# Piecewise MILP Under- and Overestimators for Global Optimization of Bilinear Programs

**Danan Suryo Wicaksono and I. A. Karimi**
Dept. of Chemical and Biomolecular Engineering, National University of Singapore, 4 Engineering Drive 4, Singapore 117576

*DOI 10.1002/aic.11425*
*Published online February 19, 2008 in Wiley InterScience (www.interscience.wiley.com).*

*Many practical problems of interest in chemical engineering and other fields can be formulated as bilinear programs (BLPs). For such problems, a local nonlinear programming solver often provides a suboptimal solution or even fails to locate a feasible one. Numerous global optimization algorithms devised for bilinear programs rely on linear programming (LP) relaxation, which is often weak, and, thus, slows down the convergence rate of the global optimization algorithm. An interesting recent development is the idea of using an ab initio partitioning of the search domain to improve the relaxation quality, which results in a relaxation problem that is a mixed-integer linear program (MILP) rather than LP, called as piecewise MILP relaxation. However, much work is in order to fully exploit the potential of such approach. Several novel formulations are developed for piecewise MILP under- and overestimators for BLPs via three systematic approaches, and two segmentation schemes. As is demonstrated and evaluated the superiority of the novel models is shown, using a variety of examples. In addition, metrics are defined to measure the effectiveness of piecewise MILP relaxation within a two-level-relaxation framework, and several theoretical results are presented, as well as valuable insights into the properties of such relaxations, which may prove useful in developing global optimization algorithms.* © 2008 American Institute of Chemical Engineers *AIChE J,* 54: 991–1008, 2008

*Keywords: global optimization, bilinear programming, mixed-integer linear programming, piecewise under- and overestimators, process network synthesis*

## Introduction

Many practical problems of interest in chemical engineering and other fields can be formulated as optimization problems involving bilinear functions of continuous decision variables. For instance, the mathematical programming formulations for the pooling problem,[1] integrated water systems synthesis,[2] process network synthesis,[3] crude oil operations scheduling,[4,5] as well as fuel gas network design and management in liquefied natural gas (LNG) plants,[6,7] all involve bilinear products of continuous decision variables, such as stream flows and compositions. The optimization formulations involving such bilinear functions, called bilinear programs (BLPs), belong to the class of nonconvex nonlinear programming problems that exhibit multiple local optima. For such problems, a local nonlinear programming (NLP) solver often provides a suboptimal solution or even fails to locate a feasible one. However, the need for obtaining a guaranteed globally optimal solution is real, essential, and often critical, in many practical problems mentioned previously. Understandably, this has led to a flurry of research activities[8,9] in the last two decades on global optimization, which involves obtaining a theoretically guaranteed globally optimal solution to a nonconvex mathematical program.

Correspondence concerning this article should be addressed to I. Karimi at cheiak@nus.edu.sg.

While several global optimization algorithms[10–14] exist today, the most common ones use the so-called spatial branch-and-bound framework.[15,16] This framework is similar to the standard branch-and-bound algorithm widely used in combinatorial optimization.[17] The main difference is that the spatial branch-and-bound branches in continuous rather than discrete variables. Tight lower and upper bounds, efficient procedures for obtaining them, and clever strategies for branching are the main challenges in this scheme. For a minimization (maximization) problem, any feasible solution acts as a valid upper (lower) bound and can be obtained by means of a local NLP solver (e.g., CONOPT, MINOS, SNOPT). For lower (upper) bounds, however, the common approach is to solve a good convex (concave), linear or nonlinear, relaxation of the original problem to global optimality using a standard LP solver (e.g., CPLEX, OSL, LINDO, XA) or a local NLP solver. If the gap between the lower and upper bounds exceeds a prespecified tolerance for any partition of the search space, that partition is branched further, until the gap reduces below the tolerance.

The development of this branch-and-bound approach has been the focus of much research during the last decade. Branch-and-reduce optimization navigator (BARON), a commercial implementation of this framework, by Sahinidis[18] has been a significant development. Ryoo and Sahinidis[19] introduced a *branch-and-reduce* approach with a range-reduction test based on Lagrangian multipliers. Zamora and Grossmann[20] proposed a *branch-and-contract* global optimization algorithm for univariate concave, bilinear, and linear fractional functions. The emphasis was on reducing the number of nodes in the branch-and-bound tree through the proper use of a contraction operator. This involved maximizing and minimizing each variable within a linear relaxation problem. Neumaier et al.[21] presented test results for the software performing complete search to solve global optimization problems and concluded that BARON is the fastest and most robust.

The success of a spatial branch-and-bound scheme depends critically on the rate at which the gap between the lower and upper bounds reduces. For faster convergence, this gap must decrease quickly and monotonically, as the search space reduces. In other words, devising efficient procedures for obtaining tight bounds is a key challenge in global optimization, as both the quality of bounds, and the time required to obtain them strongly influence the overall effectiveness and efficiency of a global optimization algorithm. As stated earlier, relaxation of the original problem is the most widely used procedure, so the quality of relaxation and the effort required for its solution are extremely critical.

Much research has focused on constructing a convex relaxation for factorable nonconvex NLP problems. This class of problems exclusively involves factorable functions, which are the ones that can be expressed as recursive sums and products of univariate functions.[22] Several researchers[23,24] proposed symbolic reformulation techniques to transform an arbitrary factorable nonconvex program into an equivalent standard form in which all nonconvex terms are expressed as special nonlinear terms, such as bilinear and concave univariate terms. This approach employs the fact that all factorable algebraic functions involve one or more unary and/or binary operations. Transcendental functions, such as the exponential and logarithm of a single variable, are examples of the former and five basic arithmetic operations of addition, subtraction, multiplication, division, and exponentiation form the latter. Therefore, these special nonlinear terms form the building blocks for factorable nonconvex problems that abound in a wide range of disciplines including chemical engineering. In addition to those mentioned earlier, many problems in process systems engineering, such as process design, operation, and control fall within this scope. Thus, by addressing bilinear programs in this work, we are essentially addressing the much wider class of factorable nonconvex programs.

LP relaxation is the most widely used technique for obtaining lower bounds for a factorable nonconvex program. McCormick[22] was the first to present convex underestimators and concave overestimators for the bilinear term on a rectangle. Later, Al-Khayyal and Falk[25] theoretically characterized these under- and overestimators as the convex envelope for a bilinear term. Foulds et al.[26] utilized the bilinear envelope embedded inside a branch-and-bound framework to solve a bilinear program for the single-component pooling problem based on total flow formulation. Tawarmalani et al.[27] showed that tighter LP relaxations can be produced by disaggregating the products of a single continuous variable and a sum of several continuous variables. LP relaxation, however, is often weak, and, thus, other forms of relaxation have also been proposed.Androulakis et al.[28] proposed a convex quadratic NLP relaxation, named *αBB underestimator*, which can be applied to general twice continuously differentiable functions. However, the tightness of such a relaxation for specific problems involving bilinear terms is inferior compared to its LP counterpart. Meyer and Floudas[29] attempted to improve the tightness of the classical αBB underestimator via a smooth piecewise quadratic, perturbation function.

Sherali and Alameddine[30] introduced a novel technique, called reformulation-linearization technique (RLT), to improve the relaxation of a bilinear program by creating redundant constraints. Ben-Tal et al.[31] proposed an alternative formulation for a bilinear program for the multicomponent pooling problem, based on individual flow formulation and employed a Lagrangian relaxation to solve it within a branch-and-bound framework. Adhya et al.[32] proposed another Lagrangian approach for generating valid relaxations for the pooling problem that are tighter than LP relaxations. Tawarmalani and Sahinidis[12] showed that the combined total and individual flow formulation for the bilinear programs of multicomponent pooling and related problems proposed by Quesada and Grossmann,[3] produces a tighter LP relaxation compared to either the Lagrangian relaxation or the LP relaxation, based on either the total or individual flow formulations alone. While the formulation of Quesada and Grossmann[3] can be derived using the RLT, no theoretical and/or systematic framework exists to date for deriving RLT formulations with predictably efficient performance for general nonconvex programs.

An interesting recent development is the idea of *ab initio* partitioning of the search domain, which results in a relaxation problem that is a mixed-integer linear program (MILP), rather than LP, called as piecewise MILP relaxation. Some recent work has shown the promise of such an approach in accelerating the convergence rate in several important appli-

cations, such as process network synthesis,[33] integrated water systems synthesis,[34] and generalized pooling problem.[35] However, much work is in order to fully exploit the potential of such an approach. All previous works have reported that the lower bounding problem in global minimization, based on piecewise MILP relaxation is the most time-consuming step. Moreover, it is solved repeatedly inside a global optimization framework (e.g., spatial branch-and-bound, outer approximation, or RLT), and, thus, many issues, such as the quality and efficiency of piecewise MILP relaxation demand further attention. In this work, we develop, analyze, compare, and improve several novel and existing formulations for piecewise MILP under- and overestimators for BLPs that may arise solely or within some mixed-integer bilinear programming (MIBLP) problems. We demonstrate the superiority of our under- and overestimators, as well as corresponding formulations using a variety of examples.

## Problem Statement

Our ultimate goal is to solve the following global optimization problem by employing piecewise MILP relaxation

$$\mathbf{P} = \left\{ \begin{array}{c} \text{Min} \\ \boldsymbol{x}^L \le \boldsymbol{x} \le \boldsymbol{x}^U \end{array} \quad f(\boldsymbol{x}) \text{ subject to } \mathbf{g}(\boldsymbol{x}) \le 0 \quad \text{and } \mathbf{h}(\boldsymbol{x}) = 0 \right\}$$

where $\boldsymbol{X} \in \Re^n$ is a vector of continuous variables with bound vectors $\boldsymbol{x}^L$ and $\boldsymbol{x}^U$, $f(\boldsymbol{x})$ is an $\Re^n \to \Re$ scalar objective function, and $\mathbf{g}(\boldsymbol{x})$ and $\mathbf{h}(\boldsymbol{x})$ are vectors of $\Re^n \to \Re$ scalar functions representing the inequality and equality constraints. All functions are twice continuously differentiable and involve linear and bilinear terms $xy$ only.

To achieve the aforementioned goal, we focus on developing several novel piecewise MILP under- and overestimators for the following nonconvex feasible region ($S$)

$$S = \{(x,y,z) \mid z = xy, x \in \Re, y \in \Re, x^L \le x \le x^U, y^L \le y \le y^U\}$$

## Relaxation

Relaxation involves outer-approximating the feasible region of a given problem and underestimating (overestimating) the objective function of a minimization (maximization) problem. A relaxation does not fully replace the original problem, but provides guaranteed bounds on its solutions. In a minimization (maximization) problem, the optimal solution of the relaxation problem provides a lower (upper) bound on the optimal objective function value of the original problem. Typically, a relaxation is achieved by bounding the complicating variables, terms, or functions in the original problem by means of under-, over-, and/or outer-estimating variables, terms, or functions.Several forms of relaxation exist in the literature. One form is the *discrete-to-continuous relaxation* employed for solving discrete optimization problems, where discrete variables are treated as continuous variables. For instance, binary variables in a MILP are relaxed to be 0-1 continuous.[17] Another form is the continuous *nonconvex-to-convex relaxation* employed for solving nonconvex NLP. For example, the bilinear envelope suggested by McCormick[22] and Al-Khayyal and Falk[25] is widely used to relax bilinear terms in nonconvex programs. This relaxation involves replacing every occurrence of the bilinear term $xy$ in the original program by an additional variable $z$, and adding the following linear (convex) underestimators (Eqs. R1 and R2), and linear (concave) overestimators (Eqs. R3 and R4)

$$z \ge xy^L + x^L y - x^L y^L \quad \text{(R1)}$$

$$z \ge xy^U + x^U y - x^U y^U \quad \text{(R2)}$$

$$z \le xy^L + x^U y - x^U y^L \quad \text{(R3)}$$

$$z \le xy^U + x^L y - x^L y^U \quad \text{(R4)}$$

Since the resulting relaxation is linear and continuous, it is called as *LP relaxation* (Figure 1).

The quality of a relaxation is the accuracy with which a relaxation approximates the original problem and/or its solution. The closer the approximation, the *tighter* is the relaxation. An important consideration in relaxation is the size of the relaxation problem. This can be measured in terms of the numbers of variables, constraints, and nonzeros involved in the formulation. Typically, a larger problem size is needed to achieve a tighter relaxation. While solving MILPs in a branch-and-bound framework, a tighter formulation is likely to require fewer nodes, while a smaller formulation is likely to require fewer iterations for each node. Therefore, the actual computational performance of a formulation is difficult to determine *a priori* because of the trade-off between tightness and size.

