Computational Comparison of Piecewise−Linear Relaxations for Pooling Problems

Chrysanthos E. Gounaris, Ruth Misener, and Christodoulos A. Floudas*

Department of Chemical Engineering Princeton University Princeton, New Jersey 08544-5263

This work discusses alternative relaxation schemes for the pooling problem, a theoretically and practically interesting optimization problem. The problem nonconvexities appear in the form of bilinear terms and can be addressed with the relaxation technique based on the bilinear convex and concave envelopes. We explore ways to improve the relaxation tightness, and thus the efficiency of a global optimization algorithm, by employing a piecewise linearization scheme that partitions the original domain of the variables involved and applies the principles of bilinear relaxation for each one of the resulting subdomains. We employ 15 different piecewise relaxation schemes with mixed-integer representations and conduct a comprehensive computational comparison study over a collection of benchmark pooling problems. For each case, various partitioning variants can be envisioned, cumulatively accounting for a total of 56 700 relaxations. The results demonstrate that some of the schemes are clearly superior to their counterparts and should, therefore, be preferred in the optimization of pooling processes.

1. Introduction

The pooling problem involves a feed-forward network topology and a set of attributes that are being monitored. The latter are also called *qualities* and constitute fundamental properties of the flowing medium. The network involves a layer of input nodes, a number of layers (usually just one) of intermediate nodes, and a final layer of output nodes. The input nodes represent the available supply streams, and the intermediate nodes represent the available transhipment facilities, while the output nodes represent the potential end-user/demand locations. It is assumed that *linear blending* (i.e., flow-weighted averaging) of all qualities occurs at each intermediate and output node. This also accounts for the use of the terms *pool* and *pooling*.

The objective is to find the most cost-effective combination of continuous flows throughout the network that will allow for the production of a set of output streams that meet certain specifications regarding the level of their qualities. Limits on the extent of the various flows, including a lack of connectivity between certain pairs of nodes, may also be taken into account. In fact, the pooling problem becomes challenging only in the presence of such topological restrictions. It is easy to show that, if network connectivity is exhaustive and each output node can be directly accessed from every input node, the intermediate nodes become unnecessary and the problem reduces into trivial linear programming (LP). A pooling network is depicted illustratively in Figure 1.

Pooling problems originated from the petroleum refining industry where various process streams with varying sulfur content, octane number, and density are blended together to form final regulated products. Although it is usually studied in the context of chemical processing (e.g., petroleum refining, wastewater treatment), the pooling problem is also abundant in many fields of engineering, including supply-chain operations and communications. The pooling problem involves nonconvexities in the form of multilinear (bilinear in the case of single intermediate layer) terms and generally exhibits multiplicity of local solutions and a high degree of difficulty in locating its global solution—particularly when larger, real-scale, instances are considered.

Early attempts to solve the pooling problem were based on recursive linear programming techniques, in which the various properties of the streams were estimated, fixed, and the resulting linear programming problem solved. The estimated properties were then recalculated from the linear programming solution, and the process was repeated. Haverly[1] observed that this procedure does not always converge and presented a series of small problems to illustrate the occurrence of nonglobal local solutions. These problems are still in use today as benchmarks for testing algorithms and assessing their computational performance.

Sequential linear programming (SLP) has also been widely used on this type of problem.[2] SLP algorithms replace the bilinear terms by first order Taylor expansions and solve the problem via a sequence of linear programs.[3−5] Floudas and Aggarwal[6] proposed an algorithm based on generalized Benders decomposition[7,8] to search for the global solutions of pooling problems. Although it did exhibit satisfactory behavior in practice, this method could not guarantee convergence to the global solution.

The first algorithm that provided such guarantees was proposed by Visweswaran and Floudas.[9−12] It was based on duality theory, Lagrangian relaxation, and a novel approach to close the duality gap. Lagrangian-based global optimization methods for pooling problems were also proposed by Ben-Tal

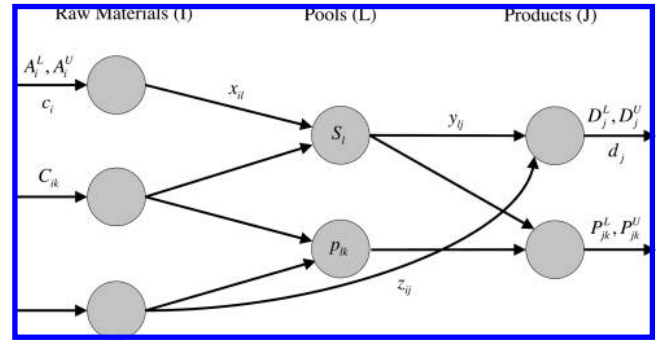

Figure 1. Illustrative pooling network.

* To whom all correspondence should be addressed. E-mail: floudas@titan.princeton.edu. Tel.: (609) 258-4595. Fax: (609) 258-0211.

10.1021/ie8016048 CCC: $40.75
Published on Web 05/07/2009

et al.,[13] Adhya et al.,[14] and Almutairi and Elhedhli.[15] Other approaches for the solution of pooling problems originated with the development of the *reformulation−linearization technique* (RLT).[16] Quesada and Grossmann[17] argued that RLT-generated cuts can boost the efficiency of pooling problem solving algorithms, and Tawarmalani and Sahinidis[18] proved that these RLT-generated cuts are at least as tight the '*P*'- and '*Q*'-formulations. Audet et al.[19] used the RLT-based quadratic programming optimization algorithm developed by Audet et al.,[20] proposed a generalized pooling problem, and customized heuristic searches have also been applied on large pooling problem instances. Meyer and Floudas[21] introduced an augmented RLT to solve a *generalized* pooling problem where the optimal topology of the network is deduced from the solution of a suitable mixed-integer nonlinear programming (MINLP) formulation.

Lodwick[22] developed methods for the preprocessing of nonlinear constraint sets that are relevant to the pooling problem, in an effort to uncover redundancies, infeasibilities, and implied bounds on decision variables. Greenberg[23] developed a computational geometry-based analysis framework that can provide exact answers to questions of sensitivity analysis and infeasibility diagnosis. Pooling problems can also be addressed by branch-and-bound global optimization methods,[18,24] where the use of available convex envelopes for bilinear terms can provide tight enclosures. For a more comprehensive survey of global optimization and its implications for the pooling problem and other applications, see the recent reviews of Floudas et al.[25] and Floudas and Gounaris,[26] the book of Floudas,[27] and the papers presented at the conferences organized by Floudas and Pardalos.[28−33]

The focus of this paper is piecewise relaxations involving ab initio domain partitioning, which have been used successfully to underestimate large-scale bilinear programming problems.[21,34] Karuppiah and Grossmann[34] implemented their piecewise−linear relaxations into a branch-and-bound algorithm and showed that these new relaxations significantly expedited solution time. On the basis of the successes of Meyer and Floudas[21] and Karuppiah and Grossmann,[34] Wicaksono and Karimi[35] proposed a number of alternative ways to formulate piecewise−linear relaxations. Note that these piecewise relaxations are similar to the one developed by Pham et al.,[36] although the large-scale algorithm of Pham et al.[36] is designed for fast computational time and does not guarantee reaching global optimality. The method of ab initio partitioning takes the opposite approach from the quadratic programming algorithm of Audet et al.[20] which optimally partitions a variable at each node of a branching tree.

The aim of this paper is to comprehensively investigate the potential for tight and efficient relaxations of the pooling problem that are based on ab initio piecewise application of convex envelopes. The results of this study can be used to better formulate large-scale problems such as the ones addressed by Meyer and Floudas[21] and Karuppiah and Grossmann.[34] The mathematical model that governs the pooling problem is presented in section 2. Sections 3 and 4 introduce the various relaxation schemes that can be constructed, while an extensive computational comparison of all the alternatives is presented in section 5. Because relaxations such as these are useful in the context of a branch-and-bound algorithm solving a large-scale pooling problem, it is important to identify the best-performing formulations for a variety of test cases. The computational experience documented in this study should be used to choose the problem formulation (e.g., '*P*'- or '*Q*'-formulation), partitioning variable, and relaxation formulation.

Table 1. Notation used for the '*P*'-Formulation of the Pooling Problem

type	name	description
indices	$i \in \{1, 2, ..., I\}$	input streams (raw materials)
	$\boldsymbol{l} \in \{1, 2, ..., L\}$	pools (blending facilities)
	$j \in \{1, 2, ..., J\}$	output streams (end products)
	$k \in \{1, 2, ..., K\}$	attributes (qualities monitored)
sets	T_X	$(i,\boldsymbol{l})$ pairs for which input to pool connection exists
	T_Y	$(\boldsymbol{l},j)$ pairs for which pool to output connection exists
	T_Z	(i,j) pairs for which input to output connection exists
variables	$x_{i\boldsymbol{l}}$	flow from ith input to $\boldsymbol{l}$th pool
	$y_{\boldsymbol{l}j}$	flow from $\boldsymbol{l}$th pool to jth output
	z_{ij}	bypass flow from ith input to jth output
	$p_{\boldsymbol{l}k}$	level of kth attribute in $\boldsymbol{l}$th pool
	$q_{i\boldsymbol{l}}$	proportion of flow from ith input to $\boldsymbol{l}$th pool
parameters	c_i, d_j	unit price of ith raw material and jth product
	A_i^{L}	minimum required consumption of ith raw material
	A_i^{U}	maximum availability of ith raw material
	$S_{\boldsymbol{l}}$	size (capacity) of $\boldsymbol{l}$th pool
	D_j^{L}	minimum required production for jth product
	D_j^{U}	maximum demand for jth product
	C_{ik}	level of kth quality in ith raw material
	$[P_{jk}^{\mathrm{L}},P_{jk}^{\mathrm{U}}]$	acceptable range for the level of kth quality in jth product

2. Problem Formulation

In this section, we present the complete set of constraints that correspond to the classical formulation of the pooling problem, called the '*P*'-formulation. This formulation traces its origins in the work of Haverly.[1] We also outline an alternative equivalent formulation, proposed by Ben-Tal et al.,[13] which is denoted as the '*Q*'-formulation. Complete analysis of two additional formulations, a simplified '*Q*'-formulation that implicitly defines one variable per pool[19] and an augmented '*Q*'-formulation that adds constraints derived using the reformulation−linearization technique,[16−18] is outside the scope of this study, but we do perform two case studies comparing all four problem formulations in section 5. The notation used is defined in Table 1.

The objective of the '*P*'-formulation is to minimize the total loss (negative profit), which is defined as the difference between the cost of purchasing the required raw materials and the returns from selling the final products. This minimization, shown in eq 1, is subject to six sets of constraints that are presented in eqs 2−7. The raw material availabilities, pool capacities, and product demand restrictions are imposed by eqs 2, 3, and 4, respectively. Equations 5 impose the specifications of the final products, while eqs 6 and 7 correspond to the mass balances around the pools (individual qualities and overall, respectively). The hard bounds of eqs 8 also apply to the variables of the problem. Note that the summations in the following equations involve only those index values for which network connectivity holds.

$$\min_{x_{i\boldsymbol{l}},y_{\boldsymbol{l}j},z_{ij},p_{\boldsymbol{l}k}} \sum_{(i,\boldsymbol{l})\in T_X} c_i \cdot x_{i\boldsymbol{l}} - \sum_{(\boldsymbol{l},j)\in T_Y} d_j \cdot y_{\boldsymbol{l}j} - \sum_{(i,j)\in T_Z} (d_j - c_i) \cdot z_{ij} \quad (1)$$

$$A_i^{\mathrm{L}} \le \sum_{\boldsymbol{l}:(i,\boldsymbol{l})\in T_X} x_{i\boldsymbol{l}} + \sum_{j:(i,j)\in T_Z} z_{ij} \le A_i^{\mathrm{U}}, \forall i \quad (2)$$

$$\sum_{i:(i,\boldsymbol{l})\in T_X} x_{i\boldsymbol{l}} \le S_{\boldsymbol{l}}, \forall \boldsymbol{l} \quad (3)$$

$$D_j^{\mathrm{L}} \le \sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \le D_j^{\mathrm{U}}, \forall j \tag{4}$$

$$\sum_{l:(l,j)\in T_Y} p_{lk}\cdot y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik}\cdot z_{ij} \begin{cases} \ge P_{jk}^{\mathrm{L}}\left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij}\right) \\ \le P_{jk}^{\mathrm{U}}\left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij}\right) \end{cases}, \forall (j,k) \tag{5}$$

$$\sum_{i:(i,l)\in T_X} C_{ik} x_{il} = p_{lk} \sum_{j:(l,j)\in T_Y} y_{lj}, \forall (l,k) \tag{6}$$

$$\sum_{i:(i,l)\in T_X} x_{il} - \sum_{j:(l,j)\in T_Y} y_{lj} = 0, \forall l \tag{7}$$

$$\begin{aligned} 0 &\le x_{il} \le \min\{A_i^{\mathrm{U}}, S_l, \sum_{j:(l,j)\in T_Y} D_j^{\mathrm{U}}\}, && \forall (i,l) \in T_X \\ 0 &\le y_{lj} \le \min\{S_l, D_j^{\mathrm{U}}, \sum_{i:(i,l)\in T_X} A_i^{\mathrm{U}}\}, && \forall (l,j) \in T_Y \\ 0 &\le z_{ij} \le \min\{A_i^{\mathrm{U}}, D_j^{\mathrm{U}}\}, && \forall (i,j) \in T_Z \\ \min_i C_{ik} &\le p_{lk} \le \max_i C_{ik}, && \forall (l,k) \end{aligned} \tag{8}$$

We observe that the only sources of nonlinearities in the problem are the bilinear terms $y_{lj}p_{lk}$, which appear in eqs 5 and 6. Since the former is an equality constraint, the problem is not convexifiable (see Remark 5.1 in the work of Gounaris and Floudas[37]). Therefore, a global optimization methodology has to be utilized, which will entail the construction of a relaxation of the problem. In section 3, we present a number of approaches that can be followed.

Instead of absolute input flow rates x_{il}, the '*Q*'-formulation[13] employs fractional input flow rates q_{il}, through the transformation $x_{il} = q_{il} \sum_{j:(l,j)\in T_Y} y_{lj}$, $\forall (i,l) \in T_X$. This transformation facilitates the complete removal of the p_{lk} variables at the expense of creating bilinear terms $q_{il}y_{lj}$ in the objective function, as well as in some of the constraints. As in the '*P*'-formulation, the '*Q*'-formulation minimizes the total loss (eq 9) subject to raw material availability (eq 10), pool capacity (eq 11), product demand restrictions (eq 12), product specifications (eq 13), proportion balances (eq 14), and appropriate hard bounds (eq 15):

$$\min_{q_{il},y_{lj},z_{ij}} \sum_{\substack{(i,l)\in T_X \\ (l,j)\in T_Y}} c_i\cdot q_{il}\cdot y_{lj} - \sum_{(l,j)\in T_Y} d_j\cdot y_{lj} - \sum_{(i,j)\in T_Z} (d_j - c_i)\cdot z_{ij} \tag{9}$$

$$A_i^{\mathrm{L}} \le \sum_{\substack{l:(i,l)\in T_X \\ (l,j)\in T_Y}} q_{il}\cdot y_{lj} + \sum_{j:(i,j)\in T_Z} z_{ij} \le A_i^{\mathrm{U}}, \forall i \tag{10}$$

$$\sum_{j:(l,j)\in T_Y} y_{lj} \le S_l, \forall l \tag{11}$$

$$D_j^{\mathrm{L}} \le \sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij} \le D_j^{\mathrm{U}}, \forall j \tag{12}$$

$$\sum_{\substack{l:(l,j)\in T_Y \\ i:(i,l)\in T_X}} C_{ik}\cdot q_{il}\cdot y_{lj} + \sum_{i:(i,j)\in T_Z} C_{ik}\cdot z_{ij} \times \begin{cases} \ge P_{jk}^{\mathrm{L}}\left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij}\right) \\ \le P_{jk}^{\mathrm{U}}\left(\sum_{l:(l,j)\in T_Y} y_{lj} + \sum_{i:(i,j)\in T_Z} z_{ij}\right) \end{cases}, \forall (j,k) \tag{13}$$

$$\sum_{i:(i,l)\in T_X} q_{il} = 1, \forall l \tag{14}$$

$$\begin{aligned} 0 &\le q_{il} \le 1, && \forall (i,l) \in T_X \\ 0 &\le y_{lj} \le \min\{S_l, D_j^{\mathrm{U}}, \sum_{i:(i,l)\in T_X} A_i^{\mathrm{U}}\}, && \forall (l,j) \in T_Y \\ 0 &\le z_{ij} \le \min\{A_i^{\mathrm{U}}, D_j^{\mathrm{U}}\}, && \forall (i,j) \in T_Z \end{aligned} \tag{15}$$

3. Bilinear Term and Piecewise Relaxation Schemes

McCormick[38] developed an efficient relaxation technique for the case of a bilinear term xy. It was later shown by Al-Khayyal and Falk[39] that this technique actually yields the *envelope* of the term, which is the tightest possible convex relaxation. This envelope is convex polyhedral, as the result by Rikun[40] dictates, and, given a domain of interest $[x^{\mathrm{low}},x^{\mathrm{upp}}] \times [y^{\mathrm{low}},y^{\mathrm{upp}}]$, its convex and concave portions are given by eqs 16 and 17.

$$\mathrm{cvx}_{\mathrm{env}} = \max\{y^{\mathrm{low}}x + x^{\mathrm{low}}y - x^{\mathrm{low}}y^{\mathrm{low}}, y^{\mathrm{upp}}x + x^{\mathrm{upp}}y - x^{\mathrm{upp}}y^{\mathrm{upp}}\} \tag{16}$$

$$\mathrm{ccv}_{\mathrm{env}} = \min\{y^{\mathrm{upp}}x + x^{\mathrm{low}}y - x^{\mathrm{low}}y^{\mathrm{upp}}, y^{\mathrm{low}}x + x^{\mathrm{upp}}y - x^{\mathrm{upp}}y^{\mathrm{low}}\} \tag{17}$$

Therefore, a plausible relaxation of a bilinear term that participates in an equality constraint would be to replace every occurrence of this term in the formulation by a new variable z and constrain this variable by appending the following linear constraints:

$$\begin{aligned} z &\ge y^{\mathrm{low}}x + x^{\mathrm{low}}y - x^{\mathrm{low}}y^{\mathrm{low}} \\ z &\ge y^{\mathrm{upp}}x + x^{\mathrm{upp}}y - x^{\mathrm{upp}}y^{\mathrm{upp}} \\ z &\le y^{\mathrm{upp}}x + x^{\mathrm{low}}y - x^{\mathrm{low}}y^{\mathrm{upp}} \\ z &\le y^{\mathrm{low}}x + x^{\mathrm{upp}}y - x^{\mathrm{upp}}y^{\mathrm{low}} \end{aligned} \tag{18}$$

Since the relaxation constructed results in underestimating the objective function and/or expanding the feasible region, its optimal solution constitutes only a lower bound to the global optimum of the original minimization problem. This lower bound can later be refined via a branch-and-bound global optimization procedure. Androulakis et al.[41] showed that the maximum difference between variable z and the bilinear term xy is equal to $d_{\max} = (x^{\mathrm{U}} - x^{\mathrm{L}})(y^{\mathrm{U}} - y^{\mathrm{L}})/4$; that is, it is proportional to the area of the domain under consideration.

