APOGEE: Global Optimization of Standard, Generalized, and Extended Pooling Problems via Linear and Logarithmic Partitioning Schemes

Ruth Misener, Jeffrey P. Thompson, and Christodoulos A. Floudas*

Department of Chemical and Biological Engineering
Princeton University
Princeton, NJ 08544-5263

December 21, 2010

Abstract

Our recent work globally optimized two classes of large-scale pooling problems: (i) a generalized pooling problem that treats the network topology as a decision variable [Misener and Floudas, 2010] and (ii) an extended pooling problem that incorporates the United States Environmental Protection Agency (EPA) *Title 40 Code of Federal Regulations Part 80.45: Complex Emissions Model* into the constraint set [Misener et al., 2010]. The large-scale pooling problems were optimized using a piecewise scheme that activates appropriate under- and overestimators with a number of binary decision variables that scales *linearly* with the number of segments in the piecewise relaxation. In this paper, inspired by recent work proposing models with a *logarithmic* number of binary variables for piecewise linear continuous functions, piecewise linear functions, and lower semicontinuous functions [Vielma and Nemhauser, 2010, Vielma et al., 2010a], we propose a novel formulation for the piecewise linear relaxation of bilinear functions with a number of binary variables that depends logarithmically on the number of binary variables and computationally compare the performance of this new formulation to the best-performing piecewise relaxations with a linear number of binary variables. We have also unified our work by developing APOGEE (Algorithms for Pooling-problem global Optimization in GEneral and Extended classes), a computational tool that globally optimizes standard, generalized, and extended pooling problems. APOGEE is freely available to the scientific community at `helios.princeton.edu/APOGEE/`.

Keywords: large-scale optimization; global optimization; MINLP; quadratically-constrained quadratic programs; pooling problem; EPA Complex Emissions Model

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

1 Introduction

The pooling problem, an optimization challenge of maximizing profit subject to feedstock availability, intermediate storage capacity, demand, and product specification constraints, has important practical applications to many process systems engineering domains, including petroleum refining, water systems, supply-chain operations, and communications [Bagajewicz, 2000, Jeżowski, 2010, Misener and Floudas, 2009, Visweswaran, 2009]. In petroleum refining, for example, when a variety of feedstocks emerging from distillation units, reformers, and catalytic crackers are combined under limited storage conditions into a plethora of final products with different specifications such as gasoline, diesel fuel, aviation jet fuel, and fuel oil, the global optimum of the pooling problem reports the most profitable hydrocarbon flowrates between the process units. Adding further complexity for petroleum refiners, the Environmental Protection Agency (EPA) has mandated the reduction of compounds in reformulated gasoline that generate airborne toxic emissions, smog-forming volatile organic compounds, and nitrous oxides (NO_X) [40CFR80.41, 2008, 40CFR80.45, 2007, Furman and Androulakis, 2008, Misener et al., 2010]. Environmental standards, coupled with limited availability of low-sulfur crude and automobiles requiring high octane fuels, necessitate global optimization to meet demand with maximum profitability. Although we assume *command-and-control* policy on environmental pollutants and place *profitability* in the objective function, the mathematical complexity of the pooling problem does not change between the regulatory frameworks described by Malcolm et al. [2006] and could be transformed to explicitly address environmental objectives as reviewed by Grossmann and Guillén-Gosálbez [2010].

The theoretical significance of the pooling problem stems from the nonconvex bilinear terms that represent stream mixing in intermediate storage nodes. These bilinear terms give rise to a multiplicity of local optima which prevent linear, convex, and stochastic solvers from certifying global optimality of the quadratically-constrained quadratic program (QCQP) that mathematically represents the standard pooling problem [Floudas, 2000, Floudas and Gounaris, 2009, Floudas and Pardalos, 1995, Floudas et al., 2005, Misener and Floudas, 2009]. Adding further complexity to industrially-relevant pooling problems are the binary decisions associated with the activation or deactivation of specific process units and/or connecting pipelines and the variety of piecewise nonconvex equations needed to mathematically model complex product qualities such as those in the EPA definition of polluting emissions [Audet et al., 2004, Meyer and Floudas, 2006, Misener and Floudas, 2010, Misener et al., 2010].

Our recent work globally optimized two classes of pooling problems: (i) a generalized pooling problem that treats the network topology as a decision variable [Misener and Floudas, 2010] and (ii) an extended pooling problem that incorporates the Environmental Protection Agency (EPA) *Title 40 Code of Federal Regulations Part 80.45: Complex Emissions Model* [40CFR80.45, 2007] into the constraint set [Misener et al., 2010]. After separately studying these two pooling problem instantiations, we have unified our work by developing APOGEE (Algorithms for Pooling-problem global Optimization in GEneral and Extended classes), a computational tool that globally optimizes standard, generalized, and extended pooling problems. APOGEE is freely available to the scientific community at `helios.princeton.edu/APOGEE/`.

To globally optimize the large-scale generalized and extended pooling problems in Misener and Floudas [2010] and Misener et al. [2010], we implemented a branch-and-bound algorithm that exploited advances in the piecewise-linear relaxation of bilinear terms. These piecewise relaxations, which were introduced by Meyer and Floudas [2006] and Karuppiah and Grossmann [2006] and extensively tested by Wicaksono and Karimi [2008], Gounaris et al. [2009], and Hasan and Karimi [2010], activate appropriate under- and overestimators using a number of binary decision variables that scales *linearly* with the number of segments in the piecewise relaxation. Vielma and Nemhauser [2010] and Vielma et al. [2010a] recently proposed modeling piecewise linear continuous functions, piecewise linear functions, and lower semicontinuous functions with a number of binary switches that scales *logarithmically* with the number of disjunctive segments. Motivated by their work, we propose a novel formulation for the piecewise relaxation of bilinear programs that features a logarithmic number of binary variables and computationally compare the performance of this new formulation to the best-performing piecewise relaxations with a linear number of binary variables [Gounaris et al., 2009, Hasan and Karimi, 2010, Wicaksono and Karimi, 2008].

In Section 2 outlines the three classes of pooling problems, reviews recent developments in solving nonconvex mixed integer nonlinear programs (MINLP) relevant to the pooling problem, and discusses the potential of modeling the piecewise-linear relaxation of QCQPs using a logarithmic number of binary variables [Vielma and Nemhauser, 2010, Vielma et al., 2010a]. Section 3 describes the mathematical formulation of the pooling problem. Section 4 introduces the novel formulation for the piecewise relaxation of QCQPs, and Section 5 computationally tests the new formulation. Section 6 describes how we have integrated our computational studies of piecewise-linear and edge-concave underestimators with our experience modeling the standard, generalized, and extended pooling problems by developing the global optimization solver APOGEE. We describe the specifics of the algorithms underlying APOGEE, demonstrate its performance on all three classes of pooling problems, and encourage readers to test its performance.

2 Literature Review

In the **standard pooling problem**, introduced by Haverly [1978], flow rates are chosen to maximize profit (or minimize cost) on a tripartite, pre-determined network structure of input, intermediate, and output nodes that represent feedstocks, storage tanks (or *pools*), and final products, respectively [Misener and Floudas, 2009]. Nonconvex bilinear terms in the standard pooling problem arise from tracking the levels of linearly-blending fuel qualities about the pooling nodes to meet constraints on the composition of the final products [Visweswaran, 2009]. Among the many notable contributions towards solving the standard pooling problem [Adhya et al., 1999, Almutairi and Elhedhli, 2009, Audet et al., 2004, Ben-Tal et al., 1994, Chakraborty, 2009, Floudas and Aggarwal, 1990, Floudas and Visweswaran, 1990, 1993, Floudas et al., 1989, Quesada and Grossmann, 1995, Foulds et al., 1992, Greenberg, 1995, Haverly, 1978, Lasdon et al., 1979, Lodwick, 1992, Pham et al., 2009, Tawarmalani and Sahinidis, 2002, Visweswaran and Floudas, 1990, 1993], the most directly relevant to the work presented in this paper and the computational tool APOGEE are those of:

Table 1: Comparison of standard, generalized, and extended problem classes [Misener and Floudas, 2009]

Problem Class	Allowed connections				EPA Complex Emissions Model	Optimally Selected Topology	Pools can Represent Treatment Plants*
	Input–pool	Input–output	Pool–output	Pool–Pool†			
Standard	✓	✓	✓				
Generalized	✓	✓	✓	✓		✓	✓
Extended	✓	✓	✓		✓		

†While pool–pool connections are allowed in the generalized problem, the optimization algorithm excludes feasible solutions in which cycles exist between pools.

*See Meyer and Floudas [2006].

Floudas and Visweswaran [Floudas and Visweswaran, 1990, 1993, Visweswaran and Floudas, 1990, 1993], who were the first to rigorously solve the pooling problem to global optimality; Foulds et al. [1992], who developed a linear relaxation of the QCQP by replacing each bilinear term with their convex and concave hulls [Al-Khayyal and Falk, 1983, McCormick, 1976]; Ben-Tal et al. [1994], who introduced an alternative *q*-formulation of the pooling problem that often has fewer nonconvex bilinear terms than the original *p*-formulation [Audet et al., 2004]; and Tawarmalani and Sahinidis [2002], who showed that augmenting the *q*-formulation with reformulation-linearization technique cuts [Sherali and Adams, 1999, Sherali and Alameddine, 1992] proposed by Quesada and Grossmann [1995] produces a linear relaxation of the pooling problem that strictly dominates both the *p*- and *q*-formulations. Depending on the formulation, the standard pooling problem can be classified as a linear objective with quadratic constraints (*p*-formulation) or a quadratic objective with quadratic constraints (*q*- and *pq*-formulations).

The **generalized pooling problem** increases the complexity of the pooling problem by allowing flow between the intermediate storage nodes and transforming the network topology from a pre-determined structure into an optimally-chosen configuration [Audet et al., 2004, Misener and Floudas, 2009, Meyer and Floudas, 2006]. Choosing the interconnections between source, intermediate, and output nodes is combinatorially complex. Because the activation or deactivation of each inter-node connection is a discrete decision and the linear mixing at the intermediate nodes leads to bilinear terms, the generalized pooling problem is a mixed-integer nonconvex program (nonconvex MINLP) with quadratic equalities and inequalities that exhibits multiple locally optimal solutions. Research addressing the generalized pooling problem includes that of Lee and Grossmann [2003], who used their global optimization algorithm for generalized disjunctive programs (GDP) to determine the activated feedstocks and pools for several examples; Audet et al. [2004], who solved case studies with pool-to-pool connections using a branch-and-cut algorithm [Audet et al., 2000]; Meyer and Floudas [2006], who introduced a piecewise-linear underestimation scheme that generates a valid lower bound on industrially-relevant problems that include both inter-pool connections and discrete topological decisions; Ruiz and Grossmann [2010], who used their linear relaxation strategies within a global optimization algorithm to solve one of the test cases of Lee and Grossmann [2003]; Misener and Floudas [2010], who revisited the generalized pooling problem test cases of Meyer and Floudas [2006] and exploited recent advances in piecewise-linear underestimation of bilinear terms [Floudas and Gounaris, 2009, Wicaksono and Karimi, 2008] within a branch-and-bound algorithm to globally optimize large-scale problems; and Li et al. [2010] who considered multiple scenarios to account for quality parameters that are realized after the design stage.

The mathematical program representing the generalized pooling problem is nearly identical to the model for the optimal synthesis of integrated water systems as proposed by Galan and Grossmann [1998], so the algorithms designed to globally optimize the water systems case studies [Bergamini et al., 2008, Karuppiah and Grossmann, 2006, Lee and Grossmann, 2003] are highly relevant to the generalized pooling problem and therefore aided our design of the solver APOGEE. The generalized pooling problem can also be recast as a combinatorial process synthesis problem that addresses both economic performance and environmental impact [Chakraborty and Linninger, 2002, 2003, Chakraborty et al., 2003]. For example, Baliban et al. [2010b] consider superstructure optimization of their Hybrid Biomass, Coal, and Natural Gas Facility [Baliban et al., 2010a, Elia et al., 2010] using a model that is closely linked to the generalized pooling problem.

The **extended pooling problem** maximizes profit on a standard pooling problem network while complying with constraints on nonlinearly blending fuel qualities such as those in the Environmental Protection Agency (EPA) *Title 40 Code of Federal Regulations Part 80.45: Complex Emissions Model* [40CFR80.45, 2007], a mathematical model that codifies reformulated gasoline (RFG) emissions using a function of eleven fuel qualities [Misener et al., 2010]. The RFG program, which impacts roughly 75 million people, reduces smog and airborne toxic pollutants (*e.g.*, benzene, a human carcinogen) in accordance with the Clean Air Act. Final RFG products must comply with standards, or upper bounds, on volatile organic, NO_X, and airborne toxic emissions. To solve the EPA extended pooling problem to global optimality, we formulated a mixed-integer nonlinear program (MINLP) representing the EPA Complex Emissions Model and appended this MINLP to a standard pooling problem network. The MINLP formulation contains nonconvex constraints that include bilinear, multilinear, exponential, and power law terms which bear many similarities to the model of Furman and Androulakis [2008]. We developed a mixed-integer linear relaxation of the MINLP using piecewise-linear [Gounaris et al., 2009, Karuppiah and Grossmann, 2006, Meyer and Floudas, 2006, Wicaksono and Karimi, 2008], edge-concave [Meyer and Floudas, 2005, Tardella, 1988/89, 2003, 2008], and outer approximation relaxations. We integrated these relaxations into a branch-and-bound algorithm and solved several large-scale instances to global optimality [Misener et al., 2010].

Because the only nonconvex terms in the standard and generalized pooling problems are bilinear products of a quality level or a proportion of a stream times a flow, the global optimization algorithms most relevant to the pooling problem are those specific to bilinear programs or (more generally) QCQPs and include the branch-and-bound method of Al-Khayyal and Falk [1983] that relaxes the bilinear terms using convex envelopes, the duality-based Global Optimization Algorithm of Floudas and Visweswaran [Floudas and Visweswaran, 1990, 1993, Visweswaran and Floudas, 1990, 1993], the RLT-based branch-and-cut algorithm of Audet et al. [2000], the branch-and-bound procedure of Linderoth [2005] that generates tight relaxations by partitioning two dimensional regions into triangles and rectangles, the integration of RLT and semidefinite programming relaxations by Anstreicher [2009], and the cutting plane algorithm of Bao et al. [2009] that generates a multiterm relaxation of a QCQP.

Note that algorithms addressing QCQPs are generically important because QCQPs commonly arise in process synthesis applications that include heat integration networks, separation systems, reactor networks, reactor-separator-recycle systems, and batch processes [Aggarwal and Floudas, 1990, Ciric and Floudas, 1989, Floudas and Anastasiadis, 1988, Floudas and Grossmann, 1987, Floudas and Paules, 1988, Kokossis and Floudas, 1991, 1994, Lin and Floudas, 2001]. Algorithms addressing quadratically-constrained quadratic GDPs are of particular relevance to the generalized pooling problem [Lee and Grossmann, 2001, 2003, Ruiz and Grossmann, 2010]. The extended pooling problem involves a number of additional multilinear, exponential, and power law equations that must be applied to each additional RFG product, so more generic global optimization techniques are needed for this final class of pooling problems [Floudas, 2000, Floudas et al., 2005, Floudas and Gounaris, 2009].

