations

$$w_d - x^U v_d \leq 0 \tag{15}$$

$$(x - w_d) - x^U (1 - v_d) \leq 0 \tag{16}$$

$$x - w_d \geq 0 \tag{17}$$

Indeed, if $v_d = 0$, Eq. 15, together with the fact that $w_d \geq 0$, renders $w_d = 0$. Conversely, if $v_d = 1$, Eqs. 16 and 17 render $w_d = x$, which is what is desired. The aforementioned scheme works well for any positive variable x, even when $x^L \neq 0$ Equation 17 could be rewritten as follows: $x + x^L(1 - v_d) - w_d \geq 0$. In addition, the following new constraint $w_d - x^L v_d \geq 0$ can be added. We believe these changes would do very little to improve computational efficiency. Indeed for $v_d = 1$ $x + x^L\ (1 - v_d) - w_d \geq 0$, renders Eq. 17 and the new proposed restriction ($w_d - x^L v_d \geq 0$) renders $w_d \geq x^L$, which is useless because $w_d = x > x^L$. Conversely, when $v_d = 0$, $x + x^L(1 - v_d) - w_d \geq 0$ and $w_d - x^L v_d \geq 0$ are trivially satisfied.

There is, however, an alternative more compact way of writing the linearization: Indeed, the following equations accomplish the same linearization.

- *Direct partitioning procedure* 2 (DPP2): In this case, the product of the binary variable and the continuous variable is linearized as follows

$$w_d \leq x^U v_d \quad \forall d = 1..D-1 \tag{18}$$

$$w_d \geq x^L v_d \quad \forall d = 1..D-1 \tag{19}$$

$$x = \sum_{d=1}^{D-1} w_d \tag{20}$$

Equations 18 and 19 guarantee that only one value of w_d (when $v_d = 1$) can be greater than zero and between bounds (all other w_d, for when $v_d = 0$, are zero). Thus, Eq. 20 sets w_d to the value of x.

- *Direct partitioning procedure* 3 (DPP3): This procedure uses the following equations to linearize Eqs. 7 and 8

$$z \leq x \hat{y}_{d+1} + x^U \left(y^U - \hat{y}_{d+1} \right) (1 - v_d) \quad \forall d = 1..D-1 \tag{21}$$

$$z \geq x \hat{y}_d - x^U \hat{y}_d (1 - v_d) \quad \forall d = 1..D-1 \tag{22}$$

$$z \leq x^U y \tag{23}$$

Let d^* be the interval for which $v_{d^*} = 1$, in the solution. Let also y^* be the corresponding value of y in the interval in quation (that is $y_{d^*} \leq y^* \leq y_{d^*+1}$. Thus, Eqs. 21 and 22 bracket z to a particular interval. Indeed, when $v_{d^*} = 1$, Eqs. 21 and 22 reduce to the following inequalities $x \hat{y}_{d^*} \leq z \leq x \hat{y}_{d^*+1}$. In turn, Eqs. 5 and 23 reduce to

$z \le x^U y^* \le x^U \hat{y}_{d^*+1}$. In the other intervals where $v_d = 0$, Eqs. 22 and 23 reduce to $(x - x^U)\hat{y}_d \le z \le x^U y^*$, which puts z between a lower negative bound and a valid upper bound. Finally, Eq. 21 reduces to $z \le x\,\hat{y}_{d+1} + x^U(y^U - \hat{y}_{d+1})$, which is a valid inequality. Indeed, recall that $x\,\hat{y}_{d^*} \le z = x\,y^* \le x\,\hat{y}_{d^*+1}$. Then, for $d \ge d^*$, we have $\hat{y}_{d^*+1} \le \hat{y}_{d+1}$, and then Eq. 21 reduces to $z \le x\,\hat{y}_{d+1} + x^U(y^U - \hat{y}_{d+1})$. We now need to prove that the righthand side is larger than or equal to $x\,\hat{y}_{d^*+1}$. Indeed, because we have $\hat{y}_{d^*+1} \le \hat{y}_{d+1}$, we have $x\,\hat{y}_{d+1} + x^U(y^U - \hat{y}_{d+1}) \ge x\hat{y}_{d^*+1} + x^U(y^U - \hat{y}_{d+1})$, which is larger than $x\,\hat{y}_{d^*+1}$ because $x^U(y^U - \hat{y}_{d+1}) >$ is positive. Conversely, when $d < d^*$, we have $\hat{y}_{d^*+1} \ge \hat{y}_{d+1}$. Thus, adding and subtracting $x\hat{y}_{d^*+1}$ to $x\,\hat{y}_{d+1} + x^U(y^U - \hat{y}_{d+1})$ and rearranging, we get $x\,\hat{y}_{d+1} + x^U(y^U - \hat{y}_{d+1}) = \hat{y}_{d^*+1}x + x^U(y^U - \hat{y}_{d+1}) + x(\hat{y}_{d+1} - \hat{y}_{d^*+1})$. However, because $y^U \ge \hat{y}_{d^*+1}$, we can then write $x\,\hat{y}_{d+1} + x^U(y^U - \hat{y}_{d+1}) \ge \hat{y}_{d^*+1}x + (x^U - x)(\hat{y}_{d^*+1} - \hat{y}_{d+1})$, which concludes the proof because the second term is positive. As in the case of DPP1, one can add constraints involving the lower bound. For example, one can add $z \ge x^L y$ or $y^L x$, which may or may not help convergence. Because we did not experiment with these options involving the lower bound, we do not discuss the issue further.

With all these substitution any MINLP model containing a bilinearity is transformed into an MILP, which is a lower bound of the original problem; this is because of the relaxation introduced.

McCormick Envelopes Variants. In this case, following Saif et al.[39] Eqs. 9–12 are substituted by the following equations

$$z \ge x^L y + \sum_{d=1}^{D-1} \left(\hat{y}_d\, w_d - x^L \hat{y}_d v_d\right) \tag{24}$$

$$z \ge x^U y + \sum_{d=1}^{D-1} \left(\hat{y}_{d+1}\, w_d - x^U \hat{y}_{d+1} v_d\right) \tag{25}$$

$$z \le x^L y + \sum_{d=1}^{D-1} \left(\hat{y}_{d+1}\, w_d - x^L \hat{y}_{d+1} v_d\right) \tag{26}$$

$$z \le x^U y + \sum_{d=1}^{D-1} \left(\hat{y}_d\, w_d - x^U \hat{y}_d v_d\right) \tag{27}$$

and several variants of how to linearize $w_d = x v_d$ follow:

- *McCormick's envelopes Procedure* 1 (MCP1): It is when Eqs. 15–17 are used.

- *McCormick's envelopes procedure* 2 (MCP2): In this case, Eqs. 18–20 are used instead of Eqs. 15–17).

- *McCormick's envelopes procedure* 3 (MCP3): In this case, Eqs. 5–6 are still used, but Eqs. 9–12 are substituted by

$$z \ge x^L y + x\ \hat{y}_d - x^L \hat{y}_d v_d - \left(x^L y^U + x^U \hat{y}_d\right)(1 - v_d) \qquad \forall d = 1..D-1 \tag{28}$$

$$z \ge x^U y + x\ \hat{y}_{d+1} - x^U \hat{y}_{d+1} v_d - x^U\left(y^U + \hat{y}_{d+1}\right)(1 - v_d) \qquad \forall d = 1..D-1 \tag{29}$$

$$z \le x^L y + x\ \hat{y}_{d+1} - x^L \hat{y}_{d+1} v_d + \left(x^U y^U - x^L\left(y^L + \hat{y}_{d+1}\right)\right)(1 - v_d) \qquad \forall d = 1..D-1 \tag{30}$$

$$z \le x^U y + x\ \hat{y}_d - x^U \hat{y}_d v_d + \left(x^U\left(y^U - y^L\right) - x^L \hat{y}_d\right)(1 - v_d) \qquad \forall d = 1..D-1 \tag{31}$$

$$z \le x^U y \tag{32}$$

The case $x^L = 0$ is a very common situation in flow sheet superstructure optimization where connections between units exist formally, but a flow rate of zero through some of these connections is almost always part of the optimal solution. If x represents the flow rates, and y the composition of the stream, when $x^L = 0$ Eq. 28 reduces to $z \ge x\,\hat{y}_d - x^U \hat{y}_d(1 - v_d)$, which is the same as Eq. 22. In turn Eq. 30 reduces to $z \le x\,\hat{y}_{d+1} + x^U y^U(1 - v_d)$, which does not reduce to Eq. 21.

As in the case of the direct partitioning, when these equations are substituted in the original MINLP, they transform it into an MILP, which is a lower bound of the original problem.

