Valid Inequalities for the Pooling Problem with Binary Variables

Claudia D'Ambrosio
Jeff Linderoth
James Luedtke

Technical Report #1682

November 2010

Valid Inequalities for the Pooling Problem with Binary Variables

Claudia D'Ambrosio, Jeff Linderoth, James Luedtke[*]

November 15, 2010

Abstract

The pooling problem consists of finding the optimal quantity of final products to obtain by blending different compositions of raw materials in pools. Bilinear terms are required to model the quality of products in the pools, making the pooling problem a non-convex continuous optimization problem. In this paper we study a generalization of the standard pooling problem where binary variables are used to model fixed costs associated with using a raw material in a pool. We derive four classes of strong valid inequalities for the problem and demonstrate that the inequalities dominate classic flow cover inequalities. The inequalities can be separated in polynomial time. Computational results are reported that demonstrate the utility of the inequalities when used in a global optimization solver.

[*]Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Mechanical Engineering Building, 1513 University Ave., Madison, WI 53706, USA, {cdambrosio;linderoth;jrluedt1}@wisc.edu

1 Introduction

The pooling problem is to optimize the net cost of products whose composition is determined by blending different raw materials. The blending network consists of three types of nodes: the input streams, representing the raw materials; the pools, where the materials are blended; and the output streams, representing the final products. By deciding the flow quantity passing through the arcs of the this network, the composition of the final products is determined.

There are many applications areas for the pooling problem, including petroleum refining, wastewater treatment, and general chemical engineering process design [2, 12, 1, 4]. Different variants of the pooling problem have been introduced in the literature. In the *standard pooling problem*, the topology of the network is fixed—the pools are fed only by the input streams and connected only to the output streams. The standard pooling problem may be modeled as a non-convex nonlinear program (NLP), where the nonlinearities are bilinear terms that are present to model the (linear) blending process that occurs. The *generalized pooling problem* involves discrete decisions, where the activation of arcs connecting different nodes of the network is to be decided. This problem can be modeled as a non-convex Mixed Integer Nonlinear Program (MINLP). In the *extended pooling problem*, the Environmental Protection Agency limits on complex emissions are considered and modeled. In this case, extra variables are introduced and the additional constraints are non-smooth. In many applications of the pooling problem finding the (certified) global optimum can result in significant monetary savings, so a significant research effort has been undertaken on global optimization approaches for the problem. For an overview of the pooling problem variants and the optimization techniques that have been applied successfully for solving such problems, the reader is referred to the recent and exhaustive survey [9].

The focus of this paper is on a special case of the generalized pooling problem, where the topology of a portion of the network is also to be decided. Specifically, there are yes-no decisions associated with connecting the input streams and the pools. For example, this variant of the pooling problem is relevant for crude oil scheduling operations [6, 14], where the input streams represent supply ships feeding the storage tanks of a refinery.

Starting from a mathematical formulation of the pooling problem called the PQ-formulation, we detect an interesting set X for which we derive valid inequalities. The set X is a generalization of the well-known single-node fixed-charge flow set [10]. We introduce three classes of inequalities for this set that we call *quality inequalities*, because they are based on the quality target for a final product. A final class of valid inequalities introduced are called *pooling flow cover inequalities*, as they are related to, but dominate, standard flow cover inequalities for the single-node fixed-charge flow set. There are exponentially many pooling flow cover inequalities, and we demonstrate that they can be separated in polynomial time.

The present work is one of the first attempts to generate valid inequalities for a variant of the standard pooling problem that uses specific structures present in the problem. Other approaches have applied general-purpose convexification techniques, such as the reformulation-linearization technique [8] or disjunctive programming [5]. Simultaneously (and independently), Papageorgiou *et al.* [11] take an approach related to ours. Specifically, they study a polyhedral set that serves as a relaxation to a blending problem with fixed-charge structure. The focus of [11] is on developing inequalities for the single product, uncapacitated case, and they are able to find facet-defining inequalities that are useful in computation.

Our new valid inequalities were validated to be quite useful computationally both on instances available in the open literature and on randomly generated instances. Inequalities generated at the root node of a branch-and-bound tree were added to a standard model for the problem and given to the state-of-the-art global optimization solver BARON. With the strengthened model on a test suite of 76 instances solvable by BARON in 2 hours, adding the inequalities resulted reducing BARON's CPU time by a factor 2. An interesting observation from this study is that these useful valid inequalities for this mixed-integer nonconvex set were derived by studying a mixed-integer *linear* relaxation of the set. This suggests that it may be a useful approach in general to study mixed-integer linear relaxations of global optimization problems.