Of particular relevance to our recent work [Misener and Floudas, 2010, Misener et al., 2010] is the ab initio piecewise relaxation of nonconvex bilinear terms as first developed by Meyer and Floudas [2006] and Karuppiah and Grossmann [2006]. Recognizing the importance of formulating these piecewise-linear relaxations in the most computationally effective manner possible, Wicaksono and Karimi [2008] introduced fifteen mathematically-equivalent alternative formulations to the two used by Meyer and Floudas [2006] and Karuppiah and Grossmann [2006] and compared the relaxation performance on several test cases. We recently proposed five additional piecewise-linear formulations and conducted a comprehensive comparative study on the computational performance of these formulations on a collection of benchmark pooling problems [Gounaris et al., 2009]. Hasan and Karimi [2010] studied the possibility of bivariate partitioning, that is, segmenting both variables participating in each bilinear term. Other groups who have used piecewise-linear underestimators include: Bergamini et al. [2008] in their Outer Approximation for Global Optimization Algorithm; Saif et al. [2008], in a reverse osmosis network case study; and Pham et al. [2009], in a fast-solving algorithm that generates near-optimal solutions.

Each of the previously-mentioned partitioning schemes requires a number of binary variables that scales linearly with the number of disjunctive segments in the relaxation. Vielma and Nemhauser [2010] and Vielma et al. [2010a] recently proposed modeling piecewise functions with a number of binary switches that scales logarithmically with the number of partitions. Inspired by their work, we formulate a novel formulation for the logarithmically-sized piecewise relaxation of QCQPs and computationally test the performance of this new formulation. Li et al. [2009] have also suggested modeling piecewise-linear functions using a logarithmic number of binary variables and demonstrated that this may be superior to using a linear number of binary variables, but their mixed integer linear model (MILP) is not sharp [Vielma et al., 2010b]; *i.e.*, the linear programming relaxation of their MILP formulation is a superset of the convex hull of their

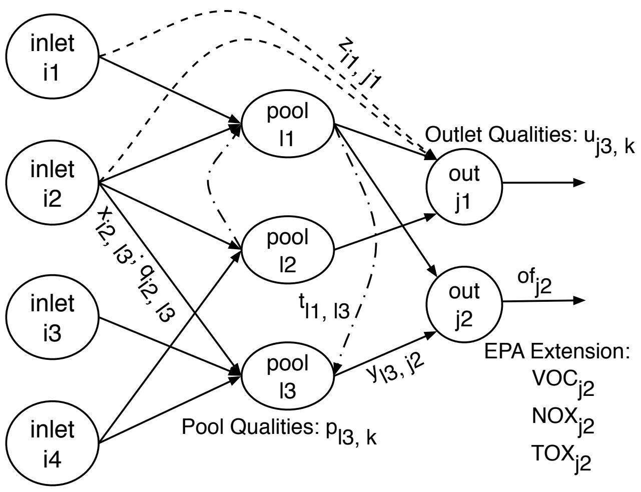


Figure 1: Variable Definitions for the Pooling Problem

MILP model [Jeroslow and Lowe, 1984]. Therefore, we base our formulation on the work of Vielma and Nemhauser [2010] and Vielma et al. [2010a].

3 Mathematical Formulation for the Pooling Problem: An Overview

Extensive descriptions of the mathematical pooling problem formulation are presented elsewhere for the standard [Misener and Floudas, 2009, Tawarmalani and Sahinidis, 2002], generalized [Audet et al., 2004, Lee and Grossmann, 2003, Meyer and Floudas, 2006, Misener and Floudas, 2010], and extended classes [Misener et al., 2010], but for completeness we outline the mathematical formulation of the pooling problem, list the pooling problem notation in Table 2, and illustrate the variable definitions in Figure 1. Table 3 lists the number of nonlinear terms in each of the benchmark pooling problems as a rough complexity estimate [Audet et al., 2004]. The nonlinear elements in the three classes of pooling problems are as follows:

- The bilinear terms in the **standard** pooling problem are either of the form $p_{l,k} \cdot y_{l,j}$ or $q_{i,l} \cdot y_{l,j}$ depending on whether Algorithm 2, which minimizes the number of bilinear terms, selects the *p*- or *q*-formulation [Ben-Tal et al., 1994, Misener and Floudas, 2009, Tawarmalani and Sahinidis, 2002].
- The **generalized** pooling problem has two sets of bilinear terms: $p_{l,k} \cdot y_{l,j}$ and $p_{l,k} \cdot t_{l,l'}$ [Misener and Floudas, 2010]. In the absence of pool to pool connections, the generalized pooling problem can be reformulated as a standard pooling problem and solved with either the *p*- or *q*-formulation [Audet et al., 2004].
- In addition to bilinear terms $q_{i,l} \cdot y_{l,j}$ and $u_{j,k} \cdot (of_j)$, the **extended** pooling problem incorporates the $u_{j,k}$ variables into nonlinear terms representing the EPA Complex Emissions Model [40CFR80.45,

Table 2: Notation for the Pooling Problem [Misener and Floudas, 2009]

Type	Name	Description
Indices	$i \in \{1, 2, \ldots, I\}$	Input streams (raw materials or feedstocks)
	$l \in \{1, 2, \ldots, L\}$	Pools (blending facilities)
	$j \in \{1, 2, \ldots, J\}$	Output streams (end products)
	$k \in \{1, 2, \ldots, K\}$	Attributes (qualities monitored)
Sets	T_X	(i, l) pairs for which input to pool connection allowed
	T_Y	(l, j) pairs for which pool to output connection allowed
	T_Z	(i, j) pairs for which input to output connection allowed
	T_T	(l, l') pairs for which pool to pool connection allowed $(l \neq l')$
Continuous Variables	$x_{i,l}$	Flow from input i to pool l
	$y_{l,j}$	Flow from intermediate pool node l to output j
	$z_{i,j}$	Direct flow from input feedstock i to product j
	$t_{l,l'}$	Flow from pool l to pool l' $(l \neq l')$
	(of_j)	Outflow of product j
	$p_{l,k}$	Level of quality attribute k in pool l
	$q_{i,l}$	Proportion of flow from input i to pool l
	$u_{j,k}$	Level of quality attribute k in product j
Binary Variables	γ_i^{inlt}	Input i is activated
	γ_l^{pool}	Pool l is activated
	$\gamma_{i,l}^x$	Active link from input i to pool l
	$\gamma_{l,j}^y$	Active link from intermediate pool node l to output j
	$\gamma_{i,j}^z$	Active link flow from input feedstock i to product j
	$\gamma_{l,l'}^t$	Active link from pool l to pool l' $(l \neq l')$
Parameters	c_i	Unit cost of raw material feedstock i
	d_j	Unit revenue for product j
	$c^\gamma,\ c$	Fixed and variable costs for the generalized problem
	$A_i^L - A_i^U$	Availability bounds of input i
	S_l	Volumetric size capacity of pool l
	$D_j^L - D_j^U$	Demand bounds for product j
	$C_{i,k}$	Level of quality k in raw material feedstock i
	$P_{j,k}^L - P_{j,k}^U$	Acceptable composition range of quality k in product j
	$r_{l,k}$	removal ratio of quality k in pool l
	f_{MIN}	Minimum flow on the network

Table 3: Test Suite of Pooling Problems

Problem Class	Problem Name		Global Optimum	Nonlinear Terms	Literature Source
Standard	Adhya	1	-549.80	20	Adhya et al. [1999]
		2	-549.80	20	
		3	-561.05	32	
		4	-877.65	40	
	BenTal	4	-450	2	Ben-Tal et al. [1994]
		5	-3500	30	
	Foulds	2	-1100	8	Foulds et al. [1992]
		3	-8	128	
		4	-8	128	
		5	-8	64	
	Haverly	1	-400	2	Haverly [1978]
		2	-600	2	
		3	-750	2	
	RT	2	-4391.83	18	Audet et al. [2004]
Generalized	Lee	1	-4640	24	Lee and Grossmann [2003]
		2	-3849	36	
	Meyer	4	$1.086 \cdot 10^6$	48	Meyer and Floudas [2006]
		10	$1.086 \cdot 10^6$	300	
		15	$9.437 \cdot 10^5$	675	Misener and Floudas [2010]
Extended	EPA	Small R1	-279.4	108	Misener et al. [2010]
		Small R2	-280.8	108	
		Med (R1/R2)	-4567	180	
		Large R1	$-1.490 \cdot 10^4$	640	
		Large R2	$-1.496 \cdot 10^4$	640	

2007, Misener et al., 2010].

The two alternative formulations for the standard pooling problem are the p- and q-formulations. The p-formulation is:

$$\min_{\substack{x_{i,l}, y_{l,j}, \\ z_{i,j}, p_{l,k}}} \sum_{(i,l)\in T_X} c_i \cdot x_{i,l} - \sum_{(l,j)\in T_Y} d_j \cdot y_{l,j} - \sum_{(i,j)\in T_Z} (d_j - c_i) \cdot z_{i,j} \tag{P1}$$

Feed Availability

$$\left[A_i^L \le \sum_{l:(i,l)\in T_X} x_{i,l} + \sum_{j:(i,j)\in T_Z} z_{i,j} \le A_i^U \quad \forall\, i \right. \tag{P2}$$

Pool Capacity

$$\left[\sum_{i:(i,l)\in T_X} x_{i,l} \le S_l \quad \forall\, l \right. \tag{P3}$$

Product Demand

$$\left[D_j^L \le \sum_{l:(l,j)\in T_Y} y_{l,j} + \sum_{i:(i,j)\in T_Z} z_{i,j} \le D_j^U \quad \forall\, j \right. \tag{P4}$$

Material Balance

$$\left[\sum_{i:(i,l)\in T_X} x_{i,l} - \sum_{j:(l,j)\in T_Y} y_{l,j} = 0 \quad \forall\, l \right. \tag{P5}$$

Quality Balance

$$\left[\sum_{i:(i,l)\in T_X} C_{i,k} \cdot x_{i,l} = p_{l,k} \cdot \sum_{j:(l,j)\in T_Y} y_{l,j} \quad \forall\, l,\, k \right. \tag{P6}$$

Product Quality

$$\left[\begin{array}{l} \sum\limits_{l:(l,j)\in T_Y} p_{l,k} \cdot y_{l,j} + \\ \sum\limits_{i:(i,j)\in T_Z} C_{i,k} \cdot z_{i,j} \end{array} \left\{ \begin{array}{l} \ge P_{j,k}^L \left(\sum\limits_{l:(l,j)\in T_Y} y_{l,j} + \sum\limits_{i:(i,j)\in T_Z} z_{i,j} \right) \\ \le P_{j,k}^U \left(\sum\limits_{l:(l,j)\in T_Y} y_{l,j} + \sum\limits_{i:(i,j)\in T_Z} z_{i,j} \right) \end{array} \right. \quad \forall\, j,\, k \right. \tag{P7}$$

Hard Bounds

$$\left[\begin{array}{rcll} 0 \le & x_{i,l} & \le \min\{A_i^U, S_l, \sum\limits_{j:(l,j)\in T_Y} D_j^U\} & \forall\, (i,l) \in T_X \\ 0 \le & y_{l,j} & \le \min\{S_l,\, D_j^U,\, \sum\limits_{i:(i,l)\in T_X} A_i^U\} & \forall\, (l,j) \in T_Y \\ 0 \le & z_{ij} & \le \min\{A_i^U,\, D_j^U\} & \forall\, (i,j) \in T_Z \\ \min\limits_i C_{i,k} \le & p_{l,k} & \le \max\limits_i C_{i,k} & \forall\;\; l,\, k \end{array} \right. \tag{P8}$$

The q-formulation, developed by Ben-Tal et al. [1994], replaces the feedstock flow rate variables $x_{i,l}$ with proportional flow rates $q_{i,l}$ using the transformation:

$$x_{i,l} = q_{i,l} \sum_{j:(l,j)\in T_Y} y_{l,j} \quad \forall\, (i,l) \in T_X.$$

The q-formulation is:

$$\min_{\substack{q_{i,l}, y_{l,j}, \\ z_{i,j}}} \sum_{\substack{(i,l)\in T_X \\ (l,j)\in T_Y}} c_i \cdot q_{i,l} \cdot y_{l,j} - \sum_{(l,j)\in T_Y} d_j \cdot y_{l,j} - \sum_{(i,j)\in T_Z} (d_j - c_i) \cdot z_{i,j} \tag{Q1}$$

Feed Availability
$$A_i^L \leq \sum_{\substack{l:(i,l)\in T_X \\ (l,j)\in T_Y}} q_{i,l} \cdot y_{l,j} + \sum_{j:(i,j)\in T_Z} z_{i,j} \leq A_i^U \quad \forall\, i \tag{Q2}$$

Pool Capacity
$$\sum_{j:(l,j)\in T_Y} y_{l,j} \leq S_l \quad \forall\, l \tag{Q3}$$

Product Demand
$$D_j^L \leq \sum_{l:(l,j)\in T_Y} y_{l,j} + \sum_{i:(i,j)\in T_Z} z_{i,j} \leq D_j^U \quad \forall\, j \tag{Q4}$$

Simplex Definition
$$\sum_{i:(i,l)\in T_X} q_{i,l} = 1 \quad \forall\, l \tag{Q5}$$

Product Quality
$$\begin{array}{l} \sum\limits_{\substack{l:(l,j)\in T_Y \\ i:(i,l)\in T_X}} C_{i,k} \cdot q_{i,l} \cdot y_{l,j} + \\ \sum\limits_{i:(i,j)\in T_Z} C_{i,k} \cdot z_{i,j} \end{array} \left\{ \begin{array}{l} \geq P_{j,k}^L \left(\sum\limits_{l:(l,j)\in T_Y} y_{l,j} + \sum\limits_{i:(i,j)\in T_Z} z_{i,j} \right) \\ \leq P_{j,k}^U \left(\sum\limits_{l:(l,j)\in T_Y} y_{l,j} + \sum\limits_{i:(i,j)\in T_Z} z_{i,j} \right) \end{array} \right. \quad \forall\, j,\, k \tag{Q6}$$

Hard Bounds
$$\begin{array}{llll} 0 \leq & q_{i,l} & \leq 1 & \forall\,(i,l) \in T_X \\ 0 \leq & y_{l,j} & \leq \min\{S_l,\, D_j^U,\, \sum\limits_{i:(i,l)\in T_X} A_i^U\} & \forall\,(l,j) \in T_Y \\ 0 \leq & z_{i,j} & \leq \min\{A_i^U,\, D_j^U\} & \forall\,(i,j) \in T_Z \end{array} \tag{Q7}$$

When using the *q*-formulation, we always append Eq. (PQ) which is redundant in Eqs. (Q1) – (Q7) but tightens the MILP relaxation of the bilinear terms $q_{i,l} \cdot y_{l,j}$ [Tawarmalani and Sahinidis, 2002].