In addition, as we shall see in the examples, which variables should be partitioned in a bilinear term is also not straightforward. For example, in the case of problems with component balances one has the option to partition the flow rates or the concentrations. Because, flow rates participate in all the balances for each unit, they generate a lot less binary variables that partitioning the composition, because each balance contains its own composition. However, although partitioning flow rates may render a smaller number of integers, but may affect speed of convergence. This is discussed in more detail later when we illustrate the method.

Partitioning of Both Bilinear Variables. Partitioning both variables has some advantages. First, the LB may improve in some schemes. If bound contraction on concentrations is successful, then further bound contraction of flow rates may take place. We reproduce the exact equations we considered for clarity and completeness in the appendix. In the few cases we tried, we did not observe improvements using the partitioning of both variables mainly because the computation time is increased and the LB was observed to be at least as tight as partitioning concentrations.

Concave functions

Univariate functions used to estimate capital cost are often concave and expressed as functions of equipment sizes as follows

$$z = \Omega\, y^{\alpha} \tag{33}$$

where α is often a value between 0 and 1, and y is the equipment capacity. This term usually shows in the objective function.

We first consider that variable y is partitioned in several intervals as shown in Eq. 4. Then we linearize this concave function in each interval following Karuppiah and Grossmann[30] as follows

$$y^{\alpha} = \bar{y} \ge \sum_{d=1}^{D-1} v_d \left((\hat{y}_d)^{\alpha} + \left(\frac{(\hat{y}_{d+1})^{\alpha} - (\hat{y}_d)^{\alpha}}{\hat{y}_{d+1} - \hat{y}_d}\right)(y - \hat{y}_d)\right) \tag{34}$$

$$z = \Omega\, \bar{y} \tag{35}$$

which we use in conjunction with Eqs. 5 and 6.

Table 1. Limiting Data of Example 1

Process	Contaminant	Mass Load (kg/h)	$C^{in,max}$ (ppm)	$C^{out,max}$ (ppm)
1	A	4	0	100
	B	2	25	75
2	A	5.6	80	240
	B	2.1	30	90

Note that, again, we have the product of a binary variable (v_d), and a continuous variable (y). The linearization of Eq. 34 is the following

$$\bar{y} \geq \sum_{d=1}^{D-1} \left((\hat{y}_d)^{\alpha} v_d + \left(\frac{(\hat{y}_{d+1})^{\alpha} - (\hat{y}_d)^{\alpha}}{\hat{y}_{d+1} - \hat{y}_d} \right) (\beta_d - \hat{y}_d v_d) \right) \quad (36)$$

$$y = \sum_{d=1}^{D-1} \beta_d \quad (37)$$

$$\beta_d \leq \hat{y}_{d+1} v_d \qquad \forall d = 1..D-1 \quad (38)$$

$$\beta_d \geq \hat{y}_d v_d \qquad \forall d = 1..D-1 \quad (39)$$

which is again used in conjunction with Eqs. 5 and 6.

When substituted in the original MINLP, they transform it into an MILP. Such MILP is a lower bound of the original problem if z only appears in the objective function as an additive term, together with the equation defining it (Eq. 33). Conversely, when z shows up in some constraint of the problem in a nonconvex manner, but not in the objective as an additive term, then one would have to add an overestimator like the following

$$\bar{y} \leq \sum_{d=1}^{D-1} v_d \left(\left(\frac{\hat{y}_d + \hat{y}_{d+1}}{2} \right)^{\alpha} + \alpha \left(\frac{\hat{y}_d + \hat{y}_{d+1}}{2} \right)^{\alpha-1} \left(y - \frac{\hat{y}_d + \hat{y}_{d+1}}{2} \right) \right) \quad (40)$$

which uses the tangent line at the middle of the interval as an upper bound. One would do this only when the constraint containing z is an equality, or when z is part of an inequality of the "less than or equal" type with a negative coefficient. Finally, Padberg[35] proposes many other different alternatives for piecewise relaxing concave terms.

Bound Contraction -Interval Elimination Strategy

After a problem has been linearized and solved, the solution from this LB is used as initial guess to obtain an upper bound (feasible), and solution (the original nonconvex problem is used in most cases). Once a lower bound and an upper bound solution have been found, there is a need to identify which interval(s) can be eliminated from consideration. The lower bound solution points at a set of intervals, one per variable. This solution not only helps to find an upper bound solution, but also guides the elimination of certain intervals. Each iteration of the procedure is as follows:

Step 1. Run the lower bound model with no forbidden intervals.

Step 2. Use the solution from the lower bound model as an initial point to solve the original NLP or MINLP problem to obtain an upper bound solution.

Step 3. If the objective function gap between the upper bound solution and the lower bound solution is lower than the tolerance, the solution was found. Otherwise go to step 4.

Step 5. Run the lower bound model, this time forbidding the interval that contains the answer for the first partitioned variable.

Step 6. If the new problem is infeasible, or if feasible, but the objective function is higher than the current upper bound, then all the intervals that have not been forbidden for this variable are eliminated. The surviving feasible region between the new bounds is partitioned again.

Step 8. Repeat steps 4 and 5 for all the other variables, one at a time.

Step 9. Go back to step 1 (a new iteration using contracted bounds starts).

Note that to guarantee the optimality, not all of the lower bound models need to be solved to zero gap. The only problems that need to have zero gap are the ones in which the lower bound of the problem (or subproblems) are obtained, which is done in step 1. The lower bound models used to eliminate intervals (step 4) can be solved to feasibility between its lower bound and the current upper bound, which is always set as the upper bound of the whole procedure.

In some cases, a preprocessing step using bound arithmetic to reduce the initial bounds of certain variables can be performed. We discuss the specifics in our article depicting various results of this method.[40]

The standard version of our interval elimination (bound contraction) in the aforementioned procedure, which we call *one-pass with one forbidden interval elimination*, because the elimination process takes place sequentially, only one variable at a time and only once for each variable.

Variations to the aforementioned elimination strategy are possible. We distinguish five specific set of alternative options:

- Options related to the amount of times all variables are considered for bound contraction:

■ *One-pass elimination*: In step 6, each variable is visited only once before a new lower bound of the whole problem is obtained.

■ *Cyclic elimination*: In step 6, once all variables are visited, the method returns to the first variable and starts the

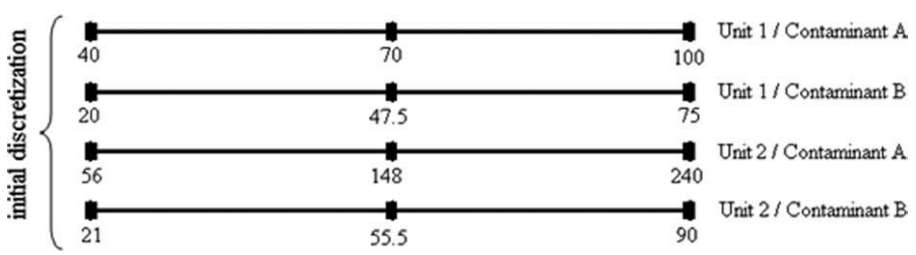

Figure 1. Illustrative example of the partitioning approach—initialization.

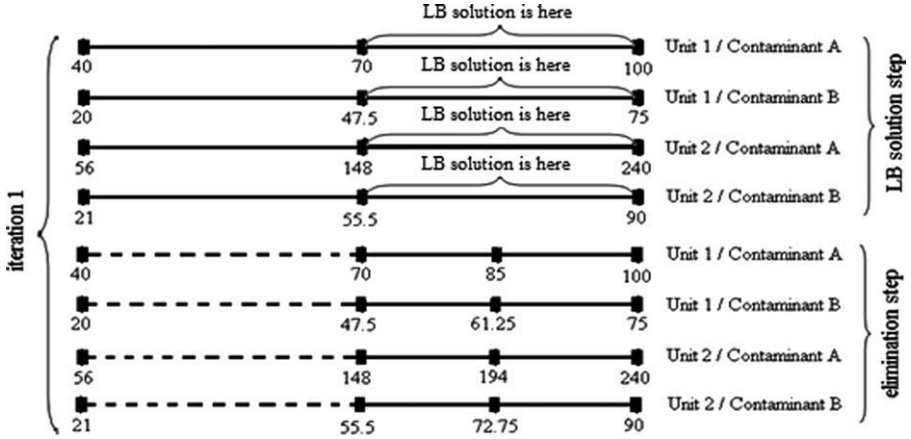

Figure 2. Illustrative example of the partitioning approach— 1st iteration.

process again, as many times as needed, until no more bound contraction is achieved.

- Options related to the amount of times each variable is bound contracted:

■ *Exhaustive elimination*: In step 6, once each variable is contracted, the process is repeated again for that same variable until no bound contraction takes place. Only then, the process moves to the next variable.

■ *Nonexhaustive elimination*: In step 6, once each variable is contracted once, the process moves to the next variable.

- Options related to the updating of the UB:

■ *Active upper bounding*: Each time elimination takes place, the upper bound is calculated again. This helps when the gap between lower and upper bound (feasible solution) improves too slowly.

