Journal of Global Optimization (2022) 84:843–867
https://doi.org/10.1007/s10898-022-01179-3

Discretization and global optimization for mixed integer bilinear programming

Xin Cheng[1] · **Xiang Li**[1]

Received: 3 August 2021 / Accepted: 19 April 2022 / Published online: 10 June 2022

Abstract
We consider global optimization of mixed-integer bilinear programs (MIBLP) using discretization-based mixed-integer linear programming (MILP) relaxations. We start from the widely used radix-based discretization formulation (called R-formulation in this paper), where the base R may be any natural number, but we do not require the discretization level to be a power of R. We prove the conditions under which R-formulation is locally sharp, and then propose an R^+-formulation that is always locally sharp. We also propose an H-formulation that allows multiple bases and prove that it is also always locally sharp. We develop a global optimization algorithm with adaptive discretization (GOAD) where the discretization level of each variable is determined according to the solution of previously solved MILP relaxations. The computational study shows the computational advantage of GOAD over general-purpose global solvers BARON and SCIP.

Keywords Global optimization · Discretization · Mixed-integer bilinear programming · MILP relaxation · Sharp formulation

1 Introduction

Many process systems engineering problems can be cast as mixed-integer bilinear programs (MIBLP), such as crude oil scheduling [15, 21, 22, 26], multi-period gasoline blending [5, 18, 19, 23], water network design and operation [1, 2, 14, 16, 17, 20, 31], supply chain management [27] and hydrogen network optimization [13]. In this paper, we consider MIBLP in the following form:

$$\begin{aligned} \min \ & c(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}), \\ s.t. \ & \boldsymbol{g}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}) \leq 0, \end{aligned} \tag{MIBLP}$$

✉ Xiang Li
xiang.li@queensu.ca

Xin Cheng
x.cheng@queensu.ca

[1] Department of Chemical Engineering, Queen's University, 19 Division Street, Kingston, ON K7L 3N6, Canada

$$\boldsymbol{\delta} \in \{0,1\}^n,$$
$$(w_{l,m}, x_m, y_l) \in \mathcal{S}, \quad \forall (l,m) \in \Theta,$$

where c, $\boldsymbol{g}$ are linear functions $\boldsymbol{x}$, $\boldsymbol{y}$ are normalized variables involved in bilinear terms and $\boldsymbol{w}$ represents values of the bilinear terms, index set Θ associates the components in $\boldsymbol{x}$ and $\boldsymbol{y}$ to components in $\boldsymbol{w}$, and set $S = \{(w, x, y) \in \mathbb{R} \times [0,1]^2 : w = xy\}$. Note that the binary variables $\boldsymbol{\delta}$ appear linearly in the objective and the constraints.

Due to the integer variables and the nonconvex bilinear terms, obtaining a global solution of a MIBLP is computationally challenging. In the spatial branch-and-bound framework, the solution efficiency depends heavily on the tightness of the relaxation problem. A straightforward way to generate a relaxation problem is to replace the bilinear terms with their convex envelopes [24], and then the relaxation problem is a mixed-integer linear program (MILP). Strengthening constraints derived from the Reformulation-Linearization Technique [29] can be added to tighten the MILP relaxation. A more sophisticated relaxation strategy, called piecewise McCormick relaxation, is to partition the domain of variables in a bilinear term into multiple subdomains and replace the bilinear term with its convex envelops on each individual subdomain [16, 35]. The disjunctive constraints from piecewise McCormick relaxation can be represented by a logarithmic number of binary variables and constraints [34], and this leads to a logarithmic partitioning scheme [7, 25].

Another approach to generate a tight MILP relaxation is based on variable discretization. In this approach, one variable in the bilinear term is represented by an integer part and a decimal part. The integer part can take a set of equally spaced discrete values in the variable range, and the decimal part represents the deviation of the variable value from one of the discrete values. Using a radix-based representation [19], the integer part can be expressed by a set of binary variables, then the bilinear term becomes the sum of a set of bilinear terms and most of the terms can be rigorously transformed into linear constraints via exact linearization. The base of the radix-based representation, R, can be any natural number, and the existing discretization based MILP relaxation methods differ primarily in the choice of R. To the best of our knowledge, Pham et al. [28] first considered the discretization of a continuous variable involved in a bilinear term. In order to reduce the search space within the branch and bound framework, they finitely enumerated possible values of the variable being discretized, so essentially $R = 1$ was selected. Teles et al. [32] developed a multi-parametric disaggregation technique (MDT), which is a discretization based method using $R = 10$. Following the MDT approach, Kolodziej et al. [18] constructed restricted MILP and relaxation MILP in branch-and-bound search for bilinear programming. They concluded that MDT yields better MILP relaxation than piecewise McCormick relaxation because of smaller problem size. Kolodziej et al. [19] extended MDT to the general radix-based representation. They also implemented both binary system ($R = 2$) and decimal system ($R = 10$) in their global optimization algorithms to solve multi-period pooling problems. Both methods showed significant computational advantage over commercial global optimization solvers. Castro [6] proposed a discretization method using a mixed-radix numeral system, which includes multiple bases coming from the prime factorization of b. Through a set of benchmark pooling problems, he showed the computational advantage of the mixed-radix discretization over a single-radix discretization as well as the logarithmic partitioning scheme in [25].

The discretization points divide the variable range into subintervals of equal length. Let b be the number of subintervals, then it indicates the level of discretization. The aforementioned discretization strategies assume that b is a power of the selected base R, but some other studies do not make this assumption. For example, Gupte et al. [10] considered the $R = 2$ case and allowed b to be any natural number no larger than a power of R, and they proved

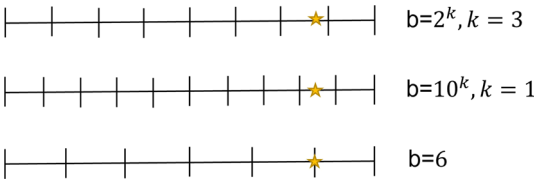

Fig. 1 A same optimal point under different discretization levels

the condition under which the continuous relaxation of the discretization formulation is a hull relaxation. Gupte et al. [11] further performed an extensive computational study on discretization based MILP approximation for the pooling problem and the results suggested that discretization seems to be a promising approach especially for large-scale standard or generalized pooling problems. Figure 1 can help explain why in some cases a smaller b is preferred and why different choices of R may affect the computational efficiency. The figure shows the same optimal variable value (indicated by the star sign) in the contexts of three different discretization levels. In the first discretization level, $R = 2$ and $b = 2^3 = 8$, so the variable range is divided into 8 subintervals. In the second discretization level, $R = 10$ and $b = 10$, so the variable range is divided into 10 subintervals. When $R = 10$, the discretization formulation requires more binary variables, but the optimal point is closer to the discretization point that it belongs to (i.e. the discretization point located to the left), implying that the MILP relaxation at the optimal solution is tighter (see Sect. 4 for more details). Therefore, $R = 10$ might be better than $R = 2$. Following this idea, if the optimal point is right on a discretization point, as with the third discretization level in the figure ($b = 6$), the MILP relaxation is equivalent to the original problem at the optimal point. Note that for realizing the third discretization level, one may choose any $R > 0$, but $R = 6$ is the most natural choice.

In this paper, we start from the general radix-based discretization formulation, called R-formulation, where R can be any natural number. We allow discretization level b to be any natural number no larger than a power of R. We prove the conditions under which R-formulation is locally sharp (i.e., its continuous relaxation leads to a hull relaxation of a part of the problem). We then propose a R^+-formulation and prove that it is always locally sharp. We further propose a H-formulation that uses hybrid base and prove that it is also always locally sharp. We also develop a global optimization method where the MILP relaxations are generated from adaptive discretization. The remaining part of the paper is organized as follows: Sect. 2 presents a general process for constructing discretization based MILP relaxations. Section 3 presents the three discretization formulations and provides theoretical results regarding the strength of the formulations. Section 4 proposes the global optimization algorithm with adaptive discretization. Section 5 performs computational study on a set of multi-period pooling problems, and the paper ends with conclusions in Sect. 6.