$$\sum_{i:(i,l)\in T_X} q_{i,l} \cdot y_{l,j} = y_{l,j} \quad \forall\, l,\, j. \tag{PQ}$$

The generalized formulation increases the complexity of the standard pooling problems by allowing pool to pool connections and transforming the topology into a decision variable. The new continuous variable $t_{l,l'}$ represents the pool-to-pool connections and the binary variables γ_i^I, γ_l^P, $\gamma_{i,l}^x$, $\gamma_{l,j}^y$, $\gamma_{i,j}^z$, $\gamma_{l,l'}^t$ activate the appropriate nodes and inter-node connections.

$$
\begin{aligned}
\min_{\substack{x_{i,l},\, y_{l,j},\\ z_{i,j},\, p_{l,k}}} & \sum_{(i,l)\in T_X} \left(c^x_{i,l} \cdot x_{i,l} + c^{\gamma,x}_{i,l} \cdot \gamma^x_{i,l} \right) + \sum_{(l,j)\in T_Y} \left(c^y_{l,j} \cdot y_{l,j} + c^{\gamma,y}_{l,j} \cdot \gamma^y_{l,j} \right) + \\
& \sum_{(i,j)\in T_Z} \left(c^z_{i,j} \cdot z_{i,j} + c^{\gamma,z}_{i,j} \cdot \gamma^z_{i,j} \right) + \sum_{(l,l')\in T_T} \left(c^t_{l,l'} \cdot t_{l,l'} + c^{\gamma,t}_{l,l'} \cdot \gamma^t_{l,l'} \right) \\
& \sum_{l} \left(c^{\text{pool}}_l \cdot \left[\sum_{l':(l,l')\in T_T} t_{l,l'} + \sum_{j:(l,j)\in T_Y} y_{l,j} \right] + c^{\gamma,\text{pool}}_l \cdot \gamma^{\text{pool}}_l \right) \\
& \sum_{i} \left(c^{\text{inlt}}_i \cdot \left[\sum_{l:(i,l)\in T_X} x_{i,l} + \sum_{j:(i,j)\in T_Z} z_{i,j} \right] + c^{\gamma,\text{inlt}}_i \cdot \gamma^{\text{inlt}}_i \right)
\end{aligned}
\tag{GEN1}
$$

Feed Availability

$$
\left[A^L_i \cdot \gamma^{\text{inlt}}_i \le \sum_{l:(i,l)\in T_X} x_{i,l} + \sum_{j:(i,j)\in T_Z} z_{i,j} \le A^U_i \cdot \gamma^{\text{inlt}}_i \quad \forall\, i \right. \tag{GEN2}
$$

Pool Capacity

$$
\left[\sum_{l':(l,l')\in T_T} t_{l,l'} + \sum_{j:(l,j)\in T_Y} y_{l,j} \le S_l \cdot \gamma^{\text{pool}}_l \quad \forall\, l \right. \tag{GEN3}
$$

Product Demand

$$
\left[D^L_j \le \sum_{l:(l,j)\in T_Y} y_{l,j} + \sum_{i:(i,j)\in T_Z} z_{i,j} \le D^U_j \quad \forall\, j \right. \tag{GEN4}
$$

Material Balance

$$
\left[\sum_{i:(i,l)\in T_X} x_{i,l} + \sum_{l':(l',l)\in T_T} t_{l',l} - \sum_{l':(l,l')\in T_T} t_{l,l'} - \sum_{j:(l,j)\in T_Y} y_{l,j} = 0 \quad \forall\, l \right. \tag{GEN5}
$$

Quality Balance

$$
\left[\begin{aligned}
& (1 - r_{l,k}) \cdot \left(\sum_{i:(i,l)\in T_X} C_{i,k} \cdot x_{i,l} + \sum_{l':(l',l)\in T_T} p_{l',k} \cdot t_{l',l} \right) = \\
& p_{l,k} \cdot \left(\sum_{j:(l,j)\in T_Y} y_{l,j} + \sum_{l':(l,l')\in T_T} t_{l,l'} \right)
\end{aligned} \quad \forall\, l,\, k \right. \tag{GEN6}
$$

Product Quality

$$
\left[\begin{aligned}
& \sum_{l:(l,j)\in T_Y} p_{l,k} \cdot y_{l,j} + \\
& \sum_{i:(i,j)\in T_Z} C_{i,k} \cdot z_{i,j}
\end{aligned}
\begin{cases}
\ge P^L_{j,k} \left(\sum_{l:(l,j)\in T_Y} y_{l,j} + \sum_{i:(i,j)\in T_Z} z_{i,j} \right) \\
\le P^U_{j,k} \left(\sum_{l:(l,j)\in T_Y} y_{l,j} + \sum_{i:(i,j)\in T_Z} z_{i,j} \right)
\end{cases} \quad \forall\, j,\, k \right. \tag{GEN7}
$$

Hard Bounds

$$
\left[\begin{array}{lll}
0 \le\ x_{i,l} & \le \min\{A^U_i, S_l, \sum_{j:(l,j)\in T_Y} D^U_j\} \cdot \gamma^x_{i,l} & \forall\, (i,\, l) \in T_X \\
0 \le\ y_{l,j} & \le \min\{S_l,\, D^U_j,\, \sum_{i:(i,l)\in T_X} A^U_i\} \cdot \gamma^y_{l,j} & \forall\, (l,\, j) \in T_Y \\
0 \le\ z_{i,j} & \le \min\{A^U_i,\, D^U_j\} \cdot \gamma^z_{i,j} & \forall\, (i,\, j) \in T_Z \\
0 \le\ t_{l,l'} & \le \min\{S_l,\, S_{l'},\, \sum_j D^U_j,\, \sum_i A^U_i\} \cdot \gamma^t_{l,l'} & \forall\, (l,\, l') \in T_T \\
0 \le\ p_{l,k} & \le (1 - r_{l,k}) \cdot \max_i C_{i,k} & \forall\ \ l,\, k
\end{array} \right. \tag{GEN8}
$$

In each problem class, we also allow a minimum flow rate f_{MIN} below which a pipe or node will be deactivated [Meyer and Floudas, 2006, Misener and Floudas, 2010]. When $f_{\text{MIN}} > 0$, we use binary variables to enforce the condition each flow is either at least the minimum flow rate or set to zero. Finally, the extended formulation builds on the standard pooling problem by appending constraints associated with the EPA Complex Emissions Model [40CFR80.45, 2007] that involve the outlet qualities $u_{j,\,k}$. For the sake of brevity, we will not include the full model in this paper and the reader is directed to Misener et al. [2010].

4 Piecewise Underestimators for a Bilinear Term: Logarithmic Number of Binary Variables

In this section, we introduce a novel piecewise linear relaxation for bilinear terms with a logarithmic number of binary terms (§4.2.2). For completeness, we also present the disjunctive program upon which all of the piecewise relaxations are based (§4.1) and a version of relaxation **nf4r** (§4.2.1) from Gounaris et al. [2009]. Both relaxations we describe assume uniform partitioning based on both the results of our computational studies [Gounaris et al., 2009] and the proof of Hasan and Karimi [2010] suggesting that segments of identical length are likely to produce the tightest relaxation.

4.1 Disjunctive Program

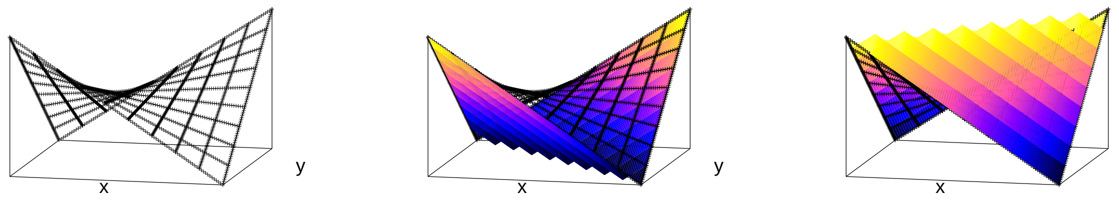
The convex hull, or tightest possible relaxation, of bilinear function $z(x,\, y)$:

$$z = x \cdot y \quad \text{such that} \quad x,\, y \in \mathbb{R},\, x^L \leq x \leq x^U,\, y^L \leq y \leq y^U \tag{1}$$

defined on a box is [Al-Khayyal and Falk, 1983, McCormick, 1976]:

$$z \geq x \cdot y^L + x^L \cdot y - x^L \cdot y^L \tag{2}$$
$$z \geq x \cdot y^U + x^U \cdot y - x^U \cdot y^U \tag{3}$$
$$z \leq x \cdot y^L + x^U \cdot y - x^U \cdot y^L \tag{4}$$
$$z \leq x \cdot y^U + x^L \cdot y - x^L \cdot y^U. \tag{5}$$

The envelope in Eqs. (2) – (5), illustrated in Figure 2, is dependent on the size of the domain, so we partition one of the variables (x) into N_P segments and come up with the disjunctive program [Gounaris et al., 2009, Karuppiah and Grossmann, 2006, Meyer and Floudas, 2006, Wicaksono and Karimi, 2008]:



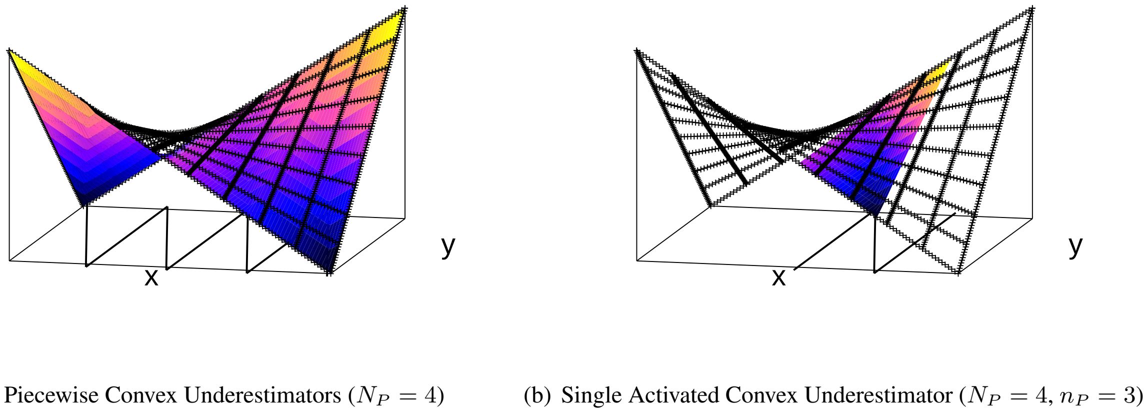
(a) Bilinear Term (b) Convex Envelope (c) Concave Envelope

Figure 2: Convex Hull of $x \cdot y$

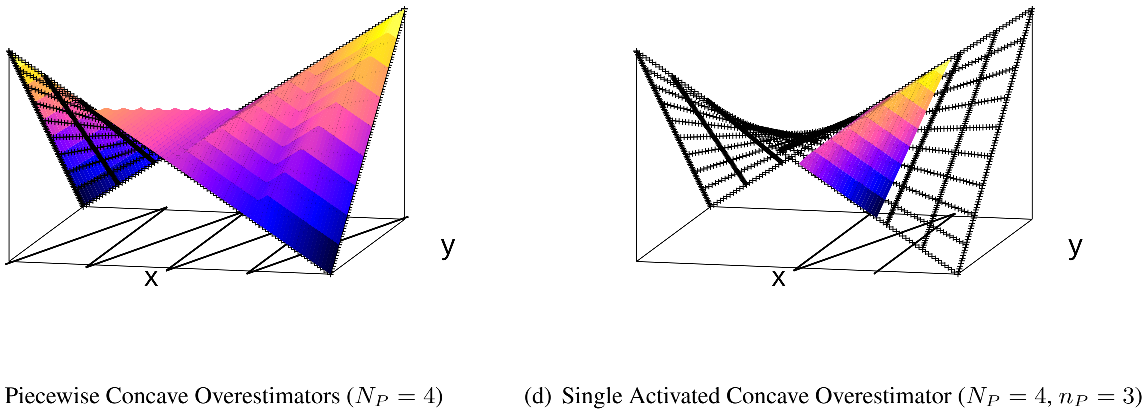
$$\bigvee n_P \in \{1, \ldots, N_P\} \left[\begin{array}{l} W(n_P) \\ \\ z \geq x \cdot y^L + (x^L + a \cdot (n_P - 1)) \cdot (y - y^L) \\ z \geq x \cdot y^U + (x^L + a \cdot \; n_P \qquad) \cdot (y - y^U) \\ z \leq x \cdot y^L + (x^L + a \cdot \; n_P \qquad) \cdot (y - y^L) \\ z \leq x \cdot y^U + (x^L + a \cdot (n_P - 1)) \cdot (y - y^U) \\ \\ x^L + a \cdot (n_P - 1) \leq x \leq x^L + a \cdot n_P \\ y^L \leq y \leq y^U \end{array}\right] \tag{6}$$

In disjunctive program (6), diagrammed in Figure 3(a) & 3(c) for the case $(N_P = 4)$, there are N_P segments on range $\left[x^L, \, x^U\right]$ and each segment is bounded by $\left[x^L + a \cdot (n_P - 1), \, x^L + a \cdot n_P\right] \;\forall n_P \in \{1, \ldots, N_P\}$ where $a = \frac{x^U - x^L}{N_P}$. Figure 3(b) & 3(d) illustrate the under- and overestimator, respectively, that are generated when a single segment is selected (in Figure 3(b) & 3(d), $n_P = 3$).

Figures 4(a) & 4(b) project Figures 2(b) & 2(c) and Figures 3(a) & 3(c), respectively, onto $y = y^L + \frac{5}{8} \cdot (y^U - y^L)$. Although the convex hull illustrated in Figure 4(a) is valid for all $x \in \left[x^L, \, x^U\right]$, each of the piecewise under- and overestimators illustrated in Figure 4(b) is valid in the domain $x \in \left[x^L + a \cdot (n_P - 1), \, x^L + a \cdot n_P\right]$ for exactly one $n_P \in \{1, \ldots, N_P\}$.