All the relaxations discussed previously are "continuous" in nature. A continuous convex relaxation can often be very weak or loose, and may be very slow in lifting the lower bounds in a global minimization algorithm. A technique[36] has been proposed for problems containing signomial terms utilizing some transformation methods, which convert a nonseparable signomial function into separable forms that are then under- and overestimated with piecewise MILP approximations. However, several examples showed that such a method does not yield better relaxation tightness than those based on continuous convex envelope (e.g., LP relaxation described previously for bilinear terms), and, hence, the use of convex envelope within a piecewise approximator is desirable whenever applicable. As a remedy, several recent

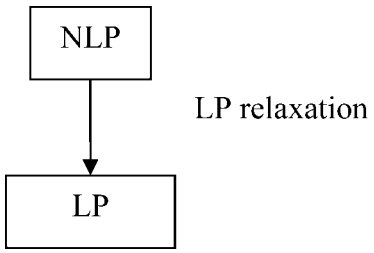


**Figure 1. LP relaxation (one-level-relaxation).**

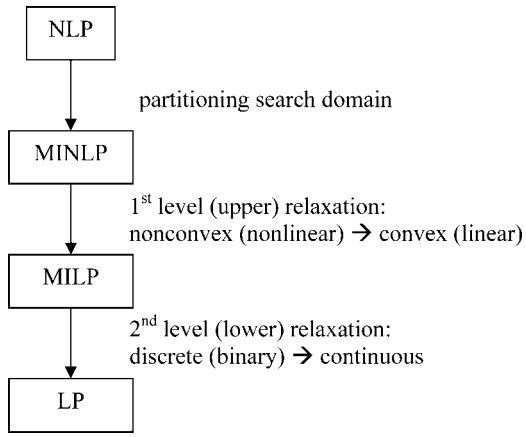


**Figure 2. Hierarchy of the piecewise MILP relaxation (two-level-relaxation).**

works[33–35] have explored the idea of piecewise MILP relaxation, based on the convex envelope for bilinear term, embedded inside a global optimization framework (e.g., outer approximation, spatial branch-and-bound, RLT), on several specific problems with promising results. The latter approach is different from the previous one, since it does not require the transformation of the original nonconvex terms into separable forms. In addition, the latter approach provides superior relaxation tightness over the continuous convex envelope as discussed later. Thus, we concentrate our work on piecewise MILP relaxation, based on the convex envelope.

The idea involves defining *a priori* several known partitions of the search space, and combining the continuous nonconvex-to-convex relaxations, based on the convex envelope of individual partitions into an overall composite relaxation. Because this involves convex relaxations of nonconvex functions over smaller regions (partitions) of the feasible region, the tightness of the overall discrete relaxation is improved as compared to the continuous relaxation over the entire feasible region. Each partition has its own distinct continuous nonconvex-to-convex relaxation, and only one partition is allowed to be active at any time. Combining these individual relaxations in a seamless manner requires switching between different partitions, and, thus, discrete decisions. Clearly, such a relaxation is discrete rather than continuous in nature, and, thus, can be formulated as a MILP problem. Because solving the resulting MILP problem generally requires some sort of discrete-to-continuous relaxation, the overall framework of piecewise MILP relaxation comprises relaxations at two levels as shown in Figure 2 (compared with LP relaxation, which only has one level as shown in Figure 1). The first one, or the *first* (*upper*) *level relaxation*, transforms the original problem with partitioned search domain into a MILP. The second one, or the *second* (*lower*) *level relaxation*, transforms the MILP into a LP (i.e., RMILP). A complex interplay of both relaxations determines the overall efficiency of the entire framework.

## Previous Formulations

The first step, as presented in the literature, in obtaining a piecewise MILP relaxation for a bilinear term is to define $N$ partitions (Figure 3) of the search space in terms of $N$ arbitrary, but exhaustive segments of the range $[x^L, x^U]$. Let $\{[a(n), a(n+1)], n = 1, 2, \ldots, N\}$ denote these segments, where $a(1) = x^L$, $a(N+1) = x^U$, and $d(n) = a(n+1) - a(n) > 0$ for all $n$. Thus, the $N$ search space partitions in the 2-D $xy$ space are $\{[a(n), a(n+1)], [y^L, y^U]\}$ for $n = 1, 2, \ldots, N$. Clearly, each point in $S$ must have its value of $x$ in one of these $N$ segments (either within the interior of a segment or at the boundary of two adjacent segments called as *grid points*). Then, using the convex envelope (Eqs. R1 – R4) for each partition, an overall piecewise relaxation of $S$ can be stated as the following special form[33] of a *disjunctive program*[37]

$$\bigvee_n \begin{bmatrix} W(n) \\ z \geq x \cdot y^L + a(n) \cdot y - a(n) \cdot y^L \\ z \geq x \cdot y^U + a(n+1) \cdot y - a(n+1) \cdot y^U \\ z \leq x \cdot y^L + a(n+1) \cdot y - a(n+1) \cdot y^L \\ z \leq x \cdot y^U + a(n) \cdot y - a(n) \cdot y^U \\ a(n) \leq x \leq a(n+1) \\ y^L \leq y \leq y^U \end{bmatrix} \quad \text{(DP)}$$

where $W(n)$ is the *boolean variable* ("true" or "false") indicating the status of disjunction $n$. The disjunctive logic OR implies that only one disjunction must hold ($W(n)$ = "true" for exactly one $n$).

One advantage of disjunctive programming is that it enables a systematic transformation of abstract disjunctive logic into a concrete mathematical programming model. Raman and Grossmann[38] showed its usefulness in modeling chemical engineering problems. While several systematic methods exist for transforming a disjunctive program into a mixed-integer program, the two most common are big-M reformulation[39] and convex-hull reformulation.[37,40,41] The pros and cons of these two reformulations are well known.[42,43] A big-M reformulation is generally smaller in size than a convex-hull reformulation, as it does not need additional disaggregated variables and constraints. However, its relaxation is typically poorer, as a convex-hull reformulation has proven tightness. In contrast, a convex-hull reformulation invariably needs additional disaggregated variables and constraints and

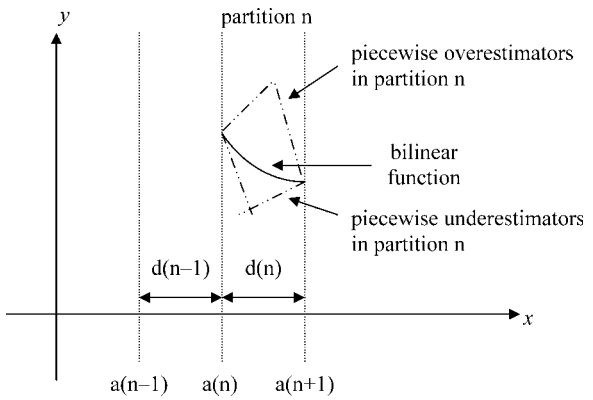


**Figure 3. *Ab initio* partitioning of the search domain.**

is typically larger, but is at least as tight as big-M reformulation. Note that the concept of a convex hull can be viewed from geometrical perspective. A convex hull of a set of points A is the smallest convex set containing A. In the sense of convex relaxation, a convex hull implies the tightest convex outer-approximation of the original problem. However, the performances of the two reformulations vary with problems, and it is not easy to judge *a priori* which will perform better for a given problem. A rigorous numerical comparison on several models, is, therefore, required to gain the insight into the actual computational performance of competitive models. For our subsequent discussion, a model/formulation is denoted with bold characters and an equation or a set of equations is denoted with plain characters preceeded by "Eq.".

For the bilinear terms arising in a generalized pooling problem, Meyer and Floudas[35] used a big-M reformulation for their piecewise MILP relaxation. Although their formulation was in the context of a specific problem, its main ideas can yield a complete big-M reformulation for DP. Such a complete formulation **BM** for an arbitrary $S$ can be stated as follows. $M$ is a sufficiently large number required for big-M reformulation

$$\lambda(n) = \begin{cases} 1 & \text{if } a(n) \le x \le a(n+1) \\ 0 & \text{otherwise} \end{cases} \quad \forall n \qquad \text{(BM-0)}$$

$$\sum_{n=1}^{N} \lambda(n) = 1 \qquad \text{(BM-1)}$$

$$x \ge a(n)\cdot\lambda(n) + x^L \cdot \left[1-\lambda(n)\right] \quad \forall n \qquad \text{(BM-2a)}$$

$$x \le a(n+1)\cdot\lambda(n) + x^U \cdot \left[1-\lambda(n)\right] \quad \forall n \qquad \text{(BM-2b)}$$

$$z \ge x\cdot y^L + a(n)\cdot(y-y^L) - M\cdot\left[1-\lambda(n)\right] \quad \forall n \qquad \text{(BM-3a)}$$

$$z \ge x\cdot y^U + a(n+1)\cdot(y-y^U) - M\cdot\left[1-\lambda(n)\right] \quad \forall n \qquad \text{(BM-3b)}$$

$$z \le x\cdot y^U + a(n)\cdot(y-y^U) + M\cdot\left[1-\lambda(n)\right] \quad \forall n \qquad \text{(BM-3c)}$$

$$z \le x\cdot y^L + a(n+1)\cdot(y-y^L) + M\cdot\left[1-\lambda(n)\right] \quad \forall n \qquad \text{(BM-3d)}$$

$$x^L \le x \le x^U, \quad y^L \le y \le y^U \qquad \text{(BM-4)}$$

Note that Meyer and Floudas[35] did not explicitly present the equivalents of Eqs. BM-3b to BM-3d for their specific generalized pooling problem.

For the bilinear terms arising in general and specific (integrated water network) process synthesis problems, Bergamini et al.[33] and Karuppiah and Grossmann[34] proposed a convex-hull reformulation. Their formulation is meant for arbitrary segment lengths (any possible arrangements of $d(n)$); hence, it is suitable for both identical (the space between the bounds of the partitioned variables is divided into equal intervals, i.e., $d(1) = \ldots = d(N)$), and nonidentical segment lengths (i.e., the space between the bounds of the partitioned variable is divided into different intervals i.e., $d(1) \neq \ldots \neq d(N)$). However, Karuppiah and Grossmann[34] mentioned some issues with the use of nonidentical segment lengths, and used identical segment length exclusively in their reported examples. Although their formulation was intended for specific process synthesis problems, its main steps can be suitably modified for $S$ in general. Then, for arbitrary segment lengths, a convex-hull formulation **CH** for $S$ based on their main ideas can be stated as follows

$$\lambda(n) = \begin{cases} 1 & \text{if } a(n) \le x \le a(n+1) \\ 0 & \text{otherwise} \end{cases} \qquad \text{(CH-0)}$$

$$\sum_{n=1}^{N} \lambda(n) = 1 \qquad \text{(CH-1)}$$

$$x = \sum_{n=1}^{N} u(n) \qquad \text{(CH-2a)}$$

$$a(n)\cdot\lambda(n) \le u(n) \le a(n+1)\cdot\lambda(n) \quad \forall n \qquad \text{(CH-2b)}$$

$$y = \sum_{n=1}^{N} v(n) \qquad \text{(CH-3a)}$$

$$y^L\cdot\lambda(n) \le v(n) \le y^U\cdot\lambda(n) \quad \forall n \qquad \text{(CH-3b)}$$

$$z \ge \sum_{n=1}^{N}\left[u(n)\cdot y^L + a(n)\cdot v(n) - a(n)\cdot y^L\cdot\lambda(n)\right] \qquad \text{(CH-4a)}$$

$$z \ge \sum_{n=1}^{N}\left[u(n)\cdot y^U + a(n+1)\cdot v(n) - a(n+1)\cdot y^U\cdot\lambda(n)\right] \qquad \text{(CH-4b)}$$

$$z \le \sum_{n=1}^{N}\left[u(n)\cdot y^L + a(n+1)\cdot v(n) - a(n+1)\cdot y^L\cdot\lambda(n)\right] \qquad \text{(CH-4c)}$$

$$z \le \sum_{n=1}^{N}\left[u(n)\cdot y^U + a(n)\cdot v(n) - a(n)\cdot y^U\cdot\lambda(n)\right] \qquad \text{(CH-4d)}$$

$$x^L \le x \le x^U, \quad y^L \le y \le y^U \qquad \text{(CH-5)}$$

The previous two formulations (**BM** and **CH**) for $S$ will serve as the bases for evaluating several novel and superior formulations that we develop next. In contrast to the literature, we use a rather intuitive and algebraic approach for our novel formulations. The first step toward our several formulations is to model the partitioning of $x$, and later, to derive the piecewise bilinear under- and overestimators (Figure 4).

## Modeling *x*-Partitions

Let $d(n) = a(n+1) - a(n)$ for $n = 1$ to $N-1$. It is clear that every value of $x$ must fall in one of the $N$ partitions.

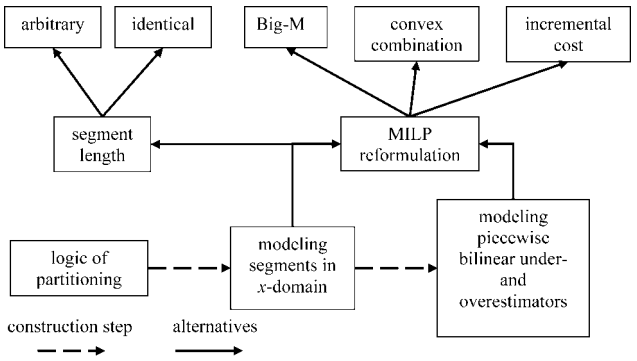


**Figure 4. Alternatives in constructing piecewise MILP under- and overestimators for bilinear programs.**

There exist several ways[44–46] of modeling this requirement. One of them is to use the binary variable $[\lambda(n)]$ as done by Eqs. CH-0 and CH-1, or Eqs. BM-0 and BM-1. Using the same variable, we can express $x$ in two different ways. One is to define a differential variable $[\Delta x(n)]$ for each segment, called as *local differential variable*, as follows

$$x = \sum_{n=1}^{N} \Big[ a(n) \cdot \lambda(n) + \Delta x(n) \Big] \tag{1a}$$

$$0 \le \Delta x(n) \le d(n) \cdot \lambda(n) \quad \forall n \tag{1b}$$

The other is to aggregate the differential variables $[\Delta x(n)]$ into a single differential variable $[\Delta x = \Delta x(1) + \Delta x(2) + \ldots + \Delta x(N)]$, called as *global differential variable*, as follows

$$x = \sum_{n=1}^{N} \Big[ a(n) \cdot \lambda(n) \Big] + \Delta x \tag{2a}$$

$$0 \le \Delta x \le \sum_{n=1}^{N} \Big[ d(n) \cdot \lambda(n) \Big] \tag{2b}$$

As far as their eventual performances in a global optimization algorithm are concerned, the differences in the aforementioned two approaches are significant. Eq. 1 requires more variables and constraints than Eq. 2, thus, models based on the former would be larger. On the other hand, since Eq. 2 can be easily derived from Eq. 1, thus, the latter cannot be tighter than the former. However, these two represent the same relaxation constructed in different variable spaces. The projections of both Eqs. 1 and 2 on the space of original variables are equivalent as can be shown easily via Fourier-Motzkin Elimination[39] of differential variables. It is indeed critical to give utmost attention to and exploit the special structure of the piecewise under- and overestimators to develop a competitive formulation/s, because as mentioned earlier, the piecewise MILP relaxations will be solved repeatedly in a global optimization algorithm, and they typically consume most of the time in each iteration. Even slight improvements will affect the overall efficiency of the global optimization algorithm, as any inefficiency in each step will propagate and eventually add up over iterations.