2 Discretization based MILP relaxation

Suppose we discretize variable x in each bilinear term, then Problem (MIBLP) can be re-written as:

$$\begin{aligned} \min \ & c(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}), \\ s.t. \ & g(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}) \leq 0, \\ & \boldsymbol{\delta} \in \{0, 1\}^n, \end{aligned} \tag{D-MIBLP}$$

$$(w_{l,m}, x_m, y_l) \in \mathcal{S}_I(b_m), \quad \forall (l,m) \in \Theta,$$

where

$$\mathcal{S}_I(b) = \left\{ \begin{array}{l} (w,x,y) \in \mathbb{R} \times [0,1]^2 : \ x = (X+\tilde{z})/b, \ \ 0 \le \tilde{z} \le 1, \ \ w = (\bar{w}+\tilde{y})/b, \\ (\bar{w}, X, y) \in P(b), \ \ \tilde{y} = \tilde{z}y \end{array} \right\},$$

and

$$\mathcal{P}(b) = \{(\bar{w}, X, y) : \ \ \bar{w} = Xy, \quad X \in \mathbb{Z}, \ \ 0 \le X \le b, \ \ 0 \le y \le 1\}.$$

Here continuous variable x is represented by integer variable $X \in [0, b]$ and a residual term $\tilde{z} \in [0, 1]$. Parameter b is the discretization level and it can be different for different bilinear terms. We can relax set $\mathcal{S}_I(b)$ by replacing $\tilde{z}y$ with its convex envelope, represented by $\text{conv}(\tilde{z}y)$, then the relaxed set is:

$$\mathcal{S}_{I-Relax}(b) = \left\{ \begin{array}{l} (w,x,y) \in \mathbb{R} \times [0,1]^2 : \ x = (X+\tilde{z})/b, \ \ 0 \le \tilde{z} \le 1, \ \ w = (\bar{w}+\tilde{y})/b, \\ (\bar{w}, X, y) \in P(b), \ \ \tilde{y} = \text{conv}(\tilde{z}y) \end{array} \right\}.$$

Since $\tilde{y} = \text{conv}(\tilde{z}y)$ only involves a set of linear constraints, set $\mathcal{S}_{I-Relax}(b)$ does not contain any bilinear terms. It can be further reformulated such that it only contains 0–1 integer variables:

$$\mathcal{S}_{B-Relax}(b) = \left\{ \begin{array}{l} (w,x,y) \in \mathbb{R} \times [0,1]^2 : \ x = (X+\tilde{z})/b, \ \ 0 \le \tilde{z} \le 1, \ \ w = (\bar{w}+\tilde{y})/b, \\ (\bar{w}, X, y) \in \mathcal{F}(b), \ \ \tilde{y} = \text{conv}(\tilde{z}y) \end{array} \right\},$$

where $\mathcal{F}(b)$ denotes any representation of set $\mathcal{P}(b)$ that uses binary variables instead of integer variables, such as a radix-based representation. Discretization based methods in the literature have different ways to represent $\mathcal{F}(b)$. Naturally, we hope the change from $\mathcal{S}_{I-Relax}(b)$ to $\mathcal{S}_{B-Relax}(b)$ does not loose tightness of the relaxation, so we assess the quality of any binary representation $\mathcal{F}(b)$ according to the following two conditions:

$$Proj_{\bar{w},X,y}\mathcal{F}(b) = \mathcal{P}(b), \tag{C1}$$

$$Proj_{\bar{w},X,y}(\text{relax}(\mathcal{F}(b))) = \text{conv}(\mathcal{P}(b)), \tag{C2}$$

where $Proj_{\bar{w},X,y}$ denotes the projection to the $(\bar{w}, X, y)$ space, and $\text{relax}(\cdot)$ denotes continuous relaxation. The first condition means that $\mathcal{F}(b)$ is a valid binary representation of $\mathcal{P}(b)$. $\mathcal{F}(b)$ usually includes additional binary and continuous variables, so it is defined on a higher-dimensional space. The second condition means that formulation $\mathcal{F}(b)$ is *sharp* for representing set $\mathcal{P}(b)$ [33]. Furthermore, we say that $\mathcal{F}(b)$ is *locally sharp* for the discretized MIBLP problem, because its continuous relaxation leads to the convex hull of part of the problem [33].

By replacing set $\mathcal{S}_I(b_m)$ in Problem (D-MIBLP) with the relaxed set $\mathcal{S}_{B-Relax}(b_m)$, we generate the following MILP relaxation for the original problem:

$$\begin{array}{ll} \min & c(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}), \\ s.t. & g(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}) \le 0, \\ & \boldsymbol{\delta} \in \{0,1\}, \\ & (w_{lm}, x_m, y_l) \in \mathcal{S}_{B-Relax}(b_m), \quad \forall (l,m) \in \Theta. \end{array} \tag{R-MILP}$$

Table 1 List of symbols

Parameters	
b	Discretization level of a variable (i.e., number of partitions of variable range)
R	Base for radix-based discretization
ϵ	Termination tolerance for the GOAD algorithm
Variables	
w	Value of bilinear term xy. The product of x_m and y_l is w_{lm}
$\bar{w}$	wb minus the residual bilinear term $\tilde{z}y$
x	Variable in bilinear term that is to be discretized, indexed by m in the MIBLP
y	Variable in bilinear term that is not to be discretized, indexed by l in the MIBLP
$\tilde{y}$	Value of residual bilinear term $\tilde{z}y$
$\hat{y}$	Auxiliary variable in discretization formulations
z	Binary variable for expressing integer X
$\tilde{z}$	Residual variable, which is the fractional part of xb
X	The integer part of xb
δ	Binary variable in the MIBLP
Sets	
$\mathcal{F}$	Binary representation of set $\mathcal{P}$
$\mathcal{H}$	H-formulation of set $\mathcal{P}$
$\mathcal{I}_1^p/\mathcal{I}_2^p$	Index sets used in the pth integer cut
$\mathcal{M}$	A relaxation of $\mathcal{P}$ (through McCormick relaxation of Xy)
$\mathcal{P}$	Graph of the discretized bilinear equation
$\mathcal{R}$	R-formulation of set $\mathcal{P}$
$\mathcal{R}^+$	R^+-formulation of set $\mathcal{P}$
$\mathbb{R}$	Set of real numbers
$\mathcal{S}$	Graph of bilinear term
$\mathcal{S}_I$	Reformulation of $\mathcal{S}$ using integer variables
$\mathcal{S}_{I-Relax}$	A relaxation of $\mathcal{S}_I$ (through McCormick relaxation of residual bilinear term)
$\mathcal{S}_{B-Relax}$	Binary representation of $\mathcal{S}_{I-Relax}$
$\mathbb{Z}$	Set of integer numbers
$\Delta^{k'}$	Set of integer cuts at GOAD iteration k'

Table 1 summarizes the list of symbols for parameters, variables and sets used in the formulations and algorithm description.

3 Binary representation formulations and their strength

In this section, we present three formulations for the binary representation $\mathcal{F}(b)$, which all lead to valid relaxations to Problem (MIBLP). We also prove the tightness of continuous relaxations of the three formulations.

3.1 *R*-formulation

We first consider the case that the binary representation $\mathcal{F}(b)$ is the R-formulation, which is the general radix-based discretization formulation widely used in the literature [19]. We write set $\mathcal{F}(b)$ as $\mathcal{R}(b)$ for this case, and this set is:

$$\mathcal{R}(b) = \left\{ \begin{array}{l} (\bar{w}, X, y, z, \hat{y}) \in \mathbb{R} \times \mathbb{Z} \times [0,1] \times \{0,1\} \times \mathbb{R} : \\ X = \sum_{i=1}^{k} R^{i-1} (\sum_{j=0}^{R-1} j \cdot z_{ij}), \quad 0 \le X \le b, \\ \bar{w} = \sum_{i=1}^{k} R^{i-1} (\sum_{j=0}^{R-1} j \cdot \hat{y}_{ij}), \\ \sum_{j=0}^{R-1} z_{ij} = 1, \quad \sum_{j=0}^{R-1} \hat{y}_{ij} = y, \quad \forall i \in \{1, \ldots, k\}, \\ 0 \le \hat{y}_{ij} \le z_{ij}, \quad \forall i \in \{1, \ldots, k\}, j \in \{1, \ldots, k\} \end{array} \right\},$$

where $k = \lceil \log_R b \rceil$ and R is the predefined base. The following proposition implies that first condition (C1) is satisfied by the R-formulation:

Proposition 1 $\mathcal{P}(b) = Proj_{\bar{w},X,y}\mathcal{R}(b)$.

Proof By construction, $\mathcal{P}(b) \subseteq \text{Proj}_{\bar{w},X,y}\mathcal{R}(b)$. Now we prove $\text{Proj}_{\bar{w},X,y}\mathcal{R}(b) \subseteq \mathcal{P}(b)$. Pick any $(\bar{w}, X, y, z, \hat{y}) \in \mathcal{R}(b)$, and we are to show $(\bar{w}, X, y) \in \mathcal{P}(b)$, or more specifically, $\bar{w} = Xy$.

First we show that $\hat{y}_{i,j} = y z_{i,j}$ $(\forall i, j)$. If $z_{i,j} = 0$, then $\hat{y}_{i,j} \le z_{i,j} = 0$, so $\hat{y}_{i,j} = y z_{i,j}$. If $z_{i,j} = 1$, then $\forall j' \ne j$, $z_{i,j'} = 0$ and therefore $\hat{y}_{i,j'} = 0$. Hence, $\hat{y}_{i,j} = y$ holds. So in this case we also have $\hat{y}_{i,j} = y z_{i,j}$.

Based on the above result,

$$\begin{aligned} \bar{w} &= \sum_{j=0}^{R-1} \sum_{i=1}^{k} j \cdot R^{i-1} \cdot \hat{y}_{i,j} \\ &= \sum_{i=1}^{k} R^{i-1} \sum_{j=0}^{R-1} j \cdot y \cdot z_{i,j} \\ &= \left(\sum_{i=1}^{k} R^{i-1} \sum_{j=0}^{R-1} j \cdot z_{i,j} \right) \cdot y \\ &= Xy. \end{aligned}$$

This completes the proof. □

To see whether $\mathcal{R}(b)$ satisfies condition (C2), we introduce an intermediate set $\mathcal{M}(b)$, which was initially introduced by Gupte et al. [10] for $R = 2$:

$$\mathcal{M}(b) = \left\{ \begin{array}{l} (\bar{w}, X, y) \in \mathbb{R} \times \mathbb{Z} \times [0,1] : \quad by + X - b \le \bar{w} \le by, \\ 0 \le \bar{w} \le X, \quad 0 \le X \le b \end{array} \right\}.$$

Then we have the following proposition that indicates the conditions under which (C2) is satisfied.