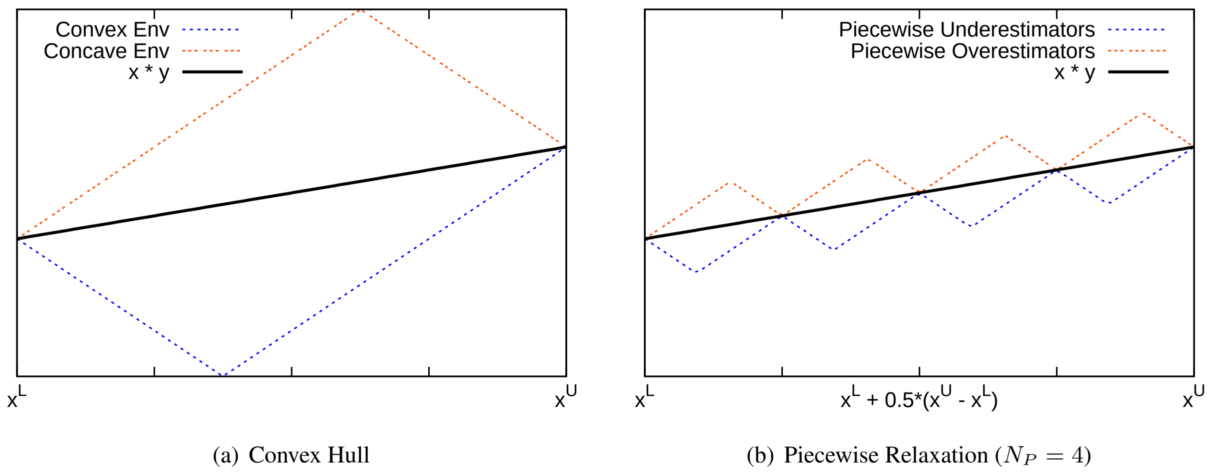


(a) Piecewise Convex Underestimators ($N_P = 4$) (b) Single Activated Convex Underestimator ($N_P = 4, n_P = 3$)



(c) Piecewise Concave Overestimators ($N_P = 4$) (d) Single Activated Concave Overestimator ($N_P = 4, n_P = 3$)

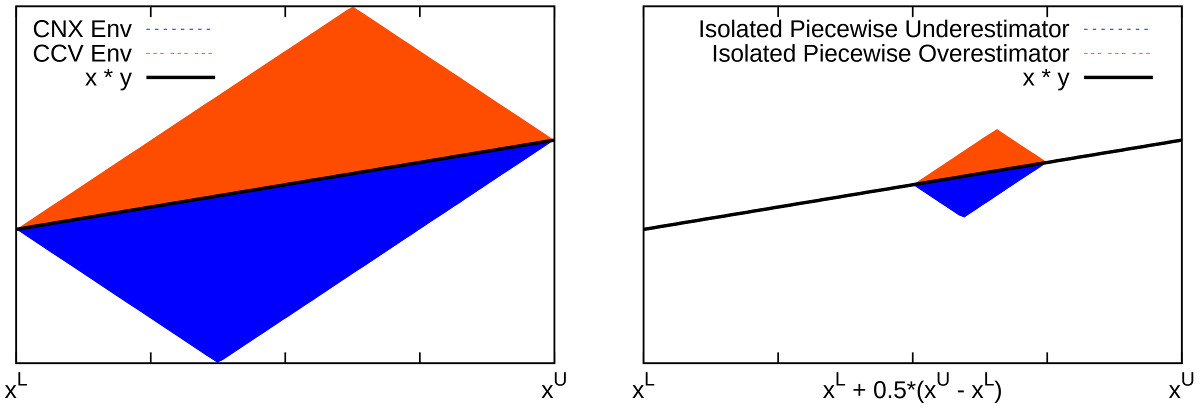
Figure 3: Piecewise Under- & Overestimators of $(x \cdot y)$ *when* $(N_P = 4)$



(a) Convex Hull (b) Piecewise Relaxation ($N_P = 4$)

Figure 4: Under- & Overestimators of $(x \cdot y)$ *Projected on* $\left(y = y^L + \frac{5}{8} \cdot (y^U - y^L)\right)$

The partitioning schemes described in Section 4.2 activate exactly one $n_P \in \{1, \ldots, N_P\}$ so that the feasible space of corresponding to the relaxation of $(x \cdot y)$ goes from the parallelogram in Figure 5(a) to a significantly smaller parallelogram illustrated in Figure 5(b) that is activated or deactivated by Disjunctive Program (6).



(a) Convex Hull (b) Single Activated Region ($N_P = 4$)

Figure 5: Feasible Set Corresponding to the Relaxation of $(x \cdot y)$ *Projected on* $\left(y = y^L + \frac{5}{8} \cdot (y^U - y^L)\right)$

4.2 Two Reformulations of the Disjunctive Program as a MILP

The linear partitioning scheme to be presented in Section 4.2.1 and the logarithmic partitioning scheme to be introduced in Sections 4.2.2 exactly represent Disjunctive Program (6) by introducing additional variables and constraints to the formulation. Table 4 compares the size of the McCormick Envelope [Al-Khayyal and Falk, 1983, McCormick, 1976] in Eqs. (2) – (5) to that of the two formulations of Disjunctive Program (6).

Table 4: Number of Additional Vars & Constraints for the McCormick, Linear, & Logarithmic Relaxation of a Single Bilinear Term

		Continuous Vars	Binary Vars	Constraints
McC Hull (no partitions)	§4.1	1	–	4
Linear Scheme	§4.2.1	$N_P + 1$	N_P	$N_P + 8$
Logarithmic Scheme	§4.2.2	$2 \cdot \lceil \log_2 N_P \rceil + 1$	$\lceil \log_2 N_P \rceil$	$3 \cdot \lceil \log_2 N_P \rceil + 7^{\dagger}$

†One constraint can be removed when the number of partitions is a power of two.

4.2.1 Linear Relaxation Scheme

This relaxation, with a linear number of binary variables, is based on **nf4r** [Gounaris et al., 2009]. Two additional variable sets are introduced in the linear partitioning scheme:

- Binary switch: $\lambda \in \{0,\, 1\}^{N_P}$
- Continuous switch: $\Delta y \in \left[0,\, y^U - y^L\right]^{N_P}$

The binary switch λ is active (*i.e.*, $\lambda(n_P) = 1$) for the segment where $(x^L + a \cdot (n_P - 1) \leq x \leq x^L + a \cdot n_P)$ and is otherwise inactive. Equations (7) & (8) mathematically describe these restrictions on λ.

$$\sum_{n_P=1}^{N_P} \lambda(n_P) = 1 \tag{7}$$

$$x^L + \sum_{n_P=1}^{N_P} a \cdot (n_P - 1) \cdot \lambda(n_P) \leq x \leq x^L + \sum_{n_P=1}^{N_P} a \cdot n_P \cdot \lambda(n_P) \tag{8}$$

Equation 7 combined with the $\lambda \in \{0,\, 1\}^{N_P}$ definition is logically equivalent to declaring λ to be a special ordered set of type 1 (SOS1) because exactly one member of the set is non-zero. However, we do not explicitly declare λ to be SOS1 in our C++/CPLEX 12.1 [ILOG, 2009] implementation of this linear relaxation scheme for the solver APOGEE.

Figure 6 illustrates Eqs. (7) & (8) for the case $N_P = 4$ and $x = x^L + 2.6 \cdot a$ which implies that $n_p = 3$, the active binary switch is $\lambda(n_P = 3) = 1$, and Eq. (8) becomes $x^L + 2 \cdot a \leq x \leq x^L + 3 \cdot a$:

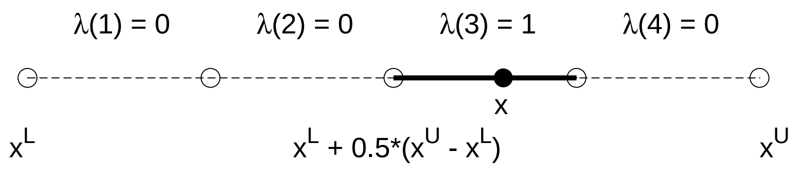


*Figure 6: Activation of a Single $\lambda(n_P)$ (*i.e., $n_P = 3 \implies \lambda(3) = 1$*) According to the Value of x ($N_P = 4$ and $x = x^L + 2.6 \cdot a$)*

Equation (9) sets continuous switch variable $\Delta y(n_P)$ to $(y - y^L)$ when $\lambda(n_P) = 1$ and 0 otherwise [Wicaksono and Karimi, 2008]:

$$y = y^L + \sum_{n_P=1}^{N_P} \Delta y(n_P) \ \text{ where } \ 0 \leq \Delta y(n_P) \leq (y^U - y^L) \cdot \lambda(n_P) \quad \forall\, n_P \in \{1, \ldots, N_P\} \tag{9}$$

Figure 7 diagrams Eq. (9) for the case depicted in Figure 6. Because the only active binary switch is $\lambda(n_P = 3) = 1$, the only nonzero Δy is $\Delta y(n_P = 3) = y - y^L$:

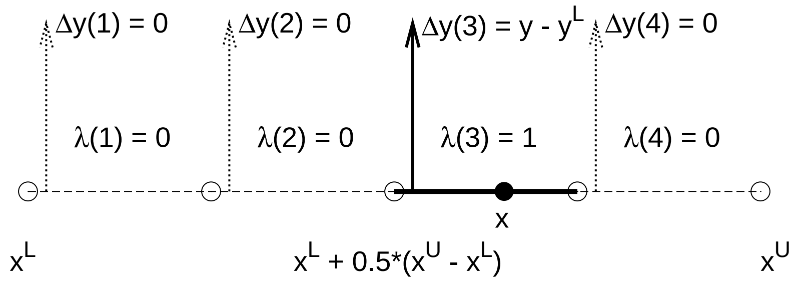


Figure 7: Activation of a Single $\Delta y(n_P)$ According to the Value of x ($N_P = 4$ and $x = x^L + 2.6 \cdot a$)

After defining these two switches λ and Δy, Equation Set (10) equivalently represents Disjunctive Program (6) and activates the correct over- and underestimators as shown in Figure 5(b):

$$z \geq x \cdot y^L + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot (n_P - 1)\right] \cdot \; \Delta y(n_P) \tag{10a}$$

$$z \geq x \cdot y^U + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot \; n_P \qquad\right] \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \lambda(n_P)\right] \tag{10b}$$

$$z \leq x \cdot y^L + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot \; n_P \qquad\right] \cdot \; \Delta y(n_P) \tag{10c}$$

$$z \leq x \cdot y^U + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot (n_P - 1)\right] \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \lambda(n_P)\right] \tag{10d}$$

$$x^L \leq x \leq x^U; \quad y^L \leq y \leq y^U$$

Notice that, after the introduction of Equation Set (10), Eq. (8) is redundant because the left hand side of Eq. (8) can be obtained up to a factor of $(y^U - y^L)$ by subtracting Eq. (10d) from Eq. (10a) and the right hand side of Eq. (8) is, up to the factor of $(y^U - y^L)$, Eq. (10b) minus Eq. (10c). Therefore, Eq. (8) follows from Equation Set (10).

4.2.2 Logarithmic Relaxation Scheme

For the logarithmic partitioning scheme, we define a parameter N_L and three additional variable sets so that the number of continuous variables, binary variables, and constraint equations all scale logarithmically.

- Number of Logarithmic Binary Variables: $N_L = \lceil \log_2 N_P \rceil$
- Binary switch: $\lambda \in \{0,\, 1\}^{N_L}$
- Continuous switch: $\Delta y \in \left[0,\, y^U - y^L\right]^{N_L}$
- Continuous slack: $s \in \left[0,\, y^U - y^L\right]^{N_L}$

As we described in Section 2, the logarithmic relaxation scheme presented in this section is inspired by the recent work of Vielma and Nemhauser [2010] and Vielma et al. [2010a]. The consistent elements between Vielma and Nemhauser [2010] and this contribution are (1) the logarithmic number of binary variables and constraint equations and (2) the injective mapping from the activation of exactly one of the N_P segments to the activation/deactivation of the N_L binary variables. However, the logarithmic number of continuous variables in this formulation differs from their linear number of continuous variables [Vielma and Nemhauser, 2010]. We can use a logarithmic number of continuous variables because the continuous switch $\Delta y(n_L)$ is effectively a linear reformulation of the bilinear term $(y - y^L) \cdot \lambda(n_L)$ [Wicaksono and Karimi, 2008]. It is important to note that since Li et al. [2009], Vielma and Nemhauser [2010], and Vielma et al. [2010a] focus on piecewise-linear functions, our contribution represents, to the best of our knowledge, the first logarithmically-scaled relaxation of continuous, nonlinear functions.

The N_L elements of λ activate or deactivate according to the binary representation of the largest grid point that is less than x. Equation (11) mathematically describes these restrictions. Although we model the mapping from the partition containing x to λ using a base-2 representation, note that any injective function $B : \{1,\, \ldots,\, N_P\} \mapsto \{0,\, 1\}^{\lceil \log_2 N_P \rceil}$ could formulate the SOS1-like constraints for the activation of exactly one of the N_P segments in Disjunctive Program (6) [Vielma and Nemhauser, 2010].

$$x^L + \sum_{n_L=1}^{N_L} 2^{n_L-1} \cdot a \cdot \lambda(n_L) \leq x \leq x^L + a + \sum_{n_L=1}^{N_L} 2^{n_L-1} \cdot a \cdot \lambda(n_L) \tag{11}$$

In the case when N_P is not a power of two (*i.e.*, $\log_2 N_P \neq \lceil \log_2 N_P \rceil$), we include an additional constraint to strengthen the linear programming relaxation of the MILP:

$$x^L + a + \sum_{n_L=1}^{N_L} 2^{n_L-1} \cdot a \cdot \lambda(n_L) \leq x^U \tag{12}$$

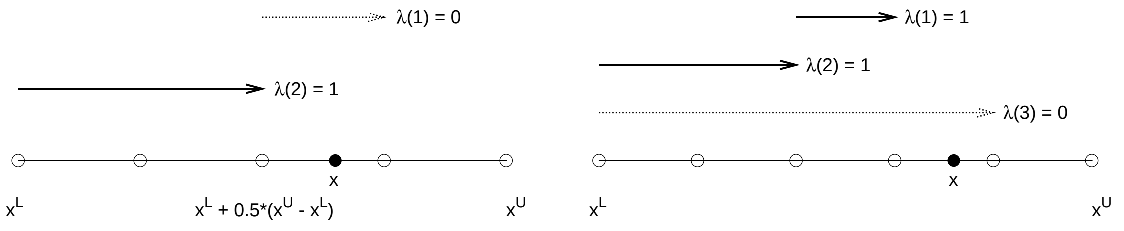
Figures 8(a) & 8(b) illustrate Eq. (11) for two example cases: $N_P = 4$ and $x = x^L + 2.6 \cdot a$ and $N_P = 5$ and $x = x^L + 3.6 \cdot a$, respectively. In Figure 8(a), which is identical to the case diagrammed in Figure 6, setting $\lambda(n_L = 1) = 1$ and $\lambda(n_L = 2) = 0$ results in the appropriate:

$$\begin{aligned} x &\geq x^L + \sum_{n_L=1}^{N_L} 2^{n_L-1} \cdot a \cdot \lambda(n_L) = x^L + 2 \cdot a \cdot \lambda(1) + a \cdot \lambda(2) = x^L + 2 \cdot a \\ x &\leq x^L + a + \sum_{n_L=1}^{N_L} 2^{n_L-1} \cdot a \cdot \lambda(n_L) = x^L + a + 2 \cdot a \cdot \lambda(1) + a \cdot \lambda(2) = x^L + 3 \cdot a \end{aligned}$$