The remainder of the paper is divided into three sections. Section 2 gives a mathematical description of the problem we study. Section 3 describes the classes of inequalities we have derived, and Section 4 gives the computational results.

2 The Pooling Problem

In this section, we introduce our variant of the standard pooling problem, in which a portion of the topology of the network must be decided. In this variant, a fixed cost is paid if an arc connecting an input stream to a pool is utilized.

2.1 Mathematical formulation

Sahinidis and Tawarmalani [13] introduced the *PQ-formulation* for the standard pooling problem and demonstrated that it provided a tighter relaxation when the standard McCormick [7] approximation is used to relax the bilinear terms and obtain a Mixed Integer Linear Programming (MILP) problem. The PQ-formulation is used as the starting point for our model.

2.1.1 Notation

Input to the pooling problem consists of a network $G = (N, A)$, where the nodes are partitioned into three sets $N = I \cup L \cup J$. The set I is the set of the input streams, the nodes representing the raw materials; the set L is the pool set, where the raw materials are blended; and J is the set of the output streams, the nodes representing the final products. For ease of notation, we will assume that there is a complete interconnection network between the nodes of I and L and between the nodes of L and J. That is, $A = \{(i,l) \mid i \in I, l \in L\} \cup \{(l,j) \mid l \in L, j \in J\}$. The inequalities we derive later can easily be generalized to sparse networks. K is a set of input and output attributes.

Each input stream $i \in I$ has an associated unit cost c_i and availability A_i. Each output stream $j \in J$ has an associated unit revenue d_j and demand bound D_j. Each pool $l \in L$ has a size capacity S_l. The parameters C_{ik} denote the level of attribute $k \in K$ found in input stream $i \in I$. For each output stream $j \in J$, P^U_{jk} is the upper bound on the composition range of attribute $k \in K$ in the final product j. Finally, there are the parameters f_{il}, which represent the fixed cost that has to be paid if the arc from input stream $i \in I$ to pool $l \in L$ is used.

Decision variables in the formulation are the following:

- q_{il}: proportion of flow from input $i \in I$ to pool $l \in L$.
- y_{lj}: flow from intermediate pool node $l \in L$ to output $j \in J$.
- v_{il}: binary variable with value 1 if the arc from input $i \in I$ to pool $l \in L$ is used, 0 otherwise.
- w_{ilj}: flow from input $i \in I$ to output $j \in J$ through pool $l \in L$.

2.1.2 The PQ-formulation

The objective is to maximize net profit:

$$\min \sum_{j \in J} \left(\sum_{l \in L} \sum_{i \in I} c_i w_{ilj} - \sum_{l \in L} d_j y_{lj} \right) + \sum_{i \in I} \sum_{l \in L} f_{il} v_{il}.$$

There are simple constraints that bound raw material availability, pool capacity, and final product demand:

$$\sum_{l \in L} \sum_{j \in J} w_{ilj} \le A_i \quad \forall i \in I; \qquad \sum_{j \in J} y_{lj} \le S_l \quad \forall l \in L; \qquad \sum_{l \in L} y_{lj} \le D_j \quad \forall j \in J.$$

A distinguishing feature of the pooling problem is that there is an upper bound on the target value for each attribute of each product:

$$\sum_{l \in L} \sum_{i \in I} C_{ik} w_{ilj} \le P^U_{jk} \sum_{l \in L} y_{lj} \quad \forall j \in J, \forall k \in K. \tag{1}$$

The q variables must satisfy properties of proportion and only be positive if the associated arc was opened:

$$\sum_{i \in I} q_{il} \le 1 \quad \forall l \in L; \qquad 0 \le q_{il} \le v_{il} \quad \forall i \in I, \forall l \in L.$$

The w variables are related to the other decision variables as

$$w_{ilj} = q_{il} y_{lj} \quad \forall i \in I, \forall l \in L, \forall j \in J. \tag{2}$$

Finally, in the PQ-formulation, two redundant sets of constraints are added to the formulation to make subsequent relaxations stronger:

$$\sum_{i \in I} w_{ilj} = y_{lj} \quad \forall l \in L, \forall j \in J; \qquad \sum_{j \in J} w_{ilj} \le q_{il} S_l \quad \forall i \in I, \forall l \in L.$$