Proposition 2 *conv($\mathcal{P}(b)$)= relax($\mathcal{M}(b)$) = Proj$_{\bar{w},X,y}$(relax($\mathcal{R}(b)$)) if and only if $b \geq R^k - 1$.*

Proof First, the convex hull of $\mathcal{M}(b)$ and the convex hull of $\mathcal{P}(b)$ are same and they are equal to relax($\mathcal{M}(b)$), as explained in Gupte et al. [10]. Therefore, relax($\mathcal{M}(b)$) = conv($\mathcal{P}(b)$).

Second, we prove relax($\mathcal{M}(b)$) $\subseteq$ Proj$_{\bar{w},X,y}$(relax($\mathcal{R}(b)$)). From $\mathcal{P}(b) =$ Proj$_{\bar{w},X,y}\mathcal{R}(b)$ (Proposition 1), we have $\mathcal{P}(b) \subseteq$ Proj$_{\bar{w},X,y}$(relax($\mathcal{R}(b)$)). And since Proj$_{\bar{w},X,y}$(relax($\mathcal{R}(b)$)) is convex, conv($\mathcal{P}(b)$) $\subseteq$ Proj$_{\bar{w},X,y}$(relax($\mathcal{R}(b)$)). And the fact relax($\mathcal{M}(b)$) = conv($\mathcal{P}(b)$) gives the result.

Next, we prove Proj$_{\bar{w},X,y}$(relax($\mathcal{R}(b)$)) $\subseteq$ relax($\mathcal{M}(b)$) if $b \geq R^k - 1$. Pick any point $(\bar{w}, X, y, z, \hat{y}) \in$ relax($\mathcal{R}(b)$), then:

$$
\begin{aligned}
\bar{w} &= \sum_{j=0}^{R-1}\sum_{i=1}^{k} j \cdot R^{i-1} \cdot \hat{y}_{i,j} \geq 0.\\
\bar{w} &= \sum_{j=0}^{R-1}\sum_{i=1}^{k} j \cdot R^{i-1} \cdot \hat{y}_{i,j} \leq \sum_{j=0}^{R-1}\sum_{i=1}^{k} j \cdot R^{i-1} \cdot z_{i,j} = X.\\
\bar{w} &= \sum_{j=0}^{R-1}\sum_{i=1}^{k} j \cdot R^{i-1} \cdot \hat{y}_{i,j} \leq \sum_{i=1}^{k} R^{i-1}(R-1)y = (R^k - 1)y \leq by.\\
\bar{w} &= \sum_{j=0}^{R-1}\sum_{i=1}^{k} j \cdot R^{i-1} \cdot (\hat{y}_{i,j} - z_{i,j} + z_{i,j})\\
&= \sum_{j=0}^{R-1}\sum_{i=1}^{k} j \cdot R^{i-1} \cdot (\hat{y}_{i,j} - z_{i,j}) + \sum_{j=0}^{R-1}\sum_{i=1}^{k} j \cdot R^{i-1} \cdot z_{i,j}\\
&\geq (\sum_{i=1}^{k} R^{i-1}) \sum_{j=0}^{R-1}(R-1)(\hat{y}_{i,j} - z_{i,j}) + X\\
&= \left(\sum_{i=1}^{k} R^{i-1}(R-1)\right)\left(\sum_{j=0}^{R-1}\hat{y}_{i,j} - \sum_{j=0}^{R-1} z_{i,j}\right) + X\\
&= (R^k - 1)(y-1) + X\\
&\geq b(y-1) + X.
\end{aligned}
$$

Therefore, $(\bar{w}, X, y) \in$ relax($\mathcal{M}(b)$). Note that the condition $b \geq R^k - 1$ implies $b = R^k - 1$ or $b = R^k$ because $k = \lceil \log_R b \rceil$.

Finally, we show that $b \geq R^k - 1$ is also necessary for Proj$_{\bar{w},X,y}$(relax($\mathcal{R}(b)$)) $\subseteq$ relax($\mathcal{M}(b)$) by an counterexample. Construct a point $(\bar{w}, X, y, z, \hat{y}) \in relax(\mathcal{R}(b))$ by setting $\forall i$

$$
\begin{aligned}
&z_{i,0} = \frac{R-1}{R}, \quad z_{i,R-1} = \frac{1}{R}, \quad z_{i,j'} = 0 \ \ (\forall j' \neq 0, R-1),\\
&\hat{y}_{i,R-1} = \frac{1}{R}, \quad \hat{y}_{i,j'} = 0 \ (\forall j' \neq R-1),
\end{aligned}
$$

then

$$X = \sum_{j=0}^{R-1} \sum_{i=1}^{k} j \cdot R^{i-1} z_{i,j} = \frac{R^k - 1}{R-1} \frac{R-1}{R} = \frac{R^k - 1}{R},$$

$$y = \sum_{j=0}^{R-1} \hat{y}_{i,j} = \frac{1}{R},$$

$$\bar{w} = \sum_{j=0}^{R-1} \sum_{i=1}^{k} j \cdot R^{i-1} \hat{y}_{i,j} = \frac{R^k - 1}{R-1} \frac{R-1}{R} = \frac{(R^k - 1)}{R}.$$

Note that $\bar{w} = \dfrac{R^k - 1}{R} = (R^k - 1)y$. Therefore, if $b < R^k - 1$, $\bar{w} > by$ and $(\bar{w}, X, y) \notin \text{relax}(\mathcal{M}(b))$. □

3.2 R^+-formulation

Proposition 2 indicates that in order for R-formulation to be locally sharp, the discretization level b needs to be a power of the base R or the power minus 1. To avoid this problem, we can add strengthening constraints $X + by - b \le \bar{w} \le by$ to $\mathcal{R}(b)$. The new formulation is called R^+-formulation, as shown below:

$$\mathcal{R}^+(b) := \left\{ \begin{array}{l} (\bar{w}, X, y, z, \hat{y}) \in \mathbb{R} \times \mathbb{Z} \times [0,1] \times \{0,1\} \times \mathbb{R} : \\ X = \sum_{i=1}^{k} R^{i-1} (\sum_{j=0}^{R-1} j \cdot z_{i,j}), \\ \bar{w} = \sum_{i=1}^{k} R^{i-1} (\sum_{j=0}^{R-1} j \cdot \hat{y}_{i,j}), \\ \sum_{j=0}^{R-1} z_{i,j} = 1, \quad \sum_{j=0}^{R-1} \hat{y}_{i,j} = y, \quad \forall i \in \{1, \ldots, k\}, \\ 0 \le \hat{y}_{i,j} \le z_{i,j}, \quad \forall i \in \{1, \ldots, k\}, \ j \in \{1, \ldots, k\}, \\ X + by - b \le \bar{w} \le by \end{array} \right\},$$

where $k = \lceil \log_R b \rceil$. By construction, binary representation $\mathcal{R}^+(b)$ satisfies condition (C1), so the following proposition holds.

Proposition 3 $\mathcal{P}(b) = Proj_{\bar{w},X,y} \mathcal{R}^+(b)$.

In addition, we can prove that $\mathcal{R}^+(b)$ always satisfies condition (C2), as stated in the following proposition.

Proposition 4 *conv($\mathcal{P}(b)$)= relax($\mathcal{M}(b)$) = Proj$_{\bar{w},X,y}$(relax($\mathcal{R}^+(b)$)).*

Proof The inequalities $\bar{w} \le by$ and $\bar{w} \ge b(y-1) + X$ in relax($\mathcal{M}(b)$) are implied by the strengthening constraints. The other inequalities in relax($\mathcal{M}(b)$) are implied by Proj$_{\bar{w},X,y}$(relax($\mathcal{R}^+(b)$)) regardless of the relationship between b and R^k, as shown in the proof for Proposition 2. □

3.3 *H*-formulation

When using the discretization based MILP relaxation for global optimization, we may want to increase the discretization level adaptively in order to avoid unnecessarily large MILP relaxations. In this case, the relationship between b and R varies during the optimization procedure. According to the previous discussions, either we use the R-formulation that may lose local sharpness or the R^+-formulation that includes additional constraints. Here we propose a novel H-formulation that uses multiple bases to represent b:

$$\mathcal{H}(b) = \left\{ \begin{array}{l} (\bar{w}, y, X, z, \hat{y}) \in \mathbb{R} \times \mathbb{Z} \times [0,1] \times \{0,1\} \times \mathbb{R} : \\ X = \sum_{i=1}^{k} \left(\prod_{n=i+1}^{k} R_n \right) \left(\sum_{j=0}^{R_i-1} z_{i,j} \cdot j \right), \ 0 \le X \le b, \\ \bar{w} = \sum_{i=1}^{k} \left(\prod_{n=i+1}^{k} R_n \right) \left(\sum_{j=0}^{R_i-1} \hat{y}_{i,j} \cdot j \right), \\ \sum_{j=0}^{R_i-1} z_{i,j} = 1, \quad \sum_{j=0}^{R_i-1} \hat{y}_{i,j} = y, \quad \forall i = 1, \ldots, k, \\ 0 \le \hat{y}_{i,j} \le z_{i,j}, \quad \forall i = 1, \ldots, k, \quad \forall j = 1, \ldots, R_i - 1 \end{array} \right\},$$

where the bases R_i are selected such that $b = \prod_{i=1}^{k} R_i$. For example, if $b = 12$, then we may select $R_1 = 3$ and $R_2 = 4$, or $R_1 = 6$ and $R_2 = 2$. Essentially $\mathcal{H}(b)$ uses an extension of the classical radix-based number system that may use different bases for different digits. Next, we prove that the H-formulation satisfies both condition (C1) and (C2) (and therefore it is locally sharp).