■ *Active lower bounding*: Each time elimination takes place, the lower bound solution is obtained again. In such case, one would allow all surviving intervals, and partition them again. If the gap between LB and UB is within the tolerance one can terminate the entire procedure.

- Options related to the amount of intervals used for forbidding:

■ *Single-interval forbidding*: This consists of forbidding only the interval that brackets the solution;

■ *Extended interval forbidding*: This consists of forbidding the interval identified originally, plus a number of adjacent ones. This is efficient when a large number of intervals are used to obtain lower bounds. Adjacent intervals, if left not forbidden, may render lower bounds that are not larger than the current upper bound. Thus, by forbidding them, other intervals are forced to be picked and those may render larger LB and lead to elimination.

- Options related to the amount of variables that are forbidden:

■ *Single-variable elimination*: This procedure is the one outlined earlier.

■ *Collective elimination*: This procedure consists of forbidding the combination of the intervals identified in the lower bound. We anticipate having problems with this strategy when the size of the problem is large. Notice that we are not simply forbidding the intervals for each variable, just their combination, through an integer cut like the one used by Balas and Jeroslow.[41]

When no interval is eliminated and the lower bound-upper bound gap is still larger than the tolerance, one can resort to increase the number of intervals and start over. This procedure normally renders better lower bounds and more efficient eliminations when the *extended interval forbidding* is

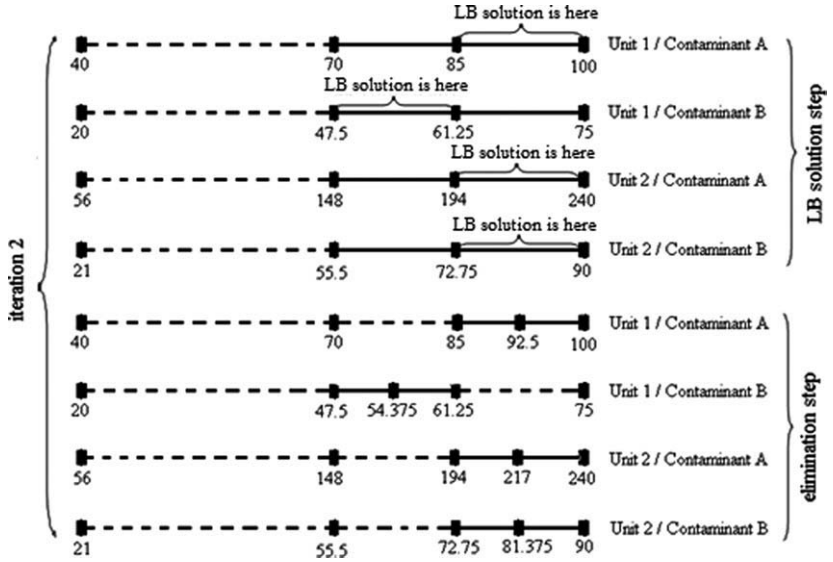

Figure 3. Illustrative example of the partitioning approach— 2nd iteration.

Table 2. Solution Progress of the Illustrative Example

Iteration	Lower Bound	Upper Bound	Relative error	Intervals eliminated
0	52.90 t/h	54.00 t/h	2.02%	NA
1	52.90 t/h	54.00 t/h	2.02%	4
2	52.90 t/h	54.00 t/h	2.02%	4
3	53.65 t/h	54.00 t/h	0.65%	4

applied. When our standard option, the *one-pass with one forbidden interval elimination* is used, an increase in the number of intervals will select a smaller part of the feasible range of each variable. Thus, increasing the number of intervals helps because it provides tighter lower bounds. However, a large number of intervals can also significantly increase the running time.

Branch and Bound Procedure

It is possible that the aforementioned interval elimination procedure fails to reduce the gap that is even using the maximum number of intervals, no interval eliminations are possible. In such a situation, we resort to a branch bound procedure. In many methods addressing bilinear terms directly,[30] the branching is normally done in the variable that is being partitioned. However, one can branch on the other or on both. In our case, we branch on the continuous variables by splitting their interval from lower to upper bound in two intervals.

For branching we use one of the following two criteria:

• The new continuous variable that is split in two is the one that has the largest deviation between the value of z_{ij}^{LB} in the parent node, and the product of the corresponding variables x_i^{LB} and y_j^{LB}, that is, choose the variable i that satisfies the following

$$ArgMax_i\left\{\left|z_{i,j}^{LB}-x_i^{LB}\,y_j^{LB}\right|\right\} \tag{41}$$

• Using information of the current upper bound solution: We do this by choosing the variable that contributes to the largest gap between z's from the lower and upper bound, that is, we choose the variable i that satisfies the following

$$ArgMax_i\left\{\left|z_{i,j}^{LB}-z_{i,j}^{UB}\right|\right\} \tag{42}$$

In addition to the B&B procedure, at each node we perform as many interval eliminations (bound contractions) as possible.

Implementation issues

Our methodology requires making several choices. These choices are:

• *Variables to be partitioned*. In water management and pooling problems these could be concentrations, flow rates, or both.

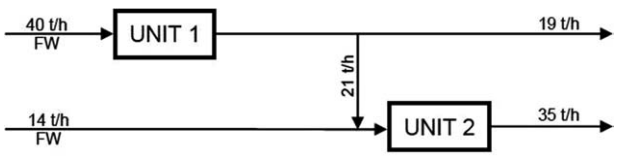

Figure 4. Optimum network of example 1.

Table 3. Solution Progress of the Illustrative Example – Using Cyclic Non-exhaustive Elimination

Iteration	Lower Bound	Upper Bound	Relative error	Number of cycles	Eliminations
0	52.90 t/h	54.00 t/h	2.02%	NA	NA
1	52.90 t/h	54.00 t/h	2.02%	4	10
2	53.67 t/h	54.00 t/h	0.62%	5	8

• *Number of intervals per variable*: It does not need to be the same for all variables.

• *LB model*: DPP1, DPP2, DPP3, MCP1, MCP2, or MCP3.

• *Variables chosen to perform bound contraction*: They need not be the same as the ones chosen to be partitioned. For example, one can partition concentrations and build a DPP1-LB model based on this partitioning, but perform bound contraction on flow rates. For this, one needs to partition the flow rates as well. The LB-Model, however, would not consider other than continuous flow rates, only including Eqs. 5 and 6 for flow rates to bracket the flow rate value and to be able to forbid it.

• *Elimination strategy*: The standard one-pass with one forbidden interval elimination, or the variants (one pass or cyclic elimination, exhaustive or not exhaustive elimination, active upper/lower bounding or not, single vs. extended intervals forbidding, or collective elimination,).

• *Variables to partition in the branch and bound procedure*.

With such a large amount of options, it is cumbersome to explore all of them. In our examples, when we report successful cases (i.e., those that seem to work fast and better than other procedures), we made no attempt to explore the possibility of other combinations being even more efficient computationally. We also make an effort to show some variant's success, even though they are less efficient. For the examples, for which the method is not as quick and efficient, we report the best we could achieve and mention a few of the failures.

Illustration of the Interval Elimination Procedure

Elimination procedure

We illustrate the details of the *one-pass with one forbidden interval elimination* procedure using a simple water network example from Wang and Smith.[42] This example optimizes only the water-using subsystem, which targets minimum freshwater consumption and has two water-using units and two contaminants. No regeneration unit exists in this example.

The nonlinear model used to describe this problem can be given by the following set of equations:

Water balance through the water-using units

$$\sum_w FWU_{w,u}+\sum_{u^*\neq u}FUU_{u^*,u}=\sum_s FUS_{u,s}+\sum_{u^*\neq u}FUU_{u,u^*}\qquad \forall u \tag{43}$$

Table 4. Number of Elimination in Each Cycle – Using Cyclic Non-exhaustive Elimination

Iteration	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5
1	4	3	2	1	NA
2	1	3	2	1	1

Table 5. Solution Progress of the Illustrative Example – Using One-pass Exhaustive Elimination

Iteration	Lower Bound	Upper Bound	Relative error	Eliminations
0	52.89 t/h	54.00 t/h	2.02%	NA
1	52.89 t/h	54.00 t/h	2.02%	10
2	53.67 t/h	54.00 t/h	0.62%	6

where $FWU_{w,u}$ is the flow rate from freshwater source w to the unit u, $FUU_{u*,u}$ is the flow rates between units u^* and u, $FUS_{u,s}$ is the flow rate from unit u and sink s.