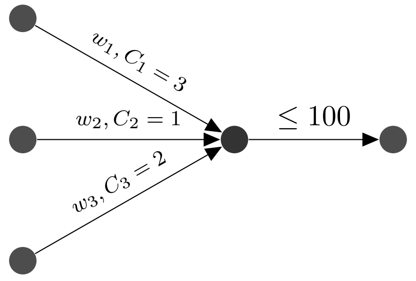
This formulation, augmented with appropriate bounds on the variables is given to the global optimization solver BARON in our subsequent computational experiments. Let $Y_{lj} \overset{\text{def}}{=} \min\{S_l, D_j, \sum_{i \in I} A_i\}$ be an upper bound on the flow on from $l \in L$ to $j \in J$. By replacing the nonlinear equations (2) with the well-known McCormick inequalities [7], a linear relaxation of the problem is formed. The McCormick inequalities in this context reduce to the following:

$$0 \le w_{ilj} \le q_{il} Y_{lj}; \qquad w_{ilj} \le y_{lj}; \qquad w_{ilj} \ge Y_{lj} q_{il} + y_{lj} - Y_{lj} \quad \forall i \in I, \forall l \in L, \forall j \in J.$$

Branching on the continuous variables q and y, as well as the binary variables v is required in order to converge to a globally optimal solution.

2.2 Example

Figure 1: Pooling problem example.



To demonstrate the valid inequalities we derive, we will use the following simple example shown in Figure 1. There is a single pool, a single product, and a single attribute ($|L| = |J| = |K| = 1$). The three input streams have input quality $C_1 = 3$, $C_2 = 1$, $C_3 = 2$, and there is an upper bound of $P^U = 2.5$ on the target quality. There is an upper bound of $Y = 100$ on the final product.

3 Valid inequalities

In this section, we extract an appropriate subset of the constraints of the formulation presented in Section 2 and derive a number of strong valid inequalities for this relaxation. To that end, we focus on a single output stream and a single attribute and define the sets

$$I^+ = \{i \in I \mid C_i - P^U \geq 0\}, \qquad \text{and } I^- = \{i \in I \mid C_i - P^U < 0\},$$

where we have dropped the irrelevant indices on the parameters. Next, we define $\alpha_i = |C_i - P^U| \; \forall i \in I$, substitute into equation (1) and use the fact that $y_l = \sum_{i \in I} w_{il}$, to extract the set

$$X = \Big\{(w, q, v) \in \mathbb{R}^{2|I||L|} \times \{0,1\}^{|I||L|} \mid \sum_{l \in L} \sum_{i \in I^+} \alpha_i w_{il} - \sum_{l \in L} \sum_{i \in I^-} \alpha_i w_{il} \leq 0;$$
$$\sum_{i \in I} q_{il} \leq 1 \; \forall l \in L; \quad w_{il} \leq Y_l q_{il}, \quad q_{il} \leq v_{il}, \qquad \forall i \in I, \forall l \in L\Big\},$$

which is a relaxation of the set of feasible solutions to the pooling problem presented in Section 2. The set X is composed of a single-node flow constraint, upper bounds on the flow variables based on the proportions q (coming from the McCormick inequalities), variable upper bounds on the proportion variables q, and the definition equations on the proportion variables q.

3.1 Quality inequalities

We define the following two classes of inequalities $\forall i \in I^+, \forall i^* \in I^-, \forall l \in L$:

$$\frac{\alpha_1}{\alpha_{i^*}} w_{il} - \sum_{i' \in I^-_>(\alpha_{i^*})} \left(\frac{\alpha_1}{\alpha_{i^*}} - 1\right) Y_l (v_{i'l} - q_{i'l}) - \frac{\alpha_1}{\alpha_i} Y_l (v_{il} - q_{il}) - \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq 0 \tag{3}$$

$$\alpha_i w_{il} - \sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} - \alpha_{i^*}) w_{i'l} - \alpha_{i^*} Y_l (v_{il} - q_{il}) - \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \leq 0 \tag{4}$$

where $\alpha_1 = \max_{i' \in I^-}\{\alpha_{i'}\}$ and $I^-_>(\alpha_{i^*}) = \{i \in I^- \mid \alpha_i > \alpha_{i^*}\}$.

