Noname manuscript No.
(will be inserted by the editor)

Relaxations and discretizations for the pooling problem

Akshay Gupte · Shabbir Ahmed · Santanu S. Dey · Myun Seok Cheon

Received: date / Accepted: date

Abstract The pooling problem is a folklore NP-hard global optimization problem that finds applications in industries such as petrochemical refining, wastewater treatment and mining. This paper assimilates the vast literature on this problem that is dispersed over different areas and gives unifying arguments and new insights on prevalent techniques. We also present new ideas for computing lower bounds on the global optimum by solving high-dimensional linear programs. Finally, we propose discretization methods for inner approximating the feasible region and obtaining good upper bounds. Valid inequalities are derived for the discretized models, which are formulated as mixed integer linear programs. The strength of our relaxations and usefulness of our discretizations is empirically validated on random test instances. We report best known upper bounds on some of the large-scale instances.

Keywords Pooling problem · Bilinear program · Lagrange relaxation · Reformulation Linearization Technique · Discretization

1 Introduction

The classical minimum cost network flow problem seeks to find the optimal way of sending raw materials from a set of suppliers to a set of customers via certain transshipment nodes in a directed capacitated network. The blending problem, which typically arises in refinery processes in the petroleum industry, is a type of minimum cost network flow problem with only two sets of nodes: suppliers and customers. The raw material at each supplier possesses different specifications, examples being concentrations of chemical compounds such as sulphur, carbon, or physical properties such as density, octane number. End products for the customers are created by directly mixing raw materials available from different suppliers. The mixing process should occur in a way such that the end products contain a certain minimum and maximum level of each specification. The objective is to minimize the total cost of producing demand.

The pooling problem, a generalization of the blending problem, combines features of both the classical network flow problem and the blending problem and can be stated in informal terms as follows: Given a list of available suppliers (inputs) with raw materials containing known specifications (specs), what is the minimum cost way of mixing these materials in intermediate tanks (pools) so as to meet the demand and spec requirements at multiple final blends (outputs)? Thus the raw materials are allowed to be first mixed in intermediate tanks referred to as pools and then sent forth from the pools to be mixed again at the output to form end products. It is also possible to send flow directly from inputs to the outputs. Figure 1 illustrates the pooling problem as a network flow problem over three sets of nodes: inputs, pools (or transshipment), and outputs. The inflows, outflows, and specification values at each pool are decision variables in the optimization model. Constraints that track specification level at each pool and that determine the level of spec

A. Gupte
Department of Mathematical Sciences, Clemson University
E-mail: agupte@clemson.edu

S. Ahmed, S. S. Dey
School of Industrial and Systems Engineering, Georgia Institute of Technology

M. S. Cheon
ExxonMobil Research and Engineering Company

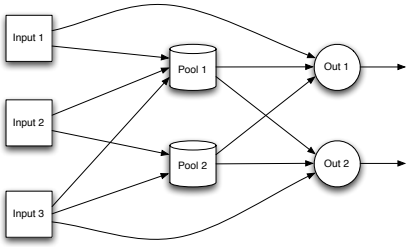


Fig. 1 A sample pooling problem

available at each output are formulated as bilinear constraints. As a result, the pooling problem is a bilinear program (BLP), which is a particular case of a nonconvex quadratic program with quadratic constraints (QCQP). In contrast, the classical blending problem, due to the absence of pools, can be formulated as a linear program (LP).

The pooling problem is a very important class of problems in the petrochemical industry [16]. The core problem features of pooling and blending appear in many different and important petrochemical optimization problems such as front-end scheduling, multi-period blending optimization, feedstock delivery scheduling with blending, and refinery planning problem. The front-end scheduling, also referred to as the crude oil operation scheduling by [38], is to find the optimal crude tank operation strategy. Crude tanks have two different roles; one is a storage place where crudes are stored and the other is a charging place where different crudes are mixed to meet specification requirements for the refining operations. The mixed crudes are discharged to a refinery unit. This optimization problem can be formulated as a mixed integer nonlinear programming problem (MINLP) where the mixed integer variables are required to represent the tank operating restrictions such as on/off or semicontinuous flows. The nonlinear terms, mainly bilinear terms that are exactly the same form as in the pooling problem, are required to keep track of the specification changes (or crude composition) in each tank. A similar problem structure can be observed for the final products and feedstock tank operations. The refinery planning problem refers to the short- or mid-term planning problem that is designed to answer the optimal process control decisions in order to maximize the profit of the complete system under a given cost structure for crudes and final products. The process control decisions include operating conditions for each unit such as temperatures and feed specifications as well as stream dispositions. The resulting mathematical model at some units, especially the splitters and mixers, is exactly same as the pooling problem. Furthermore, the refinery planning problem is often represented as a linear system with bilinear terms [14], again analogous to the pooling problem. Applications also exist in other fields of chemical engineering such as wastewater treatment [39], emissions regulation [26] and many others [37, 69].

Early efforts in solving the pooling problem were based on finding local optimal solutions using methods such as recursive LP [36], successive LP [9], and an adaptation of the generalized Benders' decomposition [22]. Sensitivity of local optimal solutions with respect to problem parameters was analyzed in [25, 28]. More recently, global optimization algorithms based on reformulation and spatial branch-and-bound [cf. 65] have been proposed by [7, 23, 58]. Studies in Lagrangian duality-based approaches were carried out by [1, 6, 13]. The state-of-the-art technique seems to be to solve the pooling problem using branch-and-cut algorithms developed for nonconvex MINLPs [12, 17] and which are well-implemented in global solvers such as `BARON`, `COUENNE`, `ANTIGONE`. A specialized branch-and-bound solver for pooling problems was implemented by [54].

Results on the pooling problem have been somewhat dispersed over different areas of optimization and chemical engineering. One of our main contributions is to gather this vast literature spread over many years and give unifying arguments. Furthermore, we provide new insights on folklore techniques and present some new ideas for relaxing the problem. Our third contribution is to empirically test discretization approaches for this problem. The paper can be divided into three main sections. In §2, we present various optimization models for the pooling problem, prove their equivalence to each other and compare their respective problem sizes. Complexity status of the problem is also discussed. The second part §3 deals with relaxations for the pooling problem. Our aim is to provide a comprehensive study of lower-bounding procedures and thereby extend the previous surveys found in [7, 30, 50, 67]. The strengths of these relaxations are analytically investigated. The third part §4 is about obtaining good upper bounds to the problem using discretization strategies for a general BLP. Additional binary variables are added to convert the discretized BLP into a mixed integer linear programming (MILP) problem. Different MILP models are obtained based on choice and representation of discretized variable. Through extensive computational experiments,

we test the viability of obtaining good feasible solutions to the pooling problem by solving a MILP approximation. Our empirical evidence demonstrates the effectiveness of this approach.

Henceforth, $\text{conv}(\cdot)$ denotes the convex hull of a set, $\mathbf{0}$ is a vector of zeros, $\Re$ is the set of reals and $\mathbb{Z}$ the set of integers. The orthogonal projection operator onto the subspace of χ variables is $\text{Proj}_{\chi}(\cdot)$. Frequently when encountering variables of the form χ_{ij}, we use $\chi_{i\cdot}$ to denote the vector of variables obtained by fixing the first index. Similarly for $x_{\cdot j}$. Cartesian product of a finite family of sets $X_1, \ldots, X_m$ is denoted by $\prod_{i=1}^{m} X_i$.

2 Problem Formulations

We use the same notation as that in [67, chap. 9]. Let $G = (\mathcal{N}, \mathcal{A})$ be a acyclic directed graph with $\mathcal{N} = I \cup L \cup J$ as the set of nodes and $\mathcal{A}$ as the set of arcs. Here I denotes the set of inputs, L the set of pools, and J the set of outputs. We assume that $\mathcal{A} \subseteq (I \times L) \cup (L \times L) \cup (L \times J) \cup (I \times J)$, i.e. there are no arcs between two inputs or two outputs and no backward arcs from pools to inputs or outputs to inputs or outputs to pools. Note that we have allowed the presence of pool-pool arcs in $\mathcal{A}$. Traditionally, problem instances with $\mathcal{A} \cap (L \times L) = \emptyset$ are referred to as *standard pooling problems*; otherwise they are *generalized pooling problems*. In this work, we mostly do not differentiate between these two cases since we wish to present a unified treatment for all problem classes. If the need arises to treat these two cases separately, then we explicitly state so. Since G is acyclic, there exists a subset $L_I := \{l \in L \colon (l', l) \notin \mathcal{A} \ \forall l' \in L \setminus l\}$ of pools with incoming arcs only from some input nodes.

For each $(i, j) \in \mathcal{A}$, let c_{ij} be the variable cost of sending a unit flow on this arc. For every $i \in \mathcal{N}$, let C_i be the capacity of this node. For a pool $l \in L$, its capacity C_l can be interpreted as the volumetric size of the pool tank, whereas for input $i \in I$, C_i is the total available supply and for output $j \in J$, C_j represents the maximum allowable flow. The upper bound on flow on arc (i, j) is denoted by u_{ij}. Typically, $u_{ij} = \min\{C_i, C_j\}$; however we allow the arcs in G to carry arbitrary upper bounds.

Let K denote the set of specifications that are tracked across the problem. For $i \in I$ and $k \in K$, λ_{ik} denotes the level of specification k in raw material at input i. Likewise, $\mu_{jk}^{\min}$ and $\mu_{jk}^{\max}$ are the lower and upper bound requirements on level of k at output j. We will mostly assume that $\lambda_{ik}, \mu_{jk}^{\min}, \mu_{jk}^{\max} \geq 0$ and $\sum_k \lambda_{ik}, \sum_k \mu_{jk}^{\min}, \sum_k \mu_{jk}^{\max} \in (0, 1]$. This assumption is consistent with the physical meaning of these parameters since the sums denote the total concentration of all specs within the flow available at a pool or output.