Contaminant balance at the water-using units

$$\sum_{w}\left(CW_{w,c}\,FWU_{w,u}\right)+\sum_{u^*\neq u}ZUU_{u^*,u,c}+\Delta M_{u,c}=\sum_{u^*\neq u}ZUU_{u,u^*,c}+\sum_{s}ZUS_{u,s,c}\qquad\forall u,c\quad(44)$$

where $CW_{w,c}$ is concentration of contaminant c in the freshwater source w, $\Delta M_{u,c}$ is the mass load of contaminant c extracted in unit u, $ZUU_{u*,u,c}$ is the mass flow of contaminant c in the stream leaving unit u^* and going to unit u, and $ZUS_{u,s,c}$ is the mass flow of contaminant c in the stream leaving unit u and going to sink s.

Maximum inlet concentration at the water-using units

$$\sum_{w}\left(CW_{w,c}\,FWU_{w,u}\right)+\sum_{u^*\neq u}ZUU_{u^*,u,c}\leq C_{u,c}^{in,max}\left(\sum_{w}FUW_{w,u}+\sum_{u^*\neq u}FUU_{u^*,u}\right)\qquad\forall u,c\quad(45)$$

where $C_{u,c}^{in,max}$ is the maximum allowed concentration of contaminant c at the inlet of unit u.

Maximum outlet concentration at the water-using units

$$\sum_{w}\left(CW_{w,c}\,FWU_{w,u}\right)+\sum_{u^*}ZUU_{u^*,u,c}+\Delta M_{u,c}\leq C_{u,c}^{out,max}\left(\sum_{u^*}FUU_{u,u^*}+\sum_{u^*}FUU_{u,u^*}\right)\qquad\forall u,c\quad(46)$$

where $C_{u,c}^{out,max}$ is the maximum allowed concentration of contaminant c at the outlet of unit u.

The contaminant mass flow rates are enough and no new variables are needed in the case of mixing nodes. However, the splitting nodes at the outlet of each unit need constraints that will reflect that all these contaminant flows and the total mass flows are consistent with the concentrations of the different contaminants. Thus, we add the corresponding relations:

Contaminant mass flow rates

$$ZUU_{u,u^*,c}=FUU_{u,u^*}\,C_{u,c}^{out}\qquad\forall u,u^*,c\quad(47)$$

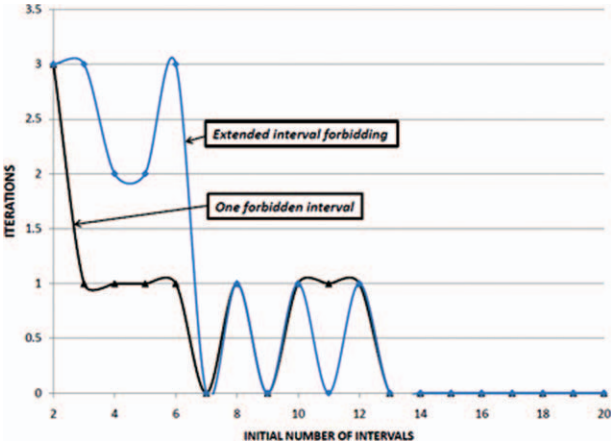

Figure 5. Influence of the number of initial intervals and the use of extended interval forbidding option—CPU time.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

$$ZUS_{u,s,c}=FUS_{u,s}\,C_{u,c}^{out}\qquad\forall u,s,c\quad(48)$$

where $C_{u,c}^{out}$ is the outlet concentration of contaminant c in unit u. Finally, we write

$$C_{u,c}^{out,Min}\leq C_{u,c}^{out}\leq C_{u,c}^{out,Max}\qquad\forall u,s,c\quad(49)$$

Objective functions (minimum freshwater consumption)

$$Min\;FW=\sum_{w}\sum_{u}FWU_{w,u}\quad(50)$$

Table 1 presents the limiting data of this problem.

For the illustration of this example, the concentration lower bound is used with two initial intervals (Figure 1), and the elimination procedure is applied on the outlet concentrations of the water-using units. The standard strategy (one-pass non-exhaustive elimination) is used.

In the upper part of Figure 2 the results of the lower bound using the preprocessed bounds, which corresponds to a value of 52.89 t/h are depicted. Using the results from this lower bound as initial points, the full problem was run and the solution obtained (54 t/h) corresponds to the first upper bound of the problem.

When the lower bound model is rerun forbidding the interval corresponding to unit 1/contaminant A, that is, the interval 70–100 ppm is forbidden, the interval from 40–70 ppm is eliminated because forcing the lower bound in this interval renders a value of the LB higher than 54 t/h. The remaining part (70–100 ppm) is partitioned again in two new intervals. Then the lower bound model is run forbidding the interval corresponding to unit 1/contaminant B, that is the interval 47.5–75 ppm. The solution is again higher than 54 t/h. Thus, the interval between 20 and 47.5 ppm is eliminated,

Table 6. Exhaustive Eliminations Progress of the Illustrative Example – Using One-pass Exhaustive Elimination

	Cout			
Iteration	Unit 1 Contaminant A	Unit 1 Contaminant B	Unit 2 Contaminant A	Unit 2 Contaminant B
1	4	2	1	3
2	-	1	3	2

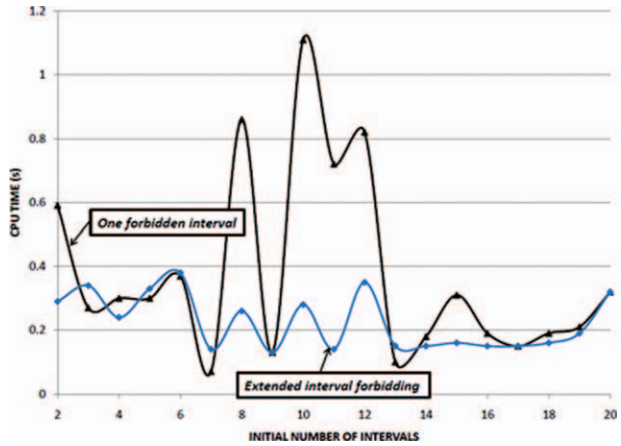

Figure 6. Influence of the number of initial intervals and the use of *extended interval forbidding* option— iterations.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

and the remaining is partitioned again. Applying this procedure to the rest of the variables renders eliminating the intervals shown in Figure 2.

After the first iteration the lower and upper bound do not change (LB = 52.90 t/h and UB = 54 t/h). The second iteration of the illustrative example is shown in Figure 3. The elimination procedure is repeated again, one variable at a time, and in all cases, the solutions found are larger than the current upper bound. Therefore, each time the corresponding interval in each variable is eliminated, the selected interval is partitioned again and the procedure moves to the next variable.

This procedure is repeated until the lower bound solution is equal (or has a given tolerance difference) to the upper bound solution. This illustrative example, using the DPP3 and partitioning concentrations in two intervals, is solved in three iterations, and 0.60 s (execution time only, not including preprocessing) using a relative tolerance of 1%. The actual solution reaches 0.65% gap.

Table 2 presents the progress of the solution through the iterations. The upper bound (54 t/h) is identified in the first iteration and is the global solution. The lower bound solution, however, does not improve until the third iteration. The optimum network of this example is presented in Figure 4.

The other option for the elimination step is cyclic nonexhaustive elimination. Tables 3 and 4 show the progress of the solution when the cyclic nonexhaustive elimination is applied.

Even though this procedure takes a smaller number of iterations, the overall running time for this example was higher (2.26 s against 0.60 s using the one-pass elimination). This is expected because this is a small problem, in which the lower bounding (step 2) is not computationally expensive. Thus, unnecessary elimination (more than the needed to achieve the given tolerance gap) may occur if the lower bound is not often verified.

The solution using one-pass exhaustive elimination is also investigated. Table 5 shows the progress of the iterations and Table 6 shows which variable had its bounds contracted and how many eliminations existed in each iteration. This strategy took 1.30 s.

Effect of the Number of Intervals

The number of initial intervals has also influence on the performance of the proposed methodology. We illustrate this for the simple aforementioned example, and we do not claim this ought to be taken as a general conclusion. Since it is known that a continuous variable can be substituted by discrete values when the number of discrete values goes to infinity, it is expected that less iterations are needed when more intervals are added. On the other hand, this generates a higher number of integer variables (what means a larger MILP model), and might make the problem computationally very expensive (increase the overall time to run it).

This influence is analyzed only for the cases of one-pass nonexhaustive elimination, which have presented the best option when only two intervals are considered. Additionally, the influence of the *extended interval forbidding* option is also verified. This option represents two main advantages: reduce the number of binary in the elimination step; and, facilitate eliminations. On the other hand, when only one interval is forbidden and elimination takes place, the disregarded portion of the variable is larger than if the *extended interval forbidding* option was used, and the stopping criteria is when the tolerance is satisfied. We performed an analysis of the influence of the number of intervals on CPU time (Figure 5), and the number of iterations (Figure 6) for this example (no branch and bound iterations were needed here).