Proposition 5 $\mathcal{P}(b) = Proj_{\bar{w},X,y}\mathcal{H}(b)$.

Proof By construction, $\mathcal{P}(b) \subseteq \text{Proj}_{\bar{w},X,y}\mathcal{H}(b)$. Now we prove $\text{Proj}_{\bar{w},X,y}\mathcal{H}(b) \subseteq \mathcal{P}(b)$. Pick any $(\bar{w}, X, y, z, \hat{y}) \in \mathcal{H}(b)$. Same to the proof for Proposition 1, we can show that $\hat{y}_{i,j} = yz_{i,j}$ $(\forall i, j)$. Consequently,

$$\begin{aligned} \bar{w} &= \sum_{i=1}^{k} \left(\prod_{n=i+1}^{k} R_n \right) \left(\sum_{j=0}^{R_i-1} \hat{y}_{i,j} \cdot j \right), \\ &= \sum_{i=1}^{k} \left(\prod_{n=i+1}^{k} R_n \right) \left(\sum_{j=0}^{R_i-1} yz_{i,j} \cdot j \right), \\ &= \sum_{i=1}^{k} \left(\prod_{n=i+1}^{k} R_n \right) \left(\sum_{j=0}^{R_i-1} z_{i,j} \cdot j \right) y, \\ &= Xy. \end{aligned}$$

This completes the proof. □

Proposition 6 $Proj_{x,y,\bar{w}}(relax(\mathcal{H}(b))) = relax(\mathcal{M}(b)) = conv(\mathcal{P}(b))$.

Proof Pick any point $(\bar{w}, y, X, z, \hat{y}) \in \text{relax}(\mathcal{H}(b))$, then

$$
\begin{aligned}
\bar{w} &= \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} \hat{y}_{i,j}\cdot j\right) \geq 0,\\
\bar{w} &= \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} \hat{y}_{i,j}\cdot j\right) \leq \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} z_{i,j}\cdot j\right) = X,\\
\bar{w} &= \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} \hat{y}_{i,j}\cdot j\right)\\
&\leq \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} y(R_i-1)\right)\\
&= y\left((R_1-1)R_2\ldots R_k + (R_2-1)R_3\ldots R_k + \ldots (R_{k-1}-1)R_k + R_k - 1\right)\\
&= y(b-1) \leq by,\\
\bar{w} &= \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} (\hat{y}_{i,j} - z_{i,j} + z_{i,j})\cdot j\right)\\
&= \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} (\hat{y}_{i,j} - z_{i,j})\cdot j\right) + \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} z_{i,j}\cdot j\right)\\
&\geq \sum_{i=1}^{k}\left(\prod_{n=i+1}^{k} R_n\right)\left(\sum_{j=0}^{R_i-1} (\hat{y}_{i,j} - z_{i,j})(R_i-1)\right) + X\\
&= \sum_{i=1}^{k}\left((R_i-1)\prod_{n=i+1}^{k} R_n\right)(y-1) + X\\
&= (b-1)(y-1) + X\\
&\geq b(y-1) + X,
\end{aligned}
$$

so $(\bar{w}, X, y) \in \text{relax}(\mathcal{M}(b))$. This proves $\text{Proj}_{\bar{w},X,y}(\text{relax}(\mathcal{H}(b))) \subseteq \text{relax}(\mathcal{M}(b))$.

We can also prove $\text{conv}(\mathcal{P}(b)) \subseteq \text{Proj}_{\bar{w},X,y}(\text{relax}(\mathcal{H}(b)))$ using the same procedure in the second part of proof for Proposition 2. In addition, according to the first part of the proof for Proposition 2, $\text{relax}(\mathcal{M}(b)) = \text{conv}(\mathcal{P}(b))$. This completes the proof. □

4 Global optimization algorithm with adaptive discretization

In this section, we develop a global optimization algorithm with adaptive discretization (GOAD) for MIBLP. The algorithm finds a global solution by iteratively solving a lower bounding problem and an upper bounding problem.

4.1 Lower and upper bounding problems

At iteration k', we solve the following lower bounding problem:

$$\begin{aligned} \min.\ & c(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}), \\ s.t.\ & g(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}) \leq 0, \\ & \boldsymbol{\delta} \in \Delta^{k'}, \\ & (w_{lm}, x_m, y_l) \in S_{B-Relax}(b_m^{k'}), \quad \forall (l,m) \in \Theta, \\ & c(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}) \leq UB. \end{aligned} \tag{L-MILP$^{k'}$}$$

This formulation is slightly different from the general form (R-MILP) introduced in Sect. 2. First, integer cuts are added to exclude previously visited $\boldsymbol{\delta}$ solutions through set $\Delta^{k'}$, which is

$$\Delta^{k'} = \left\{ \boldsymbol{\delta} \in \{0,1\}^n : \sum_{i \in \mathcal{I}_1^p} \delta_i - \sum_{j \in \mathcal{I}_0^p} \delta_j \leq |\mathcal{I}_1^p| - 1, \quad \forall p = 1, 2, \ldots, k' \right\}.$$

Index set $\mathcal{I}_1^p = \{i \in \{1, \ldots, n\} : \delta_i^p = 1\}$, and $\boldsymbol{\delta}^p$ denotes the solution generated in iteration p. Index set $\mathcal{I}_0^p = \{i \in \{1, \ldots, n\} : \delta_i^p = 0\}$. Second, the last constraint in the formulation is added to exclude any solution that is no better than the incumbent solution. UB is the current best upper bound on the optimal objective value and it is updated throughout the solution procedure. Finally, the discretization level $b_m^{k'}$ may vary over the iterations, which is explained later.

Let the value of $\boldsymbol{\delta}$ at the solution of (L-MILP$^{k'}$) be $\boldsymbol{\delta}^{k'}$, then fixing $\boldsymbol{\delta}$ to this value will yield the following upper bounding problem:

$$\begin{aligned} \min.\ & c(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}), \\ s.t.\ & g(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\delta}) \leq 0, \\ & \boldsymbol{\delta} = \boldsymbol{\delta}^{k'}, \\ & (w_{lm}, x_m, y_l) \in S, \quad \forall (l,m) \in \Theta. \end{aligned} \tag{U-NLP$^{k'}$}$$

This is a nonlinear programming (NLP) problem and has no integer variables. This NLP can be solved to global optimality much more quickly than the original MIBLP. For all problem instances in the computational study, the solution time for Problem (U-NLP$^{k'}$) is negligible in comparison to the solution time for the original MIBLP.

4.2 Adaptive discretization scheme

We choose to adaptively determine the discretization levels of variables instead of using predefined discretization levels. The adaptive scheme has two benefits. First, at the optimal solution many variables take value at the boundary of their domains, so these variables do not need to be discretized. Some variables have little impact on the optimal objective value, so a low discretization level is sufficient for them. Our proposed adaptive discretization scheme is likely to result in reasonable discretization levels for different variables. Second, the discretization level of a variable x determines the residual term $\tilde{z}$ needed to represent its

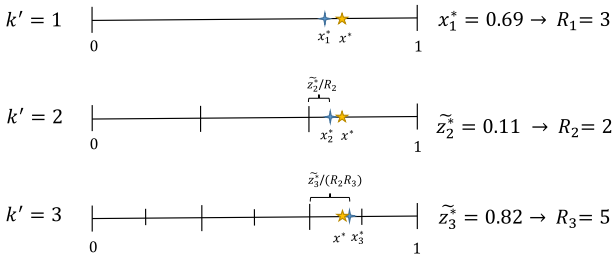

Fig. 2 Example of adaptive discretization

solution, and a smaller $\tilde{z}$ would imply a smaller relaxation gap because the gap comes from the difference between $\tilde{y} = \tilde{z}y$ in $\mathcal{S}_I(b)$ and $\tilde{y} = \text{conv}(\tilde{z}y)$ in $\mathcal{S}_{B-Relax(b)}$.

The adaptive discretization strategy perfectly matches the hybrid radix-based H-formulation, because it determines one base at each iteration. We use the example in Fig. 2 to explain the adaptive discretization scheme. For convenience, we assume there is only one normalized variable x to be discretized. The optimal solution of the problem is shown as x^* in the figure. This value is not known by the algorithm in the first several iterations, but provided in the figure as a reference. In iteration 1, there is no discretization, so the optimal solution of the lower bounding problem (L-MILP^1) only has the x value, represented by x_1^*. We want to divide the range of x into R_1 pieces such that x_1^* is close to the discretization point it belongs to. From the discretization formulation, $x = (X + \tilde{z})/b$ where b is the R_1 that we search for, so $\tilde{z} = R_1x - X$. Note that X is the largest integer that is no larger than R_1x, so $\tilde{z} = R_1x - \lfloor R_1x \rfloor$. Therefore, the residual term for x_1^* is $R_1x_1^* - \lfloor R_1x_1^* \rfloor$. On the other hand, we do not want to increase the discretization level too much within one iteration, so we limit R_1 to be between 2 and 10. Thus we can formalize the calculation of R_1 as:

$$R_1 = \underset{2 \le r \le 10}{argmin} \left(r \cdot x_1^* - \lfloor r \cdot x_1^* \rfloor \right).$$

As shown in Fig. 2, following this formula we get $R_1 = 3$ in iteration 1. Then the discretization level $b = R_1$ and the range of x is divided into 3 equal pieces for the next iteration.