At this stage, it is useful to contrast our aforementioned modeling approaches (Eqs. 1 and 2) with those (Eqs. BM-2 and CH-2) from the literature. In contrast to Eq. CH-2, Eqs. 1 and 2, use differential variables $[\Delta x(n)$ and $\Delta x)]$. While both Eqs. 1 and CH-2 disaggregate variables, Eq. 1 disaggregates the differential variable $\Delta x$, rather than $x$ itself as done by Eq. CH-2. This way, Eq. 1 uses $N+1$ constraints, and Eq. 2 uses only 3 constraints as compared to $2N+1$ for Eq. CH-2, and $2N$ for Eq. BM-2. Furthermore, Eq. 1 uses $N+1$ $[x$ and $\Delta x(n)]$, and Eq. 2 uses two variables ($x$ and $\Delta x$) as compared to $N+1$ ($x$ and $u(n)$) for Eq. CH-2, and one ($x$) for Eq. BM-2. Bilinear under- and overestimators constructed from Eqs. 1 and 2 tend to have fewer nonzeros as compared to those constructed from Eqs. CH-2 and BM-2. This is because the lower bound for each differential variable is zero. These are differences in model sizes, which as we see later, do affect the quality of relaxation and overall performance significantly.

Interestingly, the following binary variable is an equivalent alternative to $\lambda(n)$ for modeling the partitioning of $x$

$$\theta(n) = \begin{cases} 1 & \text{if } x \ge a(n+1) \\ 0 & \text{otherwise} \end{cases} \quad 1 \le n \le (N-1) \tag{NF-0}$$

$$\theta(n) \ge \theta(n+1) \quad 1 \le n \le (N-2) \tag{NF-1}$$

The aforementioned variable has been used in several works[44–47] for approximating separable nonlinear functions. In particular, Padberg[45] showed that a piecewise MILP formulation, based on $\theta(n)$ for separable nonlinear functions has the property of *total unimodularity*, which means that the corresponding polytope has more of integral extreme points. This improves the quality of such a formulation rendering it *locally ideal*.[45] Using $\theta(n)$, we can express $x$ in two ways. The first is in terms of an incremental variable $[\Delta u(n)]$ in each partition called as *local incremental variable*

$$x = x^L + \sum_{n=1}^{N} \Big[ d(n) \cdot \Delta u(n) \Big] \quad 0 \le \Delta u(n) \le 1 \tag{3a}$$

$$\begin{aligned} 0 \le \Delta u(N) \le \theta(N-1) \le \Delta u(N-1) \le \theta(N-2) \\ \le \ldots \le \Delta u(2) \le \theta(1) \le \Delta u(1) \le 1 \end{aligned} \tag{3b}$$

Note that Eq. 3b make Eq. NF-1 redundant.

The second is in terms of one incremental variable $[\Delta x]$ that is common to all partitions called as *global incremental variable*

$$x = x^L + \sum_{n=1}^{N-1} \Big[ d(n) \cdot \theta(n) \Big] + \Delta x \tag{4a}$$

$$0 \le \Delta x \le d(1) + \sum_{n=1}^{N-1} \big[ \{ d(n+1) - d(n) \} \cdot \theta(n) \big] \tag{4b}$$

Similar to Eqs. 1 and 2, Eq. 3 requires more variables and constraints than Eq. 4, thus, models based on the former would be larger. On the other hand, since Eq. 4 can be easily


**Figure 5. Comparison between convex combination ($\lambda$) formulation and incremental cost ($\theta$) formulation in modeling segments in $x$-domain**

derived from Eq. 3, the latter cannot be tighter than the former. However, both represent the same relaxation constructed in different variable spaces as can be trivially shown via Fourier-Motzkin elimination of incremental variables.

Note that $\lambda(n)$, $\theta(n)$, $\Delta x(n)$, and $\Delta u(n)$ are related by

$$\lambda(1) = 1 - \theta(1)$$
$$\lambda(n+1) = \theta(n) - \theta(n+1) \quad n = 1 \text{ to } N-2$$
$$\lambda(N) = \theta(N-1)$$
$$\Delta x(n) = d(n) \cdot \left( \Delta u(n) + \sum_{n'=1}^{n} [\lambda(n')] - 1 \right)$$

Note that we need $N-1$ $\theta(n)$ variables for modeling the segments in each $x$-domain as compared to $N$ $\lambda(n)$ (Figure 5). Furthermore, unlike $\lambda(n)$, $\theta(n)$ does not require the typical disjunctive constraint (Eq. CH-0 or BM-0), as none, one, or several $\theta(n)$ can be one simultaneously. In this approach, the incremental variable in a given partition builds up on the variables in the preceding partitions to represent $x$ as in Eqs. 3 and 4.

Our approaches for modeling $x$ differ from the existing literature in one significant manner. Instead of invoking the DP reformulation strategies behind **CH** and **BM**, Eqs. 1–4 employ rather intuitive and algebraic strategies of expressing $x$ explicitly in terms of the basic binary variables of piecewise mixed-integer linear relaxation, as well as new differential and incremental variables. Using these and some other unique modeling ideas, we now develop several novel MILP formulations for the piecewise relaxation of $S$. We allow arbitrary partitions (arbitrary or nonidentical segment lengths) first, and then we assume identical segment length.

## Big-M Formulations

In the first class of our formulations, we take Eq. 1 and reformulate the continuous convex relaxation of $S$ using the big-M constraints presented for **BM**. This gives us **NF1**, which comprises Eqs. BM-0, BM-1, 1, BM-3, and BM-4.

A straightforward alternative formulation **NF2** can be obtained by replacing Eq. 1 in **NF1** by Eq. 2. However, note that $\Delta$x can be eliminated from Eq. 2 to obtain

$$x \geq \sum_{n=1}^{N} \left[ a(n) \cdot \lambda(n) \right] \tag{5a}$$

$$x \leq \sum_{n=1}^{N} \left[ a(n+1) \cdot \lambda(n) \right] \tag{5b}$$

Then, using Eq. 5 in place of Eq. 2, we get **NF2**. **NF2** comprises Eqs. BM-0, BM-1, 5, BM-3, and BM-4.

The differences (discussed earlier) in Eqs. 1, 5, and BM-2 make **NF1** and **NF2** significantly different from **BM**. **NF1** and **NF2** use fewer constraints (see Table 1) than **BM**. While **NF2** and **BM** use the same variables, **NF1** uses $N$ more variables. Thus, **NF1** and **NF2** are smaller in size. Furthermore, and more importantly, we show later that both **NF1** and **NF2** are as tight as or tighter than **BM** for the same value of $M$. As stated earlier, **NF2** uses far fewer variables and constraints, and is smaller than **NF1**. Since smaller size is often an advantage in big-M formulations, **NF2** may actually outperform **NF1**.

## Convex Combination Formulations

While **NF1**, **NF2**, and **BM** used the BM reformulation approach for piecewise relaxation, and **CH** used the CH reformulation approach; we now build on our algebraic approach to develop several novel formulations. Our second class of formulations is constructed using the *convex combination approach* (CC), which is based on the use of $\lambda$ (Eq. CH-0) as binary variables, and is free of big-M constraints. In this sense, **CH** is also a convex combination formulation.

For our convex combination formulation, we use the differential variables $\Delta x$ and $\Delta x(n)$ as defined in Eqs. 1 and 2 combined with the following

$$y = y_L + \sum_{n=1}^{N} \Delta y(n) \tag{6a}$$

**Table 1. Characteristics and *GMRR*s of Various Model/Performance Criteria of Various Models**

| Type of segments | Arbitrary | | | | | | | | | | Identical | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model type | BM | | | CC | | | | IC | | | BM | | CC | | IC |
| Feature or criterion | BM | NF1 | NF2 | TCH | CH | NF3 | NF4 | NF5 | NF6 | NF7 | NF8 | NF9 | NF10 | NF11 | NF12 |
| *GMRR*s of models for various model / performance criteria | | | | | | | | | | | | | | | |
| CPU time (s) | 15.88 | 10.39 | 9.30 | 4.34 | 2.35 | 2.93 | 1.33 | 2.21 | 1.73 | 1.23 | 8.45 | 8.09 | 1.54 | 1.21 | 1.21 |
| nonzeros | 2.11 | 2.14 | 1.96 | 2.65 | 2.15 | 2.11 | 1.41 | 1.97 | 1.53 | 1.62 | 1.95 | 1.86 | 1.91 | 1.00 | 1.27 |
| constraints | 2.49 | 2.20 | 1.88 | 3.56 | 2.20 | 2.38 | 1.05 | 2.49 | 2.18 | 1.65 | 1.83 | 1.83 | 2.00 | 1.00 | 1.60 |
| binary variables | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.00 | 1.00 | 1.00 | 1.25 | 1.25 | 1.25 | 1.25 | 1.00 |
| continuous variables | 1.00 | 3.22 | 1.00 | 7.81 | 5.53 | 7.81 | 4.14 | 7.28 | 5.53 | 3.70 | 3.22 | 1.32 | 7.81 | 4.14 | 3.70 |
| nodes | 37.21 | 16.25 | 21.26 | 3.11 | 2.40 | 2.77 | 2.25 | 1.66 | 1.67 | 1.23 | 6.91 | 22.73 | 1.65 | 1.83 | 1.49 |
| MILP objective | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| RMILP objective | 0.89 | 0.90 | 0.90 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.90 | 0.90 | 1.00 | 1.00 | 1.00 |

Note: BM = Big-M; CC = Convex Combination; IC = Incremental Cost.

$$0 \le \Delta y(n) \le (y^U - y^L)\cdot\lambda(n) \quad \forall n \tag{6b}$$

Substituting Eqs. 1a and 6a, as well as Eqs. 2a and 6a into $z = xy$, we obtain Eqs. 7a and 7b, respectively

$$z = y^L\cdot x + \sum_{n=1}^{N}\sum_{n'=1}^{N}\left[a(n)\cdot\lambda(n)\cdot\Delta y(n')\right] + \sum_{n=1}^{N}\sum_{n'=1}^{N}\Delta x(n)\cdot\Delta y(n') \tag{7a}$$

$$z = y^L\cdot x + \sum_{n=1}^{N}\sum_{n'=1}^{N}\left[a(n)\cdot\lambda(n)\cdot\Delta y(n')\right] + \Delta x\cdot\sum_{n'=1}^{N}\Delta y(n') \tag{7b}$$

The second term in the aforementioned equations involves products of binary and continuous variables $\lambda(n)\cdot\Delta y(n')$, as well as two products of continuous variables $\Delta x(n)\cdot\Delta y(n')$ and $\Delta x\cdot\Delta y(n')$. Eqs. 6a and 7b can be simplified as the following

$$z = y^L\cdot x + \sum_{n=1}^{N}\left[a(n)\cdot\Delta y(n)\right] + \sum_{n=1}^{N}\left[\Delta x(n)\cdot\Delta y(n)\right] \tag{8a}$$

$$z = y^L\cdot x + \sum_{n=1}^{N}\left[a(n)\cdot\Delta y(n)\right] + \Delta x\cdot\sum_{n=1}^{N}\Delta y(n) \tag{8b}$$

This simplification can be explained through the following two propositions. The first states that $\sum_{n=1}^{N}\sum_{n'=1}^{N}[a(n)\cdot\lambda(n)\cdot\Delta y(n')] = \sum_{n=1}^{N}[a(n)\cdot\Delta y(n)]$ in the presence of Eqs. 6b, CH-0, and CH-1. It is proven as follows: Due to Eq. 6b, if $\lambda(n) = 0$, then $\Delta y(n) = 0$. Due to Eqs. CH-0 and CH-1, only one $\lambda(n)$ can have the value of unity, while the others $\lambda(n')$ ($n' \neq n$) have the value of zero. Based on the two previous points, $\lambda(n)\cdot\Delta y(n') = 0$ if $n' \neq n$ and $\sum_{n=1}^{N}\sum_{n'\neq n}^{N}\lambda(n)\cdot\Delta y(n') = 0$. Thus, $\sum_{n=1}^{N}\sum_{n'=1}^{N}[a(n)\cdot\lambda(n)\cdot\Delta y(n')] = \sum_{n=1}^{N}[a(n)\cdot\lambda(n)\cdot\Delta y(n)]$. Note that $\lambda(n)\cdot\Delta y(n) = \Delta y(n)$ due to Eqs. 6b and CH-0. Thus, $\sum_{n=1}^{N}\sum_{n'=1}^{N}[a(n)\cdot\lambda(n)\cdot\Delta y(n')] = \sum_{n=1}^{N}[a(n)\cdot\Delta y(n)]$ is valid in conjunction with Eqs. 6b, CH-0, and CH-1.