In Figure 2a−c, we present the plot of the bilinear term and those of its convex and concave envelope for domain $[-1,1]^2$. It should be obvious that the envelopes enjoy a great level of tightness to the function and there is no other convex enclosure that could provide a better result. In this paper, we explore ways to utilize certain nonconvex enclosures that can provide relaxations of better quality; that is, of enclosures that are tighter to the original function and lead to relaxed problems that approximate the original with better accuracy.

It is easy to deduce from eqs 16 and 17 that the envelope's level of tightness would benefit from shrinkage of the domain under consideration. This is illustrated in Figure 2d and e, where we effectively split the original domain into two symmetric subdomains and plot the convex and concave envelopes that correspond to each one of them. Each "subenvelope" is tighter than the "parent envelope" in the respective subdomain. Therefore, the combined effect of the two offer a potentially more efficient relaxation scheme, as fewer branch-and-bound refinements would be necessary for closing the optimality gap.

Although a scheme based on multiple subenvelopes is not convex, it is piecewise−affine and, thus, can be explicitly represented by a mixed-integer linear programming (MILP) formulation. The integer variables are typically binary {0,1} and are introduced into the problem so as to model the inevitable

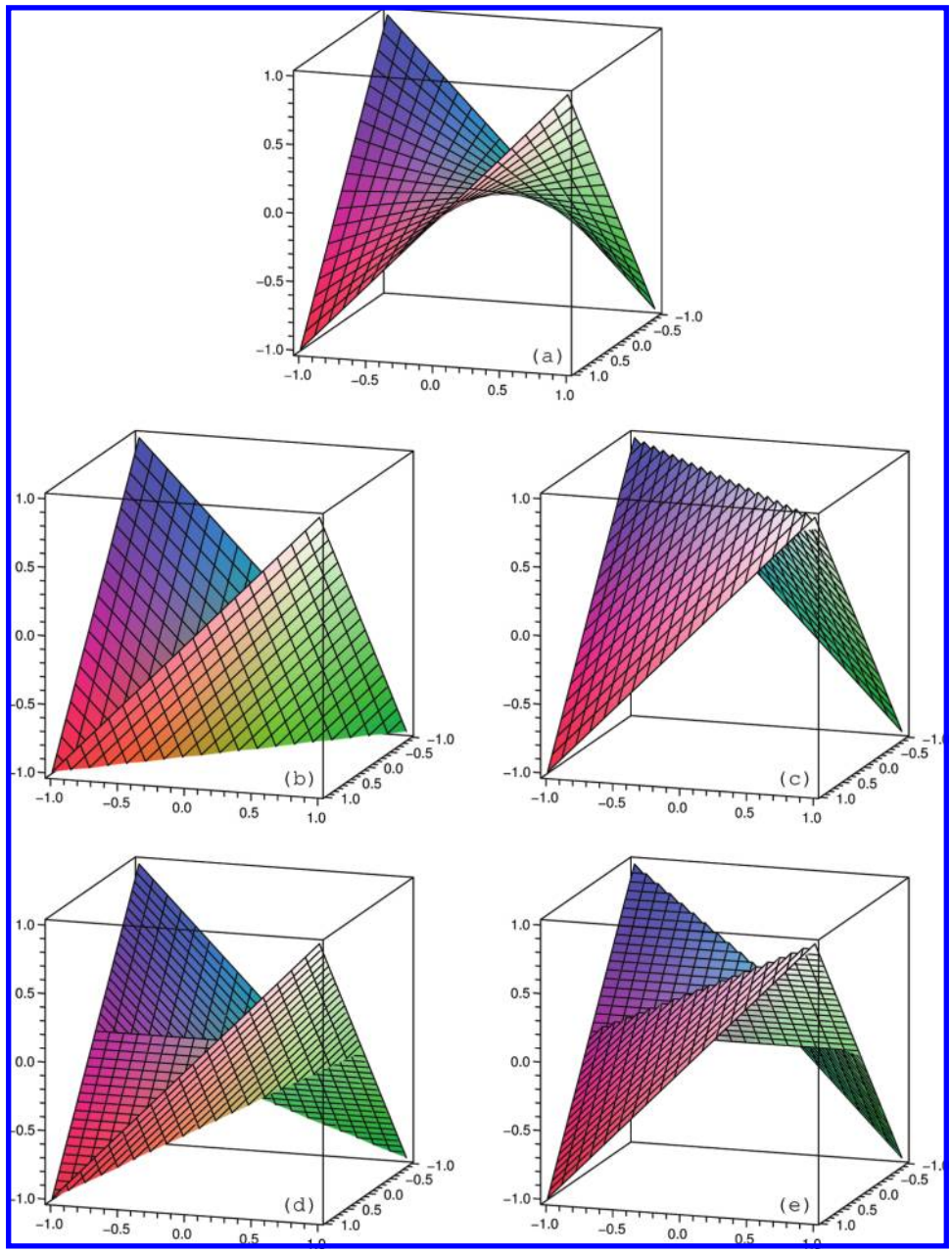

Figure 2. (a) Bilinear term in domain $[-1,1]^2$ and its (b) convex envelope, (c) concave envelope, (d) piecewise−convex underestimator, and (e) piecewise−concave overestimator.

disjunctions; that is, to provide the optimization procedure with answers regarding which subdomains and associated envelope facets are applicable at any given instance of the problem.

A number of different approaches can be followed so as to construct such representations. All of them aim at replacing the bilinearities with a linear term and then using some efficient piecewise−affine relaxation scheme to drive, at optimality, this term as close as possible to the original bilinear one. The quality of the relaxations generally benefits from a higher number of considered subdomains. In total, three distinct families of methods (*classes*) involving 15 different MILP relaxation formulations have been proposed, and the corresponding sets of constraints are presented in Table 2. Most of these formulations ('bm', 'nf1', 'nf2', 'ch', 'tch', 'nf3', 'nf4', 'nf5', 'nf6', 'nf7') have been compiled in the work of Wicaksono and Karimi,[35] with the rest ('nf2g', 'nf4l', 'nf4r', 'nf6t', 'nf7r') being variants that we developed. The derivation of these constraints is presented, in detail, in section 4. Note that Wicaksono and Karimi[35] developed five more MILP formulations ('nf8', 'nf9', 'nf10', 'nf11', and 'nf12'), but we will not consider those here as they are versions of 'nf1', 'nf2', 'nf3', 'nf4', and 'nf7' that enforce uniform partitioning and we are interested in comparing formulations using a variety of partitioning schemes.

Although they may be defined in different spaces (i.e., they may involve different variables), all of these formulations constitute valid MILP relaxations of an optimization problem that involves bilinear terms. From a mathematical programming point of view, they are all equivalent to each other (for given domain partitioning), which means that the following three properties hold for any pair of formulations "F1" and "F2": (i) Any feasible instance of F1, corresponds to a feasible instance of F2. (ii) Any infeasible instance of F1 corresponds to an infeasible instance of F2. (iii) The globally optimal instance of F1, corresponds to the globally optimal instance of F2. In short, these properties dictate MILP that the formulations, when appended to a particular bilinear program, constitute relaxations

that are either all infeasible or all feasible, at which case they also share the same solution in terms of objective value. Note that for the relaxations to be infeasible, the original problem need also be infeasible.

As it has been mentioned before, the tightness of a relaxation to the original problem is a desirable feature, as better approximations are likely to require exploration of fewer nodes in a branch-and-bound global optimization process until the optimality gap is closed. However, the size of the resulting relaxation (i.e., number of additional variables, constraints, nonzeros) is also an issue of concern. As the size increases, the solution of each node will take longer since the LP solver requires a larger number of iterations to reach the optimal solution. Typically, a tighter relaxation is also a bigger one, effectively creating an interesting trade-off. A comparative study to assess the computational performance of each formulation, within the context of the pooling problem, is presented in section 5.

Table 2 presents some size characteristics of the proposed formulations, as a function of the number of considered partitions N. In particular, it presents how many additional variables need to be defined and how many additional constraints need to be appended to the original problem. Note that when a general bilinear problem involves more than a single bilinear term, as it is typically the case, the utilized relaxation scheme needs to be applied to every term (if a complete relaxation is to be constructed). In order to account for this, the table entries have been multiplied appropriately by the number of distinct bilinear terms B present in the problem and the number of different x variables V that need to undergo partitioning. Keep in mind that two distinct bilinear terms (e.g., t_1t_2 and t_1t_3) may contain a common variable (e.g., t_1) that is selected for partitioning. Therefore, $V \leq B$ in the general case. For reference, Table 2 also includes formulation 'mc' which corresponds to the original approach by McCormick.[38] The column with the number of inequality constraints does not include the box constraints that define the hard bounds for each of the variables used.

4. Formulations

The objective of this section is to explore the various possibilities in constructing a piecewise relaxation of a bilinear term xy in the box domain $D = D_x \times D_y = [x^L,x^U] \times [y^L,y^U]$. In all of the approaches presented, it is assumed that every occurrence of this bilinear term in the original problem has been replaced by a new variable z, where $z^L \leq z \leq z^U$.

$$z^L = \min\{x^Ly^L, x^Ly^U, x^Uy^L, x^Uy^U\}$$
$$z^U = \max\{x^Ly^L, x^Ly^U, x^Uy^L, x^Uy^U\} \quad (19)$$

One of the two variables of the bilinear product (by convention, variable x) is selected for partitioning of its domain $D_x = [x^L,x^U]$ into a total of N subdomains. The partitioning is defined by a grid of points x_n, $n = 0, 1, ..., N$, such that $x_0 = x^L$ and $x_N = x^U$. We also define d_n, $n = 1, 2, ..., N$, to be the interval length of subdomain n. Since the partitioning need not be uniform, these parameters need not be all equal.

$$d_n = x_n - x_{n-1} \quad (20)$$

Domain $D_y = [y^L,y^U]$ is not subject to partitioning. Note that, in principle, both variables x and y could be selected for partitioning, in an effort to tighten the relaxation. However, this is not the approach followed in this study and it should generally be avoided as it results into relaxations of complexity $\boldsymbol{O}(2^{2N})$, instead of just $\boldsymbol{O}(2^N)$. Before we present the formulations in detail, we define here two equivalent sets of binary variables that will be used in modeling the disjunctions associated with the partitioned domain D_x.

First Set: Variables λ. The first set λ_n, $n = 1, 2, ..., N$, is a set of binary variables that are active (i.e., equal to one) if and only if the variable x lies within the nth subdomain.

$$\lambda_n = \begin{cases} 1, & \text{if } x_{n-1} \leq x \leq x_n \\ 0, & \text{otherwise} \end{cases}, \quad 1 \leq n \leq N$$

Since variable x always lies in one of the N subdomains, eq 21 is included in the formulations that involve the λ variables in order to provide coherence and tighten the relaxation.

$$\sum_{n=1}^{N} \lambda_n = 1 \quad (21)$$

Note that in the case where x lies exactly on the border between two consecutive subdomains, this equation is still valid; however, it contributes to multiplicity of optimal solutions. The set λ_n is a *special ordered set of type* 1 (sos1), which means that "at most one of the variables can be nonzero at any feasible instance". Many mixed-integer linear optimization solvers accommodate the direct declaration of this sos1 property and exploit it in a computationally beneficial manner.

There exist three ways to model the partitioning of domain D_x using the λ variables. The first way introduces a set of continuous variables δx_n, $n = 1, 2, ..., N$, where $0 \leq \delta x_n \leq d_n$. These variables have local scope and correspond to the deviation

Table 2. Formulations Used in This Study with Their Respective Sizes

class	name	equations	binary vars	continuous vars	linear ineq	linear equal
Big-M	mc		B	$4B$		
	bm	21, 29, 30	NV	B	$4NB + 2NV$	V
	nf1	21, 24, 30	NV	$B + NV$	$4N{\cdot}B + NV$	$2V$
	nf2g	21, 24, 30	NV	$B + V$	$4NB + V$	$2V$
	nf2	21, 24, 30	NV	B	$4NB + 2V$	V
conv comb	ch	21, 31, 32, 33	NV	$(N + 1)B + NV$	$(2N + 4)B + 2NV$	$B + 2V$
	tch	21, 31, 32, 34, 35	NV	$(2N + 1)B + NV$	$6NB + 2NV$	$2(B + V)$
	nf3	21, 22, 36, 39, 40	NV	$(2N + 1)B + NV$	$4NB + NV$	$2(B + V)$
	nf4l	21, 22, 36, 41, 42	NV	$(N + 2)B + NV$	$(N + 3)B + NV$	$2(B + V)$
	nf4	21, 23, 36, 41, 43	NV	$(N + 2)B + V$	$(N + 3)B + V$	$2(B + V)$
	nf4r	21, 24, 36, 44, 45	NV	$(N + 1)B$	$(N + 4)B + 2V$	$B + V$
incre cost	nf5	26, 46, 48	$(N - 1)V$	$2NB + NV$	$(4N - 1)B + 2(N - 1)V$	$B + V$
	nf6	26, 46, 49	$(N - 1)V$	$(N + 1)B + NV$	$3NB + 2(N - 1)V$	$B + V$
	nf6t	26, 46, 50	$(N - 1)V$	$(N + 1)B + NV$	$3N{\cdot}B + 2(N - 1)V$	$B + V$
	nf7	27, 51, 52, 53	$(N - 1)V$	$(N + 1)B + V$	$2(N + 1)B + V^a$	$B + V$
	nf7r	28, 53, 54, 55	$(N - 1)V$	NB	$(2N + 3)B + 2V^a$	

[a] If $N = 1$, reduce by B.

of x from grid point x_{n-1}, if $x \in [x_{n-1},x_n]$, while they are zero otherwise. This is modeled with eqs 22.

$$\begin{aligned} x &= \sum_{n=1}^{N} \{x_{n-1}\lambda_n + \delta x_n\} \\ 0 &\le \delta x_n \le d_n\lambda_n, \forall n \end{aligned} \tag{22}$$

The second way would be to aggregate all δx_n into a single variable Δx, where $0 \le \Delta x \le \max_{1\le n\le N} d_n$, that has global scope and corresponds to the deviation from the lower bound of the subdomain that is active in the given instance. Equations 23 apply.

$$\begin{aligned} x &= \sum_{n=1}^{N} \{x_{n-1}\lambda_n\} + \Delta x \\ 0 &\le \Delta x \le \sum_{n=1}^{N} d_n\lambda_n \end{aligned} \tag{23}$$

Finally, the third way would be to eliminate variable Δx overall, by solving for it in the above equality and replacing it in the above inequalities. The resulting constraints that need to be included in the formulation are shown in eq 24.

$$\sum_{n=1}^{N} x_{n-1}\lambda_n \le x \le \sum_{n=1}^{N} x_n\lambda_n \tag{24}$$

Second Set: Variables θ. The second set θ_n, $n = 1, 2, ..., N - 1$, is a set of binary variables that are active (i.e., equal to one) if and only if the variable x lies within one of the last $N - n$ subdomains.

$$\theta_n = \begin{cases} 1, & \text{if } x \ge x_n \\ 0, & \text{otherwise} \end{cases}, \quad 1 \le n \le N-1$$

The θ variables entail a particular relationship that is dictated by eq 25. Note that using the set θ_n results in fewer binary variables than using the set λ_n. However, it lacks the benefits of the sos1 property.

$$\theta_n \ge \theta_{n+1}, \forall n < N-1 \tag{25}$$

Using the θ variables, there again exist three ways to model the partitioning of domain D_x. The first calls for introducing a set of local-scope continuous variables δu_n, $n = 1, 2, ..., N$, where $0 \le \delta u_n \le 1$. These variables are equal to zero if the current instance of x is below the corresponding subdomain's lower limit ($x \le x_{n-1}$), equal to the fractional deviation of x from grid point x_{n-1}, if $x \in [x_{n-1},x_n]$, and equal to one if x is above the subdomain's upper limit ($x_n \le x$). All these are modeled with eqs 26.

$$\begin{aligned} x &= x^{\mathrm{L}} + \sum_{n=1}^{N} d_n\delta u_n \\ \delta u_n &\ge \theta_n, \forall n < N \\ \delta u_n &\le \theta_{n-1}, \forall n > 1 \end{aligned} \tag{26}$$

Note that the relationship between the various θ variables described in eq 25 is implicitly satisfied by these equations; therefore, the inclusion of eq 25 in the formulations that utilize eqs 26 is not necessary.

The second way utilizes the global-scope variable Δx that can also be used with the λ variables. Modeling the partitioning of domain D_x is achieved through eqs 27.

$$\begin{aligned} x &= x^{\mathrm{L}} + \sum_{n=1}^{N-1} \{d_n\theta_n\} + \Delta x \\ 0 &\le \Delta x \le d_1 + \sum_{n=1}^{N-1} \{(d_{n+1} - d_n)\theta_n\} \end{aligned} \tag{27}$$

Finally, a third way eliminates Δx from the above constraints, in a similar fashion with the case of the λ variables. Equations 28 apply.

$$x^{\mathrm{L}} + \sum_{n=1}^{N-1} d_n\theta_n \le x \le x_1 + \sum_{n=1}^{N-1} d_{n+1}\theta_n \tag{28}$$

4.1. Big-M Class. The Big-M approach is based on the λ variables and utilizes appropriately large constant coefficients M to activate or deactivate the various constraints. Some of the constraints contain a linear term that consists of a binary variable multiplied by a large coefficient. This term would vanish when the variable is equal to zero, thus allowing the constraint to be taken into account. On the other hand, when the binary variable is equal to one, the term would amount to a relatively large contribution that has the capacity to relax the constraint so much that it becomes redundant.