In iteration 2, we have the residual term value at the optimal solution of (L-MILP^2), represented as $\tilde{z}_2^*$. As the residual term is scaled to $[0, 1]$, $\tilde{z}_2^*$ reflects the scaled distance between the x_2^* and the largest discretized point that is no larger than x_2^*. The actual difference is $\tilde{z}_2^*/R_2$. Optimal point x_2^* is located in the 3rd subinterval of the range. Now we want to further divide the 3rd subinterval into several pieces such that x_2^* is close to the discretization point it belongs to. Following the following formula that uses $\tilde{z}_2^*$:

$$R_2 = \underset{2 \le r \le 10}{argmin} \left(r \cdot \tilde{z}_2^* - \lfloor r \cdot \tilde{z}_2^* \rfloor \right)$$

we get $R_2 = 2$, which indicates further dividing the 3rd subinterval into two pieces, and therefore dividing each other subinterval into two as well. As a result, the new discretization level is $b = R_1R_2 = 6$. For all subsequent iterations, we follow the same formula that uses the residual value. For example, in iteration 3, using the residual value R_3 should be 5, and then the discretization level becomes $b = R_1R_2R_3 = 30$. From Fig. 2, the optimal solution from the lower bound problem x_3^* is already very close to the actual optimal solution x^*, so the algorithm is likely to converge right after this iteration.

In some cases, the residual from the optimal solution of (L-MILP$^{k'}$) is close to an extreme point of the range and the above formula won't give a reasonable result. For these cases we

Table 2 The GOAD algorithm for problem (MIBLP)

Initialization

Set termination tolerance $\epsilon = 10^{-3}$.

Set iteration counter $k' = 0$.

Set bounds for Problem (MIBLP): $UBD = +\infty$, $LBD = -\infty$.

Set initial discretization level $b_m^1 = 1$ for all x_m.

Lower Bounding Problem

(1.a) Solve Problem (L-MILP$^{k'}$). If Problem L-MILP$^{k'}$ is infeasible, terminate. If no feasible solution has been found, then Problem (MIBLP) is infeasible; otherwise, the incumbent solution is an ϵ-optimal solution.

(1.b) If $obj_{L-MILP^{k'}} > LBD$, update $LBD = obj_{L-MILP^{k'}}$. Let $(\boldsymbol{w}, \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\delta}, \tilde{\boldsymbol{z}})^*$ be the optimal solution.

(1.c) Based on $\tilde{z}_m^*$ (or x_m^* when k'=1), calculate $R_{m,k'}$ according to the adaptive discretization strategy and then $b_m^{k'+1} = \prod_{i=1}^{k'} R_{m,i}$.

(1.d) If $|UBD - LBD| \leq \epsilon$, terminate and the incumbent solution is an ϵ-optimal solution; otherwise, go to step (2.a).

Upper Bounding Problem

(2.a) Solve Problem (U-NLP$^{k'}$) to global optimality.

(2.b) If $obj_{U-NLP^{k'}} < UBD$, change the incumbent solution to the current solution and update $UBD = obj_{U-NLP^{k'}}$.

(2.c) Add an integer cut to exclude $\boldsymbol{\delta}^*$.

(2.d) If $|UBD - LBD| \leq \epsilon$, terminate. If no feasible solution has been found, then Problem (MIBLP) is infeasible; otherwise, the incumbent solution is an ϵ-optimal solution. If $|UBD - LBD| > \epsilon$, $k' = k' + 1$ and go to step (1.a).

simply further divide all subintervals into two. Specifically, for iteration k', if $|\tilde{z}_{k'}^* - 0.5| \geq 0.45$ (or $|x_{k'}^* - 0.5| \geq 0.45$ when $k' = 1$), we set $R_{k'} = 2$. Therefore, the full formula for calculating $R_{k'}$ is:

$$R_{k'} = \begin{cases} \underset{2 \leq r \leq 10}{argmin} \left(r \cdot x_{k'}^* - \lfloor r \cdot x_{k'}^* \rfloor \right), & k' = 1 \text{ and } |x_{k'}^* - 0.45| < 0.45, \\ \underset{2 \leq r \leq 10}{argmin} \left(r \cdot \tilde{z}_{k'}^* - \lfloor r \cdot \tilde{z}_{k'}^* \rfloor \right), & k' \geq 2 \text{ and } |\tilde{z}_{k'}^* - 0.45| < 0.45, \\ 2, & k' = 1 \text{ and } |x_{k'}^* - 0.45| \geq 0.45, \\ 2, & k' \geq 2 \text{ and } |z_{k'}^* - 0.45| \geq 0.45. \end{cases}$$

Note that R-formulation or R^+ formulation can also be used for the adaptive discretization scheme. In this case, the discretization level b is still calculated from $R_{k'}$, but it is represented in the formulation using the predefined base R.

4.3 The GOAD algorithm

Table 2 presents the GOAD algorithm. Finite termination of the algorithm is guaranteed, because total number of integer values $\boldsymbol{\delta}$ can take is finite and the integer cuts in (L-MILP$^{k'}$) exclude all previously generated values $\boldsymbol{\delta}$.

5 Computational study

The computational study contains two parts. The first part aims to compare R- and R^+-formulations and to verify Propositions 2 and 4. The second part aims to compare the performance of the GOAD algorithm using different discretization formulations as well as commercial global optimization solvers.

We test on 20 multi-period pooling problems, including 5 benchmark problems from the literature (P146t3, P480t4, P531t4, P721t3, P852t4), and they are available in MINLP problem library at www.minlp.org. For each benchmark problem, we also construct several variants that address different numbers of time periods, so each benchmark problem is extended to 4 problem instances that have 3, 4, 5, 6 time periods (labelled as t3–t6 respectively). The parameters for all problem instances are provided in "Appendix A". We formulate all problem instances using a formulation that is essentially the source-based (SB) formulation in [23], but the blender operating modes are described using the expression in [19]. The details of the formulation are provided in "Appendix B". In this formulation, each bilinear term includes a variable that represents a fraction of inventory leaving a tank, and we always discretize this variable for the MILP relaxation. The simulation is performed on a computer with 4-core 3.40 GHz CPU, 4GB memory, and Windows 10 operating system. The binary variables $z_{i,j}$ in all formulations are set to be SOS1 variables in GAMS.

5.1 Comparison of R- and R^+-formulations

From Propositions 2 and 4, the key difference between R and R^+-formulations is that the former is locally sharp only when $b \geq R^k - 1$ while the latter is always locally sharp. In order to show this, we compare the MILP relaxation problem (L-MILP) in three scenarios: $b = 100$, $b = 512$ and $b = 750$. In each scenario, (L-MILP) adopts 4 different formulations: R formulation and $R+$ formulation with $R = 2$ and $R = 10$ respectively. The problems are modeled on GAMS 24.8.5 [3] and solved by CPLEX 12.7.1 using 6 threads, with absolute/relative tolerance 10^{-5}. We set CPLEX option "cut = − 1" so that the solver do not generate cuts that may influence the comparison. We set a time limit of 3600 s for each problem.

We also compare the root gaps of the formulations, by solving their continuous relaxations at the root node. Specifically, the root gap is calculated as

$$\text{Gap} = \left|\frac{\text{Obj}_{MILP} - \text{Obj}_{LP}}{\text{Obj}_{MILP}}\right| \times 100\%,$$

where Obj_{MILP} and Obj_{LP} represent the optimal objective values of the MILP problem and its continuous relaxation, respectively. The continuous relaxations are also solved by CPLEX 12.7.1 with absolute/relative tolerance 10^{-4}.

Tables 3, 4 and 5 show respectively the results for $b = 100$, $b = 512$, $b = 750$. For each problem instance, we highlight the lowest root gap(s) in bold unless all formulations have the same root gap. First, we discuss the second set of problem instances P480t3-P480t6 which best demonstrate the strength of the formulations. When $b = 100$ (Table 3), 2^+-formulation has a tighter root gap than 2-formulation. This verifies Proposition 2 and 4, because for $b = 100$ and $R = 2$, $k = \lceil \log_2 100 \rceil = 7$, so $b < 2^k - 1$ and 2-formulation is not locally sharp but 2^+-formulation is. On the other hand, 10-formulation and 10^+-formulation have the same root gap (which is also same to that of 2^+-formulation). This is because b is a power of 10 and therefore both formulations are locally sharp. When $b = 512$ (Table 4) where b is a power of 2, 2-formulation, 2^+-formulation, and 10^+-formulation have the same root