Proposition 3.1. *Inequality* (3) *is valid for X $\forall i \in I^+, \forall i^* \in I^-, \forall l \in L$.*

Proof. Case $v_{il} = 0$: (3) becomes $\sum_{i' \in I^-_>(\alpha_{i^*})} \left(\frac{\alpha_1}{\alpha_{i^*}} - 1\right) Y_l(v_{i'l} - q_{i'l}) + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \geq 0$.
Case $\sum_{i' \in I^-_>(\alpha_{i^*})} v_{i'l} = 0$: it reduces to

$$\begin{aligned}
\alpha_i w_{il} &\leq \alpha_{i^*} Y_l(1 - q_{il}) + \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} = \alpha_{i^*} Y_l(\sum_{i' \in I^-} q_{i'l} + \sum_{i' \in I^+ : i' \neq i} q_{i'l}) + \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \\
\alpha_i w_{il} &\leq \alpha_{i^*} Y_l \sum_{i' \in I^- \setminus I^-_>(\alpha_{i^*})} q_{i'l} + \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}, \text{ that is always valid.}
\end{aligned}$$

Case $\sum_{i' \in I^-_>(\alpha_{i^*})} v_{i'l} > 0$:

$$\begin{aligned}
\frac{\alpha_1}{\alpha_{i^*}} w_{il} - \left(\frac{\alpha_1}{\alpha_{i^*}} - 1\right) Y_l(1 - \sum_{i' \in I^-_>(\alpha_{i^*})} q_{i'l}) &\leq \frac{\alpha_1}{\alpha_i} Y_l(1 - q_{il}) + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \\
\frac{\alpha_1}{\alpha_{i^*}} w_{il} - \left(\frac{\alpha_1}{\alpha_{i^*}} - 1\right) Y_l q_{il} &\leq \frac{\alpha_1}{\alpha_i} Y_l \sum_{i' \in I^-} q_{i'l} + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \\
Y_l q_{il} &\leq \frac{\alpha_1}{\alpha_i} Y_l \sum_{i' \in I^-} q_{i'l} + \frac{\alpha_1}{\alpha_i \alpha_{i^*}} \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \\
\alpha_i Y_l q_{il} - \alpha_1 Y_l \sum_{i' \in I^-} q_{i'l} - \frac{\alpha_1}{\alpha_{i^*}} \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} &\leq 0, \text{ always valid because weaker than} \\
\sum_{i' \in I^+} \alpha_{i'} w_{i'l} - \sum_{i' \in I^-} \alpha_{i'} w_{i'l} - \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} &\leq 0.
\end{aligned}$$

□

Proposition 3.2. *Inequality* (4) *is valid for* X $\forall i \in I^+, \forall i^* \in I^-, \forall l \in L$.

Proof. Case $v_{il} = 0$: it reduces to $\sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} - \alpha_{i^*}) w_{i'l} + \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} \geq 0$.
Otherwise

$$\begin{aligned}
\alpha_i w_{il} - \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} &\leq \sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} - \alpha_{i^*}) w_{i'l} + \alpha_{i^*} Y_l(1 - q_{il}) \\
\alpha_i w_{il} - \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} &\leq \sum_{i' \in I^-_>(\alpha_{i^*})} (\alpha_{i'} w_{i'l} + \alpha_{i^*}(Y_l q_{i'l} - w_{i'l})) + \alpha_{i^*} Y_l \sum_{i' \in I^- \setminus I^-_>(\alpha_{i^*})} q_{i'l} \\
\alpha_i w_{il} - \sum_{l' \neq l} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'} &\leq \sum_{i' \in I^-} \alpha_{i'} w_{i'l}
\end{aligned}$$

□

In order to make the definition of inequalities (3) and (4) more clear, consider the example of Section 2.2. For the example, $\alpha_1 = 0.5$, $\alpha_2 = 1.5$, $\alpha_3 = 0.5$ and $I^+ = \{1\}$, $I^- = \{2, 3\}$. The inequalities (3) and (4), defined for indices $i = 1$, $i^* = 3$, respectively, are

$$\begin{aligned}
3w_1 - 200(v_2 - q_2) - 300(v_1 - q_1) &\leq 0 \qquad (5) \\
0.5w_1 - w_2 - 50(v_1 - q_1) &\leq 0. \qquad (6)
\end{aligned}$$

A final valid inequality limits the flow from inputs that exceed the target quality.

Proposition 3.3. *The following valid inequality is valid for X $\forall i \in I^+, \forall l \in L$:*

$$\alpha_i w_{il} - Y_l \alpha_1 (v_{il} - q_{il}) - v_{il} \sum_{l' \neq l} Y_{l'} \alpha_1 \leq 0. \tag{7}$$

Proof. We consider two cases. Case $v_{il} = 0$: inequality (7) reduces to $0 \leq 0$. Otherwise, using the inequality $q_{il} \leq 1 - \sum_{i' \in I^-} q_{i'l}$ we see that (7) is weaker than $\alpha_i w_{il} - Y_l \alpha_1 \sum_{i' \in I^-} q_{i'l} - \sum_{l' \neq l} Y_{l'} \alpha_1 \leq 0$ which is valid because it is weaker than the valid inequality $\sum_{l' \in L} \sum_{i' \in I^+} \alpha_{i'} w_{i'l'} \leq \sum_{l' \in L} \sum_{i' \in I^-} \alpha_{i'} w_{i'l'}$. $\square$