In the original approach, the partitioning of domain D_x is modeled according to eqs 29. For the subdomain for which $\lambda_n = 1$, these constraints tighten the bounds of variable x appropriately, while for all other subdomains these constraints become redundant as they collapse to the overall hard bounds of x.

$$\begin{aligned} x &\ge x^{\mathrm{L}} + (x_{n-1} - x^{\mathrm{L}})\lambda_n \\ x &\le x^{\mathrm{U}} - (x^{\mathrm{U}} - x_n)\lambda_n \end{aligned}, \forall n \tag{29}$$

Since we do not a priori know which value of x corresponds to the optimal solution, the formulation needs to include explicitly the facets of the corresponding subenvelope for each of the subdomains considered. Large coefficient terms are added so as, for a given problem instance, to activate only the applicable set of facets and deactivate all the rest. The envelope facets are governed by eqs 18, which are suitably adapted into eqs 30.

$$\begin{aligned} z &\ge y^{\mathrm{L}}x + x_{n-1}y - x_{n-1}y^{\mathrm{L}} - M(1 - \lambda_n) \\ z &\ge y^{\mathrm{U}}x + x_n y - x_n y^{\mathrm{U}} - M(1 - \lambda_n) \\ z &\le y^{\mathrm{U}}x + x_{n-1}y - x_{n-1}y^{\mathrm{U}} + M(1 - \lambda_n) \\ z &\le y^{\mathrm{L}}x + x_n y - x_n y^{\mathrm{L}} + M(1 - \lambda_n) \end{aligned}, \forall n \tag{30}$$

We are now in position to define our first formulation, 'bm', which consists of eqs 21, 29, and 30. These constraints, when appended to the original problem formulation, provide a valid relaxation of the bilinear term xy. Of course, every occurrence of this term in the problem should have been replaced by variable z.

Three additional Big-M formulations can be derived. They are all based on bm, their difference being the representation of the D_x domain partitioning. Instead of eqs 29, we can use either of eqs 22, 23, or 24. We shall call the resulting formulations 'nf1', 'nf2g', and 'nf2', respectively. Obviously, these formulations correspond to a substantial reduction of problem size, as $2N$ constraints are replaced by $N + 1$ (case of 'nf1') or just 2 (cases of 'nf2g' and 'nf2').

To conclude, note that the coefficient M needs to be large enough so as to be able to render completely redundant all constraints than need to be deactivated. Although a value $M \rightarrow \infty$ would be sufficient for this purpose, such a value would offer poor LP relaxations and the MILP solver will not be able to converge quickly to the solution. Therefore, a suitable value would be the one that is as small as possible, yet still serves the purpose. In the case of eqs 30, the appropriate value is $M = (x^{\mathrm{U}} - x^{\mathrm{L}})(y^{\mathrm{U}} - y^{\mathrm{L}})$.

4.2. Convex Combination Class. The convex combination class corresponds to piecewise relaxation schemes that also utilize the λ variables but do not include any big-M constraints. Wicaksono and Karimi[35] have shown, using Fourier−Motzkin elimination, that each of the formulations in this class are equivalent in a projection of the $\{x,y,z\}$. However, as we will show in section 5, the different formulations have very different computational performances.

In a first approach, the partitioning of domain D_x is modeled through a set of semicontinuous variables u_n, $n = 1, 2, ..., N$, where $u_n \in \{0\} \cup [x_{n-1},x_n]$. A semicontinuous variable is a variable that has a regular continuous range D_c (typically, $0 \notin D_c$) but can also admit additionally the value of zero. Note that some optimization solvers allow the direct declaration of such variables, alleviating the need to explicitly expand their range so as to include the value of zero (an action that usually results is looser formulations). The variables u_n are equal to x itself, if $x \in [x_{n-1},x_n]$, while they are set to zero otherwise. This is represented by eqs 31. Since only one λ variable can be active at any given instance, only one variable u_n gets to contribute to the sum.

$$\begin{aligned} x &= \sum_{n=1}^{N} u_n \\ x_{n-1}\lambda_n &\le u_n \le x_n\lambda_n, \forall n \end{aligned} \tag{31}$$

The approach also requires a second set of semicontinuous variables ν_n, $n = 1, 2, ..., N$, where $\nu_n \in \{0\} \cup [y^L,y^U]$. These variables are equal to y, if $x \in [x_{n-1},x_n]$, while equal to zero otherwise, as dictated by eqs 32.

$$\begin{aligned} y &= \sum_{n=1}^{N} \nu_n \\ y^L\lambda_n &\le \nu_n \le y^U\lambda_n, \forall n \end{aligned} \tag{32}$$

Variables v_n are used in the place of terms that are linear in variable y and have coefficients that depend on the interval n. Therefore, they can activate or deactivate, depending on which subdomain is applicable, the appropriate facets of the bilinear piecewise envelope through the use of eqs 33.

$$\begin{aligned} z &\ge \sum_{n=1}^{N} \{y^L u_n + x_{n-1}\nu_n - x_{n-1}y^L\lambda_n\} \\ z &\ge \sum_{n=1}^{N} \{y^U u_n + x_n\nu_n - x_n y^U\lambda_n\} \\ z &\le \sum_{n=1}^{N} \{y^U u_n + x_{n-1}\nu_n - x_{n-1}y^U\lambda_n\} \\ z &\le \sum_{n=1}^{N} \{y^L u_n + x_n\nu_n - x_n y^L\lambda_n\} \end{aligned} \tag{33}$$

We can now define a novel formulation, 'ch', which consists of eqs 21 and 31−33.

A first variant of formulation 'ch' utilizes a third set of semicontinuous variables w_n, $n = 1, 2, ..., N$, where $w_n \in \{0\} \cup [w_n^L,w_n^U]$, whose function is to replace the bilinear product $z = xy$, if the corresponding subdomain is applicable, while being zero otherwise. The regular bounds w^L,w^U can be derived by simple interval arithmetic on the bilinear product that these variables are meant to replace over the corresponding subdomain.

$$\begin{aligned} w_n^L &= \min\{x_{n-1}y^L, x_{n-1}y^U, x_n y^L, x_n y^U\} \\ w_n^U &= \max\{x_{n-1}y^L, x_{n-1}y^U, x_n y^L, x_n y^U\} \end{aligned}$$

Although only one w_n variable will be nonzero at the optimal instance of the problem, we do not a priori know which one this is going to be. Therefore, all of them need to be taken into account through the sum of eq 34. As an alternative, one may directly replace every occurrence of xy in the original problem by $\sum_{n=1}^{N} w_n$, alleviating the need for the presence of variable z and eq 34 in the relaxation.

$$z = \sum_{n=1}^{N} w_n \tag{34}$$

The new representation of the facets of the subenvelopes, which makes use of all three sets of semicontinuous variables (i.e., u_n, ν_n, and w_n) corresponds to eqs 35. These constraints also take care of the requirement that all the nonactive w_n variables be set to zero.

$$\begin{aligned} w_n &\ge y^L u_n + x_{n-1}\nu_n - x_{n-1}y^L\lambda_n \\ w_n &\ge y^U u_n + x_n\nu_n - x_n y^U\lambda_n \\ w_n &\le y^U u_n + x_{n-1}\nu_n - x_{n-1}y^U\lambda_n \\ w_n &\le y^L u_n + x_n\nu_n - x_n y^L\lambda_n \end{aligned}, \forall n \tag{35}$$

We define the new relaxation scheme, 'tch', as the set of eqs 21, 31, 32, 34, and 35. According to the theorem by Balas,[42] formulation 'tch' cannot be less tight than 'ch'. However, its size is substantially bigger, due to the introduction of variables w_n and the replacement of the four constraints of eqs 33 by the $4N$ constraints of eqs 35.

In the remainder of the section, we will focus on a different type of approach which is also based on formulation 'ch'. However, we first need to define a new set of continuous variables δy_n, $n = 1, 2, ..., N$, where $0 \le \delta y_n \le y^U - y^L$. These variables are used to represent the deviation of variable y from its lower bound y^L, if $x \in [x_{n-1},x_n]$, while they are to be set to zero otherwise. Equations 36 constitute the appropriate set of constraints that implements this.

$$\begin{aligned} y &= y^L + \sum_{n=1}^{N} \delta y_n \\ 0 &\le \delta y_n \le (y^U - y^L)\lambda_n, \forall n \end{aligned} \tag{36}$$

Utilizing variables δy_n, in conjunction with variables δx_n, can lead to a powerful reformulation, as the bilinear product xy becomes

$$z = y^L x + \sum_{n=1}^{N} \{x_{n-1}\delta y_n\} + \sum_{n=1}^{N} \delta x_n \delta y_n \tag{37}$$

which involves N bilinear terms of type $\delta x_n \delta y_n$.

The advantage of such a reformulation is that the bilinear terms now involve deviation variables, all of which have a lower hard bound that is equal to zero. Thus, the facets of the subenvelopes (originally represented by eqs 18), will now be governed by the special case of eqs 38, leading to an overall reduction in the size of the relaxation. This reduction will be realized in terms of number of constraints, since the first facet is now just a hard bound, as well as in terms of actual nonzeros. Both features are very desirable from a computational point of view.

$$\begin{aligned} z &\ge 0 \\ z &\ge y^{upp}x + x^{upp}y - x^{upp}y^{upp} \\ z &\le y^{upp}x \\ z &\le x^{upp}y \end{aligned} \tag{38}$$

For the relaxation of the new types of bilinear terms, we introduce variables δz_n, $n = 1, 2, ..., N$, where $0 \le \delta z_n \le d_n(y^U - y^L)$. We use these variables to replace the occurrences of the corresponding products $\delta x_n \delta y_n$ in eq 37, resulting in eq 39.

$$z = y^L x + \sum_{n=1}^{N} \{x_{n-1}\delta y_n\} + \sum_{n=1}^{N} \delta z_n \tag{39}$$

The explicit representations of the facets correspond to eqs 40, which are derived from a suitable adaptation of eqs 38. Since $\delta x_n \delta y_n$ can be not equal to zero only for the active subdomain, the same should hold for the corresponding δz_n. We observe that these constraints also serve the purpose to appropriately drive the other δz_n to zero.

$$\begin{aligned} &\delta z_n \ge d_n \delta y_n + (y^U - y^L)\delta x_n - d_n(y^U - y^L)\lambda_n \\ &\delta z_n \le (y^U - y^L)\delta x_n \\ &\delta z_n \le d_n \delta y_n \end{aligned} \quad , \forall n \tag{40}$$

We define formulation nf3 that utilizes the new approach. It consists of eqs 21, 22, 36, 39, and 40.

A variant of this relaxation can be obtained by introducing variable Δz, where $0 \le \Delta z \le \max_{1 \le n \le N} \{d_n\}(y^U - y^L)$. This variable has global scope and corresponds to the aggregate of all δz_n variables. Therefore, eq 41 applies.

$$z = y^L x + \sum_{n=1}^{N} \{x_{n-1}\delta y_n\} + \Delta z \tag{41}$$

Furthermore, aggregating eqs 40 results into eqs 42.

$$\begin{aligned} &\Delta z \ge \sum_{n=1}^{N} \{d_n \delta y_n\} + (y^U - y^L)\sum_{n=1}^{N} \{\delta x_n - d_n \lambda_n\} \\ &\Delta z \le (y^U - y^L)\sum_{n=1}^{N} \delta x_n \\ &\Delta z \le \sum_{n=1}^{N} d_n \delta y_n \end{aligned} \tag{42}$$

Formulation 'nf4l' can now be defined as the set of eqs 21, 22, 36, 41, and 42. Although being no tighter than nf3 (due to the aggregation of constraints), this formulation corresponds to a substantial reduction of problem size, as $3N$ constraints are replaced by just 3.

Given the relationship between the global and local-scope variables ($\Delta x = \sum_{n=1}^{N} \delta x_n$), another simple variant results if we replace eqs 22 and 42 with eqs 23 and 43. We name this variant formulation 'nf4'. It corresponds to eqs 21, 23, 36, 41, and 43.

$$\begin{aligned} &\Delta z \ge \sum_{n=1}^{N} \{d_n \delta y_n\} + (y^U - y^L)(\Delta x - \sum_{n=1}^{N} d_n \lambda_n) \\ &\Delta z \le (y^U - y^L)\Delta x \\ &\Delta z \le \sum_{n=1}^{N} d_n \delta y_n \end{aligned} \tag{43}$$

In order to create a final variant of formulation 'nf3', we eliminate variable Δx from the problem relaxation and utilize eq 24 instead of eqs 23. This step is similar with the step followed so as to discriminate between formulations 'nf2g' and 'nf2'. Additionally, we solve eq 41 for Δz and eliminate it from the relaxation as well. The eliminations of variables Δx and Δz result in the adaptation of eqs 43 into eqs 44.

$$\begin{aligned} &z \ge y^U x + \sum_{n=1}^{N} \{x_n \delta y_n\} - (y^U - y^L)\sum_{n=1}^{N} x_n \lambda_n \\ &z \le y^U x + \sum_{n=1}^{N} \{x_{n-1}\delta y_n\} - (y^U - y^L)\sum_{n=1}^{N} x_{n-1}\lambda_n \\ &z \le y^L x + \sum_{n=1}^{N} x_n \delta y_n \end{aligned} \tag{44}$$

Note that, since we eliminated variable Δz, we also excluded its hard bounds from the formulation. For purposes of maintaining the original tightness, we need to make sure that the information provided by these bounds is retained. The lower bound ($\Delta z \ge 0$) is converted into eq 45. The upper hard bound becomes redundant in the presence of the third constraint of eqs 44.

$$z \ge y^L x + \sum_{n=1}^{N} x_{n-1}\delta y_n \tag{45}$$

To conclude the convex combination class, we define formulation 'nf4r' that corresponds to eqs 21, 24, 36, 44 and 45.

4.3. Incremental Cost Class. Unlike the previous two classes of methods that employed the λ variables, this class of formulations employs the θ variables to model the disjunctions. However, like the convex combination class, each of the formulations in the incremental cost class are equivalent in a projection of the $\{x,y,z\}$ variables.[35]

Before we proceed with the derivation of the various formulations, we introduce a new set of continuous variables δw_n, $n = 1, 2, ..., N$, where $0 \le \delta w_n \le y^U - y^L$. These variables are defined as $\delta w_n = \delta u_n(y - y^L)$ and will be used in representing the original bilinear term $z = xy$ through the use of eq 46. This equation results after multiplying eqs 26 with $(y - y^L)$ and applying the definitions of δw_n.

$$z = y^L x + x^L y - x^L y^L + \sum_{n=1}^{N} d_n \delta w_n \tag{46}$$

In order to tighten up the formulation, it would be wise to take into account the information provided by the definition of the variables w_n and relax the bilinearities $\delta u_n(y - y^L)$ that it involves. The envelope facets for these bilinear terms are given in eqs 47. These are derived from eqs 18 using $x \leftarrow \delta u_n$, $y \leftarrow (y - y^L)$, and sets of appropriate lower and upper bounds ($\delta u_1 \in [\theta_1,1]$, $\delta u_n \in [\theta_n,\theta_{n-1}]$ $\forall$ $n \ne 1, N$, $\delta u_N \in [0,\theta_{N-1}]$, and $(y - y^L) \in [0,y^U - y^L]$).

$$\begin{aligned} &\delta w_1 \ge (y^U - y^L)\delta u_1 + y - y^U \\ &\delta w_n \ge (y - y^L)\theta_n, \forall n < N \\ &\delta w_n \ge (y^U - y^L)(\delta u_n - \theta_{n-1}) + (y - y^L)\theta_{n-1}, \forall n > 1 \\ &\delta w_1 \le y - y^L \\ &\delta w_n \le (y^U - y^L)(\delta u_n - \theta_n) + (y - y^L)\theta_n, \forall n < N \\ &\delta w_n \le (y - y^L)\theta_{n-1}, \forall n > 1 \\ &\delta w_N \le (y^U - y^L)\delta u_N \end{aligned} \tag{47}$$

We observe that the facets involve a new set of bilinearities of the type $(y - y^L)\theta_n$. Therefore, for a complete MILP relaxation, we need to further relax these as well. For this purpose, we introduce a second set of variables δv_n, $n = 1, 2, ..., N - 1$, where $0 \le \delta v_n \le y^U - y^L$. These variables are defined as $\delta v_n = (y - y^L)\theta_n$ and will replace every occurrence of the bilinearities in eqs 47, resulting in eqs 48.

$$\begin{aligned}
&\delta w_1 \geq (y^U - y^L)\delta u_1 + y - y^U\\
&\delta w_n \geq \delta v_n, \forall n < N\\
&\delta w_n \geq (y^U - y^L)(\delta u_n - \theta_{n-1}) + \delta v_{n-1}, \forall n > 1\\
&\delta w_1 \leq y - y^L\\
&\delta w_n \leq (y^U - y^L)(\delta u_n - \theta_n) + \delta v_n, \forall n < N\\
&\delta w_n \leq \delta v_{n-1}, \forall n > 1\\
&\delta w_N \leq (y^U - y^L)\delta u_N
\end{aligned} \tag{48}$$

The envelope facets of variables δv_n need to be constructed also. These are derived from eqs 18 using $x \leftarrow (y - y^L)$, $y \leftarrow \theta_n$, and sets of appropriate lower and upper bounds ($\theta_n \in [\delta u_{n+1}, \delta u_n]$ and $(y - y^L) \in [0, y^U - y^L]$). However, they all happen to coincide with some of the facets of eqs 48 and, therefore, are omitted from the relaxation as they would not offer any additional tightening.