Let y_{ij} be the flow on arc $(i, j) \in \mathcal{A}$. For notational simplicity, we will always write equations using the flow variables y_{ij} with the understanding that y_{ij} is defined only for $(i, j) \in \mathcal{A}$. At each pool $l \in L$, the total amount of incoming flow must equal the total amount of outgoing flow.

$$\sum_{i \in I \cup L} y_{il} = \sum_{j \in L \cup J} y_{lj} \quad \forall l \in L. \tag{1}$$

The capacity constraints at each node in $\mathcal{G}$ are stated as

$$\sum_{j \in L \cup J} y_{ij} \leq C_i \quad \forall i \in I, \qquad \sum_{j \in L \cup J} y_{lj} \leq C_l \quad \forall l \in L, \qquad \sum_{i \in I \cup L} y_{ij} \leq C_j \quad \forall j \in J. \tag{2}$$

Finally, flows in $\mathcal{G}$ are bounded by individual arc capacities.

$$0 \leq y_{ij} \leq u_{ij} \quad \forall (i, j) \in \mathcal{A}. \tag{3}$$

Denote $\mathcal{F} := \{y \in \Re_+^{|\mathcal{A}|} \colon (1) - (3)\}$ as the polyhedral set that defines feasible flows on G.

2.1 Concentration model : p-formulation

In the pooling problem we send flows from inputs, mix them in pool tanks, and finally send the mixture from pools to outputs. Thus the mixtures in each pool and output carry specs whose concentration values, denoted by p_{jk} for $j \in L \cup J, k \in K$, can be determined as $p_{jk} = \frac{\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L} p_{lk} y_{lj}}{\sum_{i \in I \cup L} y_{ij}}$ if $\sum_{i \in I \cup L} y_{ij} > 0$ and equal to zero otherwise. Since $0 \leq \sum_{k \in K} \lambda_{ik} \leq 1$, it follows by recursion

that $0 \leq \sum_{k \in K} p_{jk} \leq 1$, for $j \in L \cup J, k \in K$. Observe that the above expression for p_{jk} can be equivalently rewritten in a bilinear form:

$$\sum_{i \in I} \lambda_{ik} y_{il} + \sum_{l' \in L} p_{l'k} y_{l'l} = p_{lk} \sum_{j \in L \cup J} y_{lj}, \quad \forall l \in L, k \in K \tag{4a}$$

$$\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L} p_{lk} y_{lj} = p_{jk} \sum_{i \in I \cup L} y_{ij}, \quad \forall j \in J, k \in K \tag{4b}$$

The bilinear equalities in (4) will be referred to as the *spec tracking constraints* since they help determine the concentration values of specs at each pool and output.

Note 1 We have assumed here that the mixing process follows linear blending, i.e. the total amount of spec at a node is simply the sum of product of spec concentration value and total flow on each input arc into this node. More general mixing processes where this assumption may not hold true are discussed in the survey of Misener and Floudas [50]. In this paper, we will make the assumption of linear blending at pools and outputs.

For every $l \in L$, let I_l denote the subset of inputs from which there exists a directed path to l in the graph G. The set I_j is defined similarly for every $j \in J$. Since all flows originate at the inputs and the pools themselves do not have any supply or demand of flow, it is straightforward to see that the level of specification at a pool l can be bounded as follows:

$$\min_{i \in I_l} \lambda_{ik} =: \leq p_{lk}^{\min} \leq p_{lk} \leq p_{lk}^{\max} := \max_{i \in I_l} \lambda_{ik} \quad \forall l \in L, k \in K. \tag{5a}$$

The above bounds are implied by the network structure and tracking constraints (4a) but nonetheless can be added to the problem formulation. Similarly we have

$$\min_{i \in I_j} \lambda_{ik} =: \leq p_{jk}^{\min} \leq p_{jk} \leq p_{jk}^{\max} := \max_{i \in I_j} \lambda_{ik} \quad \forall j \in J, k \in K. \tag{5b}$$

We are now ready to formally state the pooling problem.

Definition 1 (Pooling problem) Given any acyclic directed graph G and its attributes, find a minimum cost feasible flow $y \in \mathcal{F}$ such that there exist some concentration values $p \in \Re^{(|L|+|J|)\times|K|}$ that satisfy (4) and $\mu_{jk}^{\min} \leq p_{jk} \leq \mu_{jk}^{\max}$ for all $j \in J, k \in K$.

$$z^* = \min_{y,p} \sum_{(i,j) \in \mathcal{A}} c_{ij} y_{ij} \text{ s.t. } y \in \mathcal{F}, \text{ (4)}, \text{ (5a)}, \mu_{jk}^{\min} \leq p_{jk} \leq \mu_{jk}^{\max} \quad \forall j \in J, k \in K. \tag{Pooling}$$

For every $j \in J, k \in K$, we can combine the spec tracking constraints (4b) and spec level requirements $\mu_{jk}^{\min} \leq p_{jk} \leq \mu_{jk}^{\max}$ to give bilinear inequality constraints of the form

$$\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L} p_{lk} y_{lj} \leq \mu_{jk}^{\max} \sum_{i \in I \cup L} y_{ij} \quad \forall j \in J, k \in K \tag{6a}$$

$$\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L} p_{lk} y_{lj} \geq \mu_{jk}^{\min} \sum_{i \in I \cup L} y_{ij} \quad \forall j \in J, k \in K. \tag{6b}$$

Note that the spec tracking constraints corresponding to the pools are retained. Thus we have the following bilinear optimization model, introduced as the p-formulation in [36].

$$z^* = \min_{y,p} \sum_{(i,j) \in \mathcal{A}} c_{ij} y_{ij} \text{ s.t. } y \in \mathcal{F}, \text{ (4a)}, \text{ (5a)}, \text{ (6)}. \tag{$\mathbb{P}$}$$

We denote the p-formulation by $\mathbb{P}$. With a slight abuse of notation and for convenience, we will sometimes also refer to the feasible set of the p-formulation by $\mathbb{P}$. A few basic observations about optimizing the pooling problem are made below.

Observation 1 $z^* = \min_y \sum_{(i,j) \in \mathcal{A}} c_{ij} y_{ij}$ *s.t.* $y \in \text{Proj}_y \mathbb{P}$.

Observation 2 $z^* \leq 0$ *since* $\mathbf{0} \in \text{Proj}_y \mathbb{P}$.

Henceforth, for nontriviality, we assume that $z^* < 0$.

Observation 3 *For any $j \in J$ such that the following system of linear inequalities*

$$\mu_{jk}^{\min} \leq \sum_{i \in I_j} \lambda_{ik}\gamma_i \leq \mu_{jk}^{\max} \quad \forall k \in K,\ \sum_{i \in I_j} \gamma_i = 1,\ \gamma \geq \mathbf{0}$$

does not admit a solution in γ, the equality $\sum_{i \in I \cup L} y_{ij} = 0$ is valid to the pooling problem and hence output node j can be deleted from the graph.

Proof Based on the network structure, tracking constraints (4b) and bounds (5b), we know that $\sum_{i \in I \cup L} y_{ij} > 0$ implies $p_{j\cdot} \in \text{conv}(\cup_{i \in I_j} \lambda_{i\cdot})$. Hence if $\text{conv}(\cup_{i \in I_j} \lambda_{i\cdot}) \cap [\mu_{j\cdot}^{\min}, \mu_{j\cdot}^{\max}] = \emptyset$ we know that $\sum_{i \in I \cup L} y_{ij}$ must be equal to zero. This feasibility check is exactly the proposed linear system. □

Henceforth, we assume that for every output j, the linear system in Observation 3 is feasible.

Observation 4 *For any $j \in J, k \in K$ such that $p_{jk}^{\max} \leq \mu_{jk}^{\max}$ (resp. $p_{jk}^{\min} \geq \mu_{jk}^{\min}$), inequality* (6a) *(resp.* (6b)*) is redundant to the pooling problem.*

We say $\mu_{jk}^{\max}$ (resp. $\mu_{jk}^{\min}$) has a trivial value if $p_{jk}^{\max} \leq \mu_{jk}^{\max}$ (resp. $p_{jk}^{\min} \geq \mu_{jk}^{\min}$).

Observation 5 *In the absence of* (6)*, or equivalently if the values of $\mu_{jk}^{\min}$ and $\mu_{jk}^{\max}$ are trivial for all j, k, the pooling problem becomes separable across pools and is simply a capacitated network flow problem that can be solved as an LP.*

Henceforth, we assume that for any $j \in J$ and $k \in K$, at least one of $\mu_{jk}^{\min}$ and $\mu_{jk}^{\max}$ has a nontrivial value.

Observation 6 *If the out-degree of each pool is exactly one, i.e. $|\{j \in \mathcal{N} \colon (l, j) \in \mathcal{A}\}| = 1$ for all $l \in L$, then the pooling problem can be solved as an LP.*

Proof The sole purpose of having the p_{lk} variables in $\mathbb{P}$ is to enforce that all the outgoing arcs from pool l carry the same concentration value for spec k. Substitute a new variable w_{lkj} for bilinear terms $p_{lk}y_{lj}$ in (4a) and (6). If each pool has only one outgoing arc, we do not need to enforce the spec consistency constraint $w_{lkj} = p_{lk}y_{lj}$ and can eliminate the p variables. Thus the $\mathbb{P}$ formulation can be completely linearized in this special case and solved as a single LP. □

2.1.1 Complexity

In general, the pooling problem is a bilinear program which is a nonconvex problem and a generalization of the strongly NP-hard linear maxmin problem [8]. Recently, a formal proof was provided for its NP-hardness.

Proposition 1 (Alfaki and Haugland [5]) *The standard pooling problem with a single pool is NP-hard.*

Proof (Sketch of proof.) The proof is via a polynomial reduction from the maximum stable set problem, which is well-known to be NP-hard, to an instance of the standard pooling problem with $|L| = 1, |I| = |J| = n$, where $n = |\mathcal{V}|$ is the number of nodes in the graph $G' = (\mathcal{V}, \mathcal{E})$ for the stable set problem. The set of arcs is $\mathcal{A} = (I \times L) \cup (L \times J)$ with all arc capacities equal to 1. Arc costs are -1 for each arc from pool to output. The set of specs is $K = \{1, \ldots, 2n\}$ and the key idea is to define a suitable set of specification values: for any $i \in I$, $\lambda_{ii} = 1, \lambda_{i,n+i} = -1$, 0 otherwise; $\mu_{jk}^{\min} = -1\ \forall j, k$; and for any $j \in J$, $\mu_{j,n+j}^{\max} = -1/n$, $\mu_{jk}^{\max} = 1$ for $k = 1, \ldots, n$ such that $(j, k) \notin \mathcal{E}$, 0 otherwise. It is not difficult to show using the constructed values of $\lambda, \mu^{\min}, \mu^{\max}$ that for any feasible solution (p', y') of the pooling problem, the subset $\mathcal{V}' := \{v = 1, \ldots, n \colon y'_{lv} > 0\}$ is a stable set in G' of cardinality at least $\sum_{v=1}^{n} y'_{lv}$. □

Remark 1 The pooling instance constructed in Proposition 1 has redundant values for $\mu_{jk}^{\min}$ and negative values for some λ_{ik} and $\mu_{jk}^{\max}$. A similar reduction but with $\lambda_{ik}, \mu_{jk}^{\max} \geq 0$ and nontrivial values of $\mu_{jk}^{\min}$ was presented in Dey and Gupte [21].