For the case $N_P = 5$ and $x = x^L + 3.6 \cdot a$ in Figure 8(b), Eq. (11) is satisfied when $\lambda(n_L = 1) = 0$, $\lambda(n_L = 2) = 1$, and $\lambda(n_L = 3) = 1$:

$$x \geq x^L + \sum_{n_L=1}^{N_L} 2^{n_L-1} \cdot a \cdot \lambda(n_L) = x^L + 4 \cdot a \cdot \lambda(1) + 2 \cdot a \cdot \lambda(2) + a \cdot \lambda(3) = x^L + 3 \cdot a$$

$$x \leq x^L + a + \sum_{n_L=1}^{N_L} 2^{n_L-1} \cdot a \cdot \lambda(n_L) = x^L + a + 4 \cdot a \cdot \lambda(1) + 2 \cdot a \cdot \lambda(2) + a \cdot \lambda(3) = x^L + 4 \cdot a$$



(a) $N_P = 4$ and $x = x^L + 2.6 \cdot a$ (b) $N_P = 5$ and $x = x^L + 3.6 \cdot a$

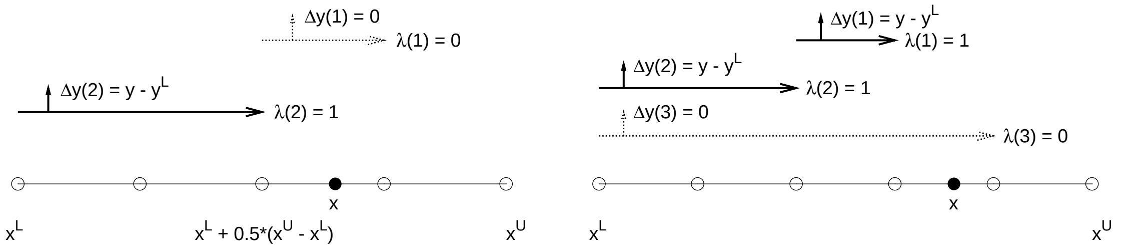
Figure 8: Activation of $\lambda(n_L)$ according to the Value of x

As in Section 4.2.1, the $\Delta y(n_L)$ variables are equal to $(y - y^L)$ for each active $\lambda(n_L)$ (*i.e.*, $\lambda(n_L) = 1 \iff \Delta y(n_L) = y - y^L$). Because $\Delta y(n_L)$ can be active for multiple $n_L \in \{1, \ldots, N_L\}$ rather than for a single partition (*i.e.*, Δy is not SOS1 in the logarithmic formulation), we introduce continuous slack $s(n_L)\ \forall\, n_L \in \{1, \ldots, N_L\}$:

$$\Delta y(n_L) \leq (y^U - y^L) \cdot \lambda(n_L) \tag{13a}$$

$$\Delta y(n_L) = (y - y^L) - s(n_L) \quad \text{where} \quad 0 \leq s(n_L) \leq (y^U - y^L) \cdot (1 - \lambda(n_L)) \tag{13b}$$

Figure 9 represents the same two examples diagrammed in Figure 8. Notice that $\lambda(n_L) = 1 \implies \Delta y(n_L) = y - y^L$ and $\lambda(n_L) = 0 \implies \Delta y(n_L) = 0$:



(a) $N_P = 4$ and $x = x^L + 2.6 \cdot a$ (b) $N_P = 5$ and $x = x^L + 3.6 \cdot a$

Figure 9: Activation of $\Delta y(n_L)$ according to the Value of x

With the definitions above, a final logarithmic partitioning scheme equivalent to Disjunctive Program (6) is:

$$z \geq x \cdot y^L + \ x^L \qquad \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L-1} \cdot \Delta y(n_L)\right] \tag{14a}$$

$$z \geq x \cdot y^U + (x^L + a) \cdot (y - y^U) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L-1} \cdot \big(\Delta y(n_L) - (y^U - y^L) \cdot \lambda(n_L)\big)\right] \tag{14b}$$

$$z \leq x \cdot y^L + (x^L + a) \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L-1} \cdot \Delta y(n_L)\right] \tag{14c}$$

$$z \leq x \cdot y^U + \ x^L \qquad \cdot (y - y^U) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L-1} \cdot \big(\Delta y(n_L) - (y^U - y^L) \cdot \lambda(n_L)\big)\right] \tag{14d}$$

$$x^L \leq x \leq x^U; \quad y^L \leq y \leq y^U$$

As in Section 4.2.1, Eq. (11) is redundant after the introduction of Equation Set (14) and unnecessary in the formulation.

4.2.3 Summary of the Relaxation Schemes

We conclude Section 4.2 by summarizing the two relaxation schemes presented in Sections 4.2.1 & 4.2.2. The two relaxations are also presented in Appendix A for the bilinear term $p_{l,k} \cdot y_{l,j}$ that appears in the p-formulation of the standard pooling problem. Appendix A assumes that the $p_{l,k}$ variables are partitioned. The linear relaxation scheme (§4.2.1) has $\lambda \in \{0, 1\}^{N_P}$ and $\Delta y \in \big[0, \, y^U - y^L\big]^{N_P}$:

$$\sum_{n_P=1}^{N_P} \lambda(n_P) = 1 \quad (\textit{i.e.}, \ \lambda \ \text{SOS1}) \tag{15a}$$

$$x^L + \sum_{n_P=1}^{N_P} a \cdot (n_P - 1) \cdot \lambda(n_P) \leq x \leq x^L + \sum_{n_P=1}^{N_P} a \cdot n_P \cdot \lambda(n_P) \tag{15b}$$

$$y = y^L + \sum_{n_P=1}^{N_P} \Delta y(n_P) \ \text{ where } \ 0 \leq \Delta y(n_P) \leq (y^U - y^L) \cdot \lambda(n_P) \quad \forall \, n_P \in \{1, \ldots, N_P\} \tag{15c}$$

$$z \geq x \cdot y^L + \sum_{n_P=1}^{N_P} \big[x^L + a \cdot (n_P - 1)\big] \cdot \ \Delta y(n_P) \tag{15d}$$

$$z \geq x \cdot y^U + \sum_{n_P=1}^{N_P} \big[x^L + a \cdot \ n_P \qquad\big] \cdot \big[\Delta y(n_P) - (y^U - y^L) \cdot \lambda(n_P)\big] \tag{15e}$$

$$z \leq x \cdot y^L + \sum_{n_P=1}^{N_P} \big[x^L + a \cdot \ n_P \qquad\big] \cdot \ \Delta y(n_P) \tag{15f}$$

$$z \leq x \cdot y^U + \sum_{n_P=1}^{N_P} \left[x^L + a \cdot (n_P - 1)\right] \cdot \left[\Delta y(n_P) - (y^U - y^L) \cdot \lambda(n_P)\right] \tag{15g}$$

$$x^L \leq x \leq x^U; \quad y^L \leq y \leq y^U \tag{15h}$$

The logarithmic relaxation scheme (§4.2.2) has $N_L = \lceil \log_2 N_P \rceil$, $\lambda \in \{0,\, 1\}^{N_L}$, $\Delta y \in \left[0,\, y^U - y^L\right]^{N_L}$, and $s \in \left[0,\, y^U - y^L\right]^{N_L}$. The equation marked with a dagger (†) is only used when N_P is not a power of two.

$$x^L + \sum_{n_L=1}^{N_L} 2^{n_L - 1} \cdot a \cdot \lambda(n_L) \leq x \leq x^L + a + \sum_{n_L=1}^{N_L} 2^{n_L - 1} \cdot a \cdot \lambda(n_L) \tag{16a}$$

$$x^L + a + \sum_{n_L=1}^{N_L} 2^{n_L - 1} \cdot a \cdot \lambda(n_L) \leq x^U \tag{16a†}$$

$$\Delta y(n_L) \leq (y^U - y^L) \cdot \lambda(n_L) \quad \forall\, n_L \in \{1,\, \ldots,\, N_L\} \tag{16b}$$

$$\Delta y(n_L) = (y - y^L) - s(n_L) \quad \forall\, n_L \in \{1,\, \ldots,\, N_L\} \tag{16c}$$

$$s(n_L) \leq (y^U - y^L) \cdot (1 - \lambda(n_L)) \quad \forall\, n_L \in \{1,\, \ldots,\, N_L\} \tag{16d}$$

$$z \geq x \cdot y^L + \ x^L \qquad \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L - 1} \cdot \Delta y(n_L)\right] \tag{16e}$$

$$z \geq x \cdot y^U + (x^L + a) \cdot (y - y^U) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L - 1} \cdot \left(\Delta y(n_L) - (y^U - y^L) \cdot \lambda(n_L)\right)\right] \tag{16f}$$

$$z \leq x \cdot y^L + (x^L + a) \cdot (y - y^L) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L - 1} \cdot \Delta y(n_L)\right] \tag{16g}$$

$$z \leq x \cdot y^U + \ x^L \qquad \cdot (y - y^U) + \left[\sum_{n_L=1}^{N_L} a \cdot 2^{n_L - 1} \cdot \left(\Delta y(n_L) - (y^U - y^L) \cdot \lambda(n_L)\right)\right] \tag{16h}$$

$$x^L \leq x \leq x^U; \quad y^L \leq y \leq y^U \tag{16i}$$

5 Comparison Between Linear & Logarithmic Relaxation Schemes

Our twofold purpose in this paper is to (1) compare the linear and logarithmic relaxation schemes presented in Section 4.2 within the context of a branch-and-bound global optimization algorithm for the three classes of pooling problems and to (2) test and report the performance of the solver APOGEE. We use the suite of twenty-five pooling problems recorded in Table 3 to test APOGEE and compare the linear and logarithmic relaxation schemes presented in Section 4.2 by globally optimizing each problem with both relaxation schemes for $N_P = 3,\, 4,\, 5,\, 6,\, 8,\, 12,\, 16$. Algorithm 1, our global optimization algorithm, is identical to the back-end of the online APOGEE application and similar to global optimization approaches that have previously addressed bilinear programs [Audet et al., 2004, Bergamini et al., 2008, Karuppiah and Grossmann, 2006, Misener and Floudas, 2010, Misener et al., 2010, Tawarmalani and Sahinidis, 2002].

The way we compare the performance of the linear and logarithmic relaxations within a global optimization algorithm differs from previous work that isolates the performance of a single MILP to compare different formulations of piecewise-linear functions [Gounaris et al., 2009, Hasan and Karimi, 2010, Meyer and Floudas, 2006, Vielma and Nemhauser, 2010, Vielma et al., 2010a, Wicaksono and Karimi, 2008]. In this paper, our goal is to design the best overall algorithm that can address any pooling problem, so we are interested in the performance of the two relaxations when they are incorporated into the APOGEE algorithm rather than in an isolated root node relaxation. Therefore, Table 8 compares the linear and logarithmic schemes within the context of Algorithm 1 and the only distinction between the linear and logarithmic test is the relaxation of the bilinear terms in the *Solve the MILP Relaxation* step of Algorithm 1. Also differing from the work of Vielma and Nemhauser [2010], we have not designed specialized branching schemes for the logarithmic representation of the piecewise functions.

In addition to the relative sizes of the linear and logarithmic relaxation of a single bilinear term $x \cdot y$ that are described in Table 4, we record in Table 5 the relative sizes of the MILP relaxations for bilinear programs such as the pooling problem. In each of the three classes of pooling problems, there is one set of variables that is partitioned based on Algorithm 2, so we reduce the number of binary variables in the MILP formulation by reusing the binary variables representing the partitions. To give a better sense of the relative sizes of the pooling problems for the twenty-five benchmark problems, we also record the sizes of the MILP relaxations at the root node for $N_P = 4$ and $N_P = 8$ in Tables 6 and 7, respectively. In Tables 6 and 7, the *Total* column represents the size of the root node pooling problem relaxation and *PW* is the contribution from the piecewise bilinear relaxation only. As should be expected, the smaller size of the logarithmic relaxation is particularly noticeable for $N_P = 8$.

Table 5: Number of Additional Vars & Constraints for the McCormick, Linear, & Logarithmic Pooling Problem Relaxations

	Continuous Vars	Binary Vars	Constraints
McC Hull	N_{BIL}	–	$4 \cdot N_{\text{BIL}}$
Lin. Rlxn	$N_{\text{BIL}} \cdot (N_P + 1)$	$N_{\text{VAR}} \cdot N_P$	$N_{\text{BIL}} \cdot (N_P + 5) + 3 \cdot N_{\text{VAR}}$
Log. Rlxn	$N_{\text{BIL}} \cdot (2 \cdot N_L + 1)$	$N_{\text{VAR}} \cdot N_L$	$N_{\text{BIL}} \cdot (3 \cdot N_L + 4) + 3 \cdot N_{\text{VAR}}^{\dagger}$

$^{\dagger}N_{\text{VAR}}$ constraints can be removed when the number of partitions N_P is a power of two

$N_L = \lceil \log_2 N_P \rceil$; $N_{\text{BIL}} \equiv$ number of bilinear terms; $N_{\text{VAR}} \equiv$ number of partitioned variables

The results of Table 8 were generated on a four-core, 2.27GHz Intel Xeon E5520 processor running Linux using a C++ program that interfaces to SNOPT 5.3 [Gill et al., 1999] and a four-threaded implementation of CPLEX 12.1 [ILOG, 2009]. Although the back-end of the APOGEE application runs the same C++ program on the same processor, APOGEE users will experience solution time variability due to currently running jobs, network load, the non-deterministic performance of the opportunistic CPLEX 12.1 [ILOG, 2009] parallel solver, and the alternative choice of SCIP 2.0.0 [Achterberg et al., 2008] linked with CLP

Algorithm 1 Globally Optimize a Standard, Generalized, or Extended Pooling Problem

```
Receive xml file from helios.princeton.edu/APOGEE/ submission and load data
if Topological Correctness Checks Fail then
    Return error message describing the infeasibility issue.
end if

Choose the formulation and relaxation scheme (Algorithm 2).

Initialize UB $= \infty$ and the branch-and-bound tree with root NODE that incorporates topological reason-
ing to determine variable bounds [Misener and Floudas, 2009, 2010, Misener et al., 2010].

while Branch-and-bound tree non-empty and running time within 7200 CPU s do

    If NODE is root or the bounds parent of NODE significantly improved with optimality-based bounds
    tightening (OBBT), tighten the variables participating nonlinearly with OBBT [Belotti et al., 2009,
    Floudas, 2000].

    Solve the MILP Relaxation of the Pooling Problem for NODE [ILOG, 2009].

    if NODE is Infeasible or $\frac{\text{UB}-\text{LB}_{\text{NODE}}}{|\text{UB}|} < \epsilon_{\text{OPT TOL}}$ then
        Discard NODE
    else
        NODE begets two progeny nodes by branching on the variable that contributes to the greatest dis-
        crepancy between the auxiliary relaxation variables and the original problem variables [Adjiman
        et al., 1998a,b, Audet et al., 2000, Misener and Floudas, 2010]. The branching point is a convex com-
        bination of the variable mid-point ($\lambda = 0.15$) and the solution to the MILP relaxation ($1-\lambda = 0.85$)
        [Adjiman et al., 1998a,b, Belotti et al., 2009, Floudas, 2000, Tawarmalani and Sahinidis, 2002].

        Use topological reasoning to tighten the variable bounds on the two new nodes [Misener and Floudas,
        2009, 2010, Misener et al., 2010].

        Locally solve the Pooling Problem for NODE using the pool of feasible solutions discovered by the
        relaxation as initial points [Gill et al., 1999].

        if $\text{UB}_{\text{NODE}} < \text{UB}$ then
            UB $\leftarrow \text{UB}_{\text{NODE}}$
        end if
    end if
    NODE $\leftarrow$ node in the tree with the smallest $\text{LB}_{\text{node}}$
end while

if Branch-and-bound tree is empty and UB $< \infty$ then
    Return $\epsilon_{\text{OPT TOL}}$ optimal solution to user.
else if Branch-and-bound tree is empty then
    Return infeasibility state to the user.
else
    Algorithm did not converge at the time limit (7200 CPU s). Return the best found feasible solution to
    user.
end if
```