The second proposition states that $\sum_{n=1}^{N}\sum_{n'=1}^{N}[\Delta x(n)\cdot\Delta y(n')] = \sum_{n=1}^{N}[\Delta x(n)\cdot\Delta y(n)]$ in the presence of Eqs. 1b, 6b, CH-0, and CH-1. It is proven as follows. Due to Eqs. 1b and 6b, if $\lambda(n) = 0$, then $\Delta x(n) = 0$ and $\Delta y(n) = 0$. Due to Eqs. CH-0 and CH-1, only one $\lambda(n)$ can have the value of unity, while the others $\lambda(n')$ ($n' \neq n$) have the value of zero. Based on the two previous points $\Delta x(n)\cdot\Delta y(n)' = 0$ if $n' \neq n$ and $\sum_{n=1}^{N}\sum_{n'\neq n}^{N}\Delta x(n)\cdot\Delta y(n') = 0$. Thus, $\sum_{n=1}^{N}\sum_{n'=1}^{N}[\Delta x(n)\cdot\Delta y(n')] = \sum_{n=1}^{N}[\Delta x(n)\cdot\Delta y(n)]$ is valid in conjunction with Eqs. 1b, 6b, CH-0, and CH-1.

Note that although it is possible to combine $y = y_L + \Delta y$ and $0 \le \Delta y \le (y^U - y^L)$ with Eqs. 1 and 2, this combination leads to a result similar to Eq. 8 albeit with additional variables and constraints to handle the binary and continuous products. Thus, the resulting formulation is in fact much larger, and, thus, not attractive. Note that we have shown that such a feature can be achieved in Eq. 8 without any additional variables and constraints, thus, reducing the size of the formulation.

Thus, we have successfully converted the original BLP represented by $S$ into a MIBLP represented by Eqs. CH-0, CH-1, CH-5, 1 or Eqs. 2, 6, and 8. However, more importantly, we have expressed $S$ in terms of one or more bilinear products of differential variables instead of one bilinear product $(x\cdot y)$ of original variables. Now, to convert this MIBLP into a MILP, we relax the bilinear terms in Eq. 8 using Eqs. R1 to R4. However, we have several options in this regard in terms of the variables that are used in expressing the various bilinear terms. We can relax either $\Delta x(n)\cdot\Delta y(n)$ (Eq. 8a) or $\Delta x\cdot\Delta y(n)$ (Eq. 8b). Note that $\Delta x(n)\cdot\Delta y$ is symmetric with $\Delta x\cdot\Delta y(n)$, and, thus, considered as one option rather than two, while $\Delta x\cdot\Delta y$ leads to a larger model, which is not attractive as explained previously. Therefore, while we must use $\Delta y(n)$, we can use either $\Delta x(n)$ or $\Delta x$ as variables. Thus, we have two possible options as follows:

1. Use $\Delta x(n)$ as the variable and relax $\Delta z(n) = \Delta x(n)\cdot\Delta y(n)$.
2. Use $\Delta x$ as the variable and relax $\Delta z = \Delta x\cdot\sum_{n=1}^{N}\Delta y(n)$.

Note that $\Delta z(n) \ge 0$, which can be implemented easily within a standard commercial MILP solver. Now, to use Eqs. R1 to R4 for the aforementioned options, we need the bounds of $\Delta x(n)$, $\Delta x$, and $\Delta y(n)$. Because the lower bounds for all are zero, Eq. R1 becomes redundant, and Eqs. R2 to R4 simplify as follows

$$z \ge y^U x + x^U y - x^U y^U \tag{9a}$$

$$z \le x^U y \tag{9b}$$

$$z \le y^U x \tag{9c}$$

This is also one significant difference between our approach and those in the literature. By transforming the lower bounds of all variables involved in the construction of the under- and overestimators for the bilinear term to zero, we reduce the size of the piecewise MILP relaxation problem in terms of both constraints and nonzeros.

From Eqs. 1b, 1c, 2b, and 6b, we identify the upper bounds of $\Delta x(n)$, $\Delta y(n)$, and $\Delta x$ as $\Delta a(n)\cdot\lambda(n)$, $(y^U - y^L)\cdot\lambda(n)$, and $\sum_{n=1}^{N}[\Delta a(n)\cdot\lambda(n)]$, respectively. Using them, we now relax. Substituting $\Delta z(n)$ for $z$, $\Delta x(n)$ for $x$, $\Delta y(n)$ for $y$, $\Delta a(n)\cdot\lambda(n)$ for $x^U$, and $(y^U - y^L)\cdot\lambda(n)$ for $y^U$ in Eq. 9, and simplifying, we obtain our next formulation **NF3**. **NF3** comprises Eqs. CH-0, CH-1, CH-5, 1, 6, NF3-1, and NF3-2

$$z = y^L\cdot x + \sum_{n=1}^{N}\left[a(n)\cdot\Delta y(n)\right] + \sum_{n=1}^{N}\Delta z(n) \tag{NF3-1}$$

$$\Delta z(n) \le (y^U - y^L)\cdot\Delta x(n) \quad \forall n \tag{NF3-2a}$$

$$\Delta z(n) \le d(n)\cdot\Delta y(n) \quad \forall n \tag{NF3-2b}$$

$$\Delta z(n) \ge (y^U - y^L)\cdot\left[\Delta x(n) - d(n)\cdot\lambda(n)\right] + d(n)\cdot\Delta y(n) \quad \forall n \tag{NF3-2c}$$

**NF3** is a novel formulation. In contrast to **CH**, **NF3** relaxes the bilinear product $[\Delta x(n)\cdot\Delta y(n)]$ of differential and disaggregated variables rather than $(x\cdot y)$ itself as in **CH**.

As discussed previously, the relaxations of $\Delta z(n) = \Delta x(n) \cdot \Delta y$ and $\Delta z(n) = \Delta x \cdot \Delta y(n)$ are identical (due to swapping the grid points positioning from $x$ to $y$ domain), therefore, both lead to **NF3**, making these two options basically identical. For the second option, that is, the relaxation of $\Delta z = \Delta x \cdot \sum_{n=1}^{N} \Delta y(n)$ using $\Delta x(n)$ as the variable, we get Eqs. CH-0, CH-1, CH-5, 1, 6, NF4-1, and NF4-2 as an alternate formulation of **NF4**

$$z = y^L \cdot x + \sum_{n=1}^{N} \left[a(n) \cdot \Delta y(n)\right] + \Delta z \qquad \text{(NF4-1)}$$

$$\Delta z \;\le\; (y^U - y^L) \cdot \sum_{n=1}^{N} \Delta x(n) \qquad \text{(NF4-2a)}$$

$$\Delta z \;\le\; \sum_{n=1}^{N} d(n) \cdot \Delta y(n) \qquad \text{(NF4-2b)}$$

$$\Delta z \ge (y^U - y^L) \cdot \left[x - \sum_{n=1}^{N-1} \left[a(n+1) \cdot \lambda(n)\right]\right] + \sum_{n=1}^{N} [d(n) \cdot \Delta y(n)] \qquad \text{(NF4-2c)}$$

However, note that using $\Delta x$ as a variable instead of $\Delta x(n)$ can simplify the aforementioned considerably $[\Delta x = \sum_{n=1}^{N} \Delta x(n)]$. This option gives us formulation **NF4**, which comprises Eqs. CH-0, CH-1, CH-5, 2, 6, NF4-1, NF4-2b, NF4-2c, and NF4-3

$$\Delta z \le (y^U - y^L) \cdot \Delta x \qquad \text{(NF4-3)}$$

Note that applying the theorem of Balas[40] to (DP), another formulation called as **TCH**, which cannot be looser than **CH**, can be constructed. **TCH** comprises of Eqs. CH-0–CH-3, and Eqs. TCH-1–TCH-2. Later, we discuss the connection between **CH** and **TCH**. Obviously, **TCH** belongs to the class of *convex combination* formulations

$$z = \sum_{n=1}^{N} w(n) \qquad \text{(TCH-1)}$$

$$w(n) \;\ge\; u(n) \cdot y^L + a(n) \cdot [v(n) - y^L \cdot \lambda(n)] \quad \forall n \qquad \text{(TCH-2a)}$$

$$w(n) \;\ge\; u(n) \cdot y^U + a(n+1) \cdot [v(n) - y^U \cdot \lambda(n)] \quad \forall n \qquad \text{(TCH-2b)}$$

$$w(n) \;\le\; u(n) \cdot y^L + a(n+1) \cdot [v(n) - y^L \cdot \lambda(n)] \quad \forall n \qquad \text{(TCH-2c)}$$

$$w(n) \;\le\; u(n) \cdot y^U + a(n) \cdot [v(n) - y^U \cdot \lambda(n)] \quad \forall n \qquad \text{(TCH-2d)}$$

In the Appendix, we show that all formulations that belong to the class of convex combination have equivalent discrete-to-continuous tightness. We also show that their 2nd level relaxations have a direct relationship with the bilinear envelope. However, in terms of model size, **NF4** is clearly more attractive than **NF3**, **CH**, and **TCH**.

## Incremental Cost Formulations

Our third class of formulations employs the use of $\theta$ (Eq. NF-0) as binary variables, and is called as incremental cost approach (IC), due to its incremental nature as described previously.

First, we use the local incremental variable in Eq. 3a

$$x = x^L + \sum_{n=1}^{N} [d(n) \cdot \Delta u(n)] \qquad 0 \;\le\; \Delta u(n) \;\le\; 1 \qquad \text{(3a)}$$

Multiplying by $y$ and defining $\Delta w(n) = \Delta u(n) \cdot \Delta y$ give us

$$z = x^L \cdot y + y^L \cdot (x - x^L) + \sum_{n=1}^{N} d(n) \cdot \Delta w(n) \qquad \text{(NF5-1)}$$

From Eq. 3b, we identify the bounds of $[\theta(1), 1]$ for $\Delta u(1)$, $[\theta(n), \theta(n-1)]$ for $\Delta u(n)$ from $n = 2$ to $n = N-1$, and $[0, \theta(N-1)]$ for $\Delta u(N)$. Using these and the bounds of $[0, y^U - y^L]$ for $\Delta y$ in Eqs. R1–R4, and defining $\Delta v(n) = \theta(n) \cdot \Delta y$ for $n < N$, we obtain

$$\Delta w(n) \;\ge\; \Delta v(n) \quad \forall n < N \qquad \text{(NF5-2a)}$$

$$\Delta w(1) \;\ge\; (y^U - y^L) \cdot \Delta u(1) + y - y^U \qquad \text{(NF5-2b)}$$

$$\Delta w(n) \;\ge\; (y^U - y^L) \cdot [\Delta u(n) - \theta(n-1)] + \Delta v(n-1) \quad \forall n > 1 \qquad \text{(NF5-2c)}$$

$$\Delta w(1) \;\le\; y - y^L \qquad \text{(NF5-2d)}$$

$$\Delta w(n) \;\le\; \Delta v(n-1) \quad \forall n > 1 \qquad \text{(NF5-2e)}$$

$$\Delta w(N) \;\le\; (y^U - y^L) \cdot \Delta u(N) \qquad \text{(NF5-2f)}$$

$$\Delta w(n) \;\le\; (y^U - y^L) \cdot [\Delta u(n) - \theta(n)] + \Delta v(n) \quad \forall n < N \qquad \text{(NF5-2g)}$$

To linearize the bilinear product $\Delta v(n) = \theta(n) \cdot \Delta y$ for $n < N$, we use the bounds of $[\Delta u(n+1), \Delta u(n)]$ for $\theta(n)$ from Eq. 3b, and $[0, y^U - y^L]$ for $\Delta y$ in Eqs. R1 – R4 to obtain Eqs. NF5-2a, NF5-2c, NF5-2e, and NF5-2g. Thus, no additional constraints are required for linearizing the bilinear product $\Delta v(n)$. Now, multiplying Eq. 3b by $\Delta y$ gives us

$$0 \;\le\; \Delta w(N) \;\le\; \Delta v(N-1) \;\le\; \Delta w(N-1) \;\le\; \Delta v(N-2) \;\le\; \ldots \;\le\; \Delta w(2) \;\le\; \Delta v(1) \;\le\; \Delta w(1) \;\le\; y - y^L \qquad \text{(3c)}$$

Interestingly, Eq. 3c are identical to Eqs. NF5-2a, NF5-2d, and NF5-2e. Thus, our new formulation **NF5** comprises Eqs. NF-0, CH-5, 3a, 3b, NF5-1, and NF5-2.

Note that the need for $\Delta v(n) = \theta(n) \cdot \Delta y$ for $n < N$ in **NF5** arose, because we used the tightest possible bounds of

$\Delta u(n)$ in terms of $\theta(n)$ from Eq. 3b. If we use the looser bounds of [0, 1] for $\Delta u(n)$, then we get the following in place of Eq. NF5-2

$$\Delta w(n) \;\le\; y - y^L \quad \forall n \tag{NF6-1a}$$

$$\Delta w(n) \;\le\; (y^U - y^L)\cdot \Delta u(n) \quad \forall n \tag{NF6-1b}$$

$$\Delta w(n) \;\ge\; (y^U - y^L)\cdot \Delta u(n) + y - y^U \quad \forall n \tag{NF6-1c}$$

Thus, our next formulation **NF6** comprises Eqs. NF-0, CH-5, 3a, 3b, NF5-1, and NF6-1. In addition, we can use the following from Eq. 3c

$$0 \;\le\; \Delta w(N) \;\le\; \Delta w(N-1) \;\le\; \ldots \;\le\; \Delta w(2) \;\le\; \Delta w(1) \;\le\; y - y^L \tag{NF6-2}$$

Note that Eq. NF6-2 makes Eq. NF6-1a redundant, and Eq. NF6-2 is already included inside Eq. NF5 through Eq. 3c.