We define now our first incremental cost formulation, 'nf5', which consists of eqs 26, 46, and 48. A variant of this relaxation can be obtained if, in the above analysis for the derivation of eqs 48, we take into account the hard bounds for variables δu_n instead of the tighter bounds that involve the θ variables. By doing so, eqs 48 can be replaced by the smaller set of eqs 49, which correspond to the new envelope facets of the bilinear term $\delta u_n(y - y^L)$. The fourth set of facets just coincides with the lower hard bounds $w_n \geq 0$ and, thus, is omitted from the presentation.

$$\begin{aligned}
&\delta w_n \geq (y^U - y^L)\delta u_n + y - y^U\\
&\delta w_n \leq y - y^L \qquad\qquad , \forall n\\
&\delta w_n \leq (y^U - y^L)\delta u_n
\end{aligned} \tag{49}$$

We define formulation 'nf6' that corresponds to eqs 26, 46, and 49. This formulation cannot be tighter than formulation 'nf5' since looser bounds were utilized for the derivation of the envelope facets. However, its size is considerably smaller as it does not require the additional δv_n variables in representing the relaxation. Tightening of this relaxation scheme can be achieved if we additionally take into account the inherent relationship $\delta w_n \geq \delta w_{n+1}$, $\forall\, n < N$, which results from $\delta u_n \geq \delta u_{n+1}$, $\forall\, n < N$ (see eqs 26). These relationships can be combined with eqs 49 into eqs 50.

$$\begin{aligned}
&\delta w_n \geq (y^U - y^L)\delta u_n + y - y^U, \forall n\\
&\delta w_1 \leq y - y^L\\
&\delta w_n \leq w_{n-1}, \forall n > 1\\
&\delta w_n \leq (y^U - y^L)\delta u_n, \forall n
\end{aligned} \tag{50}$$

Formulation 'nf6t' can be defined as the set of eqs 26, 46, and 50. Like the other members of the incremental cost class, this formulation has an equivalent projected feasible region in the space of variables $\{x,y,z\}$ with formulation 'nf5'.[35]

For the next formulation in this class, we make use of the global-scope deviation variable Δx through its definition in eqs 27. Moreover, we need to define a new continuous variable Δw, where $0 \leq \Delta w \leq \max_{1\leq n\leq N} d_n(y^U - y^L)$. This variable is defined as $\Delta w = \Delta x(y - y^L)$ and will be used in representing the original bilinear term $z = xy$ through the use of eq 51. This equation results after multiplying eqs 27 with $(y - y^L)$ and applying the definition of Δw.

$$z = y^L x + x^L y - x^L y^L + \sum_{n=1}^{N-1} \{d_n \delta v_n\} + \Delta w \tag{51}$$

Since variables Δw and δv_n participate in the problem, the information provided by their definitions should be exploited by relaxing the bilinear terms that they involve.

The envelope of the bilinear term $\Delta x(y - y^L)$ corresponds to eqs 52. These are derived from eqs 18 using $x \leftarrow \Delta x$, $y \leftarrow (y - y^L)$, and the appropriate bounds ($\Delta x \in [0, d_1 + \sum_{n=1}^{N-1} \{(d_{n+1} - d_n)\theta_n\}]$ and $(y - y^L) \in [0, y^U - y^L]$). An additional facet coincides with the hard bound $\Delta w \geq 0$ and is not taken into account as an explicit constraint.

$$\begin{aligned}
&\Delta w \geq (y^U - y^L)\Delta x + d_1(y - y^U) + \sum_{n=1}^{N-1} \{(d_{n+1} - d_n)(\delta v_n - (y^U - y^L)\theta_n)\}\\
&\Delta w \leq d_1(y - y^L) + \sum_{n=1}^{N-1} \{(d_{n+1} - d_n)\delta v_n\}\\
&\Delta w \leq (y^U - y^L)\Delta x
\end{aligned} \tag{52}$$

The envelope facets needed for the relaxation of bilinear terms $(y - y^L)\theta_n$ are given in eqs 53. They are derived from eqs 18 using $x \leftarrow (y - y^L)$, $y \leftarrow \theta_n$, and sets of appropriate lower and upper bounds ($\theta_1 \in [\theta_2, 1]$, $\theta_n \in [\theta_{n+1}, \theta_{n-1}]$ $\forall\, n \neq 1, N$, $\theta_{N-1} \in [0, \theta_{N-1}]$, and $(y - y^L) \in [0, y^U - y^L]$). Note that, unlike the case of formulation nf5, the bounds used for the θ variables are not based on variables δu_n, rather than on eq 25. This occurred simply because variables δu_n do not participate in this relaxation scheme and their inclusion would unnecessarily enlarge its size. An interesting corollary is the fact that the information provided by eq 25 is implicitly taken into account, alleviating the need for its inclusion in the formulation.

$$\begin{aligned}
&\delta v_1 \geq (y^U - y^L)\theta_1 + y - y^U\\
&\delta v_n \geq \delta v_{n+1}, \forall n < N - 1\\
&\delta v_1 \leq y - y^L\\
&\delta v_n \leq (y^U - y^L)(\theta_n - \theta_{n+1}) + \delta v_{n+1}, \forall n < N - 1\\
&\delta v_{N-1} \leq (y^U - y^L)\theta_{N-1}
\end{aligned} \tag{53}$$

An additional facet coincides with the lower hard bound $\delta v_{N-1} \geq 0$. Equations 27, 51, 52, and 53) define formulation 'nf7'.

Finally, another scheme, dubbed 'nf7r', can be derived. It calls for removing both variables Δx and Δw from the formulation by replacing eqs 27 and 51 with eqs 28 and 54. Consequently, eqs 52 become eqs 55.

$$z \geq y^L x + x^L y - x^L y^L + \sum_{n=1}^{N-1} d_n \delta v_n \tag{54}$$

Note that eq 54 corresponds to the information provided by the lower hard bound $\Delta w \geq 0$. The upper hard bound is redundant in the presence eqs 55. The complete formulation consists of eqs 28, 53, 54, and 55.

$$\begin{aligned}
&z \geq y^U x + x^L y - x^L y^U + d_1(y - y^U) + \sum_{n=1}^{N-1} \{d_{n+1}(\delta v_n - (y^U - y^L)\theta_n)\}\\
&z \leq y^L x + x^L y - x^L y^L + d_1(y - y^L) + \sum_{n=1}^{N-1} d_{n+1}\delta v_n\\
&z \leq y^U x + x^L y - x^L y^U + \sum_{n=1}^{N-1} \{d_n(\delta v_n - (y^U - y^L)\theta_n)\}
\end{aligned} \tag{55}$$

5. Computational Comparisons

The 15 MILP relaxation formulations of Table 2 have been applied to the set of benchmark pooling problems outlined in Tables 3−5. These benchmarks are widely used in the literature to assess the performance of relaxation schemes and global

Table 3. List of Benchmark Pooling Problems and Their Size

	variables						constraints						
					total		linear		nonlinear		bilinear terms		
	x_{il}/q_{il}	y_{lj}	z_{ij}	p_{lk}	P f	Q f	P f	Q f	P f	Q f	P f	Q f	global opt
Hav1[1]	2	2	2	1	7	6	3	3	3	2	2	4	−400
Hav2[1]	2	2	2	1	7	6	3	3	3	2	2	4	−600
Hav3[1]	2	2	2	1	7	6	3	3	3	2	2	4	−750
Fou2[24]	4	8	8	2	22	20	6	6	6	4	8	16	−1100
Fou3[24]	32	128	0	8	168	160	24	24	24	16	128	512	−8
Fou4[24]	32	128	0	8	168	160	24	24	24	16	128	512	−8
Fou5[24]	32	64	0	4	100	96	20	20	20	16	64	512	−8
BT4[13]	3	2	2	1	8	7	4	3	3	3	2	6	−450
BT5[13]	12	15	5	6	38	32	9	8	16	11	30	60	−3500
Adh1[14]	5	8	0	8	21	13	6	6	24	16	32	20	−549.80
Adh2[14]	5	8	0	12	25	13	6	6	36	24	48	20	−549.80
Adh3[14]	8	12	0	18	38	20	7	7	42	24	72	32	−561.05
Adh4[14]	8	10	0	8	26	18	7	7	28	20	40	40	−877.65
RT2[19]	6	6	4	8	24	16	10	7	17	15	24	18	−4391.83
EPA1[43]	4	2	8	11	25	14	14	6	30	36	22	8	−473.47

Table 4. List of Benchmark Pooling Problems and Their Topology[a]

problem	I	L	J	K	T_X	T_Y	T_Z
Hav1	3	1	2	1	$\{(i,l): i = 1, 2\}$	Ω	$\{(i,j): i = 3\}$
Hav2	3	1	2	1	$\{(i,l): i = 1, 2\}$	Ω	$\{(i,j): i = 3\}$
Hav3	3	1	2	1	$\{(i,l): i = 1, 2\}$	Ω	$\{(i,j): i = 3\}$
Fou2	6	2	4	1	{(1, 1); (2, 1); (4, 2); (5, 2)}	Ω	$\{(i,j): i = 3, 6\}$
Fou3	11	8	16	1	$\{(i,l): i - 3 \le l \le i\}$	Ω	Ø
Fou4	11	8	16	1	T_{Xf4}	Ω	Ø
Fou5	11	4	16	1	T_{Xf5}	Ω	Ø
BT4	4	1	2	1	$\{(i,l): i \neq 3\}$	Ω	$\{(i,j): i = 3\}$
BT5	5	3	5	2	$\{(i,l): i \neq 3\}$	Ω	$\{(i,j): i = 3\}$
Adh1	5	2	4	4	{(1, 1); (2, 1); (3, 2); (4, 2); (5, 2)}	Ω	Ø
Adh2	5	2	4	6	{(1, 1); (2, 1); (3, 2); (4, 2); (5, 2)}	Ω	Ø
Adh3	8	3	4	6	T_{Xa3}	Ω	Ø
Adh4	8	2	5	4	$\{(i, 1): i \le 4\} \cup \{(i; 2): i \ge 5\}$	Ω	Ø
RT2	3	2	3	4	Ω	Ω	{(1, 2); (2, 1); (2, 3); (3, 1)}
EPA1	7	1	2	11	$\{(i,l): i \le 4\}$	Ω	$\{(i,j): i \ge 4\}$

where $T_{Xf4} = \{(1, 1); (2, 2); (2, 3); (3, 3); (3, 4); (3, 5); (4, 1); (4, 4)\}$
$\cup \{(4, 6); (4, 7); (5, 2); (5, 3); (5, 5); (5, 8); (6, 3); (6, 4)\}$
$\cup \{(6, 5); (6, 6); (7, 1); (7, 4); (7, 6); (7, 7); (8, 2); (8, 5)\} \cup \{(8, 7); (8, 8); (9, 6); (9, 7); (9, 8); (10, 1); (10, 8); (11, 2)\}$
$T_{Xf5} = \{(i, 1): i \neq 5, 6, 7\} \cup \{(i, 2): i \neq 1, 6, 11\}$
$\cup \{(i, 3): i \neq 1, 2, 3\} \cup \{(i, 4): i \neq 9, 10, 11\}$
$T_{Xa3} = \{(i, 1): i \le 2\} \cup \{(i; 2): 3 \le i \le 5\} \cup \{(i, 3): i \ge 6\}$

[a] Ω denotes the universal set (i.e., all connections present). Ø denotes the empty set.

optimization algorithms. References for these problems, as well as information regarding their size and global solution, can be found in Table 3. Note that an appropriate count of the number of constraints should not consider the cases where $A_i^L = 0$, $A_i^U \to \infty$, $S_l \to \infty$, $D_j^L = 0$, $D_j^U \to \infty$, $P_{jk}^L \le \min_i C_{ik}$, and $P_{jk}^U \ge \max_i C_{ik}$, as they render the respective constraints superfluous. Accordingly, the numbers listed in Table 3 do not include such cases. The overall network topology for the benchmark problems can be found in Table 4. Table 5 lists the values for all the parameters involved.

In order to relax the bilinear terms $z \leftarrow y_{lj}p_{lk}$ in the 'P'-formulation of each problem ($z \leftarrow y_{lj}q_{il}$ in the 'Q'-formulation), two partitioning approaches can be followed. One is to map $x \leftarrow y_{lj}$, $y \leftarrow p_{lk}$ ($y \leftarrow q_{il}$), and partition the domain of each variable y_{lj} into N subdomains. The other is to map $x \leftarrow p_{lk}$ ($x \leftarrow q_{il}$), $y \leftarrow y_{lj}$, and, therefore, apply the partitioning on the p_{lk} (q_{il}) variables. We shall refer to these approaches as the 'y'-variant and the 'p'-variant ('q'-variant), respectively. Of course, there exist other approaches based on partitioning a mixed set of y_{lj} and p_{lk} (q_{il}) variables (as long as the selection covers all bilinear terms participating in the problem), as well as opportunities for selecting a different number of subdomains for each partitioned variable. However, given the numerous combinations and alternatives that would have to be considered otherwise, we will restrict this study to the two variants of each formulation as described above.

For each of the two variants in both formulations, we consider a number of different partitioning levels N. The exact values are selected from the elements of the set:

$$N = \{2, 3, 4, 5, 7, 10, 15\}$$

Depending on the particular scheme and the value of N used, the size of the relaxation formulation that has to be appended to the original problem can be calculated from the entries of Table 2. The number of distinct bilinear terms B for each of the problems, as well as the number of partitioned variables V, is given in Table 3. Note that in the case of the 'y'-variant, V equals the number of the y_{lj} variables (also equal to $|T_Y|$), while in the case of the 'p'-variant ('q'-variant), V equals the number of the p_{lk} (q_{il}) variables, i.e., $L \cdot K$ in the 'P'-formulation and $|T_X|$ in the 'Q'-formulation. The calculated size has to be added to the size of the original problem, as given in Table 3, so as to obtain the complete size of the pooling problem relaxation. Notice that the problem sizes listed in Table 4 are not only topology dependent but that the number of bilinear terms in the original problem further depends on the problem formulation. In the case of Fou3−5, there are many more bilinear terms in

Table 5. Parameter Values for the Benchmark Pooling Problems[a]

problem	c_i	d_j	A_l^L	A_l^U	S_l	D_j^L	D_j^L	C_{ik}	P_{jk}^L	P_{jk}^U
Hav1	$[6\ \ 16\ \ 10]^T$	$[9\ \ 15]^T$	0	∞	∞	0	$[100\ \ 200]^T$	$[3\ \ 1\ \ 2]^T$	0	$[2.5\ \ 1.5]^T$
Hav2	$[6\ \ 16\ \ 10]^T$	$[9\ \ 15]^T$	0	∞	∞	0	$[600\ \ 200]^T$	$[3\ \ 1\ \ 2]^T$	0	$[2.5\ \ 1.5]^T$
Hav3	$[6\ \ 13\ \ 10]^T$	$[9\ \ 15]^T$	0	∞	∞	0	$[600\ \ 200]^T$	$[3\ \ 1\ \ 2]^T$	0	$[2.5\ \ 1.5]^T$
Fou2	$[6\ \ 13\ \ 10\ \ 3\ \ 13\ \ 7]^T$	$[9\ \ 15\ \ 6\ \ 12]^T$	0	∞	∞	0	$[100\ \ 200\ \ 100\ \ 200]^T$	$[3\ \ 1\ \ 2\ \ 3.5\ \ 1.5\ \ 2.5]^T$	0	$[2.5\ \ 1.5\ \ 3\ \ 2]^T$
Fou3	$21 - i$	$20.5 - 0.5i$	0	∞	∞	0	1	$0.9 + 0.1i$	0	$1 + 0.05i$
Fou4	$21 - i$	$20.5 - 0.5i$	0	∞	∞	0	1	$0.9 + 0.1i$	0	$1 + 0.05i$
Fou5	$21 - i$	$20.5 - 0.5i$	0	∞	∞	0	1	$0.9 + 0.1i$	0	$1 + 0.05i$
BT4	$[6\ \ 16\ \ 10\ \ 15]^T$	$[9\ \ 15]^T$	0	A_{b4}^U	∞	0	$[100\ \ 200]^T$	$[3\ \ 1\ \ 2\ \ 2\ \ 1]^T$	0	$[2.5\ \ 1.5]^T$
BT5	$[6\ \ 16\ \ 10\ \ 15\ \ 12]^T$	$[18\ \ 15\ \ 19\ \ 16\ \ 14]^T$	0	A_{b5}^U	∞	0	$[100\ \ 200\ \ 100\ \ 100\ \ 100]^T$	C_{b5}	0	P_{b5}^U
Adh1	$[7\ \ 3\ \ 2\ \ 10\ \ 5]^T$	$[16\ \ 25\ \ 15\ \ 10]^T$	0	∞	∞	0	$[10\ \ 25\ \ 30\ \ 10]^T$	C_{a1}	0	P_{a1}^U
Adh2	$[7\ \ 3\ \ 2\ \ 10\ \ 5]^T$	$[16\ \ 25\ \ 15\ \ 10]^T$	0	∞	∞	0	$[10\ \ 25\ \ 30\ \ 10]^T$	C_{a2}	0	P_{a2}^U
Adh3	$[7\ \ 3\ \ 2\ \ 10\ \ 5\ \ 5\ \ 9\ \ 11]^T$	$[16\ \ 25\ \ 15\ \ 10]^T$	0	∞	∞	0	$[10\ \ 25\ \ 30\ \ 10]^T$	C_{a3}	0	P_{a3}^U
Adh4	$[15\ \ 7\ \ 4\ \ 5\ \ 6\ \ 3\ \ 5\ \ 5]^T$	$[10\ \ 25\ \ 30\ \ 6\ \ 10]^T$	0	∞	∞	0	$[15\ \ 25\ \ 10\ \ 20\ \ 15]^T$	C_{a4}	0	P_{a4}^U
RT2	$[49.2\ \ 62\ \ 300]^T$	$[190\ \ 230\ \ 150]^T$	0	A_{r2}^U	$[12.5\ \ 17.5]^T$	5	∞	C_{r2}	P_{r2}^L	P_{r2}^U
EPA1	$[2\ \ 8\ \ 10\ \ 16\ \ 2\ \ 2\ \ 5]^T$	$[6\ \ 12]^T$	A_{e1}^L	A_{e1}^U	300	100	200	C_{e1}	P_{e1}^L	P_{e1}^U