Algorithm 2 Choose the Formulation and Relaxation Scheme

```
if Standard Pooling Problem then
    if p-Formulation has Fewer Bilinear Terms than q-Formulation [Ben-Tal et al., 1994] then
        if Fewer $p_{l,k}$ Variables than $y_{l,j}$ Variables then
            Use the p-Formulation [Misener and Floudas, 2009] and partition the $p_{l,k}$ variables.
        else
            Use the p-Formulation [Misener and Floudas, 2009] and partition the $y_{l,j}$ variables.
        end if
    else
        if Fewer $q_{i,l}$ Variables than $y_{l,j}$ Variables then
            Use the pq-Formulation [Misener and Floudas, 2009] and partition the $q_{i,l}$ variables.
        else
            Use the pq-Formulation [Misener and Floudas, 2009] and partition the $y_{l,j}$ variables.
        end if
    end if
else if Generalized Pooling Problem then
    if Fewer $p_{l,k}$ Variables than $y_{l,j}$ and $t_{l,l'}$ Variables then
        Use the Generalized-Formulation [Audet et al., 2004, Lee and Grossmann, 2003, Misener and
        Floudas, 2010] and partition the $p_{l,k}$ variables.
    else
        Use the Generalized-Formulation [Audet et al., 2004, Lee and Grossmann, 2003, Misener and
        Floudas, 2010] and partition the $y_{l,j}$ and $t_{l,l'}$ variables.
    end if
else
    Use the Extended-Formulation [Misener et al., 2010] and partition the $q_{i,l}$ variables.
end if
```

Table 6: Relative Sizes of the Linear and Logarithmic Relaxation Scheme for $N = 4$

Problem	Linear Number of Binaries						Logarithmic Number of Binaries					
	# Cnt Vars		# Bin Vars		# Eqs		# Cnt Vars		# Bin Vars		# Eqs	
	Total	PW	Total	PW	Total	PW	Total	PW	Total	PW	Total	PW
Adhya 1	114	100	20	20	258	195	114	100	10	10	273	210
Adhya 2	114	100	20	20	274	195	114	100	10	10	289	210
Adhya 3	181	160	32	32	403	312	181	160	16	16	427	336
Adhya 4	219	200	32	32	465	384	219	200	16	16	497	416
BenTal 4	19	10	4	4	41	21	19	10	2	2	42	22
BenTal 5	189	150	24	24	341	288	189	150	12	12	365	312
Foulds 2	63	40	8	8	113	78	63	40	4	4	119	84
Foulds 3	809	640	32	32	1287	1176	809	640	16	16	1407	1296
Foulds 4	809	640	32	32	1287	1176	809	640	16	16	1407	1296
Foulds 5	421	320	16	16	687	588	421	320	8	8	747	648
Haverly 1	18	10	4	4	39	21	18	10	2	2	40	22
Haverly 2	18	10	4	4	39	21	18	10	2	2	40	22
Haverly 3	18	10	4	4	39	21	18	10	2	2	40	22
RT 2	107	90	24	24	227	180	107	90	12	12	239	192
Lee 1	161	120	41	32	285	240	161	120	25	16	301	256
Lee 2	225	180	57	48	415	360	225	180	33	24	439	384
Meyer 4	304	240	103	48	625	468	304	240	79	24	661	504
Meyer 10	1708	1500	307	120	3307	2790	1708	1500	247	60	3577	3060
Meyer 15	3742	3360	531	180	7162	6183	3742	3360	441	90	7789	6810
EPA Small R1/2	357	40	46	16	799	84	351	40	38	8	155	819
EPA Med R1/2	687	150	85	40	1467	300	182	150	20	20	436	320
EPA Large R1/2	2411	700	206	56	5168	1302	2393	700	178	28	5362	1428

Total is the size of the root node pooling problem relaxation. *PW* is the contribution from the bilinear relaxation.

Table 7: Relative Sizes of the Linear and Logarithmic Relaxation Scheme for $N = 8$

Problem	Linear Number of Binaries						Logarithmic Number of Binaries					
	# Cnt Vars		# Bin Vars		# Eqs		# Cnt Vars		# Bin Vars		# Eqs	
	Total	PW	Total	PW	Total	PW	Total	PW	Total	PW	Total	PW
Adhya 1	194	180	40	40	338	275	154	140	15	15	333	270
Adhya 2	194	180	40	40	354	275	154	140	15	15	349	270
Adhya 3	309	288	64	64	531	440	245	224	24	24	523	432
Adhya 4	379	360	64	64	625	544	299	280	24	24	617	536
BenTal 4	27	18	8	8	49	29	23	14	3	3	48	28
BenTal 5	309	270	48	48	461	408	249	210	18	18	455	402
Foulds 2	95	72	16	16	145	110	79	56	6	6	143	108
Foulds 3	1321	1152	64	64	1799	1688	1065	896	24	24	1791	1680
Foulds 4	1321	1152	64	64	1799	1688	1065	896	24	24	1791	1680
Foulds 5	677	576	32	32	943	844	549	448	12	12	939	840
Haverly 1	26	18	8	8	47	29	22	14	3	3	46	28
Haverly 2	26	18	8	8	47	29	22	14	3	3	46	28
Haverly 3	26	18	8	8	47	29	22	14	3	3	46	28
RT 2	179	162	48	48	299	252	143	126	18	18	293	246
Lee 1	257	216	73	64	381	336	209	168	33	24	373	328
Lee 2	369	324	105	96	559	504	297	252	45	36	547	492
Meyer 4	496	432	151	96	817	660	400	336	91	36	805	648
Meyer 10	2908	2700	427	240	4507	3990	2308	2100	277	90	4477	3960
Meyer 15	6430	6048	711	360	9850	8871	5086	4704	486	135	9805	8826
EPA Small R1/2	405	72	62	32	831	116	371	56	42	12	843	112
EPA Med R1/2	847	270	125	80	1587	420	740	210	75	30	1609	410
EPA Large R1/2	3027	1260	262	112	5728	1862	2687	980	192	42	5782	1848

Total is the size of the root node pooling problem relaxation. *PW* is the contribution from the bilinear relaxation.

Table 8: Comparing the Linear and Logarithmic Relaxation Scheme for Several Partitioning Levels (CPU s for $\epsilon_{OPT\,TOL}$ Global Optimization)

Problem	$\epsilon_{\text{OPT TOL}} = \frac{\text{UB}-\text{LB}}{\lvert\text{UB}\rvert}$	$N_P = 3$		$N_P = 4$		$N_P = 5$		$N_P = 6$		$N_P = 8$		$N_P = 12$		$N_P = 16$	
		Lin	Log	Lin	Log	Lin	Log	Lin	Log	Lin	Log	Lin	Log	Lin	Log
Adhya 1	$1 \cdot 10^{-6}$	***0.628***	0.799	0.752	***0.613***	**0.822**	**0.868**	**0.791**	**0.782**	1.141	**0.826**	2.250	**1.041**	9.450	**0.981**
Adhya 2	$1 \cdot 10^{-6}$	***0.428***	0.563	***0.469***	**0.494**	**0.585**	0.705	**0.611**	**0.585**	**0.590**	**0.627**	1.425	**0.693**	5.716	**0.650**
Adhya 3	$1 \cdot 10^{-6}$	**0.485**	0.548	***0.434***	***0.469***	**0.534**	0.653	**0.583**	0.694	0.690	**0.501**	2.074	**0.654**	–	**0.613**
Adhya 4	$1 \cdot 10^{-6}$	***0.203***	0.240	**0.226**	***0.209***	**0.249**	**0.267**	0.315	**0.259**	0.351	**0.288**	1.324	**0.384**	1.671	**0.391**
BenTal 4	$1 \cdot 10^{-6}$	***0.023***	0.038	***0.022***	0.029	***0.023***	0.038	***0.022***	0.025	**0.026**	0.030	0.034	**0.025**	0.040	**0.028**
BenTal 5	$1 \cdot 10^{-6}$	***0.082***	0.114	***0.076***	***0.081***	**0.200**	0.223	0.263	**0.174**	0.269	**0.185**	0.604	**0.187**	2.936	**0.219**
Foulds 2	$1 \cdot 10^{-6}$	***0.029***	0.040	***0.028***	0.032	***0.028***	0.060	***0.028***	0.043	***0.031***	**0.034**	**0.032**	0.044	***0.030***	0.035
Foulds 3	$1 \cdot 10^{-6}$	***0.324***	***0.328***	0.614	**0.367**	0.766	**0.683**	**0.401**	0.575	**0.428**	**0.399**	0.492	**0.409**	2.499	**0.432**
Foulds 4	$1 \cdot 10^{-6}$	***0.329***	1.698	**0.468**	1.652	***0.303***	0.449	**0.385**	**0.413**	1.207	**0.382**	0.604	**0.556**	**0.628**	1.063
Foulds 5	$1 \cdot 10^{-6}$	**0.130**	**0.121**	0.226	***0.097***	**0.209**	0.366	**0.148**	0.177	**0.127**	0.156	0.316	**0.224**	1.577	**0.526**
Haverly 1	$1 \cdot 10^{-6}$	**0.025**	**0.028**	***0.023***	0.027	***0.022***	0.038	***0.024***	**0.026**	***0.023***	0.030	0.032	**0.027**	0.038	**0.027**
Haverly 2	$1 \cdot 10^{-6}$	**0.031**	**0.028**	***0.026***	***0.024***	**0.035**	**0.036**	**0.035**	0.073	**0.029**	**0.028**	0.034	**0.029**	0.030	***0.026***
Haverly 3	$1 \cdot 10^{-6}$	***0.031***	**0.034**	***0.032***	***0.032***	***0.029***	0.033	***0.032***	**0.033**	**0.033**	***0.031***	0.039	**0.034**	0.039	**0.033**
RT 2	$1 \cdot 10^{-6}$	**0.258**	0.588	0.274	***0.208***	**0.305**	0.369	0.376	**0.334**	0.702	**0.313**	6.410	**0.532**	9.158	**0.417**
Lee 1	$1 \cdot 10^{-3}$	**3.602**	11.161	4.127	**3.556**	***2.562***	3.739	10.257	**5.285**	9.415	***2.622***	9.091	**3.665**	22.750	**5.928**
Lee 2	$1 \cdot 10^{-3}$	***38.793***	***40.189***	51.216	***36.973***	67.675	**55.808**	76.475	**49.075**	138.42	**58.585**	256.53	**70.696**	791.31	**77.835**
Meyer 4	$1 \cdot 10^{-3}$	**85.243**	136.16	***9.055***	11.787	**12.067**	27.138	**16.832**	23.869	**13.449**	16.687	**19.994**	71.906	**12.338**	37.160
Meyer 10	$1 \cdot 10^{-3}$	**3203.8**	4707.3	**369.91**	1330.0	**295.75**	2554.9	***222.35***	994.42	1298.2	**520.81**	**1193.3**	3878.4	2600.9	**2206.4**
Meyer 15	$1 \cdot 10^{-3}$	***921.06***	5841.2	**1380.1**	3345.5	**1421.2**	–	**1957.3**	6074.0	4518.9	**3332.4**	**4570.9**	–	6472.7	**3750.6**
EPA Small R1	$1 \cdot 10^{-3}$	**2.183**	**2.002**	2.052	***1.672***	**1.958**	**1.964**	**1.994**	**1.912**	**1.960**	**1.927**	**1.841**	2.247	**1.915**	2.166
EPA Small R2	$1 \cdot 10^{-3}$	***1.852***	***1.785***	***1.732***	***1.833***	2.433	**2.007**	**1.944**	**1.955**	**1.915**	**2.008**	**1.931**	**1.943**	2.082	***1.882***
EPA Med R1	$1 \cdot 10^{-3}$	***132.32***	146.01	234.30	**210.77**	***143.27***	178.15	189.54	**127.83**	207.52	**171.10**	264.35	**147.22**	443.45	**125.16**
EPA Med R2	$1 \cdot 10^{-3}$	***119.48***	160.94	250.41	**209.74**	129.33	***113.42***	**142.42**	180.75	216.00	**166.51**	254.50	**149.66**	353.60	**151.08**
EPA Large R1	$1 \cdot 10^{-2}$	4580.3	**683.85**	**3149.5**	5996.9	**880.89**	1228.4	**832.56**	3862.8	412.10	**227.01**	***187.82***	1613.0	595.49	**355.41**
EPA Large R2	$1 \cdot 10^{-2}$	482.08	**186.58**	**261.72**	303.53	**527.23**	3310.9	519.70	***89.781***	618.08	***85.127***	296.41	**236.40**	187.18	**112.22**
Best Form. (Lin/Log)		23	10	16	16	21	8	18	14	11	21	7	19	4	21
Best Part. Level		15		20		7		5		6		1		4	

Bold numbers highlight the best formulation (Lin or Log) for each partitioning level N_P. Italicized numbers represent the best time for a single problem. CPU times within 10% of the minimum are considered equivalent to the best time. Empty table entries represent runs that did not converge in the 7200 CPU s time limit.

1.13 [Forrest et al., 2010] as a MILP solver.

The first two columns of Table 8 list the twenty-five test cases and the optimality tolerance to which each of the problems was solved. In general, APOGEE solves standard problems to $\epsilon_{\text{OPT TOL}} = 1 \cdot 10^{-6}$ and the larger generalized and extended problems to $\epsilon_{\text{OPT TOL}} = 1 \cdot 10^{-3}$, but in Table 8 we have solved the two largest extended problems to $\epsilon_{\text{OPT TOL}} = 1 \cdot 10^{-2}$ so that we can better compare the formulations (these two problems do not converge to $\epsilon_{\text{OPT TOL}} = 1 \cdot 10^{-3}$ within the 7200 CPU s time limit).