Table 7. Summary of the Best Results for the Water Networks

Example	Original Solution	Our Global Solution (1% tolerance)	Iterations	Time***
1 – Wang and Smith (1994)	54.00 t/h	54.00 t/h	0	0.07 s
2 – Wang and Smith (1994)	55.50 t/h	55.47 t/h	0	0.1 s
3 – Koppol et al. (2003)	119.33 t/h	119.33 t/h	0	0.14 s
4 – Koppol et al. (2003) – NLP	33.57 t/h	33.57 t/h	0	0.56 s
4 – Koppol et al. (2003) – MINLP	33.57 t/h	33.57 t/h	1	75.71 s (1 m 15.71 s)
5 – Karuppiah and Grossmann (2006)*	117.5 t/h (37.72 s)	117.05 t/h	0	1.57 s
6 – Karuppiah and Grossmann (2006)*	$381,751.35 (13.21 s/3.75 s**)	$381,751.35	0	0.41 s
7 – Karuppiah and Grossmann (2006)*	$874,057.37 (0.9 s)	$874,057.37	0	0.25 s
8 – Karuppiah and Grossmann (2006)*	$1,033,810.95 (231.37 s)	$1,033,810.95	1	30.15 s
8 – Karuppiah and Grossmann (2006) – MINLP	N/A	$1,033,859.85	1	73.79 s (1 m 13.79 s)
9 – Faria and Bagajewicz (2009b)	$410,277	$410,277	1	3.00 s
10 – Alva-Argaez et al. (2007)	$616,824	$578,183	62	57,960 s (16 h 6 m) (Baron solves in 7 h 5 m 13 s with 1% relative gap)

*Problem originally solved for global optimality.
**The second time reported corresponds to Bergamini et al. (2008).
***We show the Execution time only.

Table 8. Summary of the Options Tried in Each Example

Example	Variable Partitioned/ Intervals	LB Model	Variables for Bound Contraction	Elimination Strategy/ Active Bounding	Variables for Branch and Bound	Time**
1 – Wang and Smith (1994)	Concent. 2 intervals	DPP3	Concent.	One-pass non-exhaustive	Not needed	0.6 s
	Concent. 2 intervals	DPP3	Concent.	Cyclic non-exhaustive	Not needed	2.26 s
	Concent. 2 interval	DPP3	Concent.	One-pass exhaustive	Not needed	1.30 s
	Concent. 7 intervals	DPP3	Concent.	One-pass non-exhaustive one forbidden interval	Not needed	0.07 s
	Concent. (9,11,13,18) intervals	DPP3	Concent.	One-pass non-exhaustive Extended interval forbidding	Not needed	0.15 s
2 – Wang and Smith (1994)- Refinery Example	Concent. 1 interval	DPP2	Not needed Solved at root node	Not needed	Not needed	0.10 s
	Flowrate 1 interval	DPP2	Not needed Solved at root node	Not needed	Not needed	0.16 s
3 – Koppol et al. (2003)	Flowrates 2 intervals	DPP2	Not needed Solved at root node	One-pass non-exhaustive	Not needed	10.67 s
	Flowrates 1 intervals	MCP2	Not needed Solved at root node	Not needed	Not needed	0.19 s
	Concent. 1 interval	DPP2	Not needed Solved at root node	Not needed	Not needed	0.17 s
	Concent. 1 interval	MCP2	Not needed Solved at root node	Not needed	Not needed	0.14 s
4 – Koppol et al. (2003) – NLP	Flowrates 1 intervals	MCP2	Not needed Solved at root node	Not needed	Not needed	0.53 s
	Concent. 1 interval	DPP2	Not needed Solved at root node	Not needed	Not needed	0.57 s
	Concent. 1 interval	MCP2	Not needed Solved at root node	Not needed	Not needed	0.56 s
4 – Koppol et al. (2003) – MINLP	Concent. 2 intervals	MCP2	Concent.	One-pass non-exhaustive active upper bounding	Not Needed	75.71 s (1 m 15.71 s)
5 – Karuppiah and Grossmann (2006)*	Concent. 10 intervals	MCP2	Concent.	Not needed	Not Needed	1.57 s
	Concent. 10 intervals	DPP2	Concent.	Not needed	Not Needed	1.59 s
	Flowrates 2 intervals	MCP2	Flowrates	One-pass non-exhaustive one forbidden interval	Not Needed	11,795 s
	Flowrates 2 intervals	MCP2	Flowrates	One-pass non-exhaustive one forbidden interval	Concent.	23.73 s
	Flowrates 2 intervals	MCP2	Flowrates	One-pass non-exhaustive one forbidden interval	Flowrates	40.93 s
6 – Karuppiah and Grossmann (2006)*	Concent. 4 intervals	MCP2	Concent. Regeneration flowrates	Not needed	Not Needed	0.41 s
7 – Karuppiah and Grossmann (2006)*	Concent. 2 intervals	DPP2 DPP3 MCP2 MCP3	Concent.	Not needed	Not Needed	0.25 s
8 – Karuppiah and Grossmann (2006)*	Concent. 2 intervals	MCP2	Concent. Regeneration flowrates	one-pass extended interval forbidding exhaustive elimination active upper bounding	Not Needed	30.15 s
8 – Karuppiah and Grossmann (2006) – MINLP	Concent. 2 intervals	MCP2	Concent. Regeneration flowrates	one-pass extended interval forbidding exhaustive elimination active upper bounding	Not Needed	73.79 s (1 m 13.79 s)
9 – Faria and Bagajewicz (2009b)	Concent. Reg. Flows Units Flows 2 intervals for each one	MCP2	Concent. Regeneration flowrates	one-pass guided exhaustive active upper bounding active lower bounding	Not Needed	3.00 s
10 – Alva-Argaez et al. (2007)	Concent.: 2 intervals All Flows: 5 intervals	MCP2	All Concentrations. All flowrates	one-pass guided exhaustive active upper bounding active lower bounding	Concentrations and All flowrates	57,960 s (16 h 6 m)

*We show the Execution time only.

Table 9. Summary of the Results for the Pooling Problems

Example	Original Solution	Global Solution	Option	Time
11a – Adhya et al. (1999)	549.80 (425 s/1.75 s*/24 s**)	549.80	pq-formulation MCP2-C	0.24 s
11b – Adhya et al. (1999)	549.80 (1115 s/1.49 s*/27 s**)	549.80	pq-formulation MCP2-C	0.30 s
12 – Adhya et al. (1999)	561.04 (19314 s/0.79 s*/10 s**)	561.04	pq-formulation DPP2-C	0.19 s
13 – Adhya et al. (1999)	877.65 (183 s/0.26 s*/20 s**)	877.65	pq-formulation MCP3-C	0.09 s

*Time reported corresponding to Tawarmalani and Sahinidis (2002).
**Time reported corresponding to Pham et al.(2009).

The results are shown for illustration purposes only and one should not draw general conclusions from them.

For the *one-pass with one forbidden interval elimination* option, the quickest solution (0.07 s) is found when the procedure is initialized with seven intervals. This is the point in which the solution is first found at the root node. For the *extended interval forbidding* case, very similar CPU times are found for the cases in which the solution is found at the root node (7, 9, 11, and 13–18 intervals), that is, computational times of approximately 0.15 s.

Although one would expect the lower bound to be higher when the number of intervals increases, we have observed occasional ups and downs. Gounaris et al.[26] explained that increasing number of intervals generally corresponds to tighter bounds, but cuts are not necessarily conserved with increasing intervals. One should always see monotonic trends in the tightness of relaxations with multiples of intervals (i.e., four intervals should always be at least as tight as two intervals).

Results

We tested the method using several water management, pooling and generalized pooling problems. In all cases we identified the global optimum and we summarize here only a few aspects of the results. Full answers (flow sheets and other details) are included in a separate publication.

Tables 7 and 8 summarizes the results for water management examples, and Table 9 shows the results of pooling problems. Finally, Table 10 summarizes the results obtained for one difficult generalized pooling problem.[23] As termination criteria we used 1% gap. We used GAMS/CPLEX (CPLEX version 10.1) on a one core 2.8 GHz workstation (PC) using Windows. The whole partition algorithm was implemented in GAMS. Although we include comparisons with the time used by other methods we warn the reader that these are not times obtained using the same software and hardware and are for information purposes only.