We summarize several results related to complexity. The standard pooling problem with

1. a single pool and no direct arcs from inputs to outputs is equivalent to a 0/1 MILP [21, Appendix A] and is polynomially time solvable for fixed $|K|$ [5],
2. a single pool is polynomially time solvable for fixed $|J|$ [35],
3. a single pool is as hard to approximate as a stable set problem [21],
4. in-degree of each node at most 2 is NP-hard [35]. Similarly for out-degree at most 2.

2.2 Alternate formulations

2.2.1 Proportion model : q-formulation

The q-formulation was proposed by Ben-Tal et al. [13] for standard pooling problems wherein (6) is modeled using proportion variables q_{il} to denote the fraction of incoming flow to pool l that is contributed by input i. By definition, we have $\sum_{i\in I} q_{il} = 1$ for all $l \in L$ and $y_{il} = q_{il}\sum_{i'\in I} y_{i'l} = q_{il}\sum_{j\in L\cup J} y_{lj}$ for all $i \in I, l \in L$. Then we can eliminate the p variables from ($\mathbb{P}$) to obtain the so-called q-formulation. In fact, (4) implies that $p_{lk} = \sum_{i\in I}\lambda_{ik}q_{il} \;\; \forall l \in L, k \in K$. For generalized pooling problems, a straightforward extension of this idea will be to define proportion variables q_{il} for $l \in L, i \in I \cup L$. However, not only does this involve using $\mathcal{O}(|L|^2 + |I||L|)$ proportion variables, but we also introduce bilinear terms of the form $q_{il}q_{i'l'}$ thereby losing the disjoint bilinear structure (i.e. all nonlinearities being $q_{il}y_{lj}$). Instead, Alfaki and Haugland [4] proposed a q-formulation by defining q_{il} as the fraction of incoming flow to pool l that originated from some input i and not distinguishing between flows that started at i and reached l along different paths. This formulation has $\mathcal{O}(|I||L|)$ proportion variables. Also, all bilinear terms are of the form $q_{il}y_{lj}$ as explained next.

Let $q_{\cdot l}$ denote the vector $(q_{il})_{i\in I_l}$. We have

$$q_{\cdot l} \in \Delta_{|I_l|} := \{q_{\cdot l} \geq \mathbf{0} \colon \sum_{i\in I_l} q_{il} = 1\} \quad \forall l \in L, \tag{7}$$

Since we send flows from inputs to outputs via pools, we can create a super-sink node that connects to all outputs and consider each input $i \in I$ to be a unique commodity[1]. The flow of commodity i on arc (l, j) is given by $v_{ilj} = q_{il}y_{lj}$ for $l \in L, j \in L \cup J, i \in I_l$. In order to ensure flow balance of commodity i at pool l, we must add the constraint

$$y_{il} + \sum_{l'\in L: i\in I_{l'}} q_{il'}y_{l'l} \;=\; q_{il}\sum_{j\in L\cup J} y_{lj} \quad \forall l \in L, i \in I_l. \tag{8}$$

Observation 7 *Equations* (7) *and* (8) *render the flow balance constraint* (1) *redundant.*

For $l \in L_I$ and $i \in I_l$, equation (8) reads: $y_{il} = q_{il}\sum_j y_{lj}$.

The spec level requirement constraints at the output are modeled as

$$\sum_{i\in I}\lambda_{ik}y_{ij} + \sum_{l\in L, i\in I_l}\lambda_{ik}q_{il}y_{lj} \geq \mu^{\min}_{jk}\sum_{i\in I\cup L} y_{ij}, \quad \forall j \in J, k \in K. \tag{9a}$$

$$\sum_{i\in I}\lambda_{ik}y_{ij} + \sum_{l\in L, i\in I_l}\lambda_{ik}q_{il}y_{lj} \leq \mu^{\max}_{jk}\sum_{i\in I\cup L} y_{ij}, \quad \forall j \in J, k \in K \tag{9b}$$

The q-formulation for pooling problem can now be stated as follows.

$$z^* = \min_{y,p} \sum_{(i,j)\in\mathcal{A}} c_{ij}y_{ij} \;\text{ s.t. }\; y \in \mathcal{F},\; (7) - (9). \tag{$\mathbb{Q}$}$$

In the case of standard pooling problems, the above formulation reduces to the one proposed by [13]. Based on the above ideas, two new formulations for the standard pooling problem were developed in [5]: ($\mathbb{TP}$) that uses proportions of flows traveling from a pool l to an output j and ($\mathbb{STP}$) that combines the proportion variables from both ($\mathbb{Q}$) and ($\mathbb{TP}$) and consequently, has more variables and bilinear terms.

2.2.2 pq-formulation

The *pq*-formulation was introduced in [67, chap. 9] for standard pooling problems and is obtained by appending some valid inequalities to ($\mathbb{Q}$). These inequalities can be derived from the Reformulation Linearization Technique (RLT) [63] by multiplying (7) with y_{lj} and the pool capacity constraints in (2) with q_{il}, respectively. For both standard and generalized problem, the valid inequalities can be stated as follows:

$$\sum_{i\in I_l} q_{il}y_{lj} = y_{lj} \;\; \forall l \in L, j \in L \cup J, \qquad \sum_{j\in L\cup J} q_{il}y_{lj} \leq C_l q_{il} \;\; \forall l \in L, i \in I_l. \tag{10}$$

[1] For the p-formulation, specifications serve the role of commodities and (4a) is a commodity balance constraint.

Addition of (10) to ($\mathbb{Q}$) yields formulation ($\mathbb{PQ}$).

$$z^* = \min_{y,p} \sum_{(i,j)\in\mathcal{A}} c_{ij}y_{ij} \text{ s.t. } y \in \mathcal{F},\ (7)-(9),\ (10). \qquad (\mathbb{PQ})$$

In §3.2.1, we give an insight into the conventional wisdom behind ($\mathbb{PQ}$) being a strong formulation for the pooling problem and hence a formulation of choice for solving (standard) problem instances.

2.2.3 Hybrid formulation

Audet et al. [7] suggested a model that combined the p and q variables along with the y variables. The motivation was to avoid having bilinear terms of the form $q_{il}q_{jl'}$ that would arise by a straightforward extension of the Ben-Tal et al. model to the case of generalized pooling problems. In this so-called hybrid model, proportion variables are used for pools in L_I and concentration variables are used for pools in $L \setminus L_I$. We skip the details of this hybrid formulation ($\mathbb{HYB}$) since it can be easily obtained by combining the previous sections.

2.2.4 Equivalence of formulations

We now formally prove the correctness of the forgoing formulations for the pooling problem. Although the proof is straightforward for standard problems, a little bit of work is required for generalized problems where we have arcs between two pools and there may exist multiple paths from an input to a pool. This tedious exercise was skipped in [4] and is formally presented here for completeness. We say that two formulations are equivalent if for every feasible point in one formulation, there exists a feasible point with same objective value in the other formulation and vice versa.

Proposition 2 *Formulations* ($\mathbb{P}$), ($\mathbb{Q}$), ($\mathbb{PQ}$), *and* ($\mathbb{HYB}$) *are equivalent.*

Proof First let us show that for any feasible point (q, y) in ($\mathbb{Q}$) there exists some p such that (p, y) satisfies (4a) and (6). Observe that if we set $p_{lk} = \sum_{i\in I_l} \lambda_{ik}q_{il}$ for all $l \in L, k \in K$, then (9) implies (6). It remains to verify that with this choice of p, (8) implies (4a). Fix some $l \in L, k \in K$. For every $i \in I_l$, multiply both sides of (8) with λ_{ik}. Summing over $i \in I_l$ gives us $\sum_{i\in I_l} \lambda_{ik}y_{il} + \sum_{l'\in L} \sum_{i\in I_{l'}\cap I_l} \lambda_{ik}q_{il'}y_{l'l} = \left(\sum_{i\in I_l} \lambda_{ik}q_{il}\right) \sum_{j\in L\cup J} y_{lj}$. Since for any $(l^{'}, l) \in \mathcal{A} \cap L \times L$, we have $I_{l'} \subseteq I_l$ and hence $I_{l'} \cap I_l = I_{l'}$, the equality becomes $\sum_{i\in I_l} \lambda_{ik}y_{il} + \sum_{l'\in L} \left(\sum_{i\in I_{l'}} \lambda_{ik}q_{il'}\right) y_{l'l} = \left(\sum_{i\in I_l} \lambda_{ik}q_{il}\right) \sum_{i\in I\cup L} y_{il}$, where on the right hand side we have also used the flow balance equality (1). It follows that (4a) is satisfied by choosing $p_{lk} = \sum_{i\in I_l} \lambda_{ik}q_{il}$.