Let us consider again the example of Section 2.2. Inequality (7) for $i = 1$ is

$$0.5 w_1 - 150(v_1 - q_1) - 0 \leq 0. \tag{8}$$

None of the three classes of inequalities introduced dominates the other, as in general, they can cut off different fractional solutions. Specifically, for the example problem,

- consider the solution $q' = (0.1; 0.9; 0)$, $y = 100$, $w' = (10; 90; 0)$, $v' = (0.15; 0.9; 0)$. The inequalities reduce to: $30 - 0 - 15 \leq 0$; $5 - 90 - 2.5 - 0 \leq 0$; $5 - 7.5 \leq 0$ and only the first inequality cuts off this infeasible solution.
- Consider the solution $q' = (0.5; 0.1; 0.4)$, $y = 100$, $w' = (50; 10; 40)$, $v' = (0.7; 0.9; 0.4)$. The inequalities reduce to: $150 - 160 - 60 \leq 0$; $25 - 10 - 10 \leq 0$; $25 - 30 \leq 0$ and only the second one excludes the fractional solution.
- Finally, consider the solution $q' = (0.5; 0.5; 0)$, $y = 100$, $w' = (50; 50; 0)$, $v' = (0.6; 1; 0)$. The inequalities reduce to: $150 - 100 - 30 \leq 0$; $25 - 50 - 5 \leq 0$; $25 - 15 \leq 0$ and only the last inequality cuts off the fractional solution.

3.2 Pooling flow cover inequalities

In this section, we focus on the case of a single pool, and hence for notational convenience drop the index l from the discussion. The primary result of the section is the generalization of a flow cover inequality:

Proposition 3.4. *Let $C^+ \subseteq I^+$ with $\lambda = \sum_{i \in C^+} \alpha_i Y > 0$, $S^- \subseteq I^-$, and define $u^*_{C^+} = \max_{i \in C^+} \alpha_i Y$. The following pooling flow cover inequality is valid for X:*

$$\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in S^-} [u^*_{C^+}(v_i - q_i)] - \sum_{i \in I^- \setminus S^-} \alpha_i w_i \leq 0. \tag{9}$$

Proof. Let $(w, q, v) \in X$ and define the set $T = \{i \mid v_i = 1\}$. If $|S^- \cap T| = 0$, inequality (9) reduces to $\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in I^- \setminus S^-} \alpha_i w_i \leq 0$ which is valid because $w_i = 0$ for all $i \in S^-$. Now suppose $|S^- \cap T| \geq 1$. Since $-\sum_{i \in I^- \setminus S^-} \alpha_i w_i \leq 0$ because $w_i \geq 0$ $\forall i \in I$, it is sufficient to prove that $\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in S^-} [u^*_{C^+}(v_i - q_i)] \leq 0$. First, observe that

$$\frac{1}{u^*_{C^+}} \left(\sum_{i \in C^+} \alpha_i w_i - \sum_{i \in S^-} [u^*_{C^+}(v_i - q_i)] \right) \leq \sum_{i \in C^+} \frac{\alpha_i w_i}{\alpha_i Y} - \sum_{i \in S^-} (v_i - q_i) \leq \sum_{i \in C^+} q_i - \sum_{i \in S^-} (v_i - q_i).$$

Thus, we will be done if we prove that $\sum_{i \in C^+} q_i - \sum_{i \in S^-} (v_i - q_i) \leq 0$, which follows from

$$\sum_{i \in C^+} q_i - \sum_{i \in S^- \cap T} (1 - q_i) = \sum_{i \in C^+} q_i + \sum_{i \in S^- \cap T} q_i - |S^- \cap T| \leq 0$$

since $|S^- \cap T| \geq 1$.

□

The next simple proposition shows that the inequalities (9) are stronger than the generalized flow cover inequalities (see [15]):

$$\sum_{i\in C^+} \alpha_i w_i - \sum_{i\in S^-} \lambda v_i - \sum_{i\in I^-\setminus S^-} \alpha_i w_i \leq 0. \tag{10}$$

Proposition 3.5. *Inequalities* (10) *are implied by* (9)*, for every $C^+ \subseteq I^+$ such that $\lambda = \sum_{i\in C^+} \alpha_i Y > 0$ and $S^- \subseteq I^-$.*

Proof. By definition, $u^*_{C^+} \leq \lambda$, and hence $-\sum_{i\in S^-} u^*_{C^+}(v_i - q_i) \geq -\sum_{i\in S^-} u^*_{C^+} v_i \geq -\sum_{i\in S^-} \lambda v_i$. □

Now let us consider the multiple-pool, one output stream and one attribute case. (Thus, we add index $l \in L$ back to the variables.)