In contrast to all previous formulations (**CH**, **BM**, and **NF1** – **NF4**), **NF5** and **NF6** use one fewer binary variable for each bilinear term $z$, and share the advantageous property mentioned earlier. Moreover, unlike the convex combination formulations presented previously, these incremental cost formulations do not require the disaggregation of $y$. It is clear that **NF5** is as tight as or tighter than **NF6**, but **NF6** uses fewer variables and constraints. Nevertheless, further study (see Appendix) shows that all the projected feasible regions of these formulations in the space of variables $\{x, y, z\}$ are equivalent.

For our next formulation, we use the following global incremental variable from Eq. 4a, which is common for all partitions. The use of such a variable makes **NF7** contain less continuous variables than **NF5** and **NF6**

$$x = x^L + \sum_{n=1}^{N-1} \left[d(n)\cdot\theta(n)\right] + \Delta x \tag{4a}$$

Using the aforementioned and defining $\Delta v(n) = \theta(n)\cdot\Delta y$ and $\Delta w = \Delta x\cdot\cdot\,\Delta y$, we obtain

$$z = y^L\cdot x + x^L\cdot y - x^L\cdot y^L + \sum_{n=1}^{N-1}[d(n)\cdot\Delta v(n)] + \Delta w \tag{NF7-1}$$

For linearizing $\Delta v(n)\cdot = \theta(n)\cdot\Delta y$, we use the bounds $[\theta(2), 1]$ for $\theta(1)$, $[\theta(n+1), \theta(n-1)]$ for $\Delta\theta(n)$ with n from 2 to $N-1$, $[0, \theta(N-1)]$ for $\theta(N)$, and $[0, y^U - y^L]$ for $\Delta y$ in Eqs. R1–R4 to obtain

$$0 \le \Delta v(N-1) \le \Delta v(N-2) \le \ldots \le \Delta v(2) \le \Delta v(1) \le y - y^L \tag{NF7-2a}$$

$$\Delta v(1) \ge (y^U - y^L)\cdot\theta(1) + y - y^U \tag{NF7-2b}$$

$$\Delta v(n) \ge (y^U - y^L)\cdot[\theta(n) - \theta(n-1)] + \Delta v(n-1) \quad 2 \le n \le N-2 \tag{NF7-2c}$$

$$\Delta v(N-1) \le \theta(N-1)\cdot[y^U - y^L] \tag{NF7-2d}$$

Finally, to linearize $\Delta w = \Delta x\cdot\Delta y$, we use the bounds $[0, d(1) + \sum_{n=1}^{N-1}[\{d(n+1) - d(n)\}\cdot\theta(n)]]$ for $\Delta x$ and $[0, y^U - y^L]$ for $\Delta y$ in Eq. 9. This gives us

$$\Delta w \le d(1)\cdot(y - y^L) + \sum_{n=1}^{N-1}[\{d(n+1) - d(n)\}\cdot\Delta v(n)] \tag{NF7-3a}$$

$$\Delta w \le (y^U - y^L)\cdot\Delta x \tag{NF7-3b}$$

$$\Delta w \ge \Delta x\cdot(y^U - y^L) + d(1)\cdot(y - y^U) + \sum_{n=1}^{N-1}[\{d(n+1) - d(n)\}\cdot\{\Delta v(n) - (y^U - y^L)\cdot\theta(n)\}] \tag{NF7-3c}$$

It is obvious that **NF7** has a more attractive size than **NF5** and **NF6**.

**NF1–NF7** all allowed arbitrary partitions of $x$. As mentioned earlier, the previous works[34,35] found it easier to use identical rather than arbitrary segment lengths in their reported case studies. However, they[33–35] did not present any fine-tuned versions of their general formulations (for arbitrary segment lengths) for the special and simplified case of identical segment length. We now develop five such formulations and show that such tailoring does indeed lead to a more compact formulation, which can have significant effect on computational performance.

## Formulations with Identical Segments

We use one single segment length ($d$) for all partitions. Thus

$$\begin{aligned} d(n) &= d = (x^U - x^L)/N \\ a(n) &= x^L + (n-1)\cdot d \quad \forall n < N \end{aligned} \tag{11}$$

One major consequence of using identical segments is that we need not use $N$ local differential variables $[\Delta x(n)]$ any more. One single global $\Delta x$ and Eq. 2 are sufficient without compromising tightness. With this, Eq. 2 reduces to

$$x = x^L + d\cdot\sum_{n=1}^{N}[(n-1)\cdot\lambda(n)] + \Delta x \tag{12a}$$

$$0 \le \Delta x \le d \tag{12b}$$

We right away reduce one constraint, as Eq. 12b, in contrast to Eq. 2b, is just a bound. Applying the identical segment length approach to the Big-M class of formulations, we can easily construct **NF8** and **NF9**. These two are the simplified versions of **NF1** and **NF2** for cases involving identical segment length, respectively. Formulation **NF8** comprises of Eqs. BM-0, BM-1, 1a, 12b, BM-3, and BM-4, while Formulation **NF9** comprises of Eqs. BM-0, BM-1, 12b, BM-3, and BM-4.

Substituting from Eqs. 11 and 12a into Eq. 9b, and relaxing $\Delta z(n) = \Delta x\cdot\Delta y(n)$ as done previously forces us to use $\Delta x(n)$ again to avoid nonlinearity. In other words, our next

formulation **NF10** for identical segments is nothing but **NF3** simplified for identical segments. It comprises Eqs. CH-0, CH-1, CH-5, 12b, 6, NF3-2a, NF8-1, NF8-2, and NF8-3

$$x = x^L + \sum_{n=1}^{N}[d \cdot (n-1) \cdot \lambda(n) + \Delta x(n)] \qquad \text{(NF8-1)}$$

$$z = y^L \cdot x + x^L \cdot y - x^L \cdot y^L + d \cdot \sum_{n=1}^{N}[(n-1) \cdot \Delta y(n)] + \sum_{n=1}^{N} \Delta z(n) \qquad \text{(NF8-2)}$$

$$\Delta z(n) \le d \cdot \Delta y(n) \quad \forall n \qquad \text{(NF8-3a)}$$

$$\Delta z(n) \ge (y^U - y^L) \cdot \Delta x(n) + d \cdot \Delta y(n) - d \cdot (y^U - y^L) \cdot \lambda(n) \quad \forall n \qquad \text{(NF8-3b)}$$

Then, by relaxing rather than, we obtain our second formulation **NF11** for identical segments. It comprises Eqs. CH-0, CH-1, CH-5, 6, NF4-3, NF9-1, and NF9-2

$$z = y^L \cdot x + x^L \cdot y - x^L \cdot y^L + d \cdot \sum_{n=1}^{N}[(n-1) \cdot \Delta y(n)] + \Delta z \qquad \text{(NF9-1)}$$

$$\Delta z \le d \cdot (y - y^L) \qquad \text{(NF9-2a)}$$

$$\Delta z \ge (y^U - y^L) \cdot \Delta x + d \cdot (y - y^U) \qquad \text{(NF9-2b)}$$

Note that **NF11** uses fewer constraints and variables than **NF10**, and it is clear that **NF11** is a simplification of **NF4** for cases with identical segment length.

The next formulation **NF12** for identical segments is obtained from **NF7**, where we can set $d(n+1) - d(n) = 0$. Due to using identical segment length, **NF12** has less number of nonzeros as compared with **NF7**. In this formulation, Eq. 4b reduces to Eq. 12b, Eq. NF7-3 reduces to Eq. NF4-3 and NF9-2, and Eq. 4a becomes

$$x = x^L + d \cdot \sum_{n=1}^{N-1} \theta(n) + \Delta x \qquad \text{(NF10-1)}$$

Therefore, **NF12** comprises Eqs. 12b, NF-1, NF4-3, NF7-1, NF7-2, NF9-2, and NF10-1. Similar with **NF8 – NF9, NF12** require only a single global variable.

While the relative sizes and tightness of **BM, CH, TCH**, and **NF1 – NF12** are clear (see Table 1), it is not possible to predict the best of them in overall computational efficiency. Therefore, it is necessary for us to evaluate them numerically on a variety of problems. However, all these novel formulations (**TCH** and **NF1 – NF12**) possess stronger discrete-to-continuous relaxations as compared to BM (see Remark and Appendix).

## Case Studies

We use two case studies from the literature to derive three test problems (Examples 1, 2a, and 2b) for a comprehensive numerical comparison of the effectiveness of various models (**BM, CH, TCH**, and **NF1–NF12**). The first case study, from which we derive Example 1, is from Karuppiah and Grossmann.[34] It involves integrated water network synthesis. The second, from which we derive Examples 2a and 2b, is from page 44–46 of Floudas et al.[48] It involves the sequencing of distillation columns for nonsharp separation of a three-component mixture (propane, isobutene, and n-butane) into two products. Since a complete global optimization algorithm is not the focus of this article, we restrict ourselves to the lower-bounding problem at the root node only. Thus, our approach in this work is to embed the various models (**BM, CH, TCH**, and **NF1–NF12**) within the respective mathematical programming formulations used by the two case studies in the literature, and solve for lower bounds at the root nodes. Since the reader can refer the original references for full details on the two test case studies, we mention only those details that are different and/or essential for an adequate understanding of this work.

A fair, well-planned, extensive, and comprehensive procedure is essential[49] for a reliable assessment of MILP models based on a numerical study. To achieve a solid comparison, we solve the three test problems (Examples 1, 2a, and 2b) for several numbers of partitions and several sets of grid-point positions. For generating the latter in a convenient manner, we use the following

$$a(n) = x^L + \left(\frac{n-1}{N}\right)^{\gamma} \cdot (x^U - x^L) \quad \gamma \ge 0, \quad \forall n < N$$
$$a(N) = x^U$$
$$d(n) = \left[\left(\frac{n}{N}\right)^{\gamma} - \left(\frac{n-1}{N}\right)^{\gamma}\right] \cdot (x^U - x^L) \quad \forall n$$

As $\gamma \to 1$, the interior grid points (those except the first $a(1) = x^L$, and last points $a(N) = x^U$) become equally distributed, and as $\gamma \to 0$ ($\infty$), they move toward $x^U$ ($x^L$). Thus, $\gamma = 1$ corresponds to the case of identical segment length.

For all runs, we used a Dell Precision PW690 workstation with 3 GHz Intel Xeon single CPU, 64 GB RAM, and Windows XP Professional x64 as the operating system. GAMS 22.2/CPLEX 10 was used for all the LP and MILP problems, and GAMS 22.2/CONOPT 3 for all the NLP problems. The relative gap tolerance was set to zero in all cases to ensure solution optimality. We used $M = (x^U - x^L) \cdot (y^U - y^L)$ in all big-M constraints for a bilinear product $xy$.

## Example 1

We use Example 4 from Karuppiah and Grossmann[34] as the basis for our Example 1. For referencing the details in the original case study, we use KG to denote Karuppiah and Grossmann.[34] Thus, we call the original case study as Example KG-4, and use the same notation for equations, figures, tables, and sections in Karuppiah and Grossmann.[34] Example KG-4 is the largest problem in the study of Karuppiah and Grossmann,[34] and was represented as having industrial scale

by them. It involves five process units using water, three water-treatment units, and three contaminants. The problem was represented (Figure KG-18) as a superstructure similar to that in Takama et al.[2] The nodes in the superstructure are mixers, splitters, water-using processes, and water-treatment plants, and the arcs are the streams connecting the units. The mathematical programming formulation proposed by Karuppiah and Grossmann[34] employs flow and composition variables for each stream (arc), and total (Eqs. KG-2, KG-4, KG-6, and KG-8) and component mass balances (Eqs. KG-5, KG-7, and KG-9) for all unit (node). All balances are linear except the component mass balances (Eq. KG-3) for mixers, which are bilinear. Tables KG-7 and KG-8 in section KG-7.4, list all the numerical data for this example.

The problem is a BLP, where nonconvexities are due to the mixing of water streams with different compositions. It contains 348 variables, 312 constraints, and 234 bilinear terms (Table KG-9a). While Example KG-4 included concave univariate terms describing the size effect of plant design in the objective function (Eq. KG-1b), we use a linear objective function in our study, as such nonconvexities are not the primary focus of this study. Thus, our objective in Example 1 is to minimize the total amount of fresh water usage and wastewater treated (Eq. KG-1a). This is the only difference between Examples KG-4 and 1.

As for the solution algorithm, we use the nonredundant bound-strengthening cuts (Eq. KG-15), the logical cuts (Eq. KG-16), the akin bound-contraction preprocessing procedure (Step 1 of Section KG-6), and the partitioning of the total flow rate variables as done by Karuppiah and Grossmann.[34]

For this example, we used $N = 2$, $N = 3$, and $N = 4$ for each of **BM, CH, TCH**, and **NF1–NF12**, and 10 different sets of grid-point positions ($\gamma = 0.25$, 0.50, 0.75, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, and 4.00) for each $N$. In other words, we made 315 runs for Example 1.

## Example 2

The case study from page 44-46 of Floudas et al.[48] is also a BLP with nonconvexities, due to the products of flow rates and compositions. It comprises 24 variables, 18 constraints, and 12 bilinear terms. In this case, we partition along all flow rate variables. We use two sets of variable bounds in this case study. In Example 2a, we use the variable bounds (upper and lower) reported by Floudas et al.[46] In Example 2b, we contract the upper bounds on all flow rate variables to 180 kgmol/h, and use a new lower bound of 10 kgmol/h for the flow rate of stream 18. The bounds on other variables were kept unchanged. Note that these new bounds still contain the global optimum of the original problem as reported.