Now we show the converse, i.e. for any feasible point (p, y) in ($\mathbb{P}$), there exists some q satisfying (7), $p_{lk} = \sum_{i\in I_l} \lambda_{ik}q_{il}$ for all $l \in L, k \in K$, and such that (q, y) satisfies (8) and (9). As before, (6) $\Rightarrow$ (9) is obvious. Fix some $l \in L$. First suppose that $\sum_{i\in I\cup L} y_{il} = \sum_{j\in L\cup J} y_{lj} = 0$. Here (8) is trivially true. By problem definition, all flows originate with specification values only at the input nodes and since (1) balances total flow at each pool in G and we assume linear blending in (4a), it must be that $(p_{lk})_{k\in K} \in \text{conv}\{(\lambda_{ik})_{k\in K} : i \in I_l\}$. Thus, there exists a vector $(q_{il})_{i\in I_l}$ satisfying (7) and $p_{lk} = \sum_{i\in I_l} \lambda_{ik}q_{il}\ \ \forall k$. Henceforth assume that $\sum_{j\in L\cup J} y_{lj} > 0$. For $j \in L \cup J$, define $\xi_{lj} := y_{lj}/\sum_{j\in L\cup J} y_{lj}$ to be the fraction of outgoing flow from l directed towards j. Let T_{il} be the set of directed paths between $i \in I_l$ and l. Since G is acyclic, T_{il} is a finite set. Take a directed path $\tau := \{i, \tau_1, \ldots, \tau_r, l\} \in T_{il}$. Then the total flow from i that reaches l along path τ is equal to $\sigma^\tau_{il} := y_{i\tau_1}\xi_{\tau_l} \prod_{o=1}^{r-1} \xi_{\tau_o\tau_{o+1}}$. Construct the q variables as follows: $q_{il} = \frac{\sum_{\tau\in T_{il}} \sigma^\tau_{il}}{\sum_{i'\in I\cup L} y_{i'l}}$. The flow balance equations (1) imply that there is no supply at pools and all the supply originates at inputs. Hence, the quantity $\sum_{i\in I_l} \sum_{\tau\in T_{il}} \sigma^\tau_{il}$, which designates the total flow from all inputs to l must equal the total flow into l which is $\sum_{i'\in I\cup L} y_{i'l}$. Hence $\sum_{i\in I_l} q_{il} = 1$. Similarly, the total quantity of spec k at pool l is given by $\sum_{i\in I_l} \lambda_{ik} \sum_{\tau\in T_{il}} \sigma^\tau_{il}$ and hence $p_{lk} = \sum_{i\in I_l} \lambda_{ik}q_{il}$. Now the left hand side of (8) is

$$y_{il} + \sum_{l'\in L: i\in I_{l'}} \sum_{\tau\in T_{il'}} \sigma^\tau_{il'} \frac{y_{l'l}}{\sum_{i'\in I\cup L} y_{i'l'}} = y_{il} + \sum_{l'\in L: i\in I_{l'}} \sum_{\tau\in T_{il'}} \sigma^\tau_{il'}\xi_{l'l} = \sum_{\tau'\in T_{il}} \sigma^{\tau'}_{il} = q_{il} \sum_{j\in L\cup J} y_{lj}$$

where the second equality follows from the following observations: 1) y_{il} is defined if and only if $(i, l) \in \mathcal{A}$, 2) any non-direct path between i and l must pass through intermediate pools l' such that

$i \in I_{l'}$, (3) by construction of σ^τ, the quantity $\sigma^\tau_{il'}\xi_{l'l}$ denotes the total flow from i to l along the path $\tau' = \tau \cup (l', l) \in T_{il}$.

Thus, we have shown that ($\mathbb{P}$) and ($\mathbb{Q}$) are equivalent formulations of the pooling problem. The equivalence of ($\mathbb{Q}$) and ($\mathbb{PQ}$) is obvious due to the validity of (10). The correctness of ($\mathbb{HYB}$) can be shown using the above proof for pools in L_I. □

2.3 Problem sizes

The alternate formulations of §2.2 present different ways of modeling the p-formulation of the pooling problem obtained from Definition 1. All these equivalent formulations use the same flow variables on the arc set $\mathcal{A}$ and they only differ in the use of non-flow variables and additional constraints. Since bilinearities are responsible for making the problem hard to solve, Table 1 mentions the number of bilinear terms and bilinear constraints along with the number of non-flow variables.

Formulation	Non-flow variables	Bilinear terms	Bilinear constraints	
			Eq.	Ineq.
$\mathbb{P}$	$\|K\|\|L\|$	$\|K\|\|L\|\,\mathcal{O}(\|L\|+\|J\|)$	$\|K\|\|L\|$	$2\|K\|\|J\|$
$\mathbb{Q}$	$\sum_{l\in L}\|I_l\|$	$\sum_{l\in L}\|I_l\|\,\mathcal{O}(\|L\|+\|J\|)$	$\sum_{l\in L}\|I_l\|$	$2\|K\|\|J\|$
$\mathbb{PQ}$	$\sum_{l\in L}\|I_l\|$	$\sum_{l\in L}\|I_l\|\,\mathcal{O}(\|L\|+\|J\|)$	$\sum_{l\in L}\|I_l\| +$ $\|L\|\,\mathcal{O}(\|L\|+\|J\|)$	$2\|K\|\|J\|$ $+\sum_{l\in L}\|I_l\|$
$\mathbb{HYB}$	$\sum_{l\in L_I}\|I_l\|$ $+\|K\|\|L\setminus L_I\|$	$\left[\sum_{l\in L_I}\|I_l\| + \|K\|\|L\setminus L_I\|\right]$ $\times\ \mathcal{O}(\|L\setminus L_I\|+\|J\|)$	$\sum_{l\in L_I}\|I_l\| +$ $\|K\|\|L\setminus L_I\| +$ $\|L_I\|\,\mathcal{O}(\|L\setminus L_I\|+\|J\|)$	$2\|K\|\|J\|$ $+\sum_{l\in L_I}\|I_l\|$

Table 1 Comparing problem sizes for various formulations of the pooling problem.

2.4 Variants

We have already mentioned two types of pooling problems - standard and generalized, depending on the absence or presence of arcs between pools, respectively. There are many variants of these two basic types. A broader class of network flow problems with bilinear terms is described by [42, 58, 61]. Nonlinear blending rules have also been proposed, see [50] for a discussion and [59] for one specific example of nonlinear blending where the bilinear terms in the pooling problem are replaced by cubic terms. A variant of the standard problem, where total flow into each output is fixed to some nonzero constant, was studied by [62]. An extended pooling problem that imposes upper bounds on emissions from outputs was introduced in [52]. Other examples of MINLP models can be found in [20, 49, 51, 56, 69]. These MINLP variants arise mainly by including binary decision variables related to the use of each arc or node in the graph or forcing the flows to be semicontinuous. Pooling problems also find applications in the mining industry [15]. Stochastic versions of the standard problem that model uncertainty in the input specification levels λ's were proposed by [43, 44].

We describe one variant that arises after imposing finite time periods and inventory balance requirements at each node. A somewhat related model was considered in the blend scheduling problem of [41].

2.4.1 Time indexed pooling problems

Consider a generalized pooling problem and let T be a set of time periods. For each time period $t \in T$, we have to make the following decisions: 1) semicontinuous flow y_{ijt} on arc $(i, j) \in \mathcal{A}$, 2) s_{it} amounts of inventory to be held at a node $i \in \mathcal{N}$, 3) $x^{in}_{lt} = 1$ iff there is inflow at pool l, 4) $x^{out}_{lt} = 1$ iff there is outflow at pool l, 5) $z_{lt} = 1$ iff pool l is used for mixing. Some additional parameters are required for this model. Let a_{it} and d_{jt} be the supply at input $i \in I$ and demand at output $j \in J$, respectively, at time $t \in T$. Let h_l be the fixed cost of using a pool $l \in L$. The set of pools is partitioned into two categories - L_c and $L \setminus L_c$. A pool $l \in L_c$ is allowed to be leased on a contract

basis for a fixed period τ_l and can only be used under contract. Typically, $\tau_l \leq |T|$ and the contracts are renewable. For a pool $l \in L_c$, the fixed cost h_l is associated with the entire contract.

We first state the p-formulation ($\mathbb{P}$-Inv) of this problem. p_{lkt} denotes the concentration value of spec k at pool l at time t.

$$
\begin{aligned}
\min_{y,s,x,z} \ & \sum_{t\in T}\sum_{(i,j)\in A} c_{ij}y_{ijt} \ + \sum_{t\in T}\sum_{l\in L} h_l z_{lt} \\
& a_{it} + s_{i(t-1)} = \sum_{l\in L\cup J} y_{ilt} + s_{it} \quad \forall i \in I, t \in T \\
& \sum_{i\in I\cup L} y_{ilt} + s_{l(t-1)} = s_{lt} + \sum_{j\in L\cup J} y_{ljt} \quad \forall l \in L, t \in T \\
& \sum_{l\in I\cup L} y_{ljt} + s_{j(t-1)} = s_{jt} + d_{jt} \quad \forall j \in J, t \in T \\
& \sum_{i\in I} \lambda_{ik} y_{ilt} + \sum_{l'\in L} p_{l'kt} y_{l'lt} + p_{lk(t-1)} s_{l(t-1)} = p_{lkt} \left(\sum_{j\in L\cup J} y_{ljt} + s_{lt} \right) \quad \forall l \in L, k \in K, t \in T \\
& \mu_{jk}^{\min} \sum_{i\in I\cup L} y_{ijt} \ \leq \ \sum_{i\in I} \lambda_{ik} y_{ijt} + \sum_{l\in L} p_{lkt} y_{ljt} \ \leq \ \mu_{jk}^{\max} \sum_{i\in I\cup L} y_{ijt} \quad \forall j \in J, k \in K, t \in T \\
& (y, s, x^{in}, x^{out}, z) \in \mathcal{Z}, \ \ 0 \leq s_{lt} \leq C_l \ \ \forall l \in L, t \in T,
\end{aligned}
$$

where $\mathcal{Z}$ represents the set of combinatorial constraints that make this optimization model a mixed integer bilinear program (MIBLP).

$$
\begin{aligned}
\mathcal{Z} := \Big\{ (y,s,x^{in},x^{out},z) \colon & \ y_{ilt} \leq u_{il} x_{lt}^{in}, \ y_{ljt} \leq u_{lj} x_{lt}^{out} \quad \forall l \in L, i \in I \cup L, j \in L \cup J, t \in T && (11) \\
& x_{lt}^{in} + x_{lt}^{out} \leq \begin{cases} z_{lt} & \forall l \in L \setminus L_c \\ \min\left\{1, \sum_{t'=t+1-\tau_l}^{t} z_{lt'}\right\} & \forall l \in L_c \end{cases} \quad \forall t \in T && (12) \\
& 0 \leq s_{lt} \leq C_l \sum_{t'=t+1-\tau_l}^{t+1} z_{lt'} \quad \forall l \in L_c, t \in T && (13) \\
& y_{ijt} \in \{0\} \cup [\ell_{ij}, u_{ij}] \ \ \forall (i,j) \in \mathcal{A}, t \in T, \ \ x_{lt}^{in}, x_{lt}^{out}, z_{lt} \in \{0,1\} \ \ \forall l \in L, t \in T \Big\}. && (14)
\end{aligned}
$$

The combinatorial constraints can be explained as follows. Equation (14) states variable definitions for semicontinuous flows and binary variables. Here, $z_{lt} = 1$ for $l \in L_c$ implies that a new contract for pool l was started at time t whereas $z_{lt} = 1$ for $l \in L \setminus L_c$ implies that pool l was used at time t. Equation (12) models either inflow or outflow at each pool and for $l \in L_c$, ensures that there should be no flow if the contract has expired. Equation (11) imposes variable upper bound constraints on incoming and outgoing flows at each pool. Equation (13) clears inventory at a pool if its contract is not renewed.