For water management problems, it seems that using the larger number of intervals possible reduces the number of iterations when the problems are relatively small, which many times find the solution at the root node. This observation is not necessarily related to the size of the problem, but the tightness of the lower bound model. Note that example 8 and 10 have the same size, however, the latter showed to be much more difficult to solve for global optimality, where a poor lower bound is generated. In terms of problem definition, the latter case is a more detailed model than the former. Example 10 assumes water-using units with variable flow rates and also considers cost of connection with a fixed and a variable part, which is what, makes the feasible space considerable larger. Degeneracy is also an issue. The procedure

Table 10. Summary of the Options Tried for the Generalized Pooling Example (Meyer and Floudas, 2006)

Variable Partitioned (Intervals)	LB Model	Variables for Bound Contraction	Bound Contraction Settings	Variables for Branch and Bound	Time** (CPUs)	Number of sub-problems analyzed	Sub-problem where the optimum is identified
Concentrations (2 intervals)	MCP2	Concentrations (2 intervals) Regeneration processes flowrates (2 intervals)	Guided One pass Exhaustive UB updating	Connections and regeneration processes flowrates	16,336 (4 h 32 m 16 s)	16	1st
Concentrations (2 intervals)	MCP2	Concentrations (2 intervals)	Guided One pass Exhaustive UB updating	Connections and regeneration processes flowrates	25,722 (7 h 8 m 42 s)	34	8th
Concentrations (2 intervals)	MCP2	Concentrations (2 intervals) Regeneration processes flowrates (2 intervals)	Guided One pass Exhaustive UB updating LB updating	Connections and regeneration processes flowrates	21,420 (5 h 57 m),	16	2nd
Concentrations (2 intervals)	MCP2	Concentrations (2 intervals)	Guided One pass Exhaustive UB updating LB updating	Connections and regeneration processes flowrates	28,590 (7 h 56 m 30 s)	34	9th
Flowrates (2 intervals)	MCP2	Connections flowrates (2 intervals) Regeneration processes flowrates (2 intervals)	Guided One pass Exhaustive UB updating	Connections and regeneration processes flowrates	28,930 (8 h 2 m 10 s)	28	2nd

to find all these solutions is presented in detail by Faria and Bagajewicz.[43] This explains why bound contracting on the regeneration processes flow rates is enough.

Additionally, in some problems we observed that when concentration is partitioned, the LB of the *direct partitioning* is as tight as the *McCormick's envelopes*, and partitioning of concentrations normally generates tighter lower bounds than partitioning of flow rates.

As a strategy, for large problems, it seems that one should always start increasing the partitioning (larger number of intervals). If the solution is not found with the increase of number of intervals, an evaluation of the improvement behavior on each of the variables can be performed before it is decided which set of variables should be bound contracted and/or branched.

The results obtained when applying the presented methodology to pooling problems have shown that not only the GO method is an important factor, but also how the problem is modeled. Additionally, as in the WAP, the partitioned variable choice also has influence on the performance of the method.

For the generalized pooling problem Meyer and Floudas[23] present a method to obtain a lower bound, through reformulation and partitioning, but no systematic procedure to reduce the gap. To obtain upper bounds, they perform an extensive search using DICOPT with random initial values. They use 1000 runs of which 119 identify the same best answer with a cost of \$1,091,160, but they do not report the time this exercise takes. However, they report that the best known solution has an objective function value of \$1,086,430, but they do not provide the source. Using their procedure, they found a lower bound solution, which has a 1.2% gap with this given best known solution in 285,449 CPUs (79 h 17 m 29 s). The gap is slightly larger if one uses the upper bound obtained using DICOPT. This apparent success does not guarantee that the lower bound will be always this close for other problems. More modern results of this problem obtained using CPLEX v. 12 on a four-core, 2.83 GHz Intel Core 2 Quad processor indicate improvements of the order of 20–100 times depending on the number of intervals with respect to the 2006 reported times.[44]

In turn, after 120 h, the best solution found by BARON (run with 1% relative gap termination criteria) was \$1,107,905, but at this time the gap was 28.8%, so we stopped the run. Minimizing the total cost using our method, we obtained the network that has a total cost of \$1,086,187. The lowest time that the solution was obtained is 16,336 CPUs (4 h 32 m 16 s), which is substantially lower than the time used by BARON (see previous example), and the time needed to run the lower bound model proposed by Meyer and Floudas,[23] although they have been able to improve their original time substantially.

Conclusions

We presented the theory of a new bound contraction method that uses discrete portions of the feasible region (following Meyer and Floudas,[23] Wicaksono and Karimi,[28] and Pham et al.[37]) to identify if they can be part of the global solution or not. This procedure is used to contract the bounds of one (or both) variables participating in bilinearities. After the contraction, further partitioning leads to a tighter lower bound model.

When bound contraction is exhausted, the procedure can resorts to one of the following alternatives: increased partitioning of the lower bound model, or, creating subproblems using a branch and bound scheme.

We finally presented several results for water management and pooling problems that illustrate the effectiveness of the method.

Acknowledgment

Débora Faria acknowledges support from the CAPES/Fulbright Program (Brazil).

Notation

x = variable
y = variable
z = bilinear or concave function
x^L = lower bound of variable x
x^U = upper bound of variable x
y^L = lower bound of variable y
y^U = upper bound of variable y
$\hat{y}_d$ = discrete value of y
v_d = binary variable
w_d = auxiliary variable of x
Ω = constant
α = value between 0 and 1
β_d = auxiliary variable of y
$z_{i,j}^{LB}$ = value of $z_{i,j}$ found by the lower bound model
x_i^{LB} = value of y_i found by the lower bound model
y_j^{LB} = value of $z_{i,j}$ found by the lower bound model
$z_{i,j}^{UB}$ = value of found by the upper bound model
$FWU_{w,u}$ = flow rate from freshwater source w to the unit
$FUU_{u*,u}$ = flow rates between units u* and u
$FUS_{u,s}$ = flow rate from unit u and sink s
$CW_{w,c}$ = concentration of contaminant c in the freshwater source w
$\Delta M_{u,c}$ = mass load of contaminant c extracted in unit u
$ZUU_{u*,u,c}$ = mass flow of contaminant c in the stream leaving unit u* and going to unit u
$ZUS_{u,s,c}$ = mass flow of contaminant c in the stream leaving unit u and going to sink s
$C_{u,c}^{in,max}$ = maximum allowed concentration of contaminant c at the inlet of unit u
$C_{u,c}^{out,max}$ = maximum allowed concentration of contaminant c at the outlet of unit u
$C_{u,c}^{out}$ = outlet concentration of contaminant c in unit u
FW = total freshwater consumption
$FWU_{w,u}$ = freshwater consumption of unit u

Literature Cited

1. Sahinidis NV. BARON: A general purpose global optimization software package. *J Global Optimiz*. 1996;7(4):337–363.
2. Androulakis P, Maranas CD, Floudas CA. αBB: A global optimization method for general constrained nonconvex problems. *J Global Optimiz*. 1995;7(4):337–363.
3. Sherali HD, Adams WP. *A Reformulation-Linearization Technique for solving Discrete and Continuous Nonconvex Problems. Nonconvex Optimization and its Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers; 1999.
4. Floudas CA. *Deterministic Global Optimization: Theory, Methods and Applications. Nonconvex Optimization and Its Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers; 2000.
5. Tawarmalani M, Sahinidis NV. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Nonconvex Optimization and its Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers; 2002.
6. Horst R, Tuy H. *Global Optimization: Deterministic Approaches*. New York, NY: Springer-Verlag; 2003.
7. Hansen E, Walster GW. *Global Optimization Using Interval Analysis. Pure and Applied Mathematics*. New York, NY: Marcel Dekker; 2004.
8. Floudas CA. Global optimization in design and control of chemical process systems. *J Process Control*. 2000;10:125–134.
9. Pardalos PM, Romeijn HE, Tuy H. Recent developments and trends in global optimization. *J Comput Appl Math*. 2000;124(1–2):209–228.
10. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J. Global optimization in the 21st century: advances and challenges. *Comput Chem Eng*. 2005;29(6):1185–1202.