Table 3 Comparison of the fixed-base discretization formulations for b = 100

Cases	Obj_{MILP}	2		2^+		10		10^+	
		Time (s)	Root gap (%)	Time (s)	Root gap (%)	Time (s)	Root gap (%)	Time (s)	Root gap (%)
P146t3	45.3104	22.5	6.8	21.1	6.8	20.7	6.8	20.1	6.8
P146t4	54.1309	101.5	4.1	178.7	4.1	156.3	4.1	173.5	4.1
P146t5	59.7606	3548.2	4.0	*0.4%	4.0	*0.4%	4.0	3031.0	4.0
P146t6	64.6763	*2.4%	4.6	*2.8%	4.6	*1.9%	4.6	*4.7%	4.6
P480t3	7.7221	5.1	29.8	2.7	**29.1**	4.8	**29.1**	4.7	**29.1**
P480t4	9.2266	48.7	28.5	45.5	**27.8**	40.1	**27.8**	45.5	**27.8**
P480t5	9.5292	203.7	26.0	237.9	26.0	167.5	26.0	203.9	26.0
P480t6	9.8313	2525.9	25.1	*1.0%	**24.5**	2830.0	**24.5**	3282.3	**24.5**
P531t3	13.5070	0.9	62.9	1.3	62.9	1.4	62.9	1.3	62.9
P531t4	20.0390	4.9	13.9	5.6	13.9	9.9	13.9	10.8	13.9
P531t5	21.9760	5.4	12.1	3.9	12.1	16.0	12.1	16.2	12.1
P531t6	20.9018	33.2	16.3	23.1	16.3	44.0	16.3	43.3	16.3
P721t3	13.5268	2.8	9.2	3.1	9.2	4.3	9.2	4.0	9.2
P721t4	15.7348	52.6	9.7	59.6	9.7	57.1	9.7	89.0	9.7
P721t5	16.4382	394.7	8.5	286.5	8.5	746.9	8.5	624.8	8.5
P721t6	18.1974	1249.4	7.3	1395.9	7.3	1868.6	7.3	2147.7	7.3
P852t3	45.6052	2.4	2.2	3.1	2.2	2.7	2.2	3.4	2.2
P852t4	53.9627	7.9	2.4	7.6	2.4	10.3	2.4	10.8	2.4
P852t5	56.0884	274.0	3.2	251.1	3.2	289.9	3.2	227.3	3.2
P852t6	59.3350	1193.4	2.8	1047.9	2.8	834.7	2.8	1109.0	2.8

*Not converged within 3600 s, so the gap at termination is reported

Table 4 Comparison of the fixed-base discretization formulations for b = 512

Cases	Obj_{MILP}	2		2^+		10		10^+	
		Time (s)	Root gap (%)	Time (s)	Root gap (%)	Time (s)	Root gap (%)	Time (s)	Root gap (%)
P146t3	45.2996	24.4	6.8	23.7	6.8	32.6	6.8	23.1	6.8
P146t4	54.1303	132.6	4.1	141.3	4.1	252.5	4.1	143.9	4.1
P146t5	59.7444	*0.4%	4.0	*0.3%	4.0	*1.0%	4.0	*0.4%	4.0
P146t6	64.3937	*2.8%	5.0	*2.6%	5.0	*3.4%	5.0	*3.8%	5.0
P480t3	7.7221	2.5	**29.1**	4.3	**29.1**	7.2	30.3	6.9	**29.1**
P480t4	9.2266	21.2	**27.8**	46.3	**27.8**	89.6	28.9	57.0	**27.8**
P480t5	9.5292	212.1	26.0	250.4	26.0	291.3	26.0	207.8	26.0
P480t6	9.8313	2507.1	**24.5**	2330.7	**24.5**	*1.1%	25.2	*2.3%	**24.5**
P531t3	13.5070	1.1	62.9	1.0	62.9	2.5	62.9	2.2	62.9
P531t4	20.0390	5.6	13.9	7.6	13.9	15.9	13.9	18.3	13.9
P531t5	21.9760	5.2	12.1	5.0	12.1	36.8	12.1	35.6	12.1
P531t6	20.9003	15.4	16.3	30.2	16.3	80.8	16.3	83.9	16.3
P721t3	13.5268	2.1	9.2	2.5	9.2	5.3	9.2	5.4	9.2
P721t4	15.7348	48.5	9.7	47.5	9.7	99.0	9.7	72.4	9.7
P721t5	16.4382	269.0	8.5	249.5	8.5	448.1	8.5	387.4	8.5
P721t6	18.1974	873.9	7.3	1178.7	7.3	*0.2%	7.3	3358.0	7.3
P852t3	45.6052	1.4	2.2	1.7	2.2	3.7	2.2	3.3	2.2
P852t4	53.9627	5.4	2.4	5.6	2.4	32.5	2.4	26.1	2.4
P852t5	56.0884	216.6	3.2	190.2	3.2	307.6	3.2	268.0	3.2
P852t6	59.3280	1148.1	2.8	1604.4	2.8	2936.4	2.8	3086.9	2.8

*Not converged within 3600 s, so the gap at termination is reported

Table 5 Comparison of the fixed-base discretization formulations for b = 750

Cases	Obj_{MILP}	2		2^+		10		10^+	
		Time (s)	Root gap (%)	Time (s)	Root gap (%)	Time (s)	Root gap (%)	Time (s)	Root gap (%)
P146t3	45.298	28.4	6.8	29.0	6.8	31.0	6.8	23.2	6.8
P146t4	54.1308	157.6	4.1	171.9	4.1	195.1	4.1	201.0	4.1
P146t5	59.743	*0.6%	4.0	*0.9%	4.0	*0.4%	4.0	*1.0%	4.0
P146t6	63.363	*4.5%	6.7	*2.8%	6.7	*1.9%	6.7	*3.2%	6.7
P480t3	7.7221	4.9	29.9	4.3	**29.1**	7.1	29.8	6.6	**29.1**
P480t4	9.2266	55.1	28.6	59.1	**27.8**	76.1	28.6	53.1	**27.8**
P480t5	9.5292	192.8	26.6	210.8	**26.0**	283.2	26.6	188.6	**26.0**
P480t6	9.764	*3.0%	26.0	1973.0	**25.4**	2829.5	26.0	*2.8%	**25.4**
P531t3	13.507	1.0	62.9	1.8	62.9	3.0	62.9	1.9	62.9
P531t4	20.039	6.0	13.9	6.9	13.9	12.3	13.9	15.4	13.9
P531t5	21.976	14.6	12.1	7.0	12.1	33.1	12.1	19.4	12.1
P531t6	20.8993	31.3	16.3	32.7	16.3	142.8	16.3	66.8	16.3
P721t3	13.5265	3.8	9.2	3.2	9.2	4.9	9.2	4.0	9.2
P721t4	15.7348	76.9	9.7	50.9	9.7	67.9	9.7	118.4	9.7
P721t5	16.4382	348.9	8.5	255.1	8.5	698.5	8.5	996.1	8.5
P721t6	18.1974	1371.2	7.3	1467.4	7.3	1868.2	7.3	*2.4%	7.3
P852t3	45.6052	2.4	2.2	3.5	2.2	6.4	2.2	2.6	2.2
P852t4	53.9627	10.0	2.4	6.4	2.4	29.0	2.4	29.1	2.4
P852t5	56.0884	255.5	3.2	181.9	3.2	310.1	3.2	303.5	3.2
P852t6	59.3270	1115.6	2.8	1159.4	2.8	2360.1	2.8	2362.2	2.8

∗: Not converged within 3600 s, so the gap at termination is reported

gap, because they are all locally sharp. 10-formulation has a larger root gap because it is not locally sharp. When $b = 750$ (Table 5) where b is neither a power of 2 or a power of 10, only 2^+- and 10^+-formulations are locally sharp. So the root gaps of 2- and 10-formulations are larger. The different formulations for the other sets of the problem instances in Tables 3, 4 and 5 have the same root gap, so they can not be used to verify Proposition 2 and 4. But they do not contradict the propositions either.

Second, we can see that for each problem the locally sharp formulation with the smallest problem size usually has the best solution efficiency, such as the 2-formulation when $b = 512$. However, the solution efficiency is also influenced by many other factors, such as the variables to be branched on and the relaxation problems solved at the nodes. So it is hard to conclude that $R = 2$ is better than $R = 10$ or the other way. Next, we will examine whether within the GOAD framework certain formulation is better than the others.

5.2 Performance of the GOAD algorithm

We compare three GOAD algorithms that use 2^+-, 10^+-, and H-formulations, general-purpose global optimization solvers SCIP 7.0 [9] and BARON 21.1.13 [30], and Gurobi 9.1.2 [12] that can solve MIBLP problems to global optimality. We implement all solution methods via GAMS 36.2 [4]. For a GOAD algorithm, the lower bounding problem (L-MILP) is solved by CPLEX 20.1.0 with 6 threads, the upper bounding problem (U-NLP) is solved by SCIP 7.0, and for both bounding problems the time limit is 1800 s and absolute/relative tolerance is 10^{-3}. For using a global solver to a problem, we also allow the solver to use 6 threads. For either a GOAD algorithm or global solver, the total time limit is 3600 s and the absolute/relative tolerance is 10^{-3}.

Note that at an iteration we determine the next discretization level according to the optimal solution of (L-MILP). This problem has the same optimal objective value under different discretization formulations, but since it usually has multiple optimal solutions, different discretization formulations may give us different optimal solutions and therefore different discretization level for the next iteration. For a fair comparison, we use the discretization level determined from the H-formulation, no matter what discretization formulation the GOAD algorithm uses.

Figure 3 shows the performance profiles [8] of the solution methods under consideration. The horizontal axis τ denotes the relative solution time (or called performance ratio), and the vertical axis $\rho_s(\tau)$ denotes the percentage of problem instances solved within relative time τ. It can be seen that the GOAD algorithm with any of the three discretization formulations performs better than SCIP and BARON. There is no clear winner among the three discretization formulations within the GOAD framework, which is not unexpected as the three formulations are all locally sharp. In addition, the solution times for the three formulations not only depend on the tightness of the continuous relaxations, but also depend on other factors such as number of binary variables, number of constraints, the nodes explored in the branch-and-bound search. Therefore, the relative performance of the three formulations is different for different problem instances. It can also be seen that Gurobi performs better than any of the GOAD algorithm, SCIP, and BARON. This may be because Gurobi can generate additional strengthening cuts from solutions of convex relaxations, while the GOAD algorithm does not generate cuts from solutions of lower bounding problems.