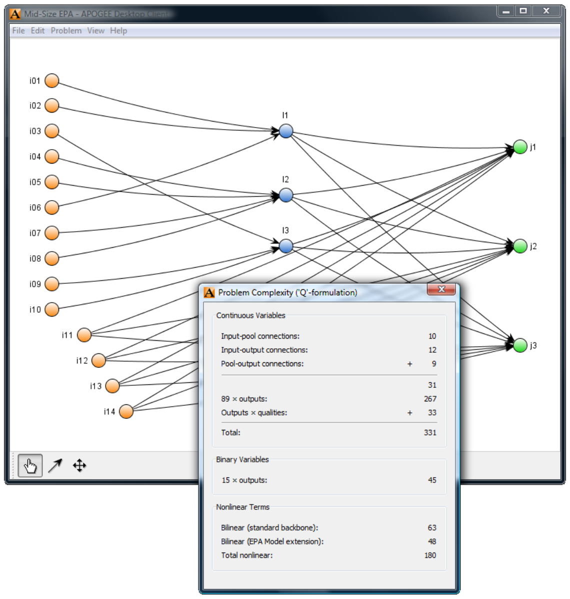
The remaining columns in Table 8 compare the linear and logarithmic formulations by highlighting in bold the formulation that is faster for each partitioning level $N_P = 3,\ 4,\ 5,\ 6,\ 8,\ 12,\ 16$. Some comparisons (*e.g.*, EPA Small R1 for $N_P = 8$) have the times corresponding to both the linear and logarithmic formulation highlighted because we consider CPU times within 10% of the minimum to be equivalent to the best time. The penultimate row in Table 8 sums the number of times that the linear and logarithmic formulations are faster for each of the seven partitioning levels. For $N_p = 8$, the linear and logarithmic formulations *win* 11 and 21 times, respectively (observe that the sum of the linear and logarithmic wins rarely adds to the number of test cases because a time within 10% of a win is also a win). The final row in Table 8 counts the number of times that the partitioning level (irrespective of formulation) is best for a specific pooling problem (again, the numbers in the final row do not sum to 25 because a tie is a win in our methodology). For example, using a partitioning level of $N_P = 8$ with either formulation is best for 6 of the 25 test instances.

Table 8 suggests several conclusions. First, corroborating the computational results of Vielma et al. [2010a], the logarithmic relaxation scheme is particularly advantageous for higher partitioning levels ($N_P \geq 8$). This observation suggests a robustness to the claims of Vielma et al. [2010a] because their suggestion of using a logarithmic number of binary variables has only been tested for individual MILP models that feature piecewise-linear functions. In contrast, the MILP relaxation in our tests is a portion of a larger global optimization algorithm and the relaxation itself includes not only the piecewise relaxation of bilinear terms but also pooling problem topological constraints and possibly the unwieldy EPA Complex Emissions Model [40CFR80.45, 2007, Misener et al., 2010].

Next, the logarithmic formulation is comparatively better when the number of partitions is a power of two. This observation follows from the size analysis in Table 5 where it is evident that our logarithmic piecewise relaxation for $N_P = 2^{n-1} + 1$ is actually larger than the $N_P = 2^n$ relaxation. For example, $\lceil \log_2 5 \rceil = \lceil \log_2 8 \rceil = 3$, so the number of continuous and binary variables are equal for the logarithmic $N_P = 5$ and $N_P = 8$ formulations, but there are more constraint equations in the $N_P = 5$ case because Eq. (12) is used to sharpen the logarithmic formulation.

Finally, despite the evident advantage of the logarithmic partitioning scheme for large N_P, the global optimization algorithm seems to converge fastest with smaller partitioning levels (*e.g.*, $N_P = 3,\ 4,\ 5$) where there is an advantage in using the linear partitioning scheme. For the case $N_P = 4$, the linear and logarithmic schemes do win an equal number of times, but three of the very large-scale problems (*i.e.*, Meyer 10, Meyer 15, and EPA Large R1) have numerical difficulty with the logarithmic scheme and our goal is that the solver APOGEE should be as stable as possible. Although the relaxations with higher partitioning levels are generally tighter than the ones with fewer partitions and are therefore likely to require fewer nodes in a branch-and-bound optimization algorithm [Gounaris et al., 2009], there is a trade-off between relaxation strength and node solution time that encourages using fewer partitions for the pooling problem [Misener and Floudas, 2010]. Therefore, we will continue using the linear relaxation scheme to solve pooling problems. Although we are currently using $N_P = 4$ to solve pooling problems within APOGEE, note from Table 8 that guessing the right partition is not a crucial element of our process. The only three instances that do *not* converge within the time limit (7200 CPU s) are two logarithmic relaxations of the largest generalized pooling problem (Meyer 15) at partitioning levels that are not powers of two $(N_P = 5, 12)$ and one linear relaxation of a standard problem (Adhya 3) at an unreasonably high partitioning level $(N_P = 16)$.

6 The Computational Tool APOGEE


Mid-Size EPA - APOGEE Desktop Client
File Edit Problem View Help
i01
i02
i03
i04
i05
i06
i07
i08
i09
i10
i11
i12
i13
i14
I1
I2
I3
j1
j2
j3
Problem Complexity ('Q'-formulation)

Continuous Variables

Input-pool connections:		10
Input-output connections:		12
Pool-output connections:	+	9
		31
89 × outputs:		267
Outputs × qualities:	+	33
Total:		331

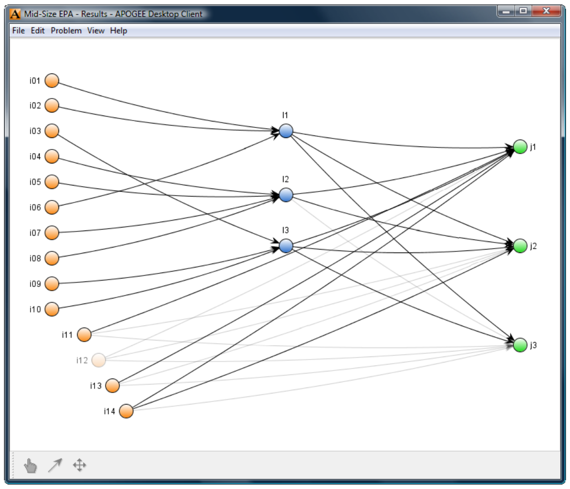
Binary Variables

15 × outputs:	45

Nonlinear Terms

Bilinear (standard backbone):	63
Bilinear (EPA Model extension):	48
Total nonlinear:	180

(a) Constructing a problem


Mid-Size EPA - Results - APOGEE Desktop Client
File Edit Problem View Help
i01
i02
i03
i04
i05
i06
i07
i08
i09
i10
i11
i12
i13
i14
I1
I2
I3
j1
j2
j3

(b) Viewing results

Figure 10: The APOGEE Desktop Client. Shown is the mid-size extended problem from Misener et al. [2010].

We have integrated our work on global optimization of pooling problems by developing the computational tool APOGEE, a generic system capable of globally optimizing the standard, generalized, and extended classes of pooling problems. The core algorithms are implemented as a C++ program that interfaces with the SNOPT 5.3 [Gill et al., 1999] and CPLEX 12.1 [ILOG, 2009] packages (Algorithm 1). APOGEE alternatively employs SCIP 2.0.0 [Achterberg et al., 2008] linked with CLP 1.13 [Forrest et al., 2010] as a MILP solver. To facilitate evaluation of the tool's performance and encourage feedback, we have also created the APOGEE Desktop Client, a cross-platform graphical user interface that enables users to construct problems interactively, submit specifications to the solver, and view results. Additionally, users are able to monitor the complexity and preferred formulation (*i.e.*, *p* or *q*) of a pooling problem as it is constructed (see Figure 10(a)). This information is relevant in predicting the computational intensity of the global optimization algorithm. The client software is freely available to the scientific community at `helios.princeton.edu/APOGEE/`.

The test suite of pooling problems presented in Table 3 is also available on the APOGEE website as pre-constructed problems. Users are encouraged to build their own problems from scratch, directly test the benchmark problems, or experiment with different scenarios by modifying the test suite of pooling problems. The desktop client warns users who submit unusually large problems that the instance is unlikely to completely converge within the time limit, but there is no size limit to accepted problems. For such large instances or problems of special interest, we recommend contacting Professor C. A. Floudas to discuss the possibility of using more specialized algorithms or more powerful computers [Misener and Floudas, 2010].

7 Conclusion

In this paper, inspired by recent work proposing modeling piecewise functions with a *logarithmic* number of binary switches [Vielma and Nemhauser, 2010, Vielma et al., 2010a], we develop a novel formulation for the piecewise relaxation of bilinear programs and computationally compare the performance of this new formulation to the best-performing piecewise relaxations with a linear number of binary variables. While we find a definitive advantage to using the logarithmic number of variables for a high partitioning level ($N_P \geq 8$), we observe that, for the pooling problem, a lower partitioning level tends to be advantageous. Therefore, using the linear partitioning level for solving large-scale pooling problems is recommended, but we conjecture that problem classes requiring finer partitioning would benefit from this alternative logarithmic relaxation scheme. We have also unified our work by developing APOGEE (Algorithms for Pooling-problem global Optimization in GEneral and Extended classes), a computational tool that globally optimizes standard, generalized, and extended pooling problems. APOGEE is freely available to the scientific community at `helios.princeton.edu/APOGEE/`.

Acknowledgments: The authors gratefully acknowledge support from the National Science Foundation (CBET – 0827907). R.M. is further thankful for her NSF Graduate Research Fellowship.

References

40CFR80.41. Code of Federal Regulations: Standards and requirements for compliance, 2008. `http://www.gpoaccess.gov/cfr/retrieve.html`.

40CFR80.45. Code of Federal Regulations: Complex emissions model, 2007. `http://www.gpoaccess.gov/cfr/retrieve.html`.

T. Achterberg, T. Berthold, T. Koch, and K. Wolter. Constraint integer programming: a new approach to integrate CP and MIP. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. CPAIOR, 2008.

N. Adhya, M. Tawarmalani, and N. V. Sahinidis. A Lagrangian approach to the pooling problem. *Ind. Eng. Chem. Res.*, 38(5):1965 – 1972, 1999.

C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice differentiable NLPs-I. Theoretical advances. *Comput. Chem. Eng.*, 22:1137 – 1158, 1998a.

C. S. Adjiman, I. P. Androulakis, and C. A. Floudas. A global optimization method, αBB, for general twice differentiable NLPs-II. Implementation and computional results. *Comput. Chem. Eng.*, 22:1159 – 1179, 1998b.

A. Aggarwal and C. A. Floudas. Synthesis of general distillation sequences - nonsharp separations. *Comput. Chem. Eng.*, 14(6):631–653, 1990.

F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Math. Oper. Res.*, 8(2):273 – 286, 1983.

H. Almutairi and S. Elhedhli. A new Lagrangean approach to the pooling problem. *J. Global Optim.*, 45: 237 – 257, 2009.

K. M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *J. Global Optim.*, 43(2-3):471 – 484, 2009.

C. Audet, P. Hansen, B. Jaumard, and G. Savard. A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Math. Program.*, 87(1):131 – 152, 2000.

C. Audet, J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenovic. Pooling problem: Alternate formulations and solution methods. *Manage. Sci.*, 50(6):761 – 776, 2004.

M. Bagajewicz. A review of recent design procedures for water networks in refineries and process plants. *Comput. Chem. Eng.*, 24:2093 – 2113, 2000.

R. C. Baliban, J. A. Elia, and C. A. Floudas. Toward novel hybrid biomass, coal, and natural gas processes for satisfying current transportation fuel demands, 1: Process alternatives, gasification modeling, process simulation, and economic analysis. *Ind. Eng. Chem. Res.*, 49(16):7343–7370, 2010a.

R. C. Baliban, J. A. Elia, and C. A. Floudas. Optimization framework for the simultaneous process synthesis, heat and power integration of a thermochemical hybrid biomass, coal, and natural gas facility. 2010b. Submitted.

X. Bao, N. V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs. *Optimization Methods and Software*, 24(4-5):485 – 504, 2009.

P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5):597–634, 2009.

A. Ben-Tal, G. Eiger, and V. Gershovitz. Global minimization by reducing the duality gap. *Math. Program.*, 63:193 – 212, 1994.

M. L. Bergamini, I. Grossmann, N. Scenna, and P. Aguirre. An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. *Comput. Chem. Eng.*, 32(3):477 – 493, 2008.

A. Chakraborty. A globally convergent mathematical model for synthesizing topologically constrained water recycle networks. *Comput. Chem. Eng.*, 33(7):1279 – 1288, 2009.

A. Chakraborty and A. A. Linninger. Plant-wide waste management. 1. synthesis and multiobjective design. *Ind. Eng. Chem. Res.*, 41(18):4591–4604, 2002.

A. Chakraborty and A. A. Linninger. Plant-wide waste management. 2. decision making under uncertainty. *Ind. Eng. Chem. Res.*, 42(2):357–369, 2003.

A. Chakraborty, R. D. Colberg, and A. A. Linninger. Plant-wide waste management. 3. long-term operation and investment planning under uncertainty. *Ind. Eng. Chem. Res.*, 42(20):4772–4788, 2003.

A. R. Ciric and C. A. Floudas. A retrofit approach for heat exchanger networks. *Comput. Chem. Eng.*, 13 (6):703 – 715, 1989.

J. A. Elia, R. C. Baliban, and C. A. Floudas. Toward novel hybrid biomass, coal, and natural gas processes for satisfying current transportation fuel demands, 2: Simultaneous heat and power integration. *Ind. Eng. Chem. Res.*, 49(16):7371–7388, 2010.

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

C. A. Floudas and A. Aggarwal. A decomposition strategy for global optimum search in the pooling problem. *ORSA J. Comput.*, 2:225 – 235, 1990.

C. A. Floudas and S. H. Anastasiadis. Synthesis of distillation sequences with several multicomponent feed and product streams. *Chem. Eng. Sci.*, 43(9):2407–2419, 1988.

C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *J. Global Optim.*, 45 (1):3 – 38, 2009.

C. A. Floudas and I. E. Grossmann. Synthesis of flexible heat-exchanger networks with uncertain flowrates and temperatures. *Comput. Chem. Eng.*, 11(4):319–336, 1987.

C. A. Floudas and P. M. Pardalos. State-of-the-art in global optimization - computational methods and applications - preface. *J. Glob. Optim.*, 7(2):113, 1995.

C. A. Floudas and G. E. Paules. A mixed-integer nonlinear programming formulation for the synthesis of heat-integrated distillation sequences. *Comput. Chem. Eng.*, 12(6):531 – 546, 1988.

C. A. Floudas and V. Visweswaran. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: I. Theory. *Comput. Chem. Eng.*, 14(12):1397 – 1417, 1990.

C. A. Floudas and V. Visweswaran. Primal-relaxed dual global optimization approach. *J. Optim. Theory Appl.*, 78(2):187 – 225, 1993.

C. A. Floudas, A. Aggarwal, and A. R. Ciric. Global optimum search for nonconvex NLP and MINLP problems. *Comput. Chem. Eng.*, 13(10):1117 – 1132, 1989.

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

J. Forrest, M. Saltzman, L. Hafer, and J. Hall. CLP. `https://projects.coin-or.org/Clp`, 2010. Version 1.13.

L. R. Foulds, D. Haughland, and K. Jornsten. A bilinear approach to the pooling problem. *Optim.*, 24:165 – 180, 1992.

K. C. Furman and I. P. Androulakis. A novel MINLP-based representation of the original complex model for predicting gasoline emissions. *Comput. Chem. Eng.*, 32:2857 – 2876, 2008.