In order to obtain a q-formulation, observe that time indexing can be treated in the same manner as pool-pool arcs. Let G' be a new graph whose nodes are partitioned into inputs I', pools L', and outputs J'. I' consists of $|I||T|$ nodes, one for each input-time pair $[i,t]$ for $i \in I, t \in T$. Similarly, L' and J' have $|L||T|$ and $|J||T|$ nodes, respectively. Consider a node $[l,t] \in L'$. Then the set of inputs in G' from which there exists a directed path to $[l,t]$ is given by $I'_{[l,t]} = \{[i,t'] \in I' : i \in I_l, t' \leq t\}$, i.e. all the input nodes in I that had a path to l and time index before t. Thus the proportion variable $q_{ilt't}$ denotes the fraction of incoming flow at pool l at time t which is contributed by input $i \in I$ from time $t' \leq t$. For any outflow arc $(l,j) \in \mathcal{A}$ from pool l, we have the bilinear terms $v_{iljt't} = q_{ilt't} y_{ljt}$ and $v^s_{ilt't} = q_{ilt't} s_{lt}$. We can now formulate the time indexed pooling problem using the q- or pq-formulations of §2.2. Note that even for medium size graphs with $|T|$ not too large, the size of the q-formulation may become prohibitively large.

3 Polyhedral relaxations

The pooling problem, as defined by its formulations in §2, involves bilinear terms in equations (4a) and (6) ((8) and (9) for $\mathbb{PQ}$) thereby making its optimization computationally challenging. A

popular methodology for solving nonconvex problems is the spatial branch-and-bound algorithm where tight relaxations of the original problem play a critical role in convergence behavior. Global optimization solvers use different bound tightening techniques that are updated at each node of the branch-and-bound tree. LP relaxations are a popular choice for lower bounding the optimum due to their scalability and ease of solvability. Semidefinite relaxations have been studied for nonconvex QCQPs [11] and applied to small-sized pooling problems [24, 56] but they do not scale well even with modest increases in problem size. In §3.1 we review known approaches for relaxing general bilinear constraints using polyhedral sets and in §3.2.1 and §3.2.2, we present some new insights into properties of the commonly used relaxations for the pooling problem. The remaining sections discuss some new relaxation techniques that, to the best of our knowledge, have not been considered before.

3.1 General bilinear programs

Given any continuous nonconvex function $f\colon C \mapsto \Re$ with a convex domain $C \subseteq \Re^n$, a popular relaxation for $\mathscr{C} := \{x \in C\colon f(x) = b\}$ is the superset $\mathscr{C}^{\text{env}} := \{x \in C\colon (\text{cvx}f)(x) \le b \le (\text{conc}f)(x)\}$, where $(\text{cvx}f)(\cdot)$ is the convex envelope (the tightest convex under-estimator) of f over C and $(\text{conc}f)(\cdot)$ is the concave envelope (the tightest concave over-estimator) of f over C. For relaxing a $\ge$ (resp. $\le$) inequality, we use only $(\text{cvx}f)(\cdot)$ (resp. $(\text{conc}f)(\cdot)$). Let $\mathcal{W}_f := \{(x,\gamma) \in C \times \Re\colon f(x) = \gamma\}$ denote the graph of f. It is well-known that $\text{conv}(\mathcal{W}_f)$ is equal to $\{(x,\gamma) \in C \times \Re\colon (\text{cvx}f)(x) \le \gamma \le (\text{conc}f)(x)\}$ and hence $\mathscr{C}^{\text{env}}$ is the strongest possible relaxation for $\mathscr{C}$ based on convexifying the graph of f. Generating envelopes of arbitrary nonlinear functions is in general a hard problem and the literature is rife with results for general and specialized functions; see for example [67] for details. For bilinear (and general multilinear) functions, several important results are known [cf. 46]. We restrict our attention to the bipartite case of the bilinear function, wherein all bilinear terms appear as a product between two types of variables - x and y, since the bilinear equality and inequality constraints in the pooling problem exhibit such structure.

First of all, a classical result due to McCormick [48] states that the envelopes of a single bilinear term $f(x,y) = axy$ with $a > 0, x \in [\ell^x, u^x], y \in [\ell^y, u^y]$ are:

$$(\text{cvx}f)(x,y) = a\ \max\{u^y x + u^x y - u^y u^x,\ \ell^y x + \ell^x y - \ell^x \ell^y\} \tag{15a}$$

$$(\text{conc}f)(x,y) = a\ \min\{u^y x + \ell^x y - \ell^x u^y,\ \ell^y x + u^x y - \ell^y u^x\}. \tag{15b}$$

Later, an equivalent statement was independently proved by [2]: the convex hull of $\{(x,y,\omega)\colon x \in [\ell^x, u^x], y \in [\ell^y, u^y], \omega = axy\}$ is described by four inequalities

$$\begin{aligned} &\gamma/a \ge u^y x + u^x y - u^y u^x,\ \gamma/a \ge \ell^y x + \ell^x y - \ell^x \ell^y, \\ &\gamma/a \le u^y x + \ell^x y - \ell^x u^y,\ \gamma/a \le \ell^y x + u^x y - \ell^y u^x. \end{aligned} \tag{16}$$

Now let $f(x,y) = \sum_{i,j} A_{ij} x_i y_j$ be a bilinear function with domain $\mathcal{X} \times \mathcal{Y}$ for some polytopes $\mathcal{X}$ and $\mathcal{Y}$. It is known [60, 64] that $(\text{cvx}f)(\cdot)$ and $(\text{conc}f)(\cdot)$ are polyhedral functions and can be evaluated at each point by optimizing an exponential sized LP formulation that triangulates f over the extreme points of $\mathcal{X} \times \mathcal{Y}$. Bao et al. [10], Misener et al. [53] have implemented and tested dynamic cut generating procedures for adding the violated facets of $\text{conv}(\mathcal{W}_f)$ by solving a separation problem over the high-dimensional LP. Thus the theoretical strength provided by $\mathscr{C}^{\text{env}}$ can be computationally obtained by recursively solving large LPs. A common choice for building an a priori compact LP relaxation is to use the McCormick envelopes separately for each bilinear term $A_{ij} x_i y_j$. It was proved in [19, 29] that if $A_{ij} \ge 0\ \forall i,j$ and $\mathcal{X} = [0,1]^m, \mathcal{Y} = [0,1]^n$, then such a single-term relaxation indeed yields the convex hull of $\mathcal{W}_f$. Later Luedtke et al. [46] generalized this result to arbitrary hyper-rectangles and also proved that for A matrices with negative or mixed sign entries or for general polytopes $\mathcal{X}$ and $\mathcal{Y}$, the McCormick relaxation can be significantly weak.

Related to finding the envelopes of f, another set of interest is the convex hull of

$$\mathcal{W} := \{(x,y,\omega)\colon x \in \mathcal{X}, y \in \mathcal{Y}, \omega_{ij} = x_i y_j\ \forall i,j\} \tag{17}$$

It is clear that $\text{conv}(\mathcal{W}_f) = \text{Proj}_{x,y,\gamma}\{(x,y,\omega,\gamma)\colon \gamma = \sum_{i,j} A_{ij}\omega_{ij}, (x,y,\omega) \in \mathcal{W}\}$. Hence for any given pair of polytopes $\mathcal{X}$ and $\mathcal{Y}$, convexifying $\mathcal{W}$ is equivalent to finding the envelopes of f over $\mathcal{X} \times \mathcal{Y}$ whereas the knowledge of $(\text{cvx}f)(\cdot)$ and $(\text{conc}f)(\cdot)$ for *specific values* of A does not imply a complete description of $\text{conv}(\mathcal{W})$. In general, an explicit closed-form representation remains elusive for the polyhedral envelopes of $\sum_{i,j} A_{ij} x_i y_j$ and for $\text{conv}(\mathcal{W})$.

The four facets in (16) are level-1 RLT inequalities [63] produced by taking pairwise multiplications between the bound factors for x ($x-\ell^x, u^x-x$) and bound factors for y ($y-\ell^y, u^y-y$). Now let $x\in\Re^m, y\in\Re^n$ for $m,n\geq 2$. If $\mathcal{X}$ and $\mathcal{Y}$ are hyper-rectangles, the corresponding $4mn$ level-1 RLT inequalities (i.e. the single-term McCormick envelopes) yield a *strict relaxation* of $\operatorname{conv}(\mathcal{W})$, as should be apparent from Luedtke et al.'s result. However when at least one of either $\mathcal{X}$ or $\mathcal{Y}$, say $\mathcal{X}$, is an arbitrary simplex, then a recent proof exploits the algebraic properties of a simplex to argue that the only nontrivial facets of $\operatorname{conv}(\mathcal{W})$ are the RLT inequalities obtained by multiplying every facet of $\mathcal{X}$ with every facet of $\mathcal{Y}$ and substituting $\omega_{ij}=x_iy_j\ \forall i,j$.

Theorem 1 (Gupte [32]) *Let $\mathcal{X}$ be a κ-dimensional simplex in $\Re^m$ for some $\kappa\leq m$ and $\mathcal{Y}$ be a polyhedron in $\Re^n$. Then the closure convex hull of $\mathcal{W}$ is equal to its level-1 RLT relaxation $\mathrm{RLT1}(\mathcal{W})$ obtained by multiplying every equation defining $\mathcal{X}$ with $y_j\ \forall j$ and every inequality defining $\mathcal{X}$ with every inequality defining $\mathcal{Y}$ and subsequently replacing $\omega_{ij}=x_iy_j$ for all i,j.*

An immediate implication of this theorem is the following corollary, which is analogous to Rikun [60, Theorem 1.4] for sum decomposition of convex envelopes.

Corollary 1 (Gupte [32]) *Let $\mathcal{X}$ be a simplex and $\mathcal{Y}=\prod_{t=1}^T\mathcal{Y}_t$ be a Cartesian product of polytopes $\mathcal{Y}_t\subset\Re^{n_t}\ \forall t$. Denote $\mathcal{W}=\cap_{t=1}^T\mathcal{W}^t$, where $\mathcal{W}^t=\{(x,(y^t)_t,(\omega^t)_t)\colon x\in\mathcal{X},y^t\in\mathcal{Y}_t,\omega^t_{ij}=x_iy^t_j\ \forall i,j\}$. Then $\operatorname{conv}(\mathcal{W})=\cap_{t=1}^T\operatorname{conv}(\mathcal{W}^t)=\cap_{t=1}^T\mathrm{RLT1}(\mathcal{W}^t)$.*

Thus bilinear functions can be convexified over simplicial constraints using a polytope in a polynomial sized extended space. This implication is pertinent to the pooling problem due to the presence of the standard simplex (7).