Table 6 provides additional results for the GOAD algorithm. The column of b_{max} shows the maximum discretization level among discretization levels of all variables at the termination of GOAD, and the column of b_{min} shows the minimum discretization level. The last

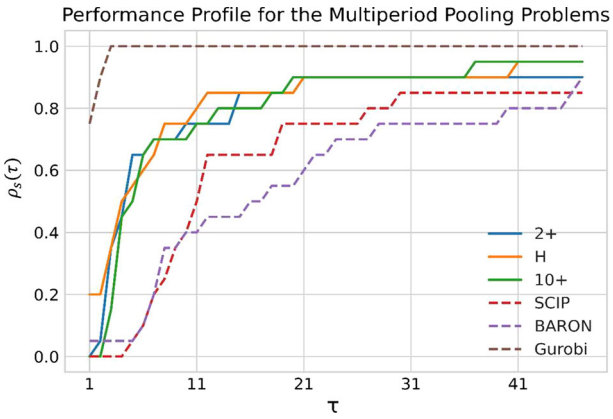

Fig. 3 Performance profiles of different solution methods

Table 6 Additional results for the GOAD algorithm

Problem	Optimal obj.	b_{max}	b_{min}	Portion of var. with b_{min}
P146t3	45.297	36	8	95.0%
P146t4	54.125	64	4	93.3%
P146t5	59.738	288	16	91.3%
P146t6	64.762†	– †	– †	– †
P480t3	7.722	10	2	95.0%
P480t4	9.227	128	32	88.3%
P480t5	9.529	7	2	90.0%
P480t6	9.831	84	16	94.0%
P531t3	13.507	49	4	89.5%
P531t4	20.039	125	8	96.5%
P531t5	21.976	40	4	94.7%
P531t6	20.901	250	8	93.7%
P721t3	13.527	10	2	90.0%
P721t4	15.735	10	2	88.3%
P721t5	16.438	24	4	92.5%
P721t6	18.197	1024	32	94.0%
P852t3	45.605	7	2	92.5%
P852t4	53.963	10	2	95.0%
P852t5	56.088	10	2	95.0%
P852t6	59.326	54	4	95.0%

†No optimum found within 1 h. The optimal obj. is from Gurobi

column includes the percentage of variables that has the minimum discretization level at the termination of GOAD. It can be seen that for all problems, the vast majority of the variables have the minimum discretization level, but some variables may need a much higher discretization level. This indicates the importance of allowing different discretization levels in the adaptive discretization scheme. In addition, the various maximum discretization levels

indicate the importance of flexible choice of bases. For example, P721t6 requires $b = 1024$ for some variable; within a traditional 10-based formulation it would have to partition the variable range into 10^4 pieces. P721t4 requires $b = 10$ for some variables and a traditional 2-based formulation may not be as good as a 10-based formulation.

6 Concluding remarks

The GOAD method we develop for MIBLP generates strong MILP relaxations via an adaptive discretization scheme. The adaptive scheme allows to increase discretization levels of different variables over the iterations and different variables may have different discretization levels. The advantage of this flexibility is demonstrated in the computational study, where different variables do have different discretization levels at the convergence of the algorithm and most of them are much smaller than the maximum discretization level. The adaptive discretization scheme demands that the discretization level b can be any natural number instead of a power of a predefined base R. In this case, the classical fixed-base R-formulation may not be locally sharp if $b < R^k - 1$ (where $k = \lceil log_R b \rceil$). We propose a R^+-formulation that includes extra strengthening constraint and prove that this formulation is locally sharp no matter what b is. We further propose a H-formulation that allows multiple bases and prove that it is also locally sharp because b is a multiplication of the bases used. Compared to a fixed-base formulation, H-formulation fits more naturally for the adaptive discretization scheme. The GOAD method solves the original MIBLP by iteratively solving the lower bounding MILP that comes from discretization and an upper bounding NLP. The time for global optimization of the NLP is negligible in comparison to the solution time for the MILP, so the latter determines the overall solution time.

The GOAD method has better performance profile than general-purpose global solvers SCIP and BARON in the computational study, no matter whether 2^+-, 10^+, or H-formulation is used for discretization. However, the GOAD method is less efficient than Gurobi, which is not a general-purpose global solver but can solve MIBLPs to global optimality. Further improvement of the GOAD method may be achieved by introduction of improved discretization formulations and strengthening cuts. For example, we may limit the bases in the H-formuation to prime numbers, such as in [6], and add more cuts in set $\mathcal{S}_{B-Relax}(b_m)$ to strengthen the lower bounding problem. We may also generate additional valid cuts from the solution of the lower bounding problem at each step and use these cuts to strengthen the problem in future steps.

Acknowledgements The authors are grateful to the Natural Sciences and Engineering Research Council of Canada for the Discovery Grant RGPIN 418411-13 and the Collaborative Research and Development Grant CRDPJ 485798-15.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Appendix A: Parameters for the multiperiod pooling problems

See Table 7.

Table 7 Supply and demand parameters

	Tank	T1	T2	T3	T4	T5	T6
F_IN (s,t)							
P146	S1	0.3	0.1	0.7	0.5	0.7	0.4
	S2	0	0.8	0.3	0.4	0.4	0.3
P480	S1	0	0.2	0.7	0.5	0	0.3
	S2	0	0.6	0.6	0.5	0	0.2
P531	S1	0.4	0.1	0.2	0.8	0.6	0.8
	S2	0.8	0.1	0.4	0.8	0.6	0.5
P721	S1	1	0.1	0.4	0.5	0.2	0
	S2	0.6	0.2	0.8	0.6	0.5	0
P852	S1	0.3	0.1	0.7	1	0.4	0.3
	S2	0	0.8	0.3	0	0.3	0.2
F_OUT(d,t)							
P146	D1	0.95	0.44	0.77	0.65	0.3	0.4
	D2	0.03	0.38	0.8	0.34	0.4	0.3
P480	D1	0.3	0.19	0.18	0.63	0	0.1
	D2	0.34	0.69	0.37	0.78	0	0.1
P531	D1	0.06	0.53	0.66	0.29	0.36	0.12
	D2	0.4	0.42	0.63	0.43	0.22	0.3
P721	D1	0.02	0.17	0.73	0.4	0.2	0.3
	D2	0.04	0.65	0.65	0.5	0.12	0.1
P852	D1	0.44	0.77	0.19	0.45	0.23	0.1
	D2	0.38	0.8	0.49	0.65	0.11	0.3

Tanks: S = {S1,S2} includes source tanks, D = {D1,D2} includes demand tanks
T1–T6: Individual time periods
F_IN (s,t): Incoming mass (10^3 kg) of source tank s in time period t
F_OUT(d,t): Outgoing mass (10^3 kg) from demand tank d in time period t

Appendix B: The SB formulation for the multiperiod pooling problems

See Table 8 for the list of symbols in the SB formulation

Table 8 List of symbols for the SB formulation

Type	Name	Description
Indices and sets	$i \in S \cup B \cup D$	Nodes
	$l \in L$	Initial liquid present in tanks
	$s \in S$	Source nodes
	$b \in B$	Pool nodes
	$d \in D$	Demand nodes
	$t \in T$	Time periods
	$q \in Q$	Components
	$(i, i') \in A$	Allowed arcs
	$(i, l) \in L$	Pairs of tank and liquid
Parameters	β_s	Cost for supply flows
	β_d	Price for product flows
	M	big-M
	$F_{s,t}^{IN}$	Incoming flow into supply tank s during time period t
	$F_{d,t}^{OUT}$	Flow withdraw from demand tank d during time period t.
	$C_{q,l}^{0}$	Concentration of component q in liquid l
	$C_{q,d}^{L}$	Lower bound on product concentration
	$C_{q,d}^{U}$	Upper bound on product concentration
	$F_{i,i'}^{L}$	Lower bound on pipe capacity
	$F_{i,i'}^{U}$	Upper bound on pipe capacity
	I_i^L	Lower bound on inventories
	I_i^U	Upper bound on inventories
	$\alpha_{i,i'}$	Fixed cost for each arc in use
	$\beta_{i,i'}$	Operation cost for the unit flow
Variables	$f_{l,i,i',t}$	Individual source flow from i to i' during t
	$I_{l,i,t}$	Inventory that source from l in tank i during t
	$x_{b,i',t}$	Split fraction variable
	$f_{l,d,t}^{OUT}$	Source flow send to d during t
Binary variables	$y_{i,i',t}$	Whether flow exists in arc (i,i') during time period t

The SB formulation

Objective:

$$\max . \sum_{t\in T}\left[\sum_{i:(i,d)\in A}\sum_{d\in D}\beta_d f_{i,d,t} - \sum_{s\in S}\sum_{i:(s,i)\in A}\beta_s f_{s,i,t} - \sum_{(i,i')\in A}(\alpha_{i,i'}y_{l,i,i',t} + \beta_{i,i'}f_{i,i',t})\right].$$

s.t.