Table 5. Continued

where

$$C_{b5} = \begin{bmatrix} 3 & 1 & 2 & 1 & 1.5 \\ 1 & 3 & 2.5 & 2.5 & 2.5 \end{bmatrix}^{\mathrm{T}}, P_{b5}^{\mathrm{U}} = \begin{bmatrix} 2.5 & 1.5 & 2 & 2 & 2 \\ 2 & 2.5 & 2.6 & 2 & 2 \end{bmatrix}^{\mathrm{T}}$$

$$C_{a1} = \begin{bmatrix} 1 & 4 & 4 & 3 & 1 \\ 6 & 1 & 5.5 & 3 & 2.7 \\ 4 & 3 & 3 & 3 & 4 \\ 0.5 & 2 & 0.9 & 1 & 1.6 \end{bmatrix}^{\mathrm{T}}, P_{a1}^{\mathrm{U}} = \begin{bmatrix} 3 & 3 & 3.25 & 0.75 \\ 4 & 2.5 & 3.5 & 1.5 \\ 1.5 & 5.5 & 3.9 & 0.8 \\ 3 & 4 & 4 & 1.8 \end{bmatrix}^{\mathrm{T}}$$

$$C_{a2} = \begin{bmatrix} & C_{a1}^{\mathrm{T}} & & & \\ 5 & 4 & 7 & 3 & 3 \\ 9 & 4 & 10 & 4 & 7 \end{bmatrix}^{\mathrm{T}}, C_{a3} = \begin{bmatrix} & & & C_{a2} & & \\ 1.8 & 2.7 & 4 & 3.5 & 6.1 & 3 \\ 5 & 1 & 1.7 & 2.9 & 3.5 & 2.9 \\ 3 & 3 & 3 & 1 & 5 & 2 \end{bmatrix}^{\mathrm{T}}, P_{a2}^{\mathrm{U}} = P_{a3}^{\mathrm{U}} = \begin{bmatrix} & P_{a1}^{\mathrm{U\,T}} & & \\ 6 & 7 & 7 & 6 \\ 5 & 6 & 6 & 6 \end{bmatrix}^{\mathrm{T}}$$

$$C_{a4} = \begin{bmatrix} 0.5 & 1.4 & 1.2 & 1.5 & 1.6 & 1.1 & 1.5 & 1.4 \\ 1.9 & 1.8 & 1.9 & 1.2 & 1.8 & 1.1 & 1.5 & 1.6 \\ 1.3 & 1.7 & 1.4 & 1.7 & 1.6 & 1.4 & 1.5 & 1.2 \\ 1 & 1.6 & 1.4 & 1.3 & 2 & 2 & 1.5 & 1.6 \end{bmatrix}^{\mathrm{T}}, P_{a4}^{\mathrm{U}} = \begin{bmatrix} 1.2 & 1.4 & 1.3 & 1.2 & 1.6 \\ 1.7 & 1.3 & 1.3 & 1.1 & 1.9 \\ 1.4 & 1.8 & 1.9 & 1.7 & 2 \\ 1.7 & 1.4 & 1.9 & 1.6 & 2.5 \end{bmatrix}^{\mathrm{T}}$$

$$C_{r2} = \begin{bmatrix} 0.82 & 3 & 99.2 & 90.5 \\ 0.62 & 0 & 87.9 & 83.5 \\ 0.75 & 0 & 114 & 98.7 \end{bmatrix}, P_{r2}^{\mathrm{L}} = \begin{bmatrix} 0.74 & 0 & 95 & 85 \\ 0.74 & 0 & 96 & 88 \\ 0.74 & 0 & 91 & 0 \end{bmatrix}, P_{r2}^{\mathrm{U}} = \begin{bmatrix} 0.79 & \infty & \infty & \infty \\ 0.79 & 0.9 & \infty & \infty \\ 0.79 & \infty & \infty & \infty \end{bmatrix}$$

$$C_{e1} = \begin{bmatrix} 0.1 & 800 & 6 & 20 & 70 & 50 & 0 & 10 & 0 & 0 & 0 \\ 0.2 & 400 & 8.8 & 60 & 85 & 30 & 0.8 & 15 & 0 & 0 & 0 \\ 0.4 & 200 & 8 & 55 & 80 & 25 & 1 & 15 & 0 & 0 & 0 \\ 0.7 & 100 & 8 & 50 & 75 & 10 & 0.2 & 5 & 0 & 0 & 0 \\ 18.2 & 0 & 8.4 & 100 & 100 & 0 & 0 & 0 & 18.2 & 0 & 0 \\ 15.7 & 0 & 8 & 100 & 100 & 0 & 0 & 0 & 0 & 15.7 & 0 \\ 34.8 & 0 & 9.6 & 100 & 100 & 0 & 0 & 0 & 0 & 0 & 34.8 \end{bmatrix}$$

$$P_{e1}^{\mathrm{L}} = \begin{bmatrix} 2.7 & 50 & 6.4 & 30 & 70 & 0 & 0 & 0 & 0 & 0 & 2.05 \\ 2.7 & 50 & 6.4 & 30 & 70 & 0 & 0 & 0 & 0 & 0 & 2.05 \end{bmatrix}$$

$$P_{e1}^{\mathrm{U}} = \begin{bmatrix} 4 & 500 & 10 & 70 & 100 & 30 & 2 & 25 & 0.91 & 0.79 & 3.48 \\ 4 & 250 & 8 & 60 & 85 & 25 & 0.5 & 10 & 0.91 & 0.79 & 3.48 \end{bmatrix}$$

$$A_{b4}^{\mathrm{U}} = [\infty \;\; \infty \;\; \infty \;\; 50]^{\mathrm{T}}, A_{b5}^{\mathrm{U}} = [\infty \;\; \infty \;\; \infty \;\; 50 \;\; \infty]^{\mathrm{T}}, A_{r2}^{\mathrm{U}} = [60.9756 \;\; 161.29 \;\; 5]^{\mathrm{T}}, A_{e1}^{\mathrm{L}} = [50 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^{\mathrm{T}}, A_{e1}^{\mathrm{U}} = [400 \;\; 200 \;\; 200 \;\; 100 \;\; 10 \;\; 10 \;\; 50]^{T}$$

[a] A scalar entry (incl. ∞) may be a vector or a matrix, depending on the applicable case.

the '*Q*'- than '*P*'-formulation (512 versus 128 or 64), but this observation reverses in the case of the Adhya problems.

In order to investigate the effect of nonuniform partitioning, we define parameter $\gamma > 0$ which is used in the selection of the grid points according to eq 56. As $\gamma \rightarrow 0$, the grid points tend to accumulate toward the upper end of the domain; that is, $x_n \rightarrow x^U$, $\forall\ n > 0$. Conversely, as $\gamma \rightarrow \infty$, the grid points are selected closer to the lower end; that is, $x_n \rightarrow x^L$, $\forall\ n > 0$. Note that the case of $\gamma = 1$ corresponds to uniform partitioning. The lengths of the intervals thus generated are given by eq 57.

$$x_n = x^L + \left(\frac{n}{N}\right)^{\gamma}(x^U - x^L), \quad 0 \le n \le N \tag{56}$$

$$d_n = \left[\left(\frac{n}{N}\right)^{\gamma} - \left(\frac{n-1}{N}\right)^{\gamma}\right](x^U - x^L), \quad 1 \le n \le N \tag{57}$$

For each of the *N* values considered in this study, ten different values of γ are used and these are selected from the elements of the set

$$\Gamma = \{0.25, 0.50, 0.75, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00\}$$

Cumulatively, a total of 56 700 different problem relaxations are considered (for each benchmark problem, there are 10 γ values, 7 partitioning levels, 15 MILP relaxation formulations, 2 sets of variables participating in bilinear terms, and 2 problem formulations). All 15 benchmark pooling problems were considered using the '*P*'-formulation, but the Fou3−5[24] problems were not considered using the '*Q*'-formulation because of the size of these formulations makes the '*P*'-formulation far better than the '*Q*'-formulation (note from Table 3 that the '*Q*'-formulation of Fou3−5 requires 512 distinct binary terms and partitioning at least 32 variables).

Given the large amount of results, we shall first clarify the objectives of this comparative study. First and foremost, it should be mentioned that it is meaningless to draw conclusions by comparing results between two different pooling problems, as these are generally of different size and entail a different degree of computational difficulty. However, the number of benchmark problems (15) is sufficient to allow us to draw reliable conclusions regarding the overall performance of the various schemes and partitioning variants. The main objectives that we will focus on are the following:

(a) understanding the effect that the partitioning level *N* has on the lower bound and quantifying the associated effect on the computational effort required,

(b) exploring the effect of nonuniform partitioning, as characterized by the parameter γ,

(c) investigating which piecewise schemes most efficiently relax the pooling problems,

(d) assessing whether the '*P*'- or '*Q*'-formulation offers a leading advantage and, within the formulation, which variant ('*y*', '*p*', or '*q*') should be used.

All of the problems were formulated in GAMS[44] and the MILP solver CPLEX[45] was used for solving all the relaxations to optimality. Each of the optimal solutions constitutes a valid lower bound to the global solution of the respective pooling problem, and corresponds to the root-node solution of a global optimization branch-and-bound procedure. Various quantities that relate to computational performance, such as total CPU time, number of nodes visited in the MILP branch-and-bound tree, and total number of LP iterations, were tracked and recorded.

In Figures 3−5, we present the lower bounds generated by the '*P*'-formulation as a function of *N*, for all the problems and variants. Similarly, Figures 6−9 present the lower bounds for the '*Q*'-formulation. The different curves correspond to different values of the γ parameter, with the special case of uniform partitioning ($\gamma = 1$) depicted by the dark solid line. Note that each of these plots corresponds to every one of the 15 piecewise relaxation schemes, as all relaxation formulations are equivalent. The relaxation formulations only differ in terms of the computational effort needed to obtain these results, an issue that will be addressed later. The plots include also the lower bound obtained in the case of $N = 1$. It should be clear that, in all cases, this bound is the one that would have been determined with the standard bilinear relaxation proposed by McCormick.[38]

In this part of our analysis, we do not take into account the '*P*'-formulation of problems Fou2−5 and BT5, as in these problems the optimality gap is closed for all the cases considered (i.e., the standard technique of $N = 1$ suffices). Their plots would correspond to sets of overlapping horizontal lines at the level of the global optimum, and thus, their presentation is omitted. However, observe from Figure 9 that closing the optimality gap is formulation-dependent and the gap does not close, even for $N = 15$, in the '*Q*'-formulation of Fou2. Further, the lower bounds generated by the '*Q*'-formulation of Fou3−5 and BT5 are not diagrammed because their prohibitive size prevents us from obtaining the relaxation values within a reasonable amount of time (1 h). Finally, note the resemblance between the plots for Hav1 and BT4. This can be attributed to the fact that BT4 has been developed as a larger instance of Hav1 (see Tables 4 and 5 for details).

We observe that the piecewise relaxations provide tighter bounds as *N* increases, asymptotically approaching the best possible value which is the global optimum of the given problem. In many of the '*P*'-formulation cases, the relaxation schemes were able to exactly converge to the global optimum, and in some cases, this occurred for even the modest partitioning level of $N = 2$ (e.g., Hav1 ('*P*'-formulation)/'*y*'-variant, $\gamma = 1$). In the branch-and-bound global optimization algorithm designed by Karuppiah and Grossmann,[34] tight relaxation schemes such as these markedly reduced the number of nodes explored in the enumeration tree.

Despite the general ascending trend, monotonic increase of the lower bound with increase in *N* does not always hold. In the '*p*'-variant of the '*P*'-formulation of the EPA1 problem, for example, $N = 2$ provides a higher lower bound than $N = 3$ for the case of $\gamma = 1$ (solid line). A similar observation holds in the case of $\gamma = 0.75$, for partitioning levels $N = 3$ and $N = 4$. However, a relaxation scheme that corresponds to a partitioning level *N* can never be tighter than one that corresponds to a partitioning level $N' = sN$, where $s \in \mathbb{N}^*$; that is, when N' is an integer multiple of *N*. The reasoning behind this is the fact that every subdomain that results from a partitioning into *sN* intervals ought to be a subset of one of the subdomains that resulted from the partitioning into only *N* intervals. Comparing eqs 16 and 17 with those that would result if $x^{low} \rightarrow x^{low} + \delta x$ or $x^{upp} \rightarrow x^{upp} - \delta x$, it follows that all applicable bilinear subenvelopes in the finer partitioning case are tighter than in the sparser one, leading to a lower bound that cannot be looser. Clearly, this does not hold for the cases highlighted above, since $N'/N = 3/2$ and $4/3$, respectively.

The effect of nonuniform partitioning appears to be strong, as there are significant differences between the various curves with respect to the rate of lower bound improvement. For example, in the case of the '*y*'-variant, we observe that the curves for the '*P*'-formulation of the Adhya problems span over a wide LB-range, with smaller values of γ exhibiting inferior convergence rates. The '*p*'-variant plot for Adh4 initially shows similar convergence rate for all curves, but this trend is not

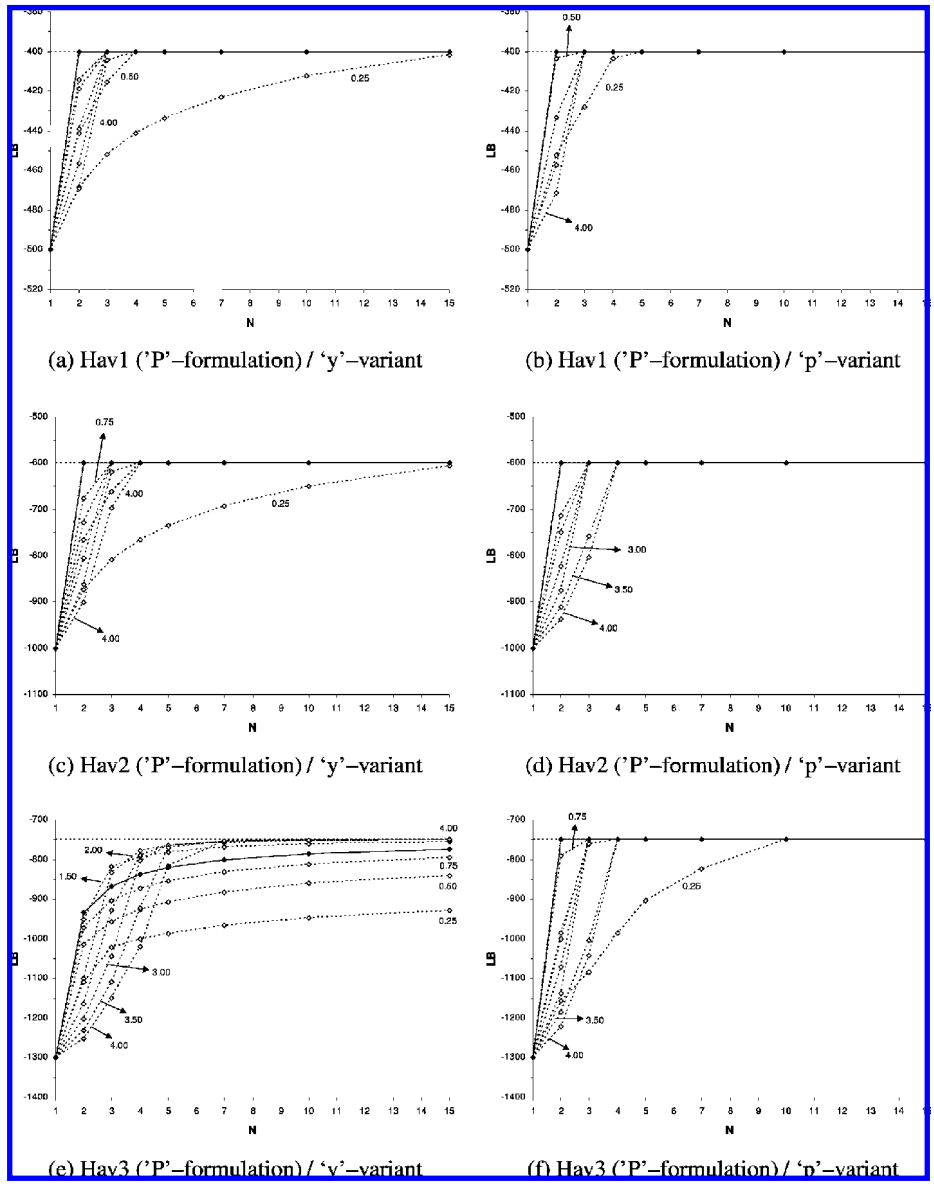

Figure 3. Lower bounds for the Haverly problems as a function of partitioning level N and parameter γ.

maintained in the large-N regime. The 'p'-variant plots for RT2 and EPA1 are highly unbalanced, with curves scattered throughout. On the other hand, there are cases where a great degree of uniformity is enjoyed. This happens in the 'p'-variant for problems Hav1, Hav2, BT4, Adh1, and Adh2, where the lower bounds are tightened relatively rapidly and the curves practically coincide after some N value.

For each of the problems, there usually exists an optimal γ value that works best, dominating the others for most partitioning levels. This is often the natural value of $\gamma = 1$, which corresponds to a uniform grid, but there are many cases where this is not true. In general, moderate values of γ (i.e., closer to the uniform partitioning level of $\gamma = 1$) perform better than extreme values. Furthermore, note that the plots maintain some level of coherence, in the sense that near values of γ correspond to neighboring curves. However, it is not uncommon for curves to cross each other. The higher γ values exhibit some hysteresis and are not as efficient for small values of N, however they quickly recover and end up being more efficient than their lower-γ counterparts in the large-N regime. Conversely, the low-γ curves do not exhibit this delay; however, they converge to the global optimum solution at a significantly smaller rate.

In any case, prior experience with a particular problem would be necessary so as to identify the optimal setting for the γ parameter. This is not a great limitation, though, as most real life problems are usually solved repeatedly over certain time cycles (e.g., the case of an oil refinery where a daily schedule of delivery operations is generated), with each fresh problem corresponding to a small perturbation of problems solved in the past. Therefore, it is not unlikely that the potential user will have developed good intuition regarding how to set the value for the γ parameter.

We will assess now the overall performance of the two different variants for both problem formulations. As a suitable metric, we will use the reduction in the optimality gap achieved as a function of partitioning level. In order to avoid a bias due to the different scaling of the problems, the gap is normalized (denoted $\tilde{gap}$) with respect to its original value; that is, $\tilde{gap} = 100\%$ for $N = 1$ and $\tilde{gap} = 0\%$ if the global optimum has been attained. Regarding the normalized gap reduction (denoted $\tilde{gap}^R$) as a function of N, the formula of eq 58 applies.

$$\tilde{gap}^R(N) = 1 - \tilde{gap}(N) = \frac{LB(N) - LB(1)}{LB(\infty) - LB(1)} \qquad (58)$$

where LB(∞) corresponds to the global optimum, and LB(1), to the lower bound obtained by the standard, non-piecewise, relaxation scheme.

Figure 10 presents the normalized gap reduction evolution over N for both variants of the 'P'- and 'Q'-formulations. The curves "avg" correspond to averaging over all problems and values for γ used in this analysis. This provides us with general intuition regarding the level of tightness expected by each of the variants. In addition, the curves "best γ" are presented. These are also averaged curves over all the different problems, but they correspond only to contributions from the values of γ that each time provide the best result (curves that dominate in Figures 3−9, for given N). This would be a more appropriate metric, if one had taken into account the previous analysis and had restricted oneself to using the best possible gridding pattern. The following formulas apply.

$$\tilde{gap}^{R}_{avg}(N) = \frac{1}{|\Pi||\Gamma|}\sum_{\pi\in\Pi}\sum_{\gamma\in\Gamma}\tilde{gap}^{R}_{\pi\gamma}(N)$$
$$\tilde{gap}^{R}_{best\gamma}(N) = \frac{1}{|\Pi|}\sum_{\pi\in\Pi}\max_{\gamma\in\Gamma}\tilde{gap}^{R}_{\pi\gamma}(N) \quad (59)$$

where Π indexes over the benchmark problems, and set Π is the set of the 10 problems that appear in Figures 3−5; that is, Π = {Hav1−3, BT4, RT2, EPA1, Adh1−4}.