We close this section with the following remark.

Remark 2 Let $\mathscr{C}=\{(x,y)\in\mathcal{X}\times\mathcal{Y}\colon\sum_{i,j}A_{ij}x_iy_j\leq b\}$. Since the bilinear function $f(x,y)=\sum_{i,j}A_{ij}x_iy_j$ can be reformulated into a nonconvex quadratic as $(x,y)^\top Q\begin{pmatrix}x\\y\end{pmatrix}$, where $Q=\frac{1}{2}\begin{bmatrix}0&A\\A^\top&0\end{bmatrix}$, the set $\mathscr{C}$ can be relaxed by applying methods that are known in literature for general QCQPs. An exhaustive study of these techniques is beyond the scope of this paper; see Burer and Saxena [18] for a review. A common method [40] is to write $Q=Q_1-Q_2$, where Q_1 and Q_2 are two positive semidefinite matrices; such a decomposition is always possible – for example, using the eigenvalues of Q. Denote $g_i(x,y):=(x,y)^\top Q_i\begin{pmatrix}x\\y\end{pmatrix}$, for $i=1,2$, as two convex quadratics. We have $f(x,y)=g_1(x,y)-g_2(x,y)$ and $\mathscr{C}=\{(x,y)\in\mathcal{X}\times\mathcal{Y}\colon g_1(x,y)\leq g_2(x,y)\}$.

For example in $\mathbb{PQ}$, inequality (9b) can be reformulated in two ways. Denote $\breve{\lambda}_{ijk}:=\lambda_{ik}-\mu^{\max}_{jk}, I^+_{ljk}:=\{i\in I_l\colon\breve{\lambda}_{ijk}>0\}$ and $I^-_{ljk}:=\{i\in I_l\colon\breve{\lambda}_{ijk}<0\}$. Then (9b) is equivalent to either of the following:

$$\sum_i\lambda_{ik}y_{ij}+\sum_{l\in L,i\in I^+_{ljk}}\breve{\lambda}_{ijk}(q_{il}+y_{lj})^2+\sum_{l\in L,i\in I^-_{ljk}}\breve{\lambda}_{ijk}(q_{il}-y_{lj})^2 \leq\sum_{l\in L,i\in I^+_{ljk}}\breve{\lambda}_{ijk}(q_{il}-y_{lj})^2+\sum_{l\in L,i\in I^-_{ljk}}\breve{\lambda}_{ijk}(q_{il}+y_{lj})^2,$$

$$\sum_{i\in I}\lambda_{ik}y_{ij}+\sum_{l\in L,i\in I_l}(\breve{\lambda}_{ijk}q_{il}+y_{lj})^2\leq\sum_{l\in L,i\in I_l}(\breve{\lambda}_{ijk}q_{il}-y_{lj})^2.$$

It can be verified that the second inequality corresponds to the eigenvalue decomposition of the Q-matrix in (9b).

The set $\mathscr{C}^{\mathrm{qcqp}}:=\{(x,y)\in\mathcal{X}\times\mathcal{Y}\colon g_1(x,y)\leq(\operatorname{conc}g_2)(x,y)\}$ gives a convex relaxation of $\mathscr{C}$. Note that $(\operatorname{conc}g_2)(\cdot)$ is a affine function obtained by convexifying $g_2(\cdot)$ over the extreme points of $\mathcal{X}\times\mathcal{Y}$. Since $g_1(\cdot)$ is convex, it follows that $\mathscr{C}^{\mathrm{qcqp}}=\{(x,y)\in\mathcal{X}\times\mathcal{Y}\colon(\operatorname{cvx}g_1)(x,y)-(\operatorname{conc}g_2)(x,y)\leq 0\}$. Now the fact that convex envelopes do not sum-decompose in general [cf. 66] leads to $(\operatorname{cvx}f)(\cdot)\geq(\operatorname{cvx}g_1)(\cdot)+(\operatorname{cvx}-g_2)(\cdot)=(\operatorname{cvx}g_1)(\cdot)-(\operatorname{conc}g_2)(\cdot)$, implying that $\mathscr{C}^{\mathrm{qcqp}}$ is a (possibly strict) superset of the envelope relaxation $\mathscr{C}^{\mathrm{env}}$. Since $\mathscr{C}^{\mathrm{env}}$ is a polyhedral relaxation for bilinear functions and can be obtained computationally through a cutting plane procedure, this decomposition method for nonconvex QCQPs does not present any additional advantages over convexifying the entire bilinear function.

3.2 Relaxing feasible sets

The special structure of the pooling problem implies that when convexifying the individual bilinear functions with respect to the variable bounds, then the single-term McCormick inequalities (16) yield the best possible relaxation. This can be observed as follows. For the $\mathbb{P}$ formulation, we have the bilinear functions $\sum_{l' \in L} p_{l'k} y_{l'l} - p_{lk} \sum_{j \in L \cup J} y_{lj}$ and $\sum_{l \in L} p_{lk} y_{lj}$ in (4a) and (6), respectively, with bounds given by (5a) and (3). Since the bilinear terms in the second function are separable, it is obvious that $\text{cvx}(\sum_l p_{lk} y_{lj}) = \sum_l \text{cvx}(p_{lk} y_{lj})$ (similarly for conc). The two summations in $\sum_{l' \in L} p_{l'k} y_{l'l} - p_{lk} \sum_{j \in L \cup J} y_{lj}$ are separable; the first summand $\sum_{l' \in L} p_{l'k} y_{l'l}$ is further completely separable whereas the second summand $p_{lk} \sum_{j \in L \cup J} y_{lj}$ obeys $\text{cvx}(p_{lk} \sum_{j \in L \cup J} y_{lj}) = \sum_{j \in L \cup J} \text{cvx}(p_{lk} y_{lj})$ due to [46, Theorem 3.11]. Analogous arguments hold for $\mathbb{Q}$ and $\mathbb{PQ}$, where we have $\sum_{l' \in L : i \in I_{l'}} q_{il'} y_{l'l}$ in (8), $\sum_{l \in L, i \in I_l} \lambda_{ik} q_{il} y_{lj}$ in (9) and $\sum_{i \in I_l} q_{il} y_{lj}, \sum_{j \in L \cup J} q_{il} y_{lj}$ in (10).

Proposition 3 (Sum decomposition rule) *For the pooling problem, envelopes of bilinear functions taken over the bounds - equations* (5a) *and* (3) *and* $q_{il} \in [0,1]\ \forall l \in L, i \in I_l$*, on the associated variables can be obtained from single term McCormick inequalities.*

Besides variable bounds, the problem formulations in §2 contain additional linear constraints given by (1), (2) and (7). If we consider the question of convexifying the associated bilinear functions over some or all of these constraints along with variable bounds, then we need (many more) inequalities in addition to the McCormick inequalities (16). Hereafter, we turn our attention to finding good relaxations for constraints in the pooling problem by discussing various relaxations that are based on convexifying sets of the form (17). Different relaxations arise depending on which subset of constraints (1) - (3), (7) is used for defining the domains $\mathcal{X}$ and $\mathcal{Y}$. The objective function, being linear in y, is left unchanged for each of these relaxations.

3.2.1 p- and pq-relaxations

First, we are interested in studying a relaxation of the feasible set that arises at each pool. For $\mathbb{P}$ and $\mathbb{Q}$, relaxations of the feasible sets corresponding to pool l are denoted by the sets $\mathcal{P}_l$ and $\mathcal{Q}_l$, respectively, and are defined as

$$\mathcal{P}_l := \big\{(p_{l\cdot}, y_{l\cdot}, w_{l\cdot\cdot})\colon w_{lkj} = p_{lk} y_{lj}\ \forall k \in K, j \in L \cup J,\ p_{lk} \in [p_{lk}^{\min}, p_{lk}^{\max}]\ \forall k \in K,\ y_{l\cdot} \in \mathcal{Y}_l\big\},$$
$$\mathcal{Q}_l := \big\{(q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot})\colon v_{ilj} = q_{il} y_{lj}\ \forall i \in I_l, j \in L \cup J,\ q_{\cdot l} \in \Delta_{|I_l|},\ y_{l\cdot} \in \mathcal{Y}_l\big\}, \text{ where}$$

where $\mathcal{Y}_l$ is the set of feasible capacitated flows at pool l and is given by $\mathcal{Y}_l := \{y_{l\cdot}\colon \sum_{j \in L \cup J} y_{lj} \leq C_l,\ y_{lj} \in [0, u_{lj}]\ \forall j \in L \cup J\}$. These single pool relaxations are constructed by dropping the (i) incoming arcs at pool l along with their respective bounds, (ii) commodity balance constraints (4a) and (8) for $\mathbb{P}$ and $\mathbb{Q}$, respectively. Observe that we have also included new variables w_{lkj} and v_{ilj} for the bilinear terms. Using these new variables, the bilinear constraints (4a) and (6) (resp. (8) and (9)) in $\mathbb{P}$ (resp. $\mathbb{Q}$) can be linearized as

$$\sum_{i \in I} \lambda_{ik} y_{il} + \sum_{l' \in L} w_{l'kl} = \sum_{j \in L \cup J} w_{lkj} \quad \forall l \in L, k \in K \tag{19a}$$
$$\mu_{jk}^{\min} \sum_{i \in I \cup L} y_{ij} \ \leq \ \sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L} w_{lkj} \ \leq \ \mu_{jk}^{\max} \sum_{i \in I \cup L} y_{ij} \quad \forall j \in J, k \in K. \tag{19b}$$

$$y_{il} + \sum_{l' \in L\colon i \in I_{l'}} v_{il'l} = \sum_{j \in L \cup J} v_{ilj} \quad \forall l \in L, i \in I_l \tag{20a}$$
$$\mu_{jk}^{\min} \sum_{i \in I \cup L} y_{ij} \ \leq \ \sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L, i \in I_l} \lambda_{ik} v_{ilj} \ \leq \ \mu_{jk}^{\max} \sum_{i \in I \cup L} y_{ij} \quad \forall j \in J, k \in K. \tag{20b}$$

The valid inequalities (10) added to $\mathbb{PQ}$ are similarly linearized as

$$\sum_{i \in I_l} v_{ilj} = y_{lj} \ \ \forall l \in L, j \in L \cup J, \tag{20c}$$
$$\sum_{j \in L \cup J} v_{ilj} \leq C_l q_{il} \ \ \forall l \in L, i \in I_l. \tag{20d}$$

Equation (19a) was also motivated by [45] using the Reduced RLT (RRLT) procedure.