For Examples 2a and 2b, we used $N = 10$, $N = 12$, and $N = 15$ for each of **BM**, **CH**, **TCH**, and **NF1–NF12,** and the same 10 sets of grid-point positions ($\gamma = 0.25$, 0.50, 0.75, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, and 4.00) for each $N$. In other words, we made 630 runs for Examples 2a and 2b together. We used larger numbers of partitions for these examples, because we wanted to examine the effects of large numbers of segments. This particular case study made it possible for us to do this, because it is much smaller than Example 1. Having larger numbers of segments also magnifies the differences in the computational performances of the various models, which makes model ranking easier and more reliable.

## Results and Discussion

For evaluating the computational performance of various models, we use relative rather than absolute solution times to eliminate the effect of problem-to-problem variation. This enables us to compare several different formulations across a variety of test problems with different numbers of segments, grid-point positions, variable bounds, and problem structures. As suggested by several researchers,[50,51] geometric mean relative rank (GMRR) is a useful measure in this regard. As suggested by Liu and Karimi,[51] GMRR can also be used to obtain relative model ranks for not just solution times, but also other criteria, such as numbers of binary variables, continuous variables, constraints, nodes, as well as optimal MILP and RMILP values. $GMRR(m,c)$ for a formulation or model $m$ and criterion $c$ is defined as

$$GMRR(m,c) = \sqrt[p]{\prod_{p=1}^{P} \frac{C(m,p)}{C^*(p)}}$$

where, $p$ refers to a test problem, $P$ is the total number of problems, $C(m,p)$ is the value of criterion $c$ for problem $p$, and $C^*(p)$ is the best value of criterion $c$ across all models used to solve problem $p$. We use the minimum as the best for criteria, such as solution times, and numbers of binary variables, continuous variables, and constraints, while maximum as the best $C^*(p)$ for MILP and RMILP values. We set 8,000 CPU s as the maximum computation time for each run. If a model fails to attain the global optimal solution within 8,000 CPU s, then we take its solution time as 8,000 CPU s.

Table 1 clearly shows that formulations differ significantly in computational performance. Since the lower-bounding problem is typically the most time-consuming step in each iteration and is solved repeatedly, even a slight improvement in the computational efficiency of a lower-bounding procedure can significantly affect the overall efficiency of a global minimization algorithm (e.g., outer-approximation, spatial branch-and-bound). Based on Table 1, **NF4**, **NF6** and **NF7** seem to be the most competitive among the models with arbitrary segment lengths. These three formulations have CPU time *GMRR*s of 1.33, 1.73, and 1.23, respectively, which are significantly better than those of the existing formulations from the literature (*GMRR*(**BM**, CPU time) = 15.88, *GMRR*(**CH**, CPU time) = 2.35). However, **NF11** and **NF12** offer even better general performance although restricted to cases involving only identical segment length (*GMRR*(**NF11**, CPU time) = 1.21, *GMRR*(**NF12**, CPU time) = 1.21). On the other hand, the big-M based formulations (i.e., **BM**, **NF1 – NF2**, **NF8 - NF9**) are not at all competitive in terms of solution efficiency, having CPU time *GMRR*s of 8.09 to 15.88. In addition, most of the BM models fail to reach the optimal solution of the 1st level relaxation within the maximum limit of computation time (8000 CPU s). TCH (*GMRR*(TCH, CPU time) = 4.34), which theoretically represents the convex hull of the 2nd level relaxation, generally performs better than BM formulations, although it is less competitive compared to other formulations.

All incremental cost models obviously use the fewest binary variables (*GMRR*(*m*, binary variable) = 1 vs. *GMRR*(*m*, binary variable) = 1.25 for others). The models employing identical segment length generally use fewer nonzeros and constraints compared to their more general counterparts. For instance, *GMRR*(**NF11**, nonzeros) = 1, and *GMRR*(**NF11**, constraint) = 1 as compared to *GMRR*(**NF4**, nonzeros) = 1.41, and *GMRR*(**NF4**, constraint) = 1.05. In fact, **NF11** has the fewest nonzeros and constraints of all models. **BM** and **NF2** have the fewest number of continuous variables (*GMRR*(**BM**, continuous variable) = *GMRR*(**NF2**, continuous variable) = 1). This is expected, since **BM** and **NF2** do not require additional variables.

All convex combination and incremental cost formulations provide the same lower bound for the 2nd level relaxation; they all have a *GMRR*(*m*, RMILP) = 1. In other words, their RMILP values are identical to the value for LP relaxation. Further study (see Appendix) shows that the 2nd level relaxation of the convex combination and incremental cost formulations recovers the convex envelope of the bilinear terms in the continuous space of original variables $\{x, y, z\}$. This issue is discussed in the following section (see Remark and Appendix). Since all these convex combination and incremental cost models exhibit similar tightness in the 2nd level relaxation, their size plays an important role in the ease of solving them. **TCH**, **NF3**, and **NF5**, do not perform better due to their considerably larger sizes, while having similar 2nd level relaxation quality as those of smaller models, that is, **NF4**, **NF6**, and **NF7** (see Table 1). This fact also serves as the incentive toward the use of identical segment length, since identical segment length formulations offer smaller size. Conversely, with their worse values of *GMRR*(*m*, RMILP), the big-M based formulations are significantly looser compared to all other formulations, which causes their poor performance. This is because the partitioned feasible regions in the piecewise relaxation are disjoint.

Nevertheless, solving these models using state-of-the-art solvers, such as CPLEX, no guarantee that a single model outperforms all others in all cases. For instance, in Example 1 with N = 4, and $\gamma = 4$, **NF4** takes 151.8 CPU s, while **NF7** takes 502.1 CPU s, although GMRR analysis ranks **NF7** better than **NF4** in the CPU time. Several other factors, other than model size and relaxation quality, such as the algorithm implemented inside CPLEX are equally important. This makes it even harder to predict accurately the performance of a certain formulation. However, as suggested by Liu and Karimi[51] for batch process scheduling problems, the idea of using competitive models or formulations in parallel in multi-CPU machines, may be worth exploring. In this *parallel lower bounding scheme*, several competitive formulations run together in each iteration to compute the lower bound for the global minimization problem, and once one of the formulations finds the optimal 1st level relaxation solution, all other formulations are stopped and the global optimization algorithm proceeds to the next step. In that sense, it is crucial to develop several competitive models as we have done in this work. The results obtained from our case studies indicate that convex combination formulations and incremental cost formulations may complement each other. Specifically for this work, **NF4**, **NF6**, and **NF7** in tandem seem the most competitive choice for arbitrary segment lengths. Moreover, **NF11** and **NF12** seem the most competitive for identical segment length.

## Remarks

Although our main aim in this study is to develop and evaluate different formulations or models for piecewise MILP relaxation, it is useful to analyze our results further to gain some valuable insights into the properties of such relaxations. These insights may prove useful in improving existing algorithms or developing novel algorithms, which we hope to report in near future.

The main goal of piecewise relaxation is to improve the quality of overall piecewise MILP relaxation compared to the conventional LP relaxation, which uses one segment. Therefore, it would be good to measure the improvement in relaxation quality by means of some metrics. Recall that the piecewise relaxation involves two levels. The 1st level involves the relaxation of a NLP into a MILP, and the 2nd level involves the relaxation of the MILP into a RMILP. To measure the extents of improvements in the qualities of these two relaxations solely due to piecewise partitioning, we define two gains, namely *piecewise gain* (*PG*), and *relaxed piecewise gain* (*RPG*) for a model $m$ in a global minimization problem as follows.

$$PG(m,p,N,g) = \begin{cases} \dfrac{MILP(m,p,N,g) - LP(p)}{|LP(p)|} & \text{if } LP(p) \neq 0 \\ (m,p,N,g), & \text{otherwise} \end{cases}$$

$$RPG(m,p,N,g) = \begin{cases} \dfrac{RMILP(m,p,N,g) - LP(p)}{|LP(p)|} & \text{if } LP(p) \neq 0 \\ RMILP(m,p,N,g), & \text{otherwise} \end{cases}$$

where, $p$ is a NLP problem solved by model $m$ that uses $N$ segments, and set $g$ of grid-point positions, $LP(p)$ is the optimal objective value of the LP (1-segment) relaxation of $p$, $MILP(m,p,N,g)$ is the optimal objective value of the 1st level relaxation (piecewise MILP model $m$), and $RMILP(m,p,N,g)$ is the optimal objective value of the 2nd level relaxation (the RMILP of $m$). For a global maximization problem, the numerators in the definitions of *PG* and *RPG* are multiplied by $-1$.

$PG = 0$ and $RPG = 0$ mean no gains from the piecewise relaxation, with higher values being more desirable. Positive value of *PG* (*RPG*) indicates that the 1st (2nd) level relaxation is stronger than the LP relaxation, while negative value of *PG* (*RPG*) points the other way. Tables 2 and 3 show the values of *PG* and *RPG*, from which we draw the following observations.

*Remark 1.* $RPG(m,p,N,g) \leq 0$ irrespective of $m$, $N$, and $g$ for every $p$, as $RMILP(m,p,N,g) \leq LP(p)$ [$RMILP(m,p,N,g) \geq LP(p)$] in a global minimization (maximization) problem as proven in Theorem 1 (see Appendix).

Note that the results from Example 1 show that this condition ($RPG \leq 0$) also applies to cases where strengthening cuts are used. Of course, this comparison is made, when the

**Table 2. Piecewise Gains (*PG*) for Various *N* (Number of Segments) and $\gamma$ (Grid Positioning)**

| Example | *N* | $\gamma = 0.25$ | $\gamma = 0.50$ | $\gamma = 0.75$ | $\gamma = 1.00$ | $\gamma = 1.50$ | $\gamma = 2.00$ | $\gamma = 2.50$ | $\gamma = 3.00$ | $\gamma = 3.50$ | $\gamma = 4.00$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0.021 | 0.045 | 0.063 | 0.075 | 0.084 |
| | 3 | 0 | 0 | 0 | 0 | 0.05 | 0.074 | 0.088 | 0.097 | 0.102 | 0.104 |
| | 4 | 0 | 0 | 0 | 0.028 | 0.074 | 0.093 | 0.103 | 0.107 | 0.108 | 0.108 |
| 2a | 10 | 0 | 0 | 0 | 0 | 0.219 | 0.348 | 0.373 | 0.307 | 0.347 | 0.291 |
| | 12 | 0 | 0 | 0 | 0 | 0.304 | 0.403 | 0.406 | 0.408 | 0.355 | 0.314 |
| | 15 | 0 | 0 | 0 | 0 | 0.454 | 0.492 | 0.508 | 0.472 | 0.485 | 0.438 |
| 2b | 10 | 0 | 0 | 0 | 0 | 0.188 | 0.189 | 0.214 | 0.18 | 0.185 | 0.136 |
| | 12 | 0 | 0 | 0 | 0.089 | 0.246 | 0.229 | 0.225 | 0.207 | 0.184 | 0.246 |
| | 15 | 0 | 0 | 0 | 0.136 | 0.262 | 0.277 | 0.25 | 0.246 | 0.243 | 0.214 |

same group of cuts is used in both the LP relaxation, as well as the piecewise MILP relaxation.

*Remark 2.* $RPG(m,p,N,g) = 0$ for all convex combination and incremental cost type models (i.e., **TCH**, **CH**, **NF3 – NF7**, **NF11 – NF12**) for all test runs in this work.

Interestingly, the bounds provided by all convex combination and incremental cost formulations in the 2nd level relaxation are the same, and they are equivalent with the bounds provided by the convex continuous envelope. Even in all the case studies performed in this work, the optimal values of all variables obtained by the 2nd level relaxation of the convex combination and incremental cost formulations are the same with the ones obtained by the LP relaxation. Further study shows that the projections of all convex combination and incremental cost formulations into the space of original variables $\{x, y, z\}$ yield the same feasible region as the one represented by the continuous convex envelope (see Appendix). This fact explains the result stated in Remark 2.

*Remark 3.* $RPG(m,p,N,g) \leq 0$ for several test runs using the big-M based models (i.e., **BM**, **NF1**, **NF2**, **NF8**, and **NF9**).

This is not surprising, as the big-M based models are known to give looser 2nd level relaxations than convex-hull reformulations in many problems, and indeed do so prominently in this work. Interestingly, big-M formulations can give $RPG = 0$, even when the variable bounds are very loose as seen in Example 2a. Moreover, *RPG* for **BM** is identical to those of **NF1** and **NF2** in Example 1 for $N = 2$ irrespective of grid-point positioning. This is possible because these three formulations become more similar for $N = 2$ compared to higher values of *N*.

*Remark 4.* $RPG(m,p,N,g)$ varies with *N*, *g*, big-M values, and variable bounds for the big-M based models (i.e., **BM**, **NF1**, **NF2**, **NF8**, and **NF9**).

This is in contrast to the convex combination and incremental cost models (i.e., **TCH**, **CH**, **NF3 – NF7**, **NF11 – NF12**) that have $RPG = 0$ irrespective of *m*, *N*, *g*, and variable bounds for the problems tested in this work. While looser 2nd level relaxations may be expected from big-M models, it is interesting to note that their relaxations could be worsened by poor choices of *N*, *g*, big-M, and variable bounds.

*Remark 5.* $RPG(m^*, p, N, g) \geq RPG(\mathbf{BM}, p, N, g)$ for $m^* =$ **NF1**, **NF2**, **NF8**, **NF9**.