Bilinear terms:

$$f_{l,b,i,t} = I_{l,b,t-1}x_{b,i,t}, \qquad \forall (b,l)\in L, (b,i)\in A, t>1. \tag{SB1}$$

Mass Balance:

$$I_{l,s,t} = I^0_{l,s}/I_{l,s,t-1} + F^{IN}_{s,t} - \sum_{i:(s,i)\in A} f_{l,s,i,t}, \qquad \forall (l,s)\in L, t\in T, x \tag{SB1}$$

$$I_{l,b,t} = I^0_{l,b}/I_{l,b,t-1} - \sum_{i:(b,i)\in A} f_{l,b,i,t}$$

$$+ \sum_{i:(i,b)\in A} f_{l,i,b,t}, \qquad \forall b\in B, (l,b)\in L, t\in T, \tag{SB3}$$

$$I_{l,d,t} = I^0_{l,d}/I_{l,d,t-1} - f^{OUT}_{l,d,t} + \sum_{i:(i,d)\in A} f_{l,i,d,t}, \qquad \forall d\in D, t\in T. \tag{SB4}$$

Quality Bounds:

$$\sum_l I_{l,b,t-1}C^0_{q,l} \geq C^L_{q,d}\sum_l I_{l,b,t-1} - M(1-y_{b,d,t}), \qquad \forall (b,d)\in A, q\in Q, t>1, \tag{SB5}$$

$$\sum_l I_{l,b,t-1}C^0_{q,l} \leq C^U_{q,d}\sum_l I_{l,b,t-1} + M(1-y_{b,d,t}), \qquad \forall (b,d)\in A, q\in Q, t>1, \tag{SB5}$$

$$C^0_{q,s}y_{s,d,t} \geq C^L_{q,d}y_{s,d,t}, \qquad \forall (s,d)\in A, t\in T, \tag{SB7}$$

$$C^0_{q,s}y_{s,d,t} \leq C^U_{q,d}y_{s,d,t}, \qquad \forall (s,d)\in A, t\in T, \tag{SB8}$$

$$C^0_{q,b}y_{b,d,t} \geq C^L_{q,d}y_{b,d,t}, \qquad \forall (b,d)\in A, t=1. \tag{SB9}$$

$$C^0_{q,b}y_{b,d,t} \leq C^U_{q,d}y_{b,d,t}, \qquad \forall (b,d)\in A, t=1. \tag{SB10}$$

Tank Capacity:

$$I^L_i \leq I_{i,t} \leq I^U_i, \qquad \forall i\in N, t\in T. \tag{SB11}$$

Pipe Capacity:

$$F^L_{i,i',t} y_{i,i',t} \leq f_{i,i',t} \leq F^U_{i,i',t} y_{i,i',t}, \qquad \forall (i,i') \in A, t \in T. \tag{SB12}$$

Individual Flow Constraints:

$$\sum_l I_{l,b,t} = I_{b,t}, \qquad \forall (b,l) \in L, t \in T, \tag{SB13}$$

$$\sum_l f^{out}_{l,d,t} = F^{OUT}_{d,t}, \qquad \forall d \in D, t \in T. \tag{SB14}$$

Operation Mode:

$$y_{i,b,t} + y_{b,i',t} \leq 1, \qquad \forall (i,b),(b,i') \in A, t \in T. \tag{SB15}$$

Variable Bounds:

$$f_{i,i',t} \geq 0, \qquad \forall (i,i') \in A, t \in T, \tag{SB16}$$

$$0 \leq x_{b,i,t} \leq 1, \qquad \forall (b,i) \in A, t \in T, \tag{SB17}$$

$$y_{i,i',t} \in \{0,1\}, \qquad \forall (i,i') \in A, t \in T. \tag{SB18}$$

References

1. Adams, W.P., Sherali, H.D.: Mixed-integer bilinear programming problems. Math. Program. **59**(1–3), 279–305 (1993)
2. Bagajewicz, M.: A review of recent design procedures for water networks in refineries and process plants. Comp. Chem. Eng. **24**(9–10), 2093–2113 (2000)
3. Bussieck, M.R., Meeraus, A.: General algebraic modeling system (GAMS). In: Kallrath, J. (ed.) Modeling Languages in Mathematical Optimization, pp. 137–157. Springer, Boston (2004)
4. Bussieck, M.R., Meeraus, A.: General algebraic modeling system (GAMS). In: Kallrath, J. (ed.) Modeling Languages in Mathematical Optimization, pp. 137–157. Springer, Boston (2004)
5. Castro, P.M.: New MINLP formulation for the multiperiod pooling problem. AIChE J. **61**(11), 3728–3738 (2015)
6. Castro, P.M.: A piecewise relaxation for quadratically constrained problems based on a mixed-radix numerical system. Comp. Chem. Eng. **153**, 107459 (2021)
7. Castro, P.M., Liao, Q., Liang, Y.: Comparison of mixed-integer relaxations with linear and logarithmic partitioning schemes for quadratically constrained problems. Optim. Eng. (2021). https://doi.org/10.1007/s11081-021-09603-5
8. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
9. Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Le Bodic, P., Maher, S.J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M.E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D., Witzig, J.: The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin (2020). http://nbn-resolving.de/urn:nbn:de:0297-zib-78023
10. Gupte, A., Ahmed, S., Cheon, M.S., Dey, S.: Solving mixed integer bilinear problems using MILP formulations. SIAM J. Optim. **23**(2), 721–744 (2013)
11. Gupte, A., Ahmed, S., Dey, S.S., Cheon, M.S.: Relaxations and discretizations for the pooling problem. J. Glob. Optim. **67**(3), 631–669 (2017)
12. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022). https://www.gurobi.com

13. Hallale, N., Liu, F.: Refinery hydrogen management for clean fuels production. Adv. Environ. Res. **6**(1), 81–98 (2001)
14. Jezowski, J.: Review of water network design methods with literature annotations. Ind. Eng. Chem. Res. **49**(10), 4475–4516 (2010)
15. Jia, Z., Ierapetritou, M., Kelly, J.D.: Refinery short-term scheduling using continuous time formulation: crude-oil operations. Ind. Eng. Chem. Res. **42**(13), 3085–3097 (2003)
16. Karuppiah, R., Grossmann, I.E.: Global optimization for the synthesis of integrated water systems in chemical processes. Comp. Chem. Eng. **30**(4), 650–673 (2006)
17. Khor, S., Chachuat, B., Shah, N.: Optimization of water network synthesis for single-site and continuous processes: milestones, challenges, and future directions. Ind. Eng. Chem. Res. **53**(25), 10257–10275 (2014)
18. Kolodziej, S., Castro, P.M., Grossmann, I.E.: Global optimization of bilinear programs with a multiparametric disaggregation technique. J. Glob. Optim. **57**(4), 1039–1063 (2013)
19. Kolodziej, S.P., Grossmann, I.E., Furman, K.C., Sawaya, N.W.: A discretization-based approach for the optimization of the multiperiod blend scheduling problem. Comp. Chem. Eng. **53**, 122–142 (2013)
20. Li, A.H., Zhang, J., Liu, Z.Y.: Design of distributed wastewater treatment networks of multiple contaminants with maximum inlet concentration constraints. J. Clean. Prod. **118**, 170–178 (2016)
21. Li, J., Li, W., Karimi, I., Srinivasan, R.: Improving the robustness and efficiency of crude scheduling algorithms. AIChE J. **53**(10), 2659–2680 (2007)
22. Li, J., Misener, R., Floudas, C.A.: Continuous-time modeling and global optimization approach for scheduling of crude oil operations. AIChE J. **58**(1), 205–226 (2012)
23. 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. Comp. Chem. Eng. **87**, 13–35 (2016)
24. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part I—convex underestimating problems. Math. Program. **10**(1), 147–175 (1976)
25. Misener, R., Thompson, J.P., Floudas, C.A.: APOGEE: global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. Comp. Chem. Eng. **35**(5), 876–892 (2011)
26. Mouret, S., Grossmann, I.: Crude-oil operations scheduling. CyberInfrastructure for MINLP [www.minlp.org, a collaboration of Carnegie Mellon University and IBM Research] www.minlp.org/library/problem/index.php (2010)
27. Nahapetyan, A.G.: Bilinear programming: applications in the supply chain management. In: Encyclopedia of Optimization, pp. 282–288. Springer US, Boston, MA (2009). https://doi.org/10.1007/978-0-387-74759-0_49
28. Pham, V., Laird, C., El-Halwagi, M.: Convex hull discretization approach to the global optimization of pooling problems. Ind. Eng. Chem. Res. **48**(4), 1973–1979 (2009)
29. Sherali, H.D., Alameddine, A.: A new reformulation-linearization technique for bilinear programming problems. J. Glob. Optim. **2**(4), 379–410 (1992)
30. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005)
31. Teles, J.P., Castro, P.M., Matos, H.A.: Global optimization of water networks design using multiparametric disaggregation. Comp. Chem. Eng. **40**, 132–147 (2012)
32. Teles, J.P., Castro, P.M., Matos, H.A.: Multi-parametric disaggregation technique for global optimization of polynomial programming problems. J. Glob. Optim. **55**, 227–251 (2013)
33. Vielma, J.: Mixed integer linear programming formulation techniques. SIAM Rev. **57**, 3–57 (2015)
34. Vielma, J., Nemhauser, G.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Math. Program. **128**, 49–72 (2011)
35. Wicaksono, D.S., Karimi, I.A.: Piecewise MILP under-and overestimators for global optimization of bilinear programs. AIChE J. **54**(4), 991–1008 (2008)