11. Floudas CA, Gounaris CE. A review of recent advances in global optimization. *J Global Optimiz*. 2009;45:3–38.
12. Ben-Tal A, Eiger G, Gershovitz V. Global minimization by reducing the duality gap. *Math Program*. 1994;63:193–212.
13. Adhya N, Tawarmalani M, Sahinidis NV. A Lagrangian approach to the pooling problem. *Ind Eng Chem Res*. 1999;38:1956–1972.
14. Kuno T, Utsunomiya T. A Lagrangian based branch-and-bound algorithm for production-transportation problems. *J Global Optimiz*. 2000;18:59–73.
15. Karrupiah R, Grossmann IE. A Lagrangian based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. *J Global Optimiz*. 2008;41:1573–2916.
16. Ruiz JP, Grossmann IE. Strengthening of lower bounds in the global optimization of bilinear and concave generalized disjunctive programs. *Comput Chem Eng*. 2010;34(6):914–930.
17. Moore RE. *Interval Analysis*. Upper Saddle River, NJ: Prentice-Hall; 1966.
18. Vaidyanathan R, El-Halwagi M. Global Optimisation of Nonconvex Programs via Interval Analysis. *Comput Chem Eng*. 1994;18:889–897.
19. Hansen ER. Global optimization using interval analysis: The one-dimensional case. *J Optimiz Theory Appl*. 1979;29:331–344.
20. Ratschek H, Rokne J. *New computer methods for global optimization*. New York, NY: Halsted Press; 1988.
21. Moore R, Hansennd ER, Leclerc A. In: Floudas CA, Pardalos PM, editors. *Logic Based Outer Approximation for Global Optimization of Synthesis of Process Networks*. Princeton, NJ: Princeton University Press, 1992:321.
22. Zamora JM, Grossmann IE. A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. *J Global Optimiz*. 1999;14:217–249.
23. Meyer CA, Floudas CA. Global optimization of a combinatorially complex generalized pooling problem. *AIChE J*. 2006;52(3):1027–1037.
24. Sherali HD, Alamedinne A. A new reformulation-linearization technique for bilinear programming problems. *J Global Optimiz*. 1992;2(4):379–410.
25. Misener R, Floudas CA. Advances for the pooling problem: modeling, global optimization, and computational studies. *Appl Comput Math*. 2009;8(1):3–22.
26. Gounaris CE, Misener R, Floudas CA. Computational comparison of piecewise-linear relaxations for poolings. *Ind Eng Chem Res*. 2009;48:5742–5766.
27. Misener R, Floudas CA. Global optimization of large-scale generalized pooling problems: Quadratically constrained MINLP models. *Ind Eng Chem Res*. 2010;49:5424–5438.
28. Wicaksono DA, Karimi IA. Piecewise MILP under- and overestimators for global optimization of bilinear programs. *AIChE J*. 2008;54(4):991–1008.
29. Misener R, Gounaris CE, Floudas CA. Mathematical modeling and global optimization of large-scale extended pooling problems with the (EPA) complex emissions constraints. *Comput Chem Eng*. 2010;34:1432–1456.
30. Karuppiah R, Grossmann IE. Global optimization for the synthesis of integrated water systems in chemical processes. *Comput Chem Eng*. 2006;30:650–673.
31. McCormick GP. Computability of global solutions to factorable nonconvex programs. *Math Program*. 1976;10:146–175.
32. Karuppiah R, Grossmann IE. Global optimization of multiscenario mixed integer nonlinear programming models arising in the synthesis of integrated water networks under uncertainty. *Comput Aided Chem Eng*. 2006;21–2:1747–1752.
33. Bergamini ML, Aguirre P, Grossmann IE. Logic based outer approximation for global optimization of synthesis of process networks. *Comput Chem Eng*. 2005;29:1914–1933.
34. Bergamini ML, Grossmann IE, Scenna N, Aguirre P. An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. *Comput Chem Eng*. 2008;32:477–493.
35. Padberg M. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Res Lett*. 2000;27:1–5.
36. Hasan MMF, Karimi IA. Piecewise linear relaxation of bilinear programs using bivariate partitioning. *AIChE J*. 2010;56(7):1880–1893.
37. Pham V, Laird C, El-Halwagi M. Convex hull discretization approach to the global optimization of pooling problems. *Ind Eng Chem Res*. 2009;48(4):1973–1979.
38. Faria DC, Bagajewicz MJ. A new approach for the design of multicomponent water/wastewater networks. *Comput Aided Chem Eng*. 2008;25:319–324.
39. Saif Y, Elkamel A, Pritzker M. Global optimization of reverse osmosis network for wastewater treatment and minimization. *Ind Eng Chem Res*. 2008;47(9):3060–3070.
40. Faria D, Bagajewicz M. Global optimization of water management problems using linear relaxations and bound contraction methods. *Ind Eng Chem Res*. 2011;50(7):3738–3753.
41. Balas E, Jeroslow R. Canonical cuts on the unit hypercube. *SIAM J Appl Math*. 1972;23:61–79.
42. Wang YP, Smith R. Wastewater minimization. *Chem Eng Sci*. 1994;49(7):981–1006.
43. Faria DC, Bagajewicz MJ. On the degeneracy of the water/wastewater allocation problem in process plants. *Ind Eng Chem Res*. 2010;49(9):4340–4351.
44. In conversation with C. A. Floudas on Nov. 2009 (2009).

Appendix

In this appendix we write the equations for the partitioning of both variables in a bilinear terms. For these we use we use binary variables v_{d_y} for y and r_{d_x} for x and variables w_{d_y} for the product xv_{d_y} and s_{d_y} for the product yr_{d_x}

For all DPP1, DPP2 and DPP3, we have

$$\sum_{d_y=1}^{D_y-1} \hat{y}_{d_y} v_{d_y} \le y \le \sum_{d_y=1}^{D_y-1} \hat{y}_{d_y+1} v_{d_y} \tag{A1}$$

$$\sum_{d_y=1}^{D_y} v_{d_y} = 1 \tag{A2}$$

$$z \le \sum_{d_y=1}^{D_y-1} \hat{y}_{d_y+1} w_{d_y} \tag{A3}$$

$$z \ge \sum_{d_y=1}^{D_y-1} \hat{y}_{d_y} w_{d_y} \tag{A4}$$

$$\sum_{d_x=1}^{D_x-1} \hat{x}_{d_x} r_{d_x} \le x \le \sum_{d_x=1}^{D_x-1} \hat{x}_{d_x+1} r_{d_x} \tag{A5}$$

$$\sum_{d_x=1}^{D_x} r_{d_x} = 1 \tag{A6}$$

$$z \le \sum_{d_x=1}^{D_x-1} s_{d_x} \hat{x}_{d_x+1} \tag{A7}$$

$$z \le \sum_{d_x=1}^{D_x-1} s_{d_x} \hat{x}_{d_x} \tag{A8}$$

Additional equations for DPP1

$$w_{d_y} - x^U v_{d_y} \le 0 \tag{A9}$$

$$(x - w_{d_y}) - x^U (1 - v_{d_y}) \le 0 \tag{A10}$$

$$x - w_{d_y} \ge 0 \tag{A11}$$

$$r_{d_x} - y^U s_{d_x} \le 0 \tag{A12}$$

$$(y - s_{d_y}) - y^U (1 - r_{d_y}) \le 0 \tag{A13}$$

$$y - s_{d_y} \ge 0 \tag{A14}$$

Additional equations for DPP2

$$w_{d_y} \le x^U v_{d_y} \quad \forall d_y = 1..D_y - 1 \tag{A15}$$

$$w_{d_y} \ge x^L v_{d_y} \quad \forall d_y = 1..D_y - 1 \tag{A16}$$

$$x = \sum_{d_y=1}^{D_y-1} w_{d_y} \tag{A17}$$

$$s_{d_x} \le y^U r_{d_x} \quad \forall d_x = 1..D_x - 1 \tag{A18}$$

$$s_{d_x} \ge y^L r_{d_x} \quad \forall d_x = 1..D_x - 1 \tag{A19}$$

$$y = \sum_{d_x=1}^{D_x-1} s_{d_x} \tag{A20}$$

Additional equations for DPP3

$$z \le x\,\hat{y}_{d_x+1} + x^U\left(y^U - \hat{y}_{d_x+1}\right)(1 - v_{d_x}) \quad \forall d_x = 1..D_x - 1 \tag{A21}$$

$$z \ge x\,\hat{y}_{d_x} - x^U \hat{y}_{d_x}(1 - v_{d_x}) \quad \forall d_x = 1..D_x - 1 \tag{A22}$$

$$z \le x^U y \tag{A23}$$

$$z \le \hat{x}_{d_y+1} y + x^U\left(y^U - \hat{x}_{d_y+1}\right)\left(1 - r_{d_y+1}\right) \quad \forall d_y = 1..D_y - 1 \tag{A24}$$

$$z \ge \hat{x}_{d_y} y - x^U \hat{x}_{d_y}\left(1 - r_{d_y}\right) \quad \forall d_y = 1..D_y - 1 \tag{A25}$$

$$z \le y^U x \tag{A26}$$

There is an alternative approach that relies on introducing a variable that will be one if both intervals are chosen. For example, DPP1 one would write

$$z \le \sum_{d_x=1}^{D_x-1} \sum_{d_y=1}^{D_y-1} \hat{x}_{d_x+1}\ \hat{y}_{d_y+1} \zeta_{d_x,d_y} \tag{A27}$$

$$z \ge \sum_{d_x=1}^{D_x-1} \sum_{d_y=1}^{D_y-1} \hat{x}_{d_x}\,\hat{y}_{d_y}\,\zeta_{d_x,d_y} \tag{A28}$$

$$\sum_{d_x,d_y} \zeta_{d_x,d_y} = 1 \quad \forall d_x, d_y \tag{A29}$$

$$\zeta_{d_x,d_y} \le r_{d_x} \quad \forall d_x, d_y \tag{A30}$$

$$\zeta_{d_x,d_y} \le v_{d_y} \quad \forall d_x, d_y \tag{A31}$$

where ζ_{d_x,d_y} can be continuous. Clearly this introduces an additional fairly large number of new variables, which we believe may not be the only disadvantage, as the lower bound is also less tight than the alternative (Eqs. A9–14).