Proposition 3.6. *For each pool $l \in L$, subset of inputs $S^- \subseteq I^-$, and cover $C^+ \subseteq I^+$, define $u^{*l}_{C^+} = \max_{i\in C^+} \alpha_i Y_l$. The following inequalities are valid for X:*

$$\sum_{i\in C^+} \alpha_i w_{il} - \sum_{i\in S^-} [u^{*l}_{C^+}(v_{il} - q_{il})] - \sum_{i\in I^-\setminus S^-} \alpha_i w_{il} - \sum_{i\in I^-} \sum_{l'\in L\setminus\{l\}} \alpha_i w_{il'} \leq 0 \tag{11}$$

Proof. The proof of 3.4 is valid also in this case. □

3.2.1 The separation problem

Given a fractional solution $(w^*, q^*, v^*) \notin X$ we want to find $C^+ \subseteq I^+$, $S^- \subseteq I^-$ and pool $l \in L$ such that (11) is violated. Let the maximum violation of such constraints be

$$z_{\text{SEP}} = \max_{C^+\subseteq I^+, l\in L} \sum_{i\in C^+} \alpha_i w^*_{il} - \sum_{i\in I^-} \min(u^*_{C^+}(v^*_{il} - q^*_{il}), \alpha_i w^*_{il}) - \sum_{i\in I^-} \sum_{l'\in L\setminus\{l\}} \alpha_i w^*_{il'}.$$

If $z_{\text{SEP}} > 0$, inequality (11) is violated for (C^+, S^-, l) with $S^- = \{i \in I^- | u^*_{C^+}(v^*_{il} - q^*_{il}) < \alpha_i w^*_{il}\}$. Note that the solution with the maximum violation has the following nice property: if $i \in C^+$, then $i' \in C^+$ $\forall i'$ such that $\alpha_{i'} \leq \alpha_i$. (This follows since, by the definition of $u^*_{C^+}$, the if $\alpha_{i'} \leq \alpha_i$, the inequality can only be made more violated by including i' in C^+.) Thus, the separation problem may be solved exactly in polynomial time by considering all the α_i ($i \in I^+$) in non-increasing order over all the pools. Algorithm 1 gives pseudocode for the separation algorithm.

4 Computational results

The utility of the quality and pooling flow cover inequalities for solving instances of our version of the generalized pooling problem was tested using a cut-and-branch approach. Models were created using the GAMS modeling language both for the original problem and for the continuous linear relaxation of the problem, wherein the nonlinear constraints $w_{ilj} = q_{il} y_{lj}$ are replaced by their McCormick envelopes (as described in Section 2.1) and the constraints $v_{il} \in \{0, 1\}$ are replaced with $0 \leq v_{il} \leq 1$. The continuous relaxation

Algorithm 1 Algorithm for solving the separation problem.

```
1: set $\hat{\sigma} = -\infty$, $\hat{c} = -\infty$, $\hat{l} = -\infty$;
2: order $\alpha_i$ such that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{|I^+|}$ $(i \in I^+)$;
3: for $c = 1, \ldots, |I^+|$ do
4:   for $l \in L$ do
5:     $\sigma = \sum_{i=1}^{c} \alpha_i w_{il}^* - \sum_{i \in I^-} \min(\alpha_c Y_l(v_{il}^* - q_{il}^*), \alpha_i w_{il}^*) - \sum_{i \in I^-} \sum_{l' \in L \setminus \{l\}} \alpha_i w_{il'}^*$;
6:     if $\sigma > \hat{\sigma}$ then
7:       set $\hat{\sigma} = \sigma$, $\hat{c} = c$ and $\hat{l} = l$;
8:     end if
9:   end for
10: end for
11: if $\hat{\sigma} > 0$ then
12:   add cut for $(C^+, S^-, l)$ with $C^+ = \{1, \ldots, \hat{c}\}$, $l = \hat{l}$ and $S^- = \{i \in I^- | u_{C^+}^*(v_{il}^* - q_{il}^*) < \alpha_i w_{il}^*\}$;
13: end if
```

is iteratively solved, where at each iteration, if the solution is fractional and the separation problems find violated inequalities, they are added to the linear relaxation, and the relaxation is resolved. The process repeats until the fractional solution can no longer be separated, after which all inequalities generated are added to the MINLP model. The augmented MINLP model is solved with BARON [13] using a CPU time limit of 2 hours. We compare against the performance of BARON on the model without adding the separated quality and pooling flow cover inequalities. Computational results were obtained on a heterogeneous cluster of computers. For each instance, the model was solved on the same machine both with and without cuts, so while the CPU times cannot be compared between instances, the *relative* times between the performance of BARON with and without cuts are comparable.