Examining the curves generated by the 'P'-formulation, we observe that the 'p'-variant better reduces the gap compared to their respective 'y'-variant counterparts, but the difference is not substantial. We conclude that when it comes to the 'P'-formulation of pooling problems, the 'p'-variant holds an advantage in terms of tightness; however, the selection of an appropriate value for the γ parameter is more important. Similarly for the 'Q'-formulation, the 'q'-variant curves are slightly elevated above the 'y'-variant, but the appropriate selection of γ is of more importance. Because the range of the p_{lk} variables can be substantially tightened using hard bounds (eq 8), the 'P'-formulation tends to reduce the gap more than the 'Q'-formulation, especially in the case of coarse partitioning.

In the remainder of this section, we will focus on comparing the performance of the different formulations, with regard to the computational effort that they require so as to obtain the lower bounds presented above. We should mention that a cutoff total CPU time of 1 h was used throughout this

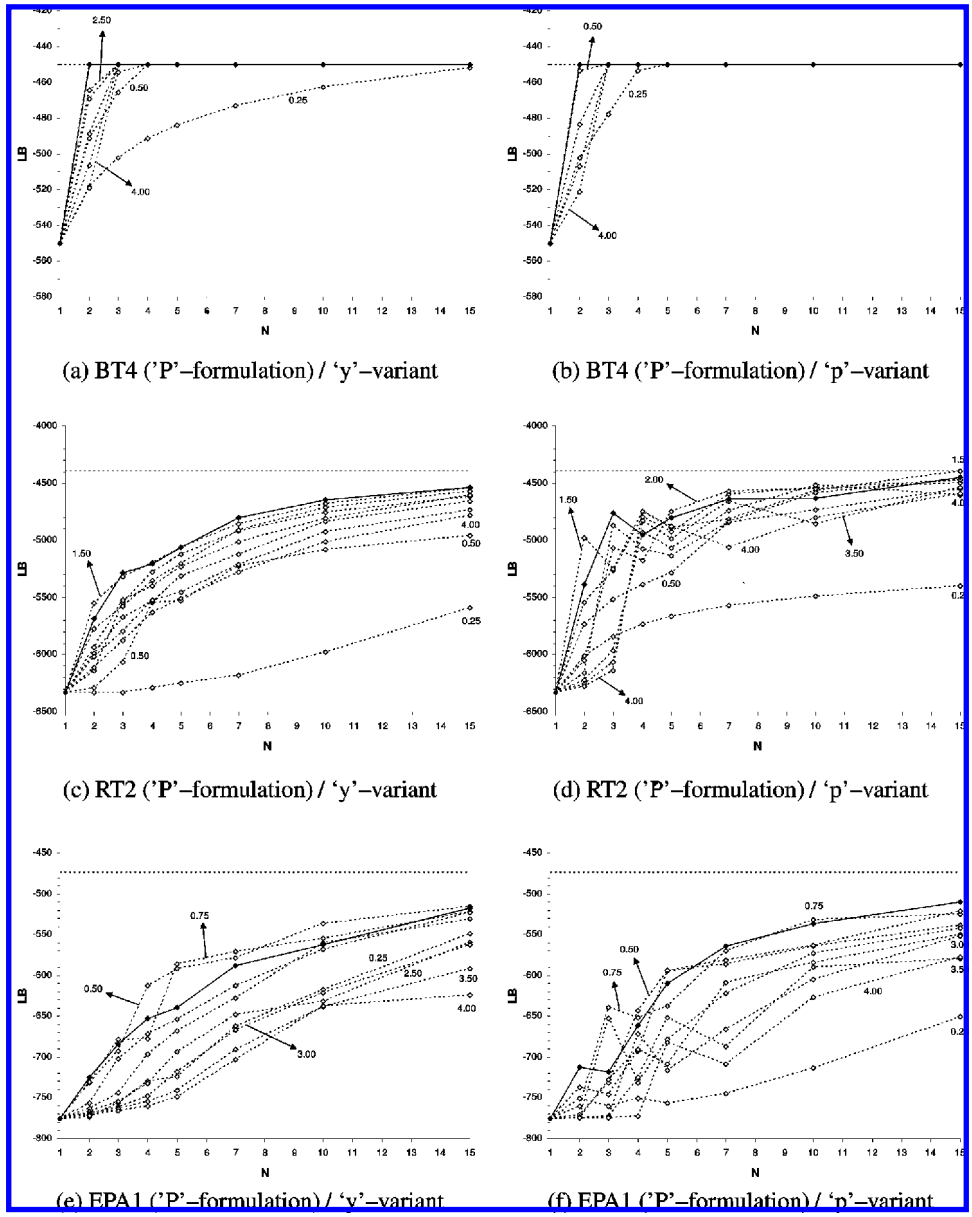

Figure 4. Lower bounds for problems BT4, RT2, and EPA1 as a function of partitioning level N and parameter γ.

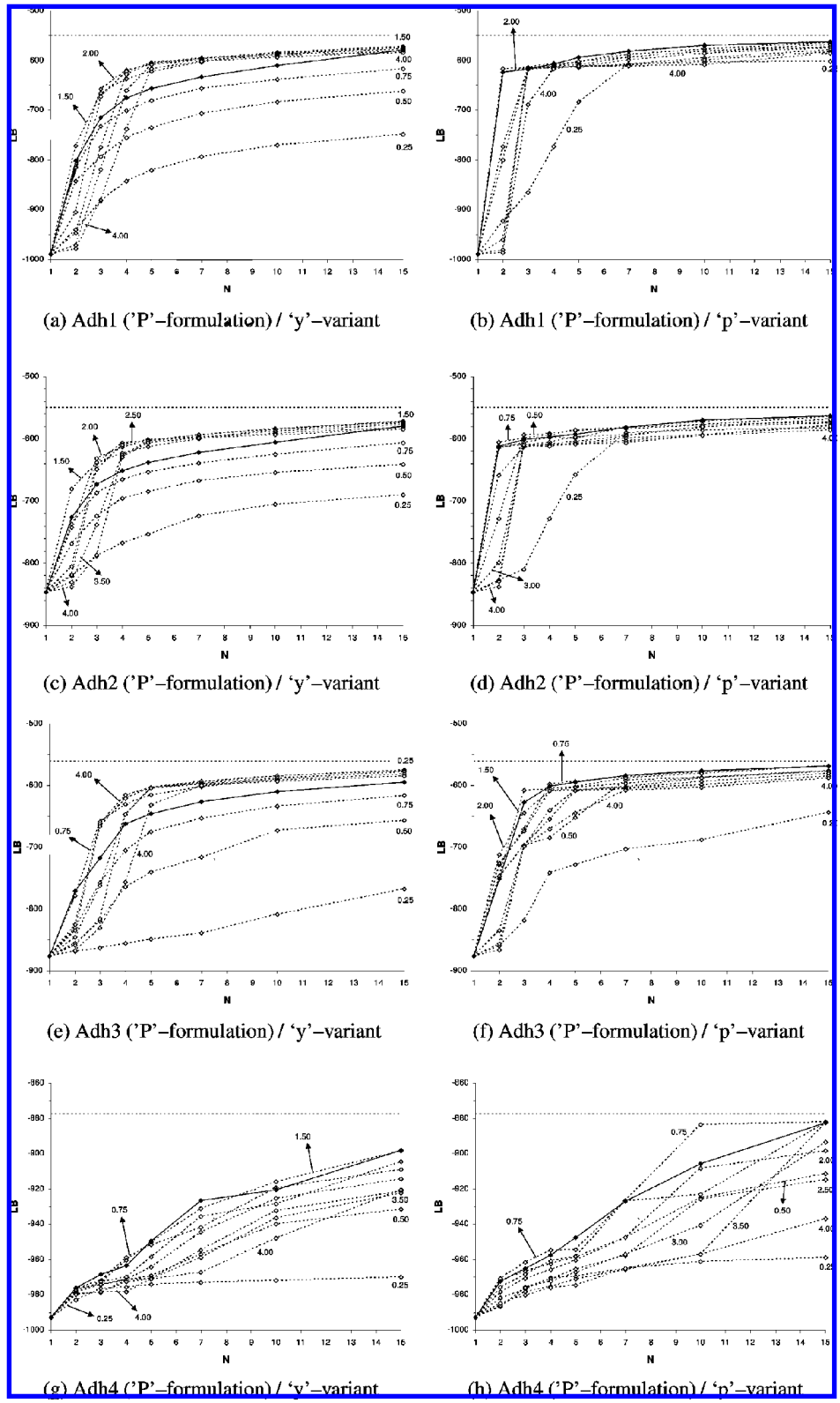

Figure 5. Lower bounds for the Adhya problems as a function of partitioning level N and parameter γ.

study. The runs that did not manage to reach the solution within this time limit were interrupted and deemed to be failed runs. Table 6 lists the number of runs that resulted into such a failure, tabulated with regard to formulation and partitioning level. As expected, more failures occur for large values of N. All the benchmark problems are represented in Table 6 except for the large Fou3−5 problems because finely partitioning (using large values of N) for those problems is prohibitive for in the 'Q'-formulation.

The failed runs generally originate from the bigger-size problems in our collection. Major contributors to failures in the 'P'-formulation are the 'y'-variant of Adh4, BT5, and Fou, the 'p'-variant of Adh2, and both variants of Adh3. In the 'Q'-formulation, major contributors to failures are the

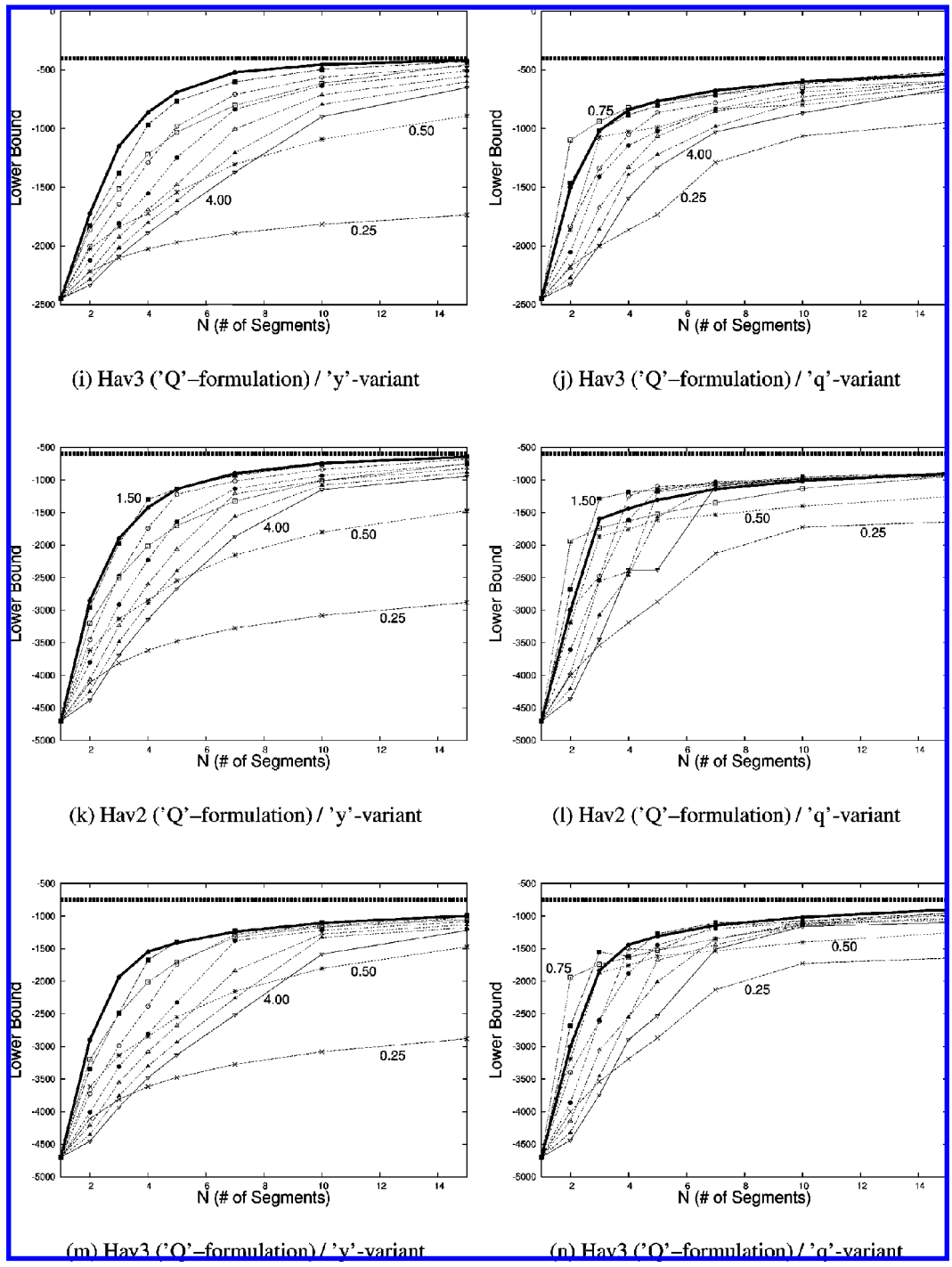

Figure 6. Lower bounds on the '*Q*'-formulation of the Haverly problems based on partitioning level N and parameter γ.

'*y*'-variant of Adh3, Adh4, and Fou2 and both variants of BT5. Referring back to Table 3, note that these test problems are the ones with the largest number of bilinear terms and, more specifically, the largest number of variables needing to be partitioned.

We observe that the Big-M formulations very often failed to obtain the solutions, and there were cases where this happened even for the modest partitioning level of $N = 4$. This can be attributed to the poor LP relaxations that these formulations typically produce. The convex combination formulations offer a significant improvement, with failures occurring only in the high-N regime. Finally, the best performance in terms of successful runs is achieved by the incremental cost formulations, where only a minimal number of failures occurs for $N = 15$.

Given the above observations, we will focus our attention only four promising MILP formulations: 'nf4l', 'nf4r', 'nf6t', and 'nf7r'. These are among the most efficient representatives of the two better-performing classes and belong to the set of formulations that we constructed as variants of those that were documented by Wicaksono and Karimi.[35]

For a given problem, variant, NLP formulation, MILP relaxation formulation, and partitioning level, we consider only the best-γ gridding pattern which requires the least amount of CPU time for the solution of the relaxation. The rationale behind this is based on the assumption that prior

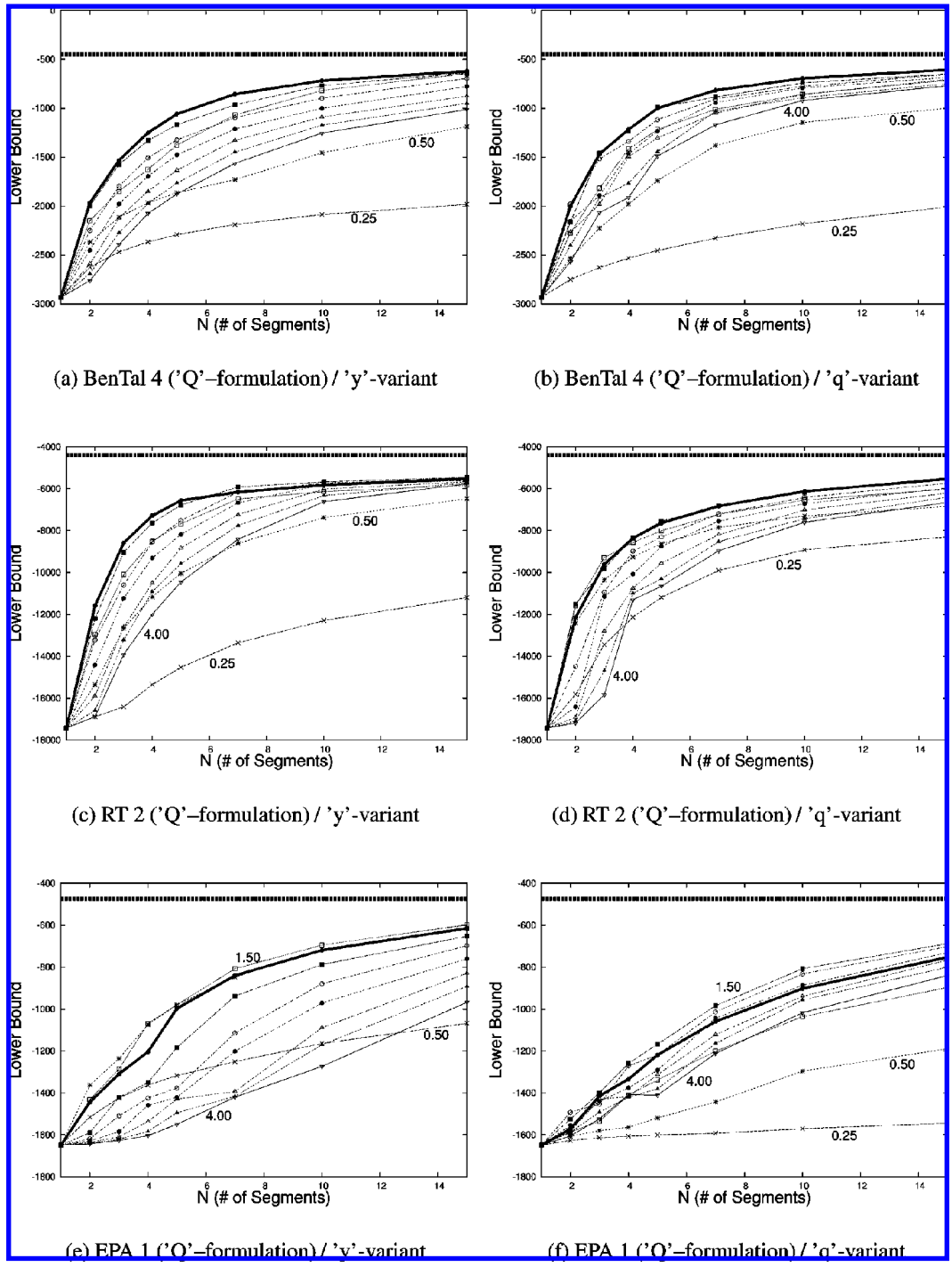

Figure 7. Lower bounds on the 'Q'-formulation of the BT4, RT2, and EPA1 problems based on partitioning level N and parameter γ.

knowledge of the optimal γ-value is available. Tables 7 and 8 present these CPU times for the most computationally intensive case, $N = 15$. Tables 9 and 10 present the required total LP iterations, and Tables 11 and 12 the required number of MILP nodes that correspond to these runs. Note that a value of 0 in either Table 10 or 11 corresponds to solution at the root node. The values in these six tables are not meant for comparison with the previously published results that solve pooling problem test cases.[18,19] Because the applications of these piecewise relaxations lie in large-scale problems, Tables 7−12 are best interpreted as comparisons of four good relaxation formulations. Within the context of a branch-and-bound algorithm solving a large-scale pooling problem, a more coarse partitioning level would be desirable, but the benefit of examining the $N = 15$ case within the context of this study is to emphasize the difference between the MILP relaxation formulations and thus generate an effective comparison.