It is obvious that $\mathbb{PQ}$ is a stronger formulation than $\mathbb{Q}$ since the former contains additional valid inequalities (10). The feasible sets of $\mathbb{P}$ and $\mathbb{PQ}$ formulations are reformulated in the lifted space as follows:

$$\mathbb{P} = \{(p, y, w) \colon y \in \mathcal{F},\ (19),\ \mathcal{P}_l\ \ \forall l \in L\}, \quad \mathbb{PQ} = \{(q, y, v) \colon y \in \mathcal{F},\ (20),\ \mathcal{Q}_l\ \ \forall l \in L\}. \tag{21}$$

The p-relaxation is obtained using the level-1 RLT relaxation of $\mathcal{P}_l$ for all l.

$$\mathscr{P} := \big\{(p, y, w) \colon y \in \mathcal{F},\ (19),\ \mathrm{RLT1}(\mathcal{P}_l)\ \ \forall l \in L\big\}. \tag{p-relax}$$

Note 2 Traditionally, the set $\mathscr{P}$ is obtained using (19) and the McCormick envelopes (15) for $w_{lkj} = p_{lk}y_{lj}$ appearing in each $\mathcal{P}_l$. However we prefer to work with our definition of $\mathscr{P}$ since we also have $\sum_j y_{lj} \le C_l$, which may not be redundant, in $\mathcal{P}_l$ and (15) is a subset of level-1 RLT inequalities.

The pq-relaxation is obtained using (15) for $v_{ilj} = q_{il}y_{lj}$ in each $\mathcal{Q}_l$ and (10).

$$\mathscr{PQ} := \big\{(q, y, v) \colon y \in \mathcal{F},\ q_l \in \Delta_{|I_l|}\ \ \forall l \in L,\ (20),\ 0 \le v_{ilj} \le u_{lj}q_{il}\ \ \forall i, l, j\big\}. \tag{pq-relax}$$

Note 3 In the definition of $\mathscr{PQ}$, note that we have included only 2 out of the 4 envelopes from (15). The concave envelope $v_{ilj} \le y_{lj}$ is redundant since it is the sum of $\sum_i v_{ilj} = y_{lj}$ and $-v_{i'lj} \le 0\ \forall i' \ne i$. The convex envelope $v_{ilj} \ge u_{lj}q_{il} + y_{lj} - u_{lj}$ is a sum of $\sum_i v_{ilj} = y_{lj}, -u_{lj}\sum_i q_{il} = -u_{lj}$ and $-v_{i'lj} + u_{lj}q_{i'l} \ge 0\ \forall i' \ne i$.

Both $\mathscr{P}$ and $\mathscr{PQ}$ are polyhedral relaxations of the respective formulations.

Proposition 4 *The strength of $\mathscr{PQ}$ is equivalent to convexifying $\mathcal{Q}_l$ for all $l \in L$ and hence $\mathscr{PQ}$ is a stronger relaxation than $\mathscr{P}$ in the sense that for any $c \in \Re^{|\mathcal{A}|}$,*

$$z^{pq} := \min\{c^\top y \colon (q, y, v) \in \mathscr{PQ}\}\ \ge\ z^p := \min\{c^\top y \colon (p, y, w) \in \mathscr{P}\}.$$

Proof Since $\Delta_{|I_l|}$ is a standard simplex, a direct application of Theorem 1 with $\mathcal{X} = \Delta_{|I_l|}$ and $\mathcal{Y} = \mathcal{Y}_l$ gives us

$$\begin{aligned}\mathrm{conv}(\mathcal{Q}_l) = \Big\{(q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot}) \colon q_l \in \Delta_{|I_l|},\ &\sum_{j \in L \cup J} v_{ilj} \le C_l q_{il}\ \ \forall i \in I_l,\ \sum_{i \in I_l} v_{ilj} = y_{lj}\ \ \forall j \in L \cup J,\\ &0 \le v_{ilj} \le u_{lj}q_{il}\ \ \forall i \in I_l, j \in L \cup J\Big\}.\end{aligned} \tag{22}$$

Note that the two additional inequalities, besides the McCormick envelopes $v_{ilj} \le u_{lj}q_{il}$ and $v_{ilj} \ge 0$, that define $\mathrm{conv}(\mathcal{Q}_l)$ are exactly the valid inequalities (20c), (20d) that were added to strengthen the $\mathbb{PQ}$ formulation. Thus $\mathscr{PQ}$ gives at least the same relaxation strength as $\mathrm{conv}(\mathcal{Q}_l)$,

$$\mathscr{PQ} = \big\{(q, y, v) \colon y \in \mathcal{F},\ (20),\ \mathrm{conv}(\mathcal{Q}_l)\ \ \forall l \in L\big\}. \tag{23}$$

As seen in the proof of Proposition 2, there exist linear mappings, of the form $p_{lk} = \sum_{i \in I_l} \lambda_{ik} q_{il}$ and $w_{lkj} = \sum_{i \in I_l} \lambda_{ik} v_{ilj}$, between points in $\mathcal{Q}_l$ and $\mathcal{P}_l$. It follows that

$$\{(q, y, v) \colon y \in \mathcal{F},\ (20),\ \mathrm{conv}(\mathcal{Q}_l)\ \ \forall l\} = \{(p, y, w) \colon y \in \mathcal{F},\ (19),\ \mathrm{conv}(\mathcal{P}_l)\ \ \forall l\}.$$

The set $\mathcal{P}_l$ has similar structure to that of the set $\mathcal{W}$ (cf. (17)) with the vector $y_{l\cdot}$ playing the role of y and $p_{l\cdot}$ playing the role of χ. The McCormick inequalities for $w_{lkj} = p_{lk}y_{lj}\ \forall k, j$ yield a relaxation of $\mathrm{conv}(\mathcal{P}_l)$ and this inclusion is strict because the bounds $p_{l\cdot} \in [p_{l\cdot}^{\min}, p_{l\cdot}^{\max}]$ define a hyper-rectangle Hence

$$\begin{aligned}\mathscr{P} &\supseteq \big\{(p, y, w) \colon y \in \mathcal{F},\ (19),\ \mathrm{conv}(\mathcal{P}_l)\ \ \forall l \in L\big\}\\ \Longrightarrow \mathrm{Proj}_y\, \mathscr{P} &\supseteq \mathrm{Proj}_y \big\{(p, y, w) \colon y \in \mathcal{F},\ (19),\ \mathrm{conv}(\mathcal{P}_l)\ \ \forall l \in L\big\}\\ &= \mathrm{Proj}_y\{(q, y, v) \colon y \in \mathcal{F},\ (20),\ \mathrm{conv}(\mathcal{Q}_l)\ \ \forall l\}\\ &= \mathrm{Proj}_y\, \mathscr{PQ}.\end{aligned}$$

Thus $\mathrm{Proj}_y\, \mathscr{PQ} \subseteq \mathrm{Proj}_y\, \mathscr{P}$, which is equivalent to the proposed statement. □

Remark 3 Based on (23), it follows that $\mathscr{PQ}$ is equivalent in strength to the relaxation obtained by convexifying the bilinear functions in $\mathbb{PQ}$ over the domain defined by simplex (7), pool capacities in (2) and flow upper bounds (3).

Since Proposition 4 argued that $\mathrm{conv}(\mathcal{Q}_l)$ is a tighter relaxation than the McCormick relaxation of $\mathcal{P}_l$, our result is stronger than previous results for standard [67, chap. 9] and generalized [4] pooling problems. The single pool argument that we adopted here extends to the hybrid formulation of §2.2.3 where the relaxations corresponding to pools with proportion variables are stronger than the relaxations of these pools in the p-formulation. Hence it follows that the strength of the $\mathbb{HYB}$ formulation is between that of $\mathbb{P}$ and $\mathbb{PQ}$. There is no dominance between $\mathbb{P}$ and $\mathbb{Q}$ formulations.

3.2.2 Piecewise linear relaxations

The strength of the McCormick envelopes (15) for a single bilinear term $x_i y_j$ depends on the bounds $[\ell_i^x, u_i^x]$ and $[\ell_j^y, u_j^y]$ for x_i and y_j, respectively. Tighter bounds lead to stronger relaxations. Hence, partitioning the intervals of one or both the variables and then constructing McCormick envelopes in each interval gives a much stronger relaxation than simply including equations (15) based on the entire interval. Of course, the level of partitioning determines the strength of this new relaxation. To enforce validity of this relaxation, we need to add extra binary variables to turn on/off each partition with exactly one partition being turned on. This gives rise to a piecewise linear MILP relaxation of $\{(x_i, y_j, \omega_{ij}) \in [\ell_i^x, u_i^x] \times [\ell_j^y, u_j^y] \times \Re\colon \omega_{ij} = x_i y_j\}$ for every i, j. Note that $\mathscr{P}\mathscr{Q}$, which is an LP, can be interpreted as a trivial piecewise linear relaxation wherein the domain of each variable has a single partition. Such piecewise linear McCormick relaxations were used by [49, 51] to solve some generalized pooling problems and [27] performed an extensive computational study on small scale standard pooling problems to investigate different partitioning levels and MILP models. Recently, [54] implemented a branch-and-bound based solver for pooling problems that uses piecewise linear MILP relaxations to generate lower bounds in the enumeration tree.

An interesting theoretical question is to determine the error introduced by partitioning the variable domains as a function of the number of partitions. There are two related questions here: the first one is to determine the distance between partitions so that we minimize the total error calculated as sum of squares (or absolute values) of errors between $\omega_{ij} = x_i y_j$ and the McCormick envelopes (15) in each partition. It was shown in [34] that the best strategy is to locate the partitions of equal length and this result is applicable to any bilinear problem. A more pertinent question is to find out how the piecewise linear relaxation schemes affect the quality of the lower bound with respect to z^*, the optimal value of the pooling problem. This was recently answered by Dey and Gupte [21, Theorem 1] who proved that for standard problems, the ratio of z^* to the optimal value of any piecewise linear McCormick relaxation is at most $|J|$, where $|J|$ denotes the number of output nodes. Notice that this performance guarantee is *independent of the number of and the distance between partitions in each variable domain.* They also proved that this approximation factor is tight, i.e. there are problem instances where this ratio gets arbitrarily close to $|J|$ for all piecewise linear relaxations.