Although all these models (i.e., **BM**, **NF1**, **NF2**, **NF8**, and **NF9**) use big-M constraints, the difference between **BM** and the others is that the latter do not use big-M constraints for modeling the partitioning of *x*. As discussed previously, a MILP piecewise relaxation model for *S* involves two disjunctive modeling, one for partitioning *x*, and the other for the bilinear under- and overestimators. While all three models (BM, NF1, and NF2) use the same constraints for the bilinear under- and overestimators, BM uses big-M type constraints for modeling *x* partitions with $a(n) - x^L$ and $x^U - a(n+1)$ as big-M values. By modeling *x* partitions without using the big-M constraints, **NF1**, **NF2**, **NF8**, and **NF9** achieve better relaxation quality in the 2nd level relaxation, and, thus, better RPG values. As has been discussed previously, these four fomulations exhibit similar tightness in the 2nd level relaxation.

*Remark 6.* $PG(m,p,N,g) \geq 0$ irrespective of *m*, *N*, and *g* for every *p*, as $MILP(m,p,N,g) \geq LP(p)$ [$MILP(m,p,N,g) \leq$

**Table 3. Relaxed Piecewise Gains (*RPG*) for Various *N* (number of segments) and $\gamma$ (Grid Positioning)**

| Example | 1 | | | 2b | | |
|---|---|---|---|---|---|---|
| $\gamma$ | N | BM | NF1–NF2, NF8–NF9 | N | BM | NF1–NF2, NF8–NF9 |
| 0.25 | 2 | −0.055 | −0.055 | 10 | −0.210 | −0.177 |
| 0.50 | | −0.060 | −0.060 | | −0.212 | −0.191 |
| 0.75 | | −0.064 | −0.064 | | −0.214 | −0.202 |
| 1.00 | | −0.068 | −0.068 | | −0.216 | −0.209 |
| 1.50 | | −0.075 | −0.075 | | −0.218 | −0.216 |
| 2.00 | | −0.082 | −0.082 | | −0.219 | −0.218 |
| 2.50 | | −0.087 | −0.087 | | −0.219 | −0.219 |
| 3.00 | | −0.091 | −0.091 | | −0.220 | −0.220 |
| 3.50 | | −0.094 | −0.094 | | −0.220 | −0.220 |
| 4.00 | | −0.097 | −0.097 | | −0.220 | −0.220 |
| 0.25 | 3 | −0.072 | −0.058 | 12 | −0.212 | −0.178 |
| 0.50 | | −0.076 | −0.065 | | −0.214 | −0.193 |
| 0.75 | | −0.080 | −0.071 | | −0.215 | −0.204 |
| 1.00 | | −0.083 | −0.076 | | −0.216 | −0.211 |
| 1.50 | | −0.089 | −0.085 | | −0.218 | −0.217 |
| 2.00 | | −0.094 | −0.092 | | −0.219 | −0.219 |
| 2.50 | | −0.097 | −0.096 | | −0.220 | −0.219 |
| 3.00 | | −0.099 | −0.098 | | −0.220 | −0.220 |
| 3.50 | | −0.100 | −0.100 | | −0.220 | −0.220 |
| 4.00 | | −0.101 | −0.101 | | −0.220 | −0.220 |
| 0.25 | 4 | −0.080 | −0.060 | 15 | −0.213 | −0.180 |
| 0.50 | | −0.084 | −0.068 | | −0.215 | −0.195 |
| 0.75 | | −0.087 | −0.075 | | −0.216 | −0.206 |
| 1.00 | | −0.090 | −0.082 | | −0.217 | −0.212 |
| 1.50 | | −0.094 | −0.091 | | −0.219 | −0.218 |
| 2.00 | | −0.097 | −0.096 | | −0.219 | −0.219 |
| 2.50 | | −0.099 | −0.099 | | −0.220 | −0.220 |
| 3.00 | | −0.101 | −0.100 | | −0.220 | −0.220 |
| 3.50 | | −0.101 | −0.101 | | −0.220 | −0.220 |
| 4.00 | | −0.102 | −0.101 | | −0.220 | −0.220 |

Note: Convex Combination and Incremental Cost Formulations give RPG = 0 for all cases tested. All formulations give RPG = 0 for Example 2a.

$LP(m,p)$] in a global minimization (maximization) problem as proven in Theorem 2 (see Appendix).

*Remark 7.* For given $p$, $N$, $g$, and variable bounds, $PG(m,p,N,g)$ is the same for all $m$, but computational efficiency varies.

This is because all models have the same optimal MILP objective for any given $p$ irrespective of $N$ and $g$, and, thus, equivalent tightness in the 1st level relaxation. It is important to note that the optimal here refers to a MILP solution with zero relative gap. However, generally big-M based models (i.e., **BM**, **NF1**, **NF2**, **NF8**, and **NF9**) require significantly more computational time than the remaining models as discussed before.

*Remark 8.* For a given $p$, $PG(m,p,N,g)$ depends on $N$, $g$, and variable bounds.

Normally $PG(m,p,N,g)$ increases with larger $N$ for given $m$, $p$, $g$, and variable bounds as the tightness of the 1st level relaxation improves. However, when the variable bounds are loose, as for instance in Example 2a where identical segment length was used ($\gamma = 1$), increasing $N$ from 10 to 15 has no effect on $PG$ ($PG$ remains at the value of 0). Therefore, tight variable bounds are crucial. For instance, consider Example 2b, where identical segment length was used with tighter bounds, increasing $N$ from 10 ($PG = 0$) to 15 ($PG = 0.136$) has relatively significant effect on $PG$. Nevertheless, $PG$ may not improve through tighter bounds, even though the 1st level relaxation becomes tighter. This is because the variable bounds change the $LP(p)$ value as well. For instance, where $N = 10$ and $\gamma = 1.50$, $PG = 0.219$ in Example 2a (looser variable bounds), while $PG = 0.188$ in Example 2b (tighter variable bounds).

It is expected, and clear from our results, that grid-point positioning affects $PG$. However, more significantly, the use of identical segment length (via special formulations such as **NF11** – **NF12**) is not necessarily the best for attaining a higher $PG$, even though it seems more efficient for each lower bounding computation. This is evident from the results on Examples 1 and 2 in Table 2. For instance, examine Example 2a for $N = 15$ and grid points positioned via $\gamma = 2.50$ gives a considerably higher $PG$ (0.508) as compared to that for identical segment length positioning ($PG = 0$). Thus, note that using identical segments need not be the best overall for a global optimization algorithm, because of the trade-off between PG and the computational efficiency for each lower bounding problem.

## Conclusion

In this article, we presented 13 novel formulations (i.e., **TCH** and **NF1**–**NF12**) for piecewise MILP under- and overestimators for global optimization of bilinear programs. These were derived using three systematic approaches: big-M, convex combination, and incremental cost, and two segmentation schemes: arbitrary and identical segment lengths. These systematic approaches and segmentation schemes can also derive the existing formulations. We compared their performance with the existing formulations in the literature (i.e., **BM** and **CH**) using two case studies arising in process network synthesis: integrated water systems synthesis and nonsharp distillation column sequencing. Based on 945 runs with various numbers of segments, grid-points positioning, and variable bounds, we demonstrated that our novel formulations give superior relaxation tightness as compared to existing **BM** formulations, especially those constructed via convex combination and incremental cost approaches. Some of them are much more compact than those existing in the literature, with no loss of relaxation quality. Note that the incremental cost formulations require one less binary variable for modeling segments in each $x$-domain as compared to other formulations. Further reduction in formulation size is achieved through the use of identical segment length. In general, our novel formulations are more efficient than all existing formulations. **NF4**, **NF6**, and **NF7** in tandem seem the most competitive choice for arbitrary segment lengths, while **NF11** and **NF12** seem the best for identical segment length. Parallel use of convex combination and incremental cost formulations with compact size seem favorable for practical purpose. On the other hand, big-M formulations, with their considerably inferior relaxation quality, seem not competitive for piecewise relaxation in this study.

Piecewise relaxation involves two levels. The 1st level transforms the original nonconvex NLP into a MILP. The 2nd level transforms the MILP into a LP. In this work, we introduced $PG$ (piecewise gain) and $RPG$ (relaxed piecewise gain) as metrics to measure the effectiveness of piecewise relaxation by comparing the tightness of the LP relaxation with those of the 1st and 2nd level relaxations, respectively. Number of segments, grid-points positioning, and variable bounds are among the factors that affect $PG$ and $RPG$. The use of identical segment length is not necessarily the best for attaining a higher $PG$ while $PG$ does not depend on formulation. All formulations are shown and proved to have $PG \geq 0$ and $RPG \leq 0$ with $RPG = 0$ is a necessary condition for those representing convex hull type relaxation of the 2nd level relaxation. All convex combination and incremental cost formulations have $RPG = 0$.

## Acknowledgments

We would like to thank the National University of Singapore and the Japan International Cooperation Agency through the ASEAN University Network/South East Asia Engineering Development Network (AUN/SEED-Net) scholarship program, as well as the Ministry of Education, Singapore, through AcRF Tier 1 grant R-279-000-182-112 for providing financial support for this work.

## Literature Cited

1. Haverly CA. Studies of the behaviour of recursion for the pooling problem. *ACM Sigmap Bul*. 1978;25:19.
2. Takama N, Kuriyama Y, Shiroko K, Umeda T. Optimal water allocation in a petrochemical refinery. *Comput Chem Eng*. 1980;4:251.
3. Quesada I, Grossmann IE. Global optimization of bilinear process networks with multicomponent flows. *Comput Chem Eng*. 1995;19:1219.
4. Reddy PCP, Karimi IA, Srinivasan R. Novel solution approach for optimizing crude oil operations. *AIChE J*. 2004;50:1177.
5. Reddy PCP, Karimi IA, Srinivasan R. A new continuous-time formulation for scheduling crude oil operations. *Chem Eng Sci*. 2004;59:325.
6. Wicaksono DS, Karimi IA, Alfadala H, Al-Hatou, OI. Optimization of fuel gas network in an LNG plant. In: Proceedings of AIChE Annual Meeting. San Francisco, CA; November 12–17, 2006.
7. Wicaksono DS, Karimi IA, Alfadala H, Al-Hatou OI. Integrating recovered jetty boil-off gas as a fuel in an LNG plant. In: Proceedings of the 17th European Symposium of Computer Aided Process Engineering. Bucharest, Romania; May 29–31, 2007.

8. Biegler LT, Grossmann IE. Retrospective on optimization. *Comput Chem Eng*. 2004;28:1169.
9. Floudas CA, Akrotirianakis IG, Caratzoulas S, Meyer CA, Kallrath J. Global optimization in the 21st century: Advances and challenges. *Comput Chem Eng*. 2005;29:1185.
10. Editor: Grossmann IE. *Global Optimization in Engineering Design*. Dordrecht, The Netherlands: Kluwer Academic; 1996.
11. Floudas CA. *Deterministic Global Optimization: Theory, Methods, and Applications*. Dordrecht, The Netherlands: Kluwer Academic; 2000.
12. Tawarmalani M, Sahinidis NV. *Convexification and Global Optimization in Continuous and Mixed-integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Dordrecht, The Netherlands: Kluwer Academic, 2002.
13. Editor: Floudas CA, Pardalos PM. *Frontiers in Global Optimization*. Dordrecht, The Netherlands: Kluwer Academic; 2004.
14. Neumaier A. *Complete Search in Continuous Global Optimization and Constraint Satisfaction*. Cambridge University Press; 2004.
15. Horst R, Tuy H. *Global Optimization: Deterministic Approaches*. 2nd ed. Germany: Springer-Verlag; 1993.
16. Tuy H. *Convex Analysis and Global Optimization*. The Netherlands: Kluwer Academic; 1998.
17. Nemhauser G, Wolsey L. *Integer and Combinatorial Optimization*. New York: John Wiley and Sons; 1988.
18. Sahinidis NV. BARON: A general purpose global optimization software package. *J of Global Optimization*. 1996;8:201.
19. Ryoo HS, Sahinidis NV. A branch-and-reduce approach to global optimization. *J of Global Optimization*. 1996;8:107.
20. Zamora JM, Grossmann IE. A branch and contract algorithm for problems with concave univariate, bilinear, and linear fractional terms. *J of Global Optimization*. 1999;14:217.
21. Neumaier A, Shcherbina O, Huyer W, Vinko T. A comparison of complete global optimization solvers. *Mathematical Programming*. 2005;103:335.
22. McCormick GP. Computability of global solutions to factorable nonconvex programs: part i-convex underestimating problems. *Mathematical Programming*. 1976;10:146.
23. Kearfott B. Decomposition of arithmetic expressions to improve the behavior of interval iteration for nonlinear systems. *Computing*. 1991;47:169.
24. Smith EMB, Pantelides CC. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Comput Chem Eng*. 1999;23:457.
25. Al-Khayyal FA, Falk JE. Jointly constrained biconvex programming. *Math Oper Res*. 1983;8:273.
26. Foulds LR, Haughland D, Jornsten K. A bilinear approach to the pooling problem. *Optimization*. 1992;24:165.
27. Tawarmalani M, Ahmed S, Sahinidis NV. Product disaggregation in global optimization and relaxations of rational programs. *J of Global Optimization*. 2002;3:281.
28. Androulakis IP, Maranas CD, Floudas CA. αBB: A global optimization method for general constrained nonconvex problems. *J of Global Optimization*. 1995;7:4.
29. Meyer CA, Floudas CA. Convex underestimation of twice continuously differentiable functions by piecewise quadratic perturbation: spline αbb underestimators. *J of Global Optimization*. 2005;32:221.
30. Sherali H, Alameddine A. A new reformulation-linearization technique for bilinear programming problems. *J of Global Optimization*. 1992;2:379.
31. Ben-Tal A, Eiger G, Gershovitz V. Global minimization by reducing duality gap. *Mathematical Programming*. 1994;63:193.
32. Adhya N, Tawarmalani M, Sahinidis NV. A Lagrangian approach to the pooling problems. *Ind Eng Chem Res*. 1999;38:1956.
33. Bergamini ML, Aguirre P, Grossmann IE. Logic-based outer approximation for globally optimal synthesis of process networks. *Comput Chem Eng*. 2005;29:1914.
34. Karuppiah R, Grossmann IE. Global optimization for the synthesis of integrated water systems in chemical processes. *Comput Chem Eng*. 2006;30:650.
35. Meyer CA, Floudas CA. Global optimization of a combinatorially complex generalized pooling problem. *AIChE J*. 2006;52:1027.
36. Bjork KM, Lindberg PO, Westerlund T. Some convexifications in global optimization of problems containing signomial terms. *Comput Chem Eng*. 2003;27:669.
37. Balas, E. Disjunctive programming. *Annals of Discrete Mathematics*. 1979;5:3.
38. Raman R, Grossmann IE. Modeling and computational techniques for logic based integer programming. *Comput Chem Eng*. 1994;18:563.
39. Williams HP. *Model building in mathematical programming*. John Wiley and Sons; 1985.
40. Balas E. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J on Algebraic and Discrete Methods*. 1985;6:466.
41. Balas E. On the convex hull of the union of certain polyhedra. *Operations Research Letters*. 1988;7:279.
42. Hooker JN. *Logic-based methods for optimization: combining optimization and constraint satisfaction*. John Wiley and Sons; 2000.
43. Vecchietti A, Lee S, Grossmann IE. Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. *Comput Chem Eng*. 2003;27:433.
44. Dantzig GB. *Linear Programming and Extensions*. Princeton, USA: Princeton University Press; 1963.
45. Padberg M. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters*. 2000;27:1.
46. Keha AB, de Farias IR, Nemhauser GL. Models for representing piecewise linear cost functions. *Operations Research Letters*. 2004; 32:44.
47. Oh HC, Karimi IA. Planning production on a single processor with sequence-dependent setups: Part 1-Determination of champaigns. *Comput Chem Eng*. 2001;25:1021.
48. Floudas, CA, Pardalos PM, Adjiman CS, Esposito WR, Gumus ZH, Harding ST, Klepeis JL, Meyer CA, Schweiger CA. *Handbook of Test Problems in Local and Global Optimization*. Dordrecht, The Netherlands: Kluwer Academic; 1999.
49. Karimi IA, Tan ZYL, Bhushan S. An improved formulation for scheduling an automated wet-etch station. *Comput Chem Eng*. 2004; 29:217.
50. Bixby R. Mixed-integer programming – It works out of the box. In: Proceedings of the Sixth International Conference on Optimization. Ballarat, Australia, December 9–11, 2004.
51. Liu Y, Karimi IA. Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. *Chem Eng Sci*. 2007;62:1549.