We observe that the four MILP formulations efficiently solve a relaxation of the benchmark problems. Even the largest problem instances could be addressed in less that 1 min (for at least one of the two variants in each of the 'P'- and 'Q'-formulation). This compares favorably against other relaxation formulations that cannot achieve this within the time limit of 1 h. All four formulations considered seem to be practically equivalent.

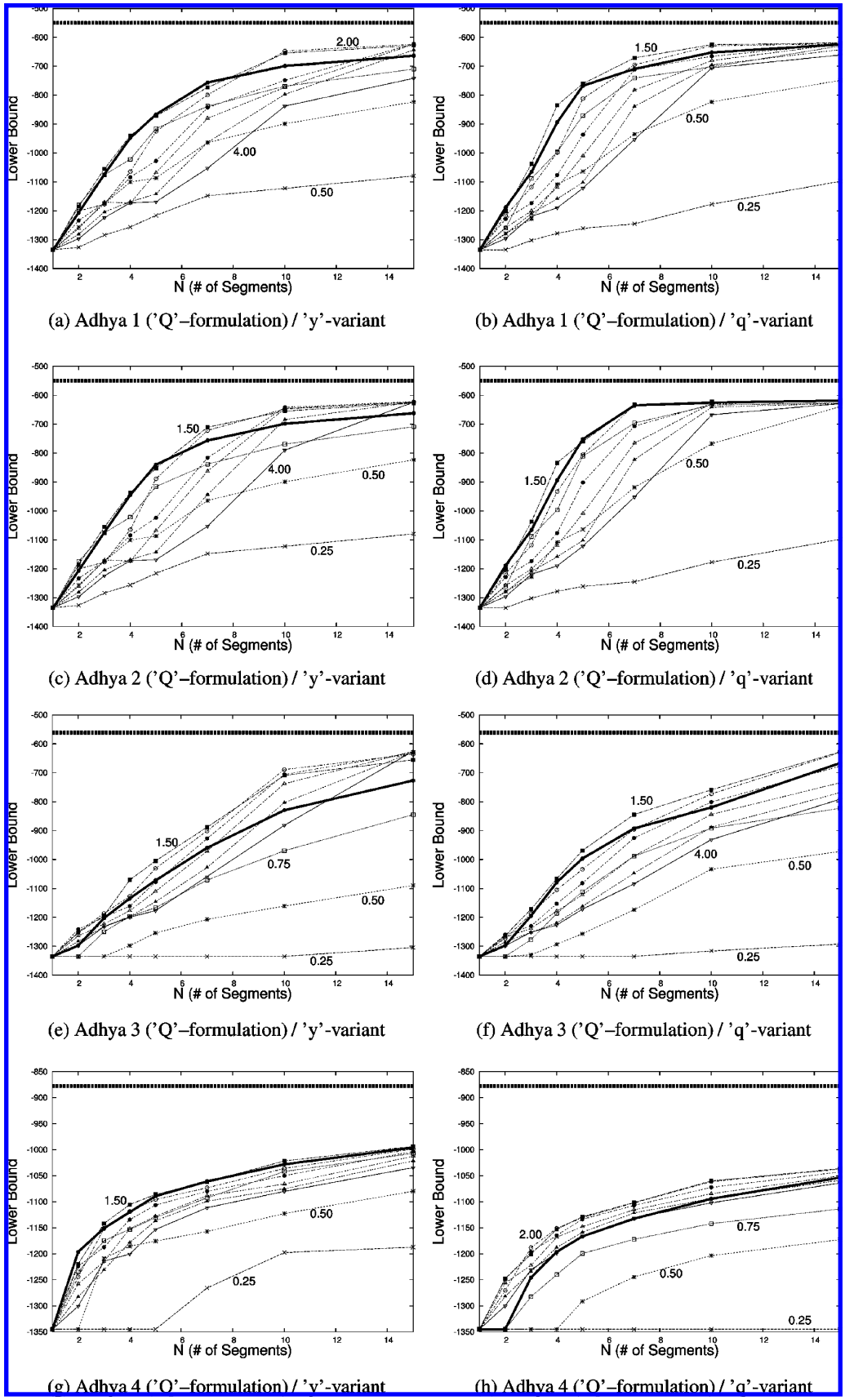

Figure 8. Lower bounds on the 'Q'-formulation of the Adhya problems based on partitioning level N and parameter γ.

As for analyzing which of the 'P'- or 'Q'-formulation is more desirable or which variant ('y', 'p', or 'q') is most advantageous, we observe that in some of the problems (e.g., Fou3), the 'p'-variant of the 'P'-formulation is clearly superior; in other examples (e.g., BT4) either variant of the 'P'-formulation is advantageous; and in others (e.g., Adh3) the 'Q'-formulation is best. *Although the mapping is not one-to-one, note as a general rule that the most efficient*

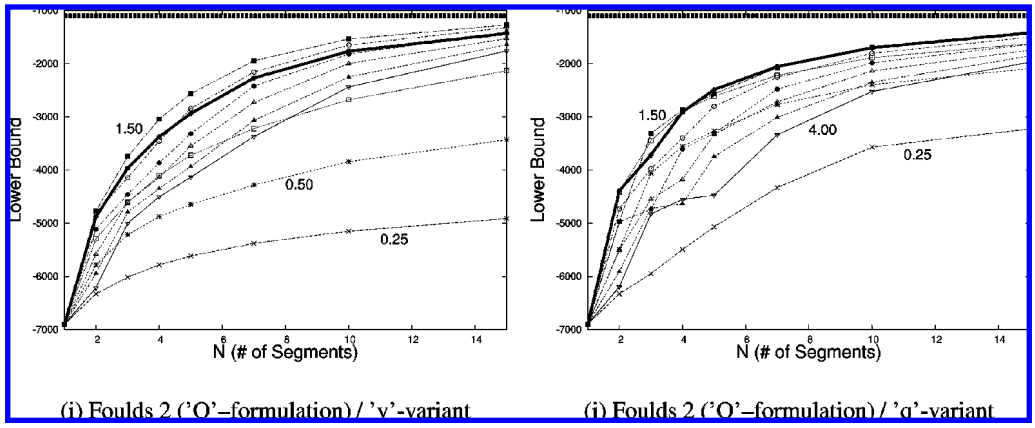

Figure 9. Lower bounds on the '*Q*'-formulation of the Foulds 2 problem based on partitioning level *N* and parameter γ.

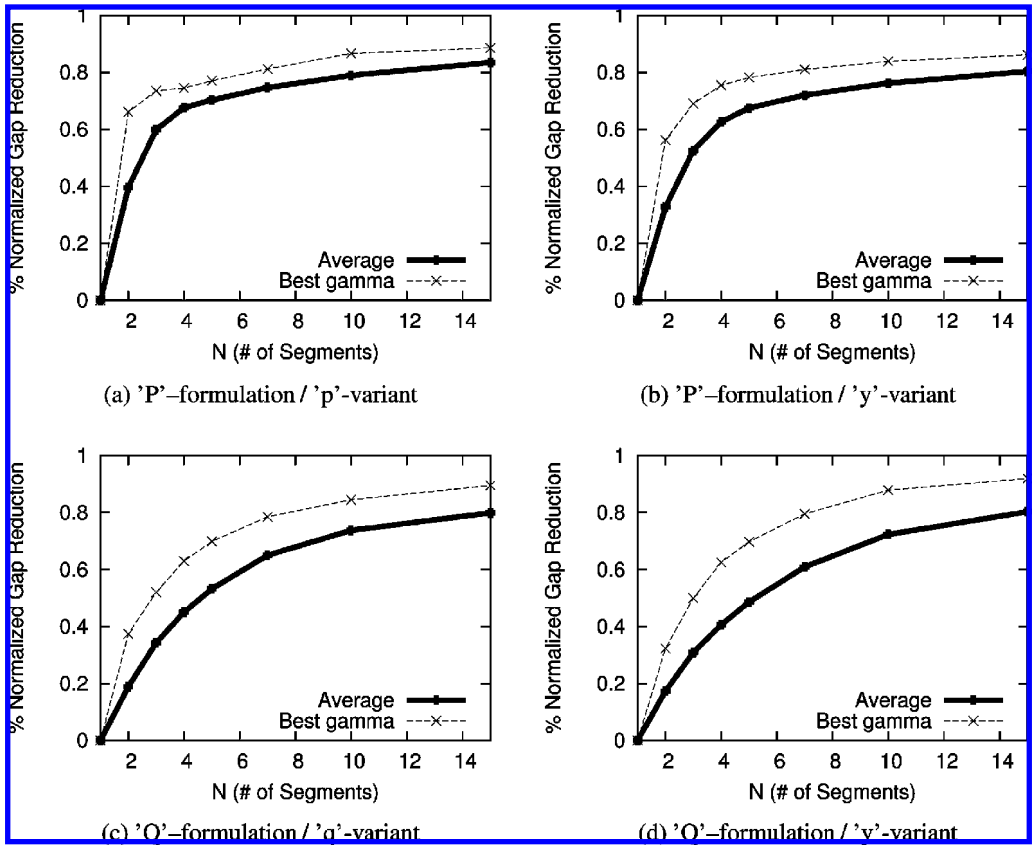

Figure 10. Normalized gap reductions as a function of partitioning level. All curves correspond to averages over problem set Π.

formulation is the one with fewer bilinear terms and the better-performing variant is the one with fewer variables to partition. Because the difference between the two variants involves examining pool qualities (the 'P'-formulation) or flows from the feed stocks (the 'Q'-formulation), it is usually best to choose the 'P'-formulation for problems involving few qualities relative to the number of feed stocks and the 'Q'-formulation for problems involving many qualities relative to the number of feed stocks.

As a final item of analysis, recall that there are many alternative ways to formulate the pooling problem.[17−19] Although this study focused on comparing 15 relaxation formulations, using the best possible problem formulation is another important step toward solving large-scale pooling problems. Figure 11 illustrates the importance of choosing the problem formulation well for the Adhya 3 test case. Although the '*Q*'-formulation of the Adh3 test case is smaller than the '*P*'-formulation, the optimality gap is closed much more quickly by the '*P*'-formulation because of the hard bounds imposed by the feed stock qualities. We can combine the advantages of a small problem size with tight relaxations by using the simplified '*Q*'-formulation of Audet et al.,[19] a formulation that eliminates one flow variable per pool, thus eliminating some of the bilinear terms and implicitly bounding the q_{il} variables. The '*PQ*'-formulation also tightens the relaxation relative to both the '*P*'- and '*Q*'-formulation without increasing the number of bilinear terms over the '*Q*'-formulation.[18]

For another problem, the Ben-Tal 4 test case, all the relaxations of all the formulations solved within our time limit of 1 h, but the relaxation optimality gaps were quite disparate. Figure 12 diagrams the relaxations resulting from the four problem formulations. The '*PQ*'- and '*P*'-formulations are clearly tighter than the other two. These results

Table 6. Number of Runs[a] That Failed to Reach Optimality within the Total CPU Time Limit of 1 h

		partitioning level N						
class	formulation	2	3	4	5	7	10	15
Big-M	bm			11	20	33	66	140
	nf1			11	22	31	63	124
	nf2g			11	21	33	59	114
	nf2			10	21	31	62	118
convex combination	ch			1	5	8	18	43
	tch			1	5	12	27	71
	nf3				4	7	15	53
	nf4l			1	5	7	11	28
	nf4				4	7	10	25
	nf4r			1	5	7	8	23
incremental cost	nf5				4	6	9	32
	nf6				3	6	8	20
	nf6t				3	6	9	20
	nf7				-	5	8	19
	nf7r				1	6	8	16

[a] Out of a total of 480 runs for each entry.

Table 7. Smallest CPU Times (s) Required for Solving the $N = 15$ 'P'-Formulation Instances

	formulations ('y'-variant)				formulations ('p'-variant)			
problem	nf4l	nf4r	nf6t	nf7r	nf4l	nf4r	nf6t	nf7r
Hav1	0.01	0.00	0.02	0.00	0.00	0.00	0.00	0.00
Hav2	0.01	0.01	0.04	0.00	0.01	0.00	0.00	0.00
Hav3	0.02	0.00	0.04	0.02	0.01	0.00	0.02	0.01
Fou2	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00
Fou3	13.65	2.64	0.72	0.22	0.67	0.52	0.32	0.16
Fou4	5.23	0.98	1.42	0.34	0.48	0.61	0.69	0.67
Fou5	1.61	0.46	3.66	0.14	0.14	0.17	0.25	0.23
BT4	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.00
BT5	0.05	0.07	0.25	0.17	0.30	0.68	0.92	0.23
Adh1	6.45	3.37	8.55	6.37	3.58	3.00	4.37	4.79
Adh2	8.80	5.16	13.66	10.21	8.41	7.21	8.34	10.91
Adh3	24.92	15.83	43.02	27.39	94.96	75.87	42.13	43.17
Adh4	20.53	15.38	25.85	11.07	3.95	3.35	8.91	8.07
RT2	1.73	1.37	1.26	1.21	1.03	0.20	0.66	0.45
EPA1	0.40	0.34	0.45	0.34	0.75	0.61	1.11	0.59

Table 8. Smallest CPU Times (s) Required for Solving the $N = 15$ 'Q'-Formulation Instances

	formulations ('y'-variant)				formulations ('q'-variant)			
problem	nf4l	nf4r	nf6t	nf7r	nf4l	nf4r	nf6t	nf7r
Hav1	0.04	0.04	0.08	0.05	0.14	0.05	0.14	0.08
Hav2	0.04	0.05	0.08	0.04	0.11	0.07	0.16	0.08
Hav3	0.04	0.04	0.08	0.05	0.14	0.05	0.11	0.08
Fou2	0.66	1.09	1.87	0.53	1.04	0.66	1.46	1.15
BT4	0.05	0.05	0.11	0.10	0.25	0.23	0.20	0.13
BT5	1.17	0.27	0.95	1.03	0.05	0.05	0.11	0.07
Adh1	3.52	2.91	3.47	2.25	0.80	0.42	1.12	0.91
Adh2	2.89	0.93	2.27	2.20	0.40	0.45	1.15	1.10
Adh3	0.38	0.50	1.19	0.65	3.54	2.64	1.95	1.56
Adh4	5.29	4.25	12.10	3.64	0.05	0.04	0.05	0.04
RT2	0.81	0.75	1.10	1.33	1.61	1.54	2.31	1.52
EPA1	0.22	0.17	0.23	0.22	0.25	0.19	0.27	0.17

indicate that it is important to consider problem formulation in designing tight relaxations that solve efficiently.

6. Conclusions

We discussed plausible relaxation schemes for the pooling problem, which is a very interesting optimization problem from both a theoretical and application point of view. The nonconvexities of this problem appear in the form of bilinear terms and can be addressed with the long-familiar technique by McCormick[38] that is based on the bilinear convex and concave envelopes.[39] In an effort to improve the tightness of this relaxation, and thus the efficiency of a global optimization algorithm, a piecewise scheme can be employed. In such a scheme, the original domain of one of the two variables in the bilinear product is partitioned into many subdomains and the principles of bilinear relaxation are applied for each one.

We derived 15 different piecewise relaxation schemes and compared them over a collection of 15 benchmark pooling problems. For each case, multiple problem formulations with two different variants can be envisioned, as one can opt to partition the domain of either the p_{lk} or the y_{lj} variable set in the 'P'-formulation or the q_{il} or the y_{lj} in the 'Q'-formulation. The study took into account 7 different partitioning levels and 10 different ways to select the location of the grid points, cumulatively accounting for a total of 56 700 relaxations.