Our first test suite consisted of 11 instances from the literature, collected by Sahinidis and Tawarmalani [13]. For these instances, a fixed cost of 500 was added to every input arc. Some of these instances contain bypass arcs, or arcs that connect input streams directly to outputs are present. Bypass arcs are treated as additional pools with only one input in the inequalities. Results of the experiment are given in Table 1, where we report for each instance, the value of the global optimum, the number of nodes and the CPU time needed for BARON to find the optimum both with and without the addition of the violated quality and pooling flow cover inequalities, and the number of inequalities found. In general, the instances from the literature were far too small to draw any meaningful conclusions. The number of nodes is fewer in 4 of 11 cases, in 6 cases it is the same, and in one case, more nodes are taken after adding the inequalities.

A second test set consisted of 90 randomly generated instances of various sizes. The random instances were parameterized by the average graph density β, the number of inputs $|I|$, the number of pools $|L|$, the number of outputs $|J|$, and the number of attributes $|K|$, and named $\beta - |I| - |L| - |J| - |K|$ based on their size. For each combination of tested parameters, 10 instances were generated, and we report average performance results over all instances in the family. All the instances are available at the webpage `http://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm`. Results of this experiment are summarized in Table 2.

Without adding inequalities, BARON is able to solve 76 of the 90 instances. Adding the inequalities, BARON is able to solve three additional instances of the 90. Averages in Table 2 are taken only over the instances that both methods can solve. Complete results detailed the computational performance on specific instances are given on the web site `http://www.or.deis.unibo.it/research_pages/`

Table 1: BARON Performance With and Without Cutting Planes

		no cuts			with cuts	
	GO	# nodes	CPU time	# cuts	# nodes	CPU time
adhya1	630.0	7	0.14	20	7	0.14
adhya2	630.0	1	0.07	15	5	0.11
adhya3	978.0	17	0.43	8	1	0.12
adhya4	529.2	17	0.84	12	7	0.54
bental4	500.0	9	0.02	4	1	0.02
bental5	-2000.0	1	0.12	0	1	0.13
foulds2	266.7	1	0.04	6	1	0.02
haverly1	500.0	9	0.01	4	1	0.01
haverly2	400.0	9	0.01	4	9	0.01
haverly3	100.0	1	0.01	4	1	0.01
rt2	-1672.6	356	1.02	0	356	1.05

Table 2: Average Performance of BARON With and Without Cutting Planes

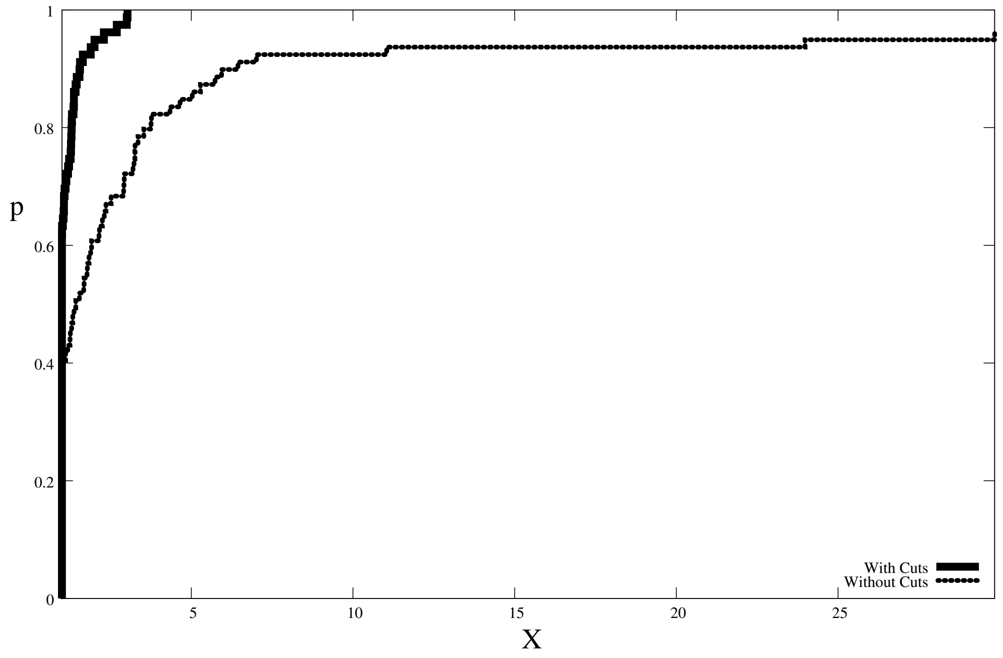
		No Cuts			With Cuts	
Instance Family	# Solved	Avg Nodes	Avg Time	Avg # Cuts	Avg Nodes	Avg Time
20-15-10-10-1	9	1052	14.9	11.1	383	7.3
20-15-10-10-2	10	7850	638.2	15.3	3338	440.3
20-15-10-10-4	10	2637	109.5	11.9	2241	168.5
30-15-5-10-1	9	22009	520.5	12.8	13095	367.4
30-15-5-10-2	8	7384	406.8	19.6	3988	239.0
30-15-5-10-4	10	6041	361.1	27.0	1884	109.9
30-15-8-10-1	3	21971	1689.6	11.7	6504	478.3
30-15-8-10-2	9	15663	823.9	19.7	4303	337.6
30-15-8-10-4	8	30424	1035.3	22.0	5472	457.2