## Appendix: Theoretical Results

Let $m$ denotes a piecewise MILP under- and overestimators model for bilinear term $xy$ over rectangle $x^L \le x \le x^U$, $y^L \le y \le y^U$. $m$ is based on partitioning the bilinear envelope of the original domain with arbitrary number of segments on $x$ (e.g., **BM**, **CH**, **TCH**, and **NF1-NF12** presented in this article). Let $p$ denotes a BLP problem. Let $LR(p)$ denotes the feasible region representing the 1-segment LP relaxation (convex envelope for bilinear term represented by Eqs. R1–R4) of $p$ in the space of variables $\{x, y, z\}$. Furthermore, let $RMR(m,p)$ and $MR(m,p)$, respectively, denote the projected feasible regions of the 2nd and 1st level relaxations of $m$ for $p$ in the same space of variables $\{x, y, z\}$.

**Theorem 1.** $LP(p) \subseteq RMR(m,p)$

**Proof:** Let $u(x,y) \le z$ and $o(x,y) \ge z$ denote the continuous linear under- and overestimators for the bilinear term $xy$ over rectangle $x^L \le x \le x^U$, $y^L \le y \le y^U$ representing the facets of $RMR(m,p)$, respectively. Comparing them to the bilinear envelope (Eqs. R1–R4), it is clear[25] that

$$u(x,y) \le \max(x^L \cdot y + x \cdot y^L - x^L \cdot y^L, x^U \cdot y + x \cdot y^U - x^U \cdot y^U)$$

and

$$o(x,y) \ge \min(x^U \cdot y + x \cdot y^L - x^U \cdot y^L, x^L \cdot y + x \cdot y^U - x^L \cdot y^U).$$

over $x^L \le x \le x^U$, $y^L \le y \le y^U$. Thus, $LP(p)$ is either contained within $RMR(m,p)$ or equivalent to $RMR(m,p)$.

**Proposition 1.** *RMR*(CC,*p*) = *LP*(*p*) (Projection of a convex combination model (CC) in the space of variables {*x*, *y*, *z*} produces the facet constraints described by Eqs. R1–R4.)

**Proof:** This can be done via Fourier-Motzkin elimination.

Consider model CH. All $u(n)$, $v(1)$, and $\lambda(1)$ from CH-4a and CH-4c, as well as all $u(n)$, $v(N)$ and $\lambda(N)$ from Eqs. CH-4b and CH-4d, can be eliminated by utilizing Eqs. CH-1, CH-2a, and CH-3a

$$z \geq x \cdot y^L + x^L \cdot y - x^L \cdot y^L + \sum_{n=2}^{N}[(a(n) - a(1)) \cdot \Delta v(n)] - \sum_{n=2}^{N}\left[(a(n) - a(1)) \cdot y^L \cdot \lambda(n)\right] \quad \text{(A-1)}$$

$$z \geq x^U \cdot y + x \cdot y^U - x^U \cdot y^U + \sum_{n=1}^{N-1}[(a(n+1) - a(N+1) \cdot \Delta v(n)] - \sum_{n=1}^{N-1}\left[(a(n+1) - a(N+1) \cdot y^U \cdot \lambda(n)\right] \quad \text{(A-2)}$$

$$z \leq x \cdot y^L + x^U \cdot y - x^U \cdot y^L + \sum_{n=1}^{N-1}[a(n+1) - a(N+1)] \cdot \Delta v(n) - \sum_{n=1}^{N-1}\left[(a(n+1) - a(N+1)) \cdot y^L \cdot \lambda(n)\right] \quad \text{(A-3)}$$

$$z \leq x \cdot y^U + x^L \cdot y - x^L \cdot y^U + \sum_{n=2}^{N}[(a(n) - a(1)) \cdot \Delta v(n)] - \sum_{n=2}^{N}\left[(a(n) - a(1) \cdot y^U \cdot \lambda(n)\right] \quad \text{(A-4)}$$

Multiply both sides of $y^L \cdot \lambda(n) \leq v(n)$ from Eq. CH-2b with $a(n) - a(1)$ and $a(n+1) - a(N+1)$

$$[a(n) - a(1)] \cdot \Delta y(n) - [a(n) - a(1)] \cdot y^L \cdot \lambda(n) \geq 0 \quad \forall n > 2 \quad \text{(A-5)}$$

$$[a(n+1) - a(N+1)] \cdot \Delta v(n) - [a(n+1) - a(N+1)] \cdot y^L \cdot \lambda(n) \leq 0 \quad \forall n < N+1 \quad \text{(A-6)}$$

Multiply both sides of from Eq. CH-2b $v(n) \leq y^U \cdot \lambda(n)$ with $a(n+1) - a(N+1)$ and $a(n) - a(1)$

$$[a(n+1) - a(N+1)] \cdot \Delta y(n) - [a(n+1) - a(N+1)] \cdot y^U \cdot \lambda(n) \geq 0 \quad \forall n < N+1 \quad \text{(A-7)}$$

$$[a(n) - a(1)] \cdot \Delta y(n) - [a(n) - a(1)] \cdot y^U \cdot \lambda(n) \leq 0 \quad \forall n > 2 \quad \text{(A-8)}$$

The last two terms of Eqs. A-1–A-4 can be eliminated by using Eqs. A-5, A-7, A-6, and A-8, respectively. The final result is equivalent with Eqs. R1–R4. Since the latter represents the bilinear envelope, it is clear that the remaining facets generated via Fourier-Motzkin Elimination are redundant.

Consider model (**TCH**). Consider the following one facet of the piecewise MILP under- and overestimators of model **TCH**. $\Delta z(N)$ from Eq. TCH-2a can be eliminated by utilizing $\Delta z(N) = z - \sum_{n=1}^{N-1} \Delta z(n)$ from Eq. TCH-1

$$\Delta z(n) \geq \Delta x(N) \cdot y^L + a(N) \cdot \Delta y(N) - a(N) \cdot y^L \cdot \lambda(N) \quad \forall n < N \quad \text{(A-9a)}$$

$$z - \sum_{n=1}^{N-1} \Delta z(n) \geq \Delta x(N) \cdot y^L + a(N) \cdot \Delta y(N) - a(N) \cdot y^L \cdot \lambda(N) \quad \text{(A-9b)}$$

It is clear that $\Delta z(N-1)$ from Eq. A-9a can be eliminated by utilizing Eq. A-9b resulting in

$$\Delta z(n) \geq \Delta x(N) \cdot y^L + a(N) \cdot \Delta y(N) - a(N) \cdot y^L \cdot \lambda(N) \quad \forall n < N-1 \quad \text{(A-10a)}$$

$$z - \sum_{n=1}^{N-2} \Delta z(n) \geq \sum_{n=N-1}^{N} \Delta x(n) \cdot y^L + a(n) \cdot \Delta y(n) - a(n) \cdot y^L \cdot \lambda(n) \quad \text{(A-10b)}$$

Repeating the same steps until the entire remaining $\Delta z(n)$ are eliminated produces Eq. CH-4a. Using the same steps for the other three facets of **TCH** produces Eqs. CH-4b–CH-4d. From this point, the next steps follow directly from those of **CH** described previously.

Consider **NF3**. **NF3** is related to **TCH** via

$$u(n) = a(n) \cdot \lambda(n) + \Delta x(n) \quad \forall n$$

$$v(n) = y^L \cdot \lambda(n) + \Delta y(n) \quad \forall n$$

$$w(n) = y^L \cdot \Delta x(n) + a(n) \cdot \Delta y(n) + y^L \cdot a(n) \cdot \lambda(n) + \Delta z(n) \quad \forall n$$

Thus, it is clear that the same result applies for **NF3**. It can be easily shown as well that the same result also applies for the other convex combination models.

**Corrolary.** *LP*(*p*) ⊆ *RMR*(BM,*p*)

**Proof:** It is clear[42] that for a disjunctive program that the discrete-to-continuous relaxation quality of a model obtained from convex hull reformulation is tighter than or as tight as those of big-M. We already showed that *RMR*(CC,*p*) = *LP*(*p*) in Proposition 1. Since one model of CC can be obtained from convex hull reformulation, it is clear that *LP*(*p*) ⊆ *RMR*(BM,*p*).

**Proposition 2.** *RMR*(IC,*p*) = *LP*(*p*) (Projection of an incremental cost model (IC) in the space of variables {*x*, *y*, *z*} produces the facet constraints described by Eqs. R-1–R-4.)

**Proof:** This can be done using Fourier - Motzkin Elimination via similar arguments used in the proof of Proposition 1.

**Remark:** Theorem 1 and Propositions 1–3 implies that $RPG(m,p,N,g) \leq 0$, $RPG(BM,p,N,g) \leq 0$ and $RPG(\mathrm{CC},p,N,g) = RPG(\mathrm{IC},p,N,g) = 0$.

**Theorem 2.** $MR(m,p) \subseteq LP(p)$.

**Proof:** As depicted by DP, $MR(m,p)$ is defined as the bilinear envelope over partition $n^*$ which $\lambda(\mathrm{n}^*) = 1$ for Big-M and convex combination models, or equivalently $\theta(1) = \ldots = \theta(n^*) = 1, \theta(n^* + 1) = \ldots = \theta(N - 1) = 0$ for incremental cost models. Hence, $MR(m,p)$ is defined as the following

$$z \geq x \cdot y^L + a(n^*) \cdot y - a(n^*) \cdot y^L$$

$$z \geq x \cdot y^U + a(n^* + 1) \cdot y - a(n^* + 1) \cdot y^U$$

$$z \leq x \cdot y^L + a(n^* + 1) \cdot y - a(n^* + 1) \cdot y^L$$

$$z \leq x \cdot y^U + a(n^*) \cdot y - a(n^*) \cdot y^U$$

$$a(n^*) \leq x \leq a(n^* + 1), \quad y^L \leq y \leq y^U$$

Since $x^L \leq a(n) \leq x^U$ for any $n$, and, thus, $x^L \leq a(n^*) \leq x^U$ and $x^L \leq a(n^* + 1) \leq x^U$, it is clear that

$$x \cdot y^L + a(n^*) \cdot y - a(n^*) \cdot y^L \geq x \cdot y^L + x^L \cdot y - x^L \cdot y^L$$

$$x \cdot y^U + a(n^* + 1) \cdot y - a(n^* + 1) \cdot y^U \geq x \cdot y^U + x^U \cdot y - x^U \cdot y^U$$

and

$$x \cdot y^L + a(n^* + 1) \cdot y - a(n^* + 1) \cdot y^L \leq x \cdot y^L + x^U \cdot y - x^U \cdot y^L$$

$$x \cdot y^U + a(n^*) \cdot y - a(n^*) \cdot y^U \leq x \cdot y^U + x^L \cdot y - x^L \cdot y^U$$

over $a(n^*) \leq x \leq a(n^* + 1)$, $y^L \leq y \leq y^U$. This completes the proof.

*Remark.* Theorem 2 implies that $PG(m,p,N,g) \leq 0$, which supports the use of piecewise MILP under- and overestimators.

*Manuscript received Jun 18, 2007, and revision received Nov. 1, 2007.*