Table 9. Number of Total LP Iterations Required for Solving the $N = 15$ '*P*'-Formulation Instances

	formulations ('*y*'-variant)				formulations ('*p*'-variant)			
problem	nf4l	nf4r	nf6t	nf7r	nf4l	nf4r	nf6t	nf7r
Hav1	58	39	180	110	33	23	75	67
Hav2	55	53	162	113	61	67	119	65
Hav3	205	91	252	109	109	111	185	94
Fou2	89	212	225	111	90	60	165	121
Fou3	5116	4014	2686	1096	1364	1296	1483	887
Fou4	1926	2240	4026	1313	1340	1332	1992	1303
Fou5	1406	984	2871	609	987	765	1146	1127
BT4	62	40	182	105	42	24	74	60
BT5	356	528	704	564	534	1474	1238	595
Adh1	35078	17280	18542	17039	20925	19419	12917	16627
Adh2	47499	25899	20127	26191	40616	36627	16006	33515
Adh3	99747	56818	64317	55631	278892	241168	55934	99056
Adh4	99715	74061	36286	28851	16207	15590	16212	20243
RT2	15859	13215	3810	5748	8552	1446	3068	1097
EPA1	1709	754	1326	1617	1396	1274	1486	727

Table 10. Number of Total LP Iterations Required for Solving the $N = 15$ '*Q*'-Formulation Instances

	formulations ('*y*'-variant)				formulations ('*p*'-variant)			
problem	nf4l	nf4r	nf6t	nf7r	nf4l	nf4r	nf6t	nf7r
Hav1	78	107	336	151	56	274	58	342
Hav2	19	127	19	120	83	333	46	277
Hav3	81	135	28	135	216	258	614	449
Fou2	1758	1694	2557	731	3863	2897	3814	2579
BT4	69	122	197	253	2405	145	181	131
BT5	1392	395	533	1044	275	76	246	102
Adh1	13846	11451	12736	9248	4917	3094	2978	1923
Adh2	13986	7190	8083	9687	1547	2060	2976	2410
Adh3	226	331	1889	682	15041	8410	4751	3285
Adh4	22094	21602	30231	11100	162	63	140	62
RT2	3294	2122	2168	3219	6244	5120	5693	6891
EPA1	792	814	835	763	61	69	58	57

Table 11. Number of MILP Nodes Required for Solving the $N = 15$ '*P*'-Formulation Instances

	formulations ('*y*'-variant)				formulations ('*p*'-variant)			
Problem	nf4l	nf4r	nf6t	nf7r	nf4l	nf4r	nf6t	nf7r
Hav1	0	0	0	0	0	0	0	0
Hav2	0	0	1	0	2	2	0	0
Hav3	14	2	12	1	0	0	2	2
Fou2	0	3	0	0	0	0	0	0
Fou3	0	0	0	0	0	0	0	0
Fou4	0	0	0	0	0	0	0	0
Fou5	0	0	50	0	0	0	0	0
BT4	0	0	0	0	0	0	0	0
BT5	0	0	0	0	0	10	10	0
Adh1	490	332	170	272	547	542	338	270
Adh2	490	321	200	248	1830	1807	484	603
Adh3	665	454	261	259	6636	5544	928	1260
Adh4	2084	1573	933	598	257	330	261	269
RT2	355	296	73	137	162	49	54	14
EPA1	36	24	18	21	55	46	60	21

Table 12. Number of MILP Nodes Required for Solving the $N = 15$ '*Q*'-Formulation Instances

	formulations ('*y*'-variant)				formulations ('*p*'-variant)			
problem	nf4l	nf4r	nf6t	nf7r	nf4l	nf4r	nf6t	nf7r
Hav1	0	0	24	0	6	22	2	42
Hav2	0	15	0	0	6	25	1	16
Hav3	10	16	0	0	21	19	31	32
Fou2	72	63	119	29	354	171	132	117
BT4	4	4	8	9	174	0	0	0
BT5	98	0	22	29	0	0	0	0
Adh1	740	714	368	293	330	88	81	42
Adh2	644	523	354	364	46	98	64	61
Adh3	16	55	80	14	694	607	144	87
Adh4	793	723	904	276	0	0	0	0
RT2	93	94	92	83	349	340	251	293
EPA1	25	28	25	29	0	1	0	0

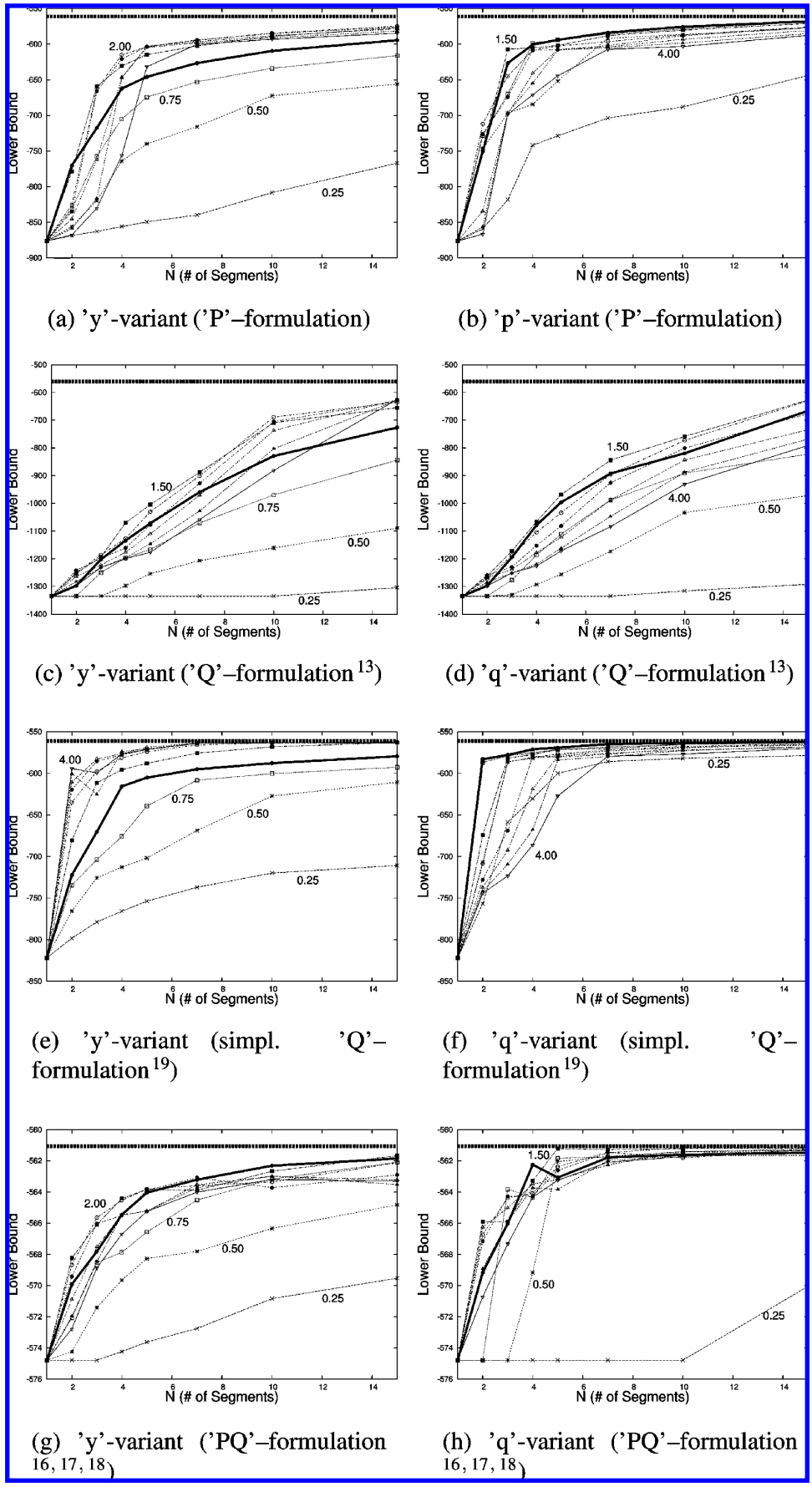

Figure 11. Comparison of four problem formulations using the Adhya 3 test case.

The computational studies demonstrated that piecewise relaxation schemes can considerably improve the results in terms of lower bounding. As expected, this benefit comes at the expense of computational effort and one should be cautious with

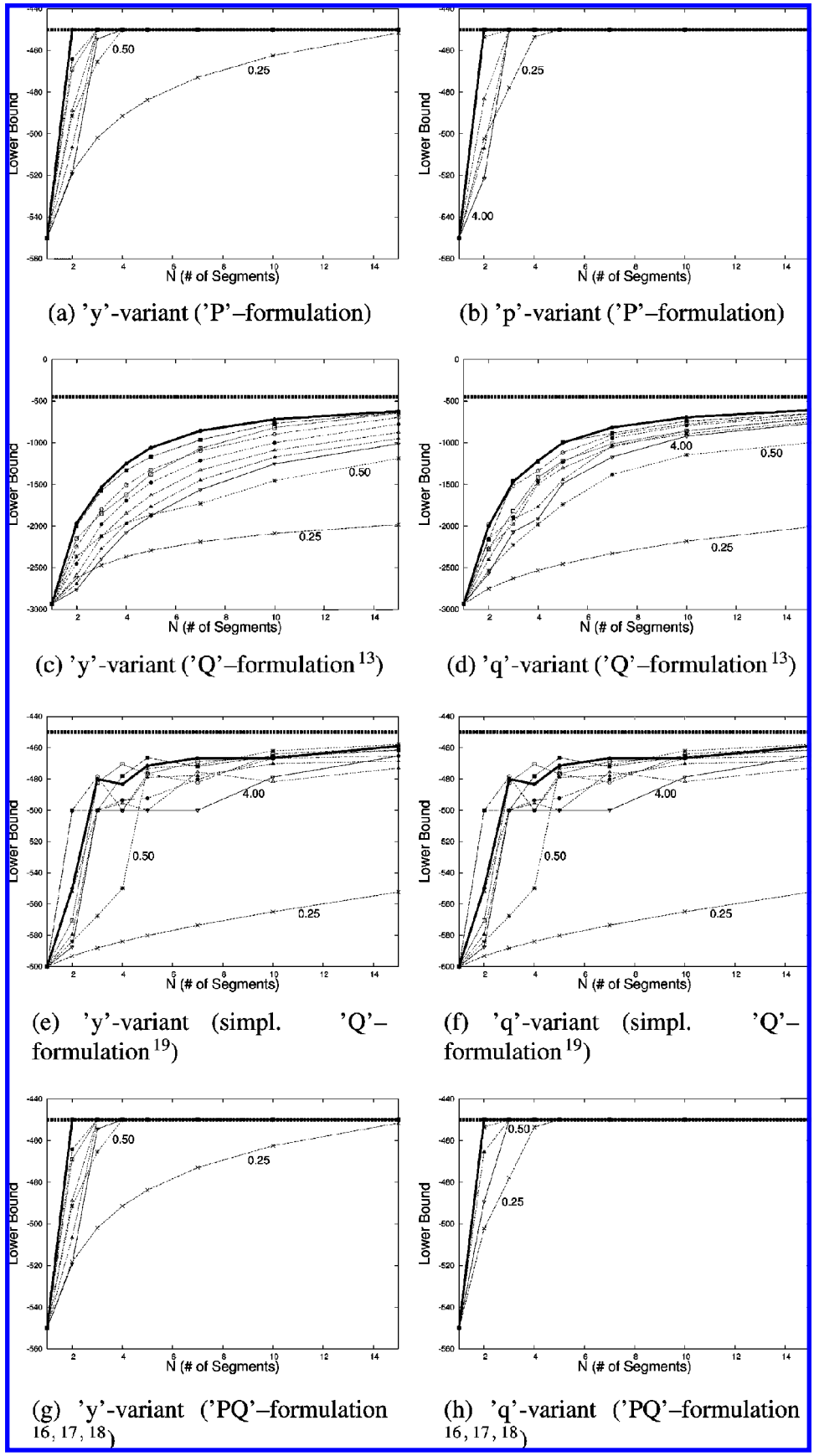

Figure 12. Comparison of four problem formulations using the Ben-Tal 4 test case.

the extent of the employed schemes. The novel formulations we introduced—'nf4l', 'nf4r', 'nf6t', and 'nf7r'—turned out to be reliable and efficient in reaching the solution. All of them were able to address every benchmark problem in a few seconds.

Acknowledgment

The authors gratefully acknowledge support from the National Science Foundation. R.M. is further thankful for her National Science Foundation Graduate Research Fellowship.

Literature Cited

(1) Haverly, C. A. Studies of the Behaviour of Recursion for the Pooling Problem. *ACM SIGMAP Bull.* **1978**, *25*, 19–32.

(2) Lasdon, L. S.; Waren, A. D.; Sarkar, S.; Palacios-Gomez, F. Solving the Pooling Problem using Generalized Reduced Gradient and Successive Linear Programming. *ACM SIGMAP Bull.* **1979**, *27*, 9–15.

(3) Palacios-Gomez, F.; Lasdon, L. S.; Engquist, M. Nonlinear Optimization by Successive Linear Programming. *Manage. Sci.* **1982**, *28*, 1106–1120.

(4) Ahang, J.; Kim, N.; Lasdon, L. S. An Improved Successive Linear Programming Algorithm. *Manage. Sci.* **1985**, *31*, 1312–1331.

(5) Baker, T. E.; Lasdon, L. S. Successive Linear Programming at Exxon. *Manage. Sci.* **1985**, *31*, 264–274.

(6) Floudas, C. A.; Aggarwal, A. A Decomposition Strategy for Global Optimum Search in the Pooling Problem. *ORSA J. Comput.* **1990**, *2*, 225–235.

(7) Benders, J. F. Partitioning Procedures for Solving Mixed Variables Programming Problems. *Numer. Math.* **1962**, *4*, 238.

(8) Geoffrion, A. M. Generalized Benders Decomposition. *J. Optim. Theory Appl.* **1972**, *10*, 237–260.

(9) Floudas, C. A.; Visweswaran, V. A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs. I. Theory. *Comput. Chem. Eng.* **1990**, *14*, 1397–1417.

(10) Visweswaran, V.; Floudas, C. A. A Global Optimization Algorithm (GOP) for Certain Classes of Nonconvex NLPs. II. Application of Theory and Test Problems. *Comput. Chem. Eng.* **1990**, *14*, 1419–1434.

(11) Floudas, C. A.; Visweswaran, V. Primal-relaxed Dual Global Optimization Approach. *J. Optim. Theory Appl.* **1993**, *78*, 187–225.

(12) Visweswaran, V.; Floudas, C. A. New Properties and Computational Improvement of the GOP Algorithm for Problems with Quadratic Objective Functions and Constraints. *J. Global Optim.* **1993**, *3*, 439–462.

(13) Ben-Tal, A.; Eiger, G.; Gershovitz, V. Global Minimization by Reducing the Duality Gap. *Math. Prog.* **1994**, *63*, 193–212.

(14) Adhya, N.; Tawarmalani, M.; Sahinidis, N. V. A Lagrangian Approach to the Pooling Problem. *Ind. Eng. Chem. Res.* **1999**, *38*, 1956–1972.

(15) Almutairi, H.; Elhedhli, S. A New Lagrangean Approach to the Pooling Problem *J. Global Optim.*, in press.

(16) Sherali, H. D.; Adams, W. P. *A Reformulation-Linearization Technique for solving Discrete and Coninuous Nonconvex Problems, Nonconvex Optimization and its Applications*; Kluwer Academic Publishers: Dordrecht, Netherlands, 1999.

(17) Quesada, I.; Grossmann, I. E. Global Optimization of Bilinear Process Networks and Multicomponent Flows. *Comput. Chem. Eng.* **1995**, *19*, 1219–1242.

(18) Tawarmalani, M.; Sahinidis, N. V. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications, Nonconvex Optimization and its Applications*; Kluwer Academic Publishers: Norwell, MA, 2002.

(19) Audet, C.; Brimberg, J.; Hansen, P.; Le Digabel, S.; Mladenovic, N. Pooling Problem: Alternate Formulations and Solution Methods. *Manage. Sci.* **2004**, *50*, 761–776.

(20) Audet, C.; Hansen, P.; Jaumard, B.; Savard, G. A Branch and Cut Algorithm for Nonconvex Quadratically Constrained Quadratic Programming. *Math. Prog.* **2000**, *87*, 131–152.

(21) Meyer, C. A.; Floudas, C. A. Global Optimization of a Combinatorially Complex Generalized Pooling Problem. *AIChE J.* **2006**, *52*, 1027–1037.

(22) Lodwick, W. A. Preprocessing Nonlinear Functional Constraints with Applications to the Pooling Problem. *ORSA J. Comput.* **1992**, *4*, 119–131.

(23) Greenberg, H. J. Analyzing the Pooling Problem. *ORSA J. Comput.* **1995**, *7*, 205–217.

(24) Foulds, L. R.; Haugland, D.; Jörnsten, K. A Bilinear Approach to the Pooling Problem. *Optimization* **1992**, *24*, 165–180.

(25) Floudas, C. A.; Akrotirianakis, I. G.; Caratzoulas, S.; Meyer, C. A.; Kallrath, J. Global optimization in the 21st century: Advances and challenges. *Comput. Chem. Eng.* **2005**, *29*, 1185–1202.

(26) Floudas, C. A.; Gounaris, C. E. A Review of recent advances in global optimization. *J. Global Optim.*, in press.

(27) Floudas, C. A. *Deterministic Global Optimization: Theory, Methods and Applications, Nonconvex Optimization and Its Applications*; Kluwer Academic Publishers: Dordrecht, Netherlands, 2000.

(28) Floudas, C. A., Pardalos, P. M., Eds. *Recent Advances In Global Optimization*; Princeton University Press: Princeton, NJ, 1992.

(29) Floudas, C. A.; Pardalos, P. M. State of the Art In Global Optimization: Computational Methods and Applications - Preface. *J. Global Optim.* **1995**, *7*, 113–113.

(30) Floudas, C. A., Pardalos, P. M., Eds. *State of the Art In Global Optimization: Computational Methods and Applications*; Kluwer Academic Publishers: Dordrecht, Netherlands, 1996.

(31) Floudas, C. A., Pardalos, P. M., Eds. *Optimization in Computational Chemistry and Molecular Biology: Local and Global Approaches, Nonconvex Optimization and Its Applications*; Kluwer Academic Publishers: Dordrecht, Netherlands, 2000.

(32) Floudas, C. A., Pardalos, P. M., Eds. *Frontiers in Global Optimization, Nonconvex Optimization and Its Applications*; Kluwer Academic Publishers: Dordrecht, Netherlands, 2004.

(33) Floudas, C. A.; Pardalos, P. M. *J. Global Optim.* **2009**, *43* (2).

(34) Karuppiah, R.; Grossmann, I. E. Global Optimization for the Synthesis of Integrated Water Systems in Chemical Processes. *Comput. Chem. Eng.* **2006**, *30*, 650–673.

(35) Wicaksono, D. S.; Karimi, I. A. Piecewise MILP Under- and Overestimators for Global Optimization of Bilinear Programs. *AIChE J.* **2008**, *54*, 991–1008.

(36) Pham, V.; Laird, C.; El-Halwagi, M. Convex Hull Discretization Approach to the Global Optimization of Pooling Problems. *Ind. Eng. Chem. Res.* **2009**, *48*, 1973–1979.

(37) Gounaris, C. E.; Floudas, C. A. Convexity of Products of Univariate Functions and Convexification Transformations for Geometric Programming. *J. Optim. Theory Appl.* **2008**, *138*, 407–427.

(38) McCormick, G. P. Computability of Global Solutions to Factorable Nonconvex Programs: Part I - Convex Underestimating Problems. *Math. Prog.* **1976**, *10*, 147–175.

(39) Al-Khayyal, F. A.; Falk, J. E. Jointly Constrained Biconvex Programming. *Math. Oper. Res.* **1983**, *8*, 273–286.

(40) Rikun, A. D. A Convex Envelope Formula for Multilinear Functions. *J. Global Optim.* **1997**, *10*, 425–437.

(41) Androulakis, I. P.; Maranas, C. D.; Floudas, C. A. α BB: A Global Optimization Method for General Constrained Nonconvex Problems. *J. Global Optim.* **1995**, *7*, 337–363.

(42) Balas, E. Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems. *SIAM J. Algebraic Discret. Methods* **1985**, *6*, 466–486.

(43) Gounaris, C. E.; Floudas, C. A. Formulation and Relaxation of an Extended Pooling Problem. Advances in Optimization II. *AIChE Annual Meeting*, Salt Lake City, UT, November 4–9, 2007.

(44) Brooke, A.; Kendrick, D.; Meeraus, A.; Raman, R. *GAMS: A Users Guide*; GAMS Development Corporation: Washington, DC, 2005.

(45) *CPLEX 9.0 User's Manual*; ILOG, Inc.: Mountain View, CA, 2005.

Received for review October 22, 2008
Revised manuscript received March 30, 2009
Accepted April 7, 2009

IE8016048