ORinstances/ORinstances.htm.

For the 76 solved instances, the total CPU time required to solve the instances reduces from 39927 seconds to 20602 seconds when the cuts are added to the model before calling BARON. The total number of nodes required to solve the 76 instances reduces from 882173 to 329851 by adding cuts. The most significant performance improvement seems to occur on the instances that have 30% network density and 8 pools (in the 30-15-8-10-x family), indicating that larger, denser instances may benefit most from the additional of the quality and pooling flow cover inequalities.

A performance profile of the CPU time of the 79 instances solvable by either method is given in Figure 2. In the curves shown in Figure 2, the point (X, p) is on the graph for the associated method if a fraction p of the 79 instances solved by either method were solved within a factor X of the best of the two methods. For a more complete description of performance profiles, the reader is referred to [3]. From the results of the second experiment, it is clear that the addition of the inequalities has a significant beneficial impact on computational performance.

Figure 2: Performance Profile of Solution Time



Acknowledgement

The authors would like to thank Ahmet Keha for bringing reference [11] to their attention. This work benefitted from numerous insightful discussions with Andrew Miller.

References

[1] M. Bagajewicz. A review of recent design procedures for water networks in refineries and process plants. *Computers & Chemical Engineering*, 24:2093–2113, 2000.

[2] C.W. DeWitt, L.S. Lasdon, A.D. Waren, D.A. Brenner, and S. Melham. OMEGA: An improved gasoline blending system for Texaco. *Interfaces*, 19:85–101, 1989.

[3] Elizabeth Dolan and Jorge Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[4] J. Kallrath. Mixed integer optimization in the chemical process industry: Experience, potential and future perspectives. *Chemical Engineering Research and Design*, 78:809–822, 2000.

[5] R. Karuppiah and I.E. Grossmann. Global optimization for the synthesis of integrated water systems in chemical processes. *Computers & Chemical Engineering*, 30:650–673, 2006.

[6] H.M. Lee, J.M. Pinto, I.E. Grossmann, and S. Park. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. *Industrial & Engineering Chemistry Research*, 35:1630–1641, 1996.

[7] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part 1 - convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[8] C.A. Meyer and C.A. Floudas. Global optimization of a combinatorially complex generalized pooling problem. *AIChE Journal*, 52:1027–1037, 2006.

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

[10] M. Padberg, T. J. Van Roy, and L. Wolsey. Valid linear inequalities for fixed charge problems. *Operations Research*, 33:842–861, 1985.

[11] D. J. Papageorgiou, A. Toriello, G. L. Nemhauser, and M.W.P. Savelsbergh. Fixed-charge transportation with product blending. Unpublished manuscript.

[12] B. Rigby, L.S. Lasdon, and A.D. Waren. The evolution of Texaco's blending systems: From OMEGA to StarBlend. *Interfaces*, 25:64–83, 1995.

[13] N.V. Sahinidis and M. Tawarmalani. Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. *Journal of Global Optimization*, 32:259–280, 2005.

[14] N. Shah. Mathematical programming techniques for crude oil scheduling. *Computers & Chemical Engineering*, 20:S1227–S1232, 1996.

[15] L.A. Wolsey and G.L. Nemhauser. *Integer and Combinatorial Optimization*. Wiley, New York, NY, USA, 1988.