B. Galan and I. E. Grossmann. Optimal design of distributed wastewater treatment networks. *Ind. Eng. Chem. Res.*, 37(10):4036 – 4048, 1998.

P. E. Gill, W. Murray, and M. A. Saunders. SNOPT. `http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm`, 1999. Version 5.3.

C. E. Gounaris, R. Misener, and C. A. Floudas. Computational comparison of piecewise-linear relaxations for pooling problems. *Ind. Eng. Chem. Res.*, 48(12):5742 – 5766, 2009.

H. J. Greenberg. Analyzing the pooling problem. *ORSA J. Comput.*, 7:205 – 217, 1995.

I. E. Grossmann and G. Guillén-Gosálbez. Scope for the application of mathematical programming techniques in the synthesis and planning of sustainable processes. *Comput. Chem. Eng.*, 34(9):1365 – 1376, 2010.

M. M. F. Hasan and I. A. Karimi. Piecewise linear relaxation of bilinear programs using bivariate partitioning. *AIChE J.*, 56(7):1880 – 1893, 2010.

C. A. Haverly. Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, 25:19 – 28, 1978.

ILOG. CPLEX. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, 2009. Version 12.1.

R. G. Jeroslow and J. K. Lowe. Modelling with integer variables. *Math. Prog. Stud.*, 22:167 – 184, 1984.

J. Jeżowski. Review of water network design methods with literature annotations. *Ind. Eng. Chem. Res.*, 49 (10):4475 – 4516, 2010.

R. Karuppiah and I. E. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Comput. Chem. Eng.*, 30:650 – 673, 2006.

A. C. Kokossis and C. A. Floudas. Synthesis of isothermal reactor–separator–recycle systems. *Chem. Eng. Sci.*, 46(5 - 6):1361 – 1383, 1991.

A. C. Kokossis and C. A. Floudas. Optimization of complex reactor networks–II. nonisothermal operation. *Chem. Eng. Sci.*, 49(7):1037 – 1051, 1994.

L. S. Lasdon, A. D. Waren, S. Sarkar, and F. Palacios. Solving the pooling problem using generalized reduced gradient and successive linear programming algorithms. *ACM SIGMAP Bull.*, 27:9 – 15, 1979.

S. Lee and I. E. Grossmann. A global optimization algorithm for nonconvex generalized disjunctive programming and applications to process systems. *Comput. Chem. Eng.*, 25(11-12):1675 – 1697, 2001.

S. Lee and I. E. Grossmann. Global optimization of nonlinear generalized disjunctive programming with bilinear equality constraints: applications to process networks. *Comput. Chem. Eng.*, 27(11):1557 – 1575, 2003.

H. L. Li, H. C. Lu, C. H. Huang, and N. Z. Hu. A superior representation method for piecewise linear functions. *INFORMS J. Comput.*, 21(2):314–321, 2009.

X. Li, E. Armagan, A. Tomasgard, and P. I. Barton. Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE J.*, 2010. In Press.

X. Lin and C. A. Floudas. Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. *Comput. Chem. Eng.*, 25(4 - 6):665 – 674, 2001.

J. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Math. Program.*, 103(2):251 – 282, 2005.

W. A. Lodwick. Preprocessing nonlinear functional constraints with applications to the pooling problem. *ORSA J. Comput.*, 4(2):119–131, 1992.

A. Malcolm, L. Zhang, and A. A. Linninger. Design of environmental regulatory policies for sustainable emission reduction. *AIChE J.*, 52(8):2792 – 2804, 2006.

G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part 1-convex underestimating problems. *Math. Program.*, 10(1):147 – 175, 1976.

C. A. Meyer and C. A. Floudas. Global optimization of a combinatorially complex generalized pooling problem. *AIChE J.*, 52(3):1027 – 1037, 2006.

C. A. Meyer and C. A. Floudas. Convex envelopes for edge-concave functions. *Math. Program.*, 103(2): 207–224, 2005.

R. Misener and C. A. Floudas. Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, 8(1):3 – 22, 2009.

R. Misener and C. A. Floudas. Global optimization of large-scale pooling problems: Quadratically constrained MINLP models. *Ind. Eng. Chem. Res.*, 49(11):5424 – 5438, 2010.

R. Misener, C. E. Gounaris, and C. A. Floudas. Mathematical modeling and global optimization of large-scale extended pooling problems with the (EPA) complex emissions constraints. *Comput. Chem. Eng.*, 34 (9):1432 – 1456, 2010.

V. Pham, C. Laird, and M. El-Halwagi. Convex hull discretization approach to the global optimization of pooling problems. *Ind. Eng. Chem. Res.*, 48:1973 – 1979, 2009.

I. Quesada and I. E. Grossmann. Global optimization of bilinear process networks with multicomponent flows. *Comput. Chem. Eng.*, 19:1219 – 1242, 1995.

J. P. Ruiz and I. E. Grossmann. Strengthening of lower bounds in the global optimization of bilinear and concave generalized disjunctive programs. *Comput. Chem. Eng.*, 34(6):914 – 930, 2010.

Y. Saif, A. Elkamel, and M. Pritzker. Global optimization of reverse osmosis network for wastewater treatment and minimization. *Ind. Eng. Chem. Res.*, 47(9):3060 – 3070, 2008.

H. D. Sherali and W. P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Dordrecht, Netherlands, 1999.

H. D. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *J. Global Optim.*, 2:379 – 410, 1992.

F. Tardella. On a class of functions attaining their maximum at the vertices of a polyhedron. *Discret. Appl. Math.*, 22:191–195, 1988/89.

F. Tardella. On the existence of polyhedral convex envelopes. In C. A. Floudas and P. M. Pardalos, editors, *Frontiers in Global Optimization*, pages 563–573. Kluwer Academic Publishers, 2003.

F. Tardella. Existence and sum decomposition of vertex polyhedral convex envelopes. *Optim. Lett.*, 2: 363–375, 2008.

M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Applications, Software, and Applications*. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

J. P. Vielma and G. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Math. Program.*, 2010. In Press (DOI: 10.1007/s10107-009-0295-4).

J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Oper. Res.*, 58(2):303 – 315, 2010a.

J. P. Vielma, S. Ahmed, and G. Nemhauser. A note on “A superior representation method for piecewise linear functions”. *INFORMS J. Comput.*, 22(3):493 – 497, 2010b.

V. Visweswaran. MINLP: Applications in blending and pooling. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2114 – 2121. Springer Science, 2 edition, 2009.

V. Visweswaran and C. A. Floudas. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs: II. application of theory and test problems. *Comput. Chem. Eng.*, 14(12):1419 – 1434, 1990.

V. Visweswaran and C. A. Floudas. New properties and computational improvement of the GOP algorithm for problems with quadratic objective functions and constraints. *J. Global Optim.*, 3:439 – 462, 1993.

D. S. Wicaksono and I. A. Karimi. Piecewise MILP under-and overestimators for global optimization of bilinear programs. *AIChE J.*, 54(4):991 – 1008, 2008.

A Linearly and Logarithmically Dependent Relaxations Applied to the Pooling Problem

This appendix summarizes the two relaxation schemes presented in Sections 4.2.1 & 4.2.2 for the bilinear term $p_{l,k} \cdot y_{l,j}$ that appears in the *p*-formulation of the standard pooling problem. We assume that the $p_{l,k}$ variables are partitioned and $a_{l,k} = \frac{p^U_{l,k} - p^L_{l,k}}{N_P}$.

A.1 Linearly Dependent Relaxations

The linear relaxation scheme (§4.2.1) has $\lambda_{l,k} \in \{0, 1\}^{N_P}$ and $\Delta y_{l,j,k} \in \left[0,\, y^U_{l,j} - y^L_{l,j}\right]^{N_P}$:

$$\sum_{n_P=1}^{N_P} \lambda_{l,k}(n_P) = 1 \quad \forall\, l,\, k \tag{17a}$$

$$p^L_{l,k} + \sum_{n_P=1}^{N_P} a_{l,k} \cdot (n_P - 1) \cdot \lambda_{l,k}(n_P) \le p_{l,k} \le p^L_{l,k} + \sum_{n_P=1}^{N_P} a_{l,k} \cdot n_P \cdot \lambda_{l,k}(n_P) \quad \forall\, l,\, k \tag{17b}$$

$$y_{l,j} = y^L_{l,j} + \sum_{n_P=1}^{N_P} \Delta y_{l,j,k}(n_P) \quad \forall\, (l,\, j) \in T_Y,\, k \tag{17c}$$

$$0 \le \Delta y_{l,j,k}(n_P) \le (y^U_{l,j} - y^L_{l,j}) \cdot \lambda_{l,k}(n_P) \quad \forall\, (l,\, j) \in T_Y,\, k,\, n_P \in \{1,\, \dots,\, N_P\} \tag{17d}$$

$$z_{l,j,k} \ge p_{l,k} \cdot y^L_{l,j} + \sum_{n_P=1}^{N_P} \left[p^L_{l,k} + a_{l,k} \cdot (n_P - 1)\right] \cdot\ \Delta y_{l,j,k}(n_P) \quad \forall\, (l,\, j) \in T_Y,\, k \tag{17e}$$

$$z_{l,j,k} \ge p_{l,k} \cdot y^U_{l,j} + \sum_{n_P=1}^{N_P} \begin{array}{l} \left[p^L_{l,k} + a_{l,k} \cdot\ n_P \qquad\right] \cdot \\ \left[\Delta y_{l,j,k}(n_P) - (y^U_{l,j} - y^L_{l,j}) \cdot \lambda_{l,k}(n_P)\right] \end{array} \quad \forall\, (l,\, j) \in T_Y,\, k \tag{17f}$$

$$z_{l,j,k} \le p_{l,k} \cdot y^L_{l,j} + \sum_{n_P=1}^{N_P} \left[p^L_{l,k} + a_{l,k} \cdot\ n_P \qquad\right] \cdot\ \Delta y_{l,j,k}(n_P) \quad \forall\, (l,\, j) \in T_Y,\, k \tag{17g}$$

$$z_{l,j,k} \le p_{l,k} \cdot y^U_{l,j} + \sum_{n_P=1}^{N_P} \begin{array}{l} \left[p^L_{l,k} + a_{l,k} \cdot (n_P - 1)\right] \cdot \\ \left[\Delta y_{l,j,k}(n_P) - (y^U_{l,j} - y^L_{l,j}) \cdot \lambda_{l,k}(n_P)\right] \end{array} \quad \forall\, (l,\, j) \in T_Y,\, k \tag{17h}$$

$$p^L_{l,k} \le p_{l,k} \le p^U_{l,k}; \quad y^L_{l,j} \le y_{l,j} \le y^U_{l,j} \tag{17i}$$

A.2 Logarithmically Dependent Relaxations

The logarithmic relaxation scheme (§4.2.2) has $N_L = \lceil \log_2 N_P \rceil$, $\lambda_{l,k} \in \{0,\, 1\}^{N_L}$, $\Delta y_{l,j,k} \in \left[0,\, y^U_{l,j} - y^L_{l,j}\right]^{N_L}$, and $s_{l,j,k} \in \left[0,\, y^U_{l,j} - y^L_{l,j}\right]^{N_L}$. The equation marked with a dagger (†) is only used when N_P is not a power of two.

$$p^L_{l,k} + \sum_{n_L=1}^{N_L} 2^{n_L - 1} \cdot a_{l,k} \cdot \lambda_{l,k}(n_L) \le p_{l,k} \le p^L_{l,k} + a_{l,k} + \sum_{n_L=1}^{N_L} 2^{n_L - 1} \cdot a_{l,k} \cdot \lambda_{l,k}(n_L) \quad \forall\, l,\, k \tag{18a}$$

$$p^L_{l,k} + a_{l,k} + \sum_{n_L=1}^{N_L} 2^{n_L - 1} \cdot a_{l,k} \cdot \lambda_{l,k}(n_L) \le p^U_{l,k} \quad \forall\, l,\, k \tag{18a†}$$

$$\Delta y_{l,j,k}(n_L) \le (y^U_{l,j} - y^L_{l,j}) \cdot \lambda_{l,k}(n_L) \quad \forall\, (l,\, j) \in T_Y,\, k,\, n_L \in \{1,\, \dots,\, N_L\} \quad \forall\, l,\, k \tag{18b}$$

$$\Delta y_{l,j,k}(n_L) = (y_{l,j} - y^L_{l,j}) - s_{l,j,k}(n_L) \quad \forall\, (l,\, j) \in T_Y,\, k,\, n_L \in \{1,\, \dots,\, N_L\} \tag{18c}$$

$$s_{l,j,k}(n_L) \le (y^U_{l,j} - y^L_{l,j}) \cdot (1 - \lambda_{l,k}(n_L)) \quad \forall\, (l,\, j) \in T_Y,\, k,\, n_L \in \{1,\, \dots,\, N_L\} \tag{18d}$$

$$z_{l,j,k} \ge \begin{array}{l} p_{l,k} \cdot y^L_{l,j} + \ p^L_{l,k} \qquad \cdot (y_{l,j} - y^L_{l,j}) + \\ \left[\sum_{n_L=1}^{N_L} a_{l,k} \cdot 2^{n_L - 1} \cdot \Delta y_{l,j,k}(n_L)\right] \end{array} \quad \forall\, (l,\, j) \in T_Y,\, k \tag{18e}$$

$$z_{l,j,k} \geq \; p_{l,k} \cdot y^U_{l,j} + (p^L_{l,k} + a_{l,k}) \cdot (y_{l,j} - y^U_{l,j}) + \left[\sum_{n_L=1}^{N_L} a_{l,k} \cdot 2^{n_L - 1} \cdot \Big(\Delta y_{l,j,k}(n_L) - (y^U_{l,j} - y^L_{l,j}) \cdot \lambda_{l,k}(n_L) \Big) \right] \quad \forall\, (l,\, j) \in T_Y,\, k \tag{18f}$$

$$z_{l,j,k} \leq \; p_{l,k} \cdot y^L_{l,j} + (p^L_{l,k} + a_{l,k}) \cdot (y_{l,j} - y^L_{l,j}) + \left[\sum_{n_L=1}^{N_L} a_{l,k} \cdot 2^{n_L - 1} \cdot \Delta y_{l,j,k}(n_L) \right] \quad \forall\, (l,\, j) \in T_Y,\, k \tag{18g}$$

$$z_{l,j,k} \leq \; p_{l,k} \cdot y^U_{l,j} + \; p^L_{l,k} \cdot (y_{l,j} - y^U_{l,j}) + \left[\sum_{n_L=1}^{N_L} a_{l,k} \cdot 2^{n_L - 1} \cdot \Big(\Delta y_{l,j,k}(n_L) - (y^U_{l,j} - y^L_{l,j}) \cdot \lambda_{l,k}(n_L) \Big) \right] \quad \forall\, (l,\, j) \in T_Y,\, k \tag{18h}$$

$$p^L_{l,k} \leq p_{l,k} \leq p^U_{l,k}; \quad y^L_{l,j} \leq y_{l,j} \leq y^U_{l,j} \tag{18i}$$