3.2.3 An extended pq-relaxation

Recall the $\mathbb{PQ}$ formulation. The k^{th} spec requirement constraints at output j are given by (20b) where $v_{ilj} = q_{il} y_{lj}$ and q_{il} denotes the ratio of incoming flow to pool l that originated at input i. Notice that each output j may itself be treated as a pool node since linear blending takes place at j as per equation (20b). Suppose that we introduce a new variable q_{ij} to denote the ratio of incoming flow to output j that originated at input i. Then the k^{th} spec level at j is $\sum_{i \in I_j} \lambda_{ik} q_{ij}$. Equation (20b) can now be modeled as

$$\mu_{jk}^{\min} \leq \sum_{i \in I_j} \lambda_{ik} q_{ij} \leq \mu_{jk}^{\max}, \quad q_{\cdot j} \in \Delta_{|I_j|} \; \forall j \in J. \tag{24a}$$

To guarantee correctness, we introduce new commodity balance constraints and bilinear terms

$$y_{ij} + \sum_{l \in L\colon i \in I_l} v_{ilj} = \sum_{t \in \mathcal{N}} \xi_{itj} \; \forall j \in J, i \in I_j, \tag{24b}$$

$$\xi_{itj} = q_{ij} y_{tj} \; \forall j \in J, (t,j) \in \mathcal{A}, i \in I_j. \tag{24c}$$

We let $\mathbb{PQ}' := \{(q, y, v, \xi)\colon y \in \mathcal{F}, (20a), (24), \mathcal{Q}_l \;\; \forall l \in L\}$ denote this extended *pq*-formulation where all the bilinear terms are present in equality constraints (20a) and (24b).

Proposition 5 *For any $(y, v) \in \mathrm{Proj}_{y,v}\, \mathscr{P}\mathscr{Q}$, there exist values for q_{ij} and ξ_{itj} for all $j \in J, (t,j) \in \mathcal{A}, i \in I_j$ such that (q, y, v, ξ) satisfies* (24).

Proof Choose some $j \in J$. If $\sum_t y_{tj} = 0$, which implies left hand side of (24b) is zero, set $\xi_{itj} = 0 \; \forall i, t, j$ and we know from Observation 3 that there exists some $q_{\cdot j} \in \Delta_{|I_j|}$ that satisfies (24a). Let $\sum_t y_{tj} > 0$. Construct $q_{ij} = (y_{ij} + \sum_{l \in L\colon i \in I_l} v_{ilj}) / \sum_t y_{tj}$ and $\xi_{itj} = q_{ij} y_{tj}$, thus satisfying (24b). Equation (20b) of $\mathscr{P}\mathscr{Q}$ and construction of q_{ij} imply that $\mu_{jk}^{\min} \leq \sum_{i \in I_j} \lambda_{ik} q_{ij} \leq \mu_{jk}^{\max}$ is satisfied. Finally,

$$\sum_{i \in I_j} q_{ij} = \frac{1}{\sum_t y_{tj}} \left[\sum_{i \in I} y_{ij} + \sum_{i \in I_j} \sum_{l \in L\colon i \in I_l} v_{ilj} \right] = \frac{1}{\sum_t y_{tj}} \left[\sum_{i \in I} y_{ij} + \sum_{l \in L, i \in I_l} v_{ilj} \right] = 1,$$

where the last equality is due to $\sum_{i\in I_l} v_{ilj} = y_{lj}$ being valid to $\mathscr{P}\mathscr{Q}$. □

This tells us that since $\mathscr{P}\mathscr{Q}$ convexifies the set $\mathcal{Q}_l$ for each $l \in L$, we would not gain any additional strength by convexifying the set defined by the constraints in (24). That being said, since $\mathscr{P}\mathscr{Q}$ relaxes $v_{ilj} = q_{il}y_{lj}$, we may be able to improve the lower bound by $\mathscr{P}\mathscr{Q}$ using valid inequalities for

$$\sum_{i\in I} \lambda_{ik} y_{ij} + \sum_{l\in L, i\in I_l} \lambda_{ik} q_{il} y_{lj} = \left[\sum_{i\in I_j} \lambda_{ik} q_{ij}\right] \sum_{t\in\mathcal{N}} y_{tj}, \;\; \sum_{t\in\mathcal{N}} y_{tj} \le C_j, \;\; y_{\cdot j} \ge \mathbf{0}, \;\; \text{(24a)},$$

for a given $j \in J, k \in K$. We leave this idea open for future research. From Remark 2 it follows that the convex hull of the above set will present a stronger relaxation than simple relaxation methods for nonconvex QCQPs.

3.3 Partial RLT relaxations

Recall that $\mathscr{P}\mathscr{Q}$ is a partial level-1 RLT relaxation since the valid inequalities (20c), (20d) for $\text{conv}(\mathcal{Q}_l)$ are generated via a RLT procedure. Here we discuss some new partial level-1 RLT relaxations to further strengthen $\mathscr{P}\mathscr{Q}$, albeit at the expense of adding many more auxiliary variables.

First observe that in the definition of $\mathcal{Q}_l$, we dropped the variables y_{il} for $i \in I \cup L$. We could have retained these variables along with their bounds and applied Theorem 1 to obtain a tighter relaxation than the one presented in (22). However, this stronger relaxation comes at a cost of introducing McCormick inequalities for new bilinear terms of the form $v'_{i'il} = q_{il}y_{i'l}$ for $i, i' \in I_l$, which are not present elsewhere in the $\mathbb{PQ}$ formulation. This increases the size of the relaxation considerably.

$$\begin{aligned}\mathscr{R}_1 := \big\{(q,y,v,v') \colon (q,y,v) \in \mathscr{P}\mathscr{Q},\; 0 \le v'_{ii'l} \le u_{i'l}q_{il} \;\; \forall l \in L, i, i' \in I_l,\\ \sum_{i\in I_l} v_{ii'l} = y_{i'l} \;\; \forall l \in L, i' \in I_l\big\}.\end{aligned}$$

Second, recall that in Proposition 4, $\mathscr{P}\mathscr{Q}$ was shown to be equivalent in strength to $\prod_{l\in L} \text{conv}(\mathcal{Q}_l)$ where $\mathcal{Q}_l$ is a relaxation of the feasible set corresponding to pool l. A second strengthening over $\mathscr{P}\mathscr{Q}$ can be obtained by performing a level-1 RLT over multiple pools that are all connected to the same output. For every $j \in J$, let $\mathcal{Y}_j := \{y_{\cdot j} \colon \sum_{i\in I\cup L} y_{ij} \le C_j, y_{ij} \in [0, u_{ij}] \;\; \forall i \in I \cup J\}$ denote the flow set corresponding to input flows at j. We use level-1 RLT inequalities for the convex hull of

$$\mathcal{S}_j := \big\{(q, y_{\cdot j}, v_{\cdot j}) \colon q_{\cdot l} \in \Delta_{|I_l|} \;\; \forall l \in L \colon (l,j) \in \mathcal{A},\; y_{\cdot j} \in \mathcal{Y}_j,\; v_{ilj} = q_{il}y_{lj} \;\; \forall l \in L \colon (l,j) \in \mathcal{A}, i \in I_l\big\}$$

for every $j \in J$. Note that $\text{conv}(\mathcal{S}_j)$ is equivalent in strength to the relaxation obtained by convexifying the bilinear functions in $\mathbb{PQ}$ over $\prod_l \Delta_{|I_l|} \times \mathcal{Y}_j$. Since a Cartesian product of simplices is not a simplex itself, we cannot apply Theorem 1 to obtain $\text{conv}(\mathcal{S}_j)$ and we only have $\text{conv}(\mathcal{S}_j) \subseteq \text{RLT1}(\mathcal{S}_j)$, unless $u_{ij} \ge C_j \;\; \forall i$ in which case $\mathcal{Y}_j$ becomes a simplex and we get $\text{conv}(\mathcal{S}_j) = \text{RLT1}(\mathcal{S}_j)$. Our second RLT relaxation is

$$\mathscr{R}_2 := \big\{(q,y,v,v') \colon (q,y,v) \in \mathscr{P}\mathscr{Q},\; (q, y_{\cdot j}, v, v'_{\cdot j}) \in \text{RLT1}(\mathcal{S}_j) \;\; \forall j \in J\big\},$$

$$\begin{aligned}\text{where } \text{RLT1}(\mathcal{S}_j) := \big\{(q,y_{\cdot j},v,v'_{\cdot j}) \colon\; & v'_{ilij} = v_{ilj} \;\; \forall l \in L \colon (l,j) \in \mathcal{A}, i \in I_l,\\ & \sum_{i\in I_l} v'_{ili'j} = y_{i'j} \;\; \forall l \in L \colon (l,j) \in \mathcal{A}, (i',j) \in \mathcal{A}\\ & 0 \le v'_{ili'j} \le u_{i'j}q_{il} \;\; \forall l \in L \colon (l,j) \in \mathcal{A}, i \in I_l, (i',j) \in \mathcal{A}\\ & \sum_{i' \colon (i',j)\in\mathcal{A}} v_{ili'j} \le C_j q_{il} \;\; \forall l \in L \colon (l,j) \in \mathcal{A}, i \in I_l\big\}.\end{aligned}$$

The third polyhedral relaxation, denoted as $\mathscr{R}_3$, is the RLT relaxation for the convex hull of

$$\tilde{\mathcal{S}} := \{(q,y,v) \colon q_{\cdot l} \in \Delta_{|I_l|} \;\; \forall l \in L,\; y \in \mathcal{F},\; v_{ilj} = q_{il}y_{lj} \;\; \forall j \in J, l \in L, i \in I_l\}, \tag{25}$$

Once again, the set $\tilde{\mathcal{S}}$ does not have the simplicial structure of Theorem 1 and in general, we do not know an inequality description of $\text{conv}(\tilde{\mathcal{S}})$ (simple examples suggest that this set has exponentially many facets). $\mathscr{R}_3$ is a stronger relaxation than $\mathscr{R}_1$ and $\mathscr{R}_2$ but also has the highest number of auxiliary variables.

3.4 Value function and Lagrangian relaxations