The McCormick double partitioning schemes (MCP1 and MCP2) make use of the cross variable selecting variable ζ_{d_x,d_y}. For MCP1, we write

$$z \ge \sum_{d_x=1}^{D-1} (\hat{x}_{d_x} s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y}\right) - \sum_{d_x=1} \sum_{d_y=1} \left(\hat{x}_{d_x}\,\hat{y}_{d_y} \zeta_{d_x,d_y}\right) \tag{A32}$$

$$z \ge \sum_{d_x=1}^{D-1} (\hat{x}_{d_x+1} s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y+1}\right) - \sum_{d_x=1}^{D-1} \sum_{d_y=1}^{D-1} \left(\hat{x}_{d_x+1}\,\hat{y}_{d_y+1} \zeta_{d_x,d_y}\right) \tag{A33}$$

$$z \le \sum_{d_x=1}^{D-1} (\hat{x}_{d_x+1}\, s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y}\right) - \sum_{d_x=1} \sum_{d_y=1} \left(\hat{x}_{d_x+1}\,\hat{y}_{d_y} \zeta_{d_x,d_y}\right) \tag{A34}$$

$$z \le \sum_{d_x=1}^{D-1} (\hat{x}_{d_x}\, s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y+1}\right) - \sum_{d_x=1} \sum_{d_y=1} \left(\hat{x}_{d_x}\,\hat{y}_{d_y+1} \zeta_{d_x,d_y}\right) \tag{A35}$$

$$\sum_{d_x,d_y} \zeta_{d_x,d_y} = 1 \quad \forall d_x, d_y \tag{A36}$$

$$\zeta_{d_x,d_y} \le r_{d_x} \quad \forall d_x, d_y \tag{A37}$$

$$\zeta_{d_x,d_y} \le v_{d_y} \quad \forall d_x, d_y \tag{A38}$$

$$w_{d_y} - x^U v_{d_y} \le 0 \tag{A39}$$

$$(x - w_{d_y}) - x^U(1 - v_{d_y}) \le 0 \tag{A40}$$

$$x - w_{d_x} \ge 0 \tag{A41}$$

$$s_d - y^U r_{d_y} \le 0 \tag{A42}$$

$$(y - s_{d_y}) - y^U(1 - r_{d_y}) \le 0 \tag{A43}$$

$$y - s_{d_y} \ge 0 \tag{A44}$$

For MCP2 we use Eqs. A32–38, and replace A39–44 by

$$w_{d_x} \le x^U v_{d_x} \quad \forall d_x = 1..D_x - 1 \tag{A45}$$

$$w_{d_x} \ge x^L v_{d_x} \quad \forall d_x = 1..D_x - 1 \tag{A46}$$

$$x = \sum_{d_x=1}^{D-1} w_{d_x} \tag{A47}$$

$$k_{d_y} \le y^U v_{d_y} \quad \forall d_y = 1..D_y - 1 \tag{A48}$$

$$k_{d_y} \ge y^L v_{d_y} \quad \forall d_y = 1..D_y - 1 \tag{A49}$$

$$y = \sum_{d_y=1}^{D-1} k_{d_y} \tag{A50}$$

An alternative scheme without the cross variable ζ_{d_x,d_y} can be constructed as follows: For MCP1

$$z \ge \sum_{d_x=1}^{D-1} (\hat{x}_{d_x} s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y}\right) - \sum_{d_y=1} \left(t_{d_y}\,\hat{y}_{d_y}\right) \tag{A51}$$

$$z \geq \sum_{d_x=1}^{D-1} (\hat{x}_{d_x+1} s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y+1} \right) - \sum_{d_y=1} \left(t_{d_y+1} \hat{y}_{d_y+1} \right) \quad \text{(A52)}$$

$$z \leq \sum_{d_x=1}^{D-1} (\hat{x}_{d_x+1} s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y} \right) - \sum_{d_y=1} \left(q_{d_y} \hat{y}_{d_y} \right) \quad \text{(A53)}$$

$$z \leq \sum_{d_x=1}^{D-1} (\hat{x}_{d_x} s_{d_x}) + \sum_{d_y=1}^{D-1} \left(w_{d_y} \hat{y}_{d_y+1} \right) - \sum_{d_y=1} \left(q_{d_y+1} \hat{y}_{d_y+1} \right) \quad \text{(A54)}$$

where t_{d_y} is given by

$$t_{d_y} \leq \sum_{d_x=1}^{D-1} (\hat{x}_{d_x} r_{d_x}) + \Gamma v_{d_y} \quad \text{(A55)}$$

$$t_{d_y} \geq \sum_{d_x=1}^{D-1} (\hat{x}_{d_x} r_{d_x}) - \Gamma(1 - v_{d_y}) \quad \text{(A56)}$$

$$t_{d_y} \leq \hat{y}_{d_y} v_{d_y} \quad \text{(A57)}$$

$$t_{d_y} \leq \hat{x}_{d_x} r_{d_x} \quad \text{(A58)}$$

$$t_{d_y} \geq 0 \quad \text{(A59)}$$

In these equations, Γ is a sufficiently large number. Similar equations can be written for

$$q_{d_y} \leq \sum_{d_x=1}^{D-1} \left(\hat{x}_{d_x+1} r_{d_x} \right) + \Gamma v_{d_y} \quad \text{(A60)}$$

$$q_{d_y} \geq \sum_{d_x=1}^{D-1} \left(\hat{x}_{d_x+1} r_{d_x} \right) - \Gamma(1 - v_{d_y}) \quad \text{(A61)}$$

$$q_{d_y} \leq \hat{y}_{d_y} v_{d_y} \quad \text{(A62)}$$

$$q_{d_y} \leq \hat{x}_{d_x+1} r_{d_x} \quad \text{(A63)}$$

$$q_{d_y} \geq 0 \quad \text{(A64)}$$

Equations A55–58 can also be replaced by

$$t_{d_x} \leq \hat{x}_{d_x} r_{d_x} + \Gamma(2 - v_{d_y} - r_{d_x}) \quad \text{(A65)}$$

$$t_{d_x} \geq \hat{x}_{d_x} r_{d_x} - \Gamma(2 - v_{d_y} - r_{d_x}) \quad \text{(A66)}$$

Similar substitutions can be made for Eqs. A60–A-61. We omit showing the alternative equations for MCP2, which use a similar scheme than the one for MCP1.

Finally, for MCP3 we use

$$z \geq \hat{x}_{d_x} y + x \hat{y}_{d_y} - \hat{x}_{d_x} \hat{y}_{d_y} - \left(\hat{x}_{d_x} \hat{y}_{d_y+1} + \hat{x}_{d_x+1} \hat{y}_{d_y} - \hat{x}_{d_x} \hat{y}_{d_y} \right) \left(2 - r_{d_x} - v_{d_y} \right) \quad \forall d_x = 1..D_x, d_y = 1..D_y \quad \text{(A67)}$$

$$z \geq \hat{x}_{d_x+1} y + x \hat{y}_{d_y+1} - \hat{x}_{d_x+1} \hat{y}_{d_y+1} - \left(\hat{x}_{d_x+1} \hat{y}_{d_y+1} + \hat{x}_{d_x+1} \hat{y}_{d_y+1} - \hat{x}_{d_x+1} \hat{y}_{d_y+1} \right) \left(2 - r_{d_x} - v_{d_y} \right) \quad \forall d_x = 1..D_x, d_y = 1..D_y \quad \text{(A68)}$$

$$z \leq \hat{x}_{d_x+1} y + x \hat{y}_{d_y} - \hat{x}_{d_x+1} \hat{y}_{d_y} + \left(\hat{x}_{d_x+1} \hat{y}_{d_y+1} + \hat{x}_{d_x+1} \hat{y}_{d_y} \right) \left(2 - r_{d_x} - v_{d_y} \right) \quad \forall d_x = 1..D_x, d_y = 1..D_y \quad \text{(A69)}$$

$$z \geq \hat{x}_{d_x} y + x \hat{y}_{d_y+1} - \hat{x}_{d_x} \hat{y}_{d_y+1} + \left(\hat{x}_{d_x+1} \hat{y}_{d_y+1} + \hat{x}_{d_x} \hat{y}_{d_y+1} \right) \left(2 - r_{d_x} - v_{d_y} \right) \quad \forall d_x = 1..D_x, d_y = 1..D_y \quad \text{(A70)}$$

$$z \leq x^U y \quad \text{(A71)}$$

$$z \leq x y^U \quad \text{(A72)}$$

From the few tests we performed using the direct partitioning options, we observed that all the aforementioned schemes for both variables did not present real advantages. We suspect that using McCormick partitioning options the results may be similar. However, a more thorough checking is needed, which we leave for future work.

Manuscript received Oct. 26, 2009, revision received Nov. 21, 2010, and final revision received Aug. 8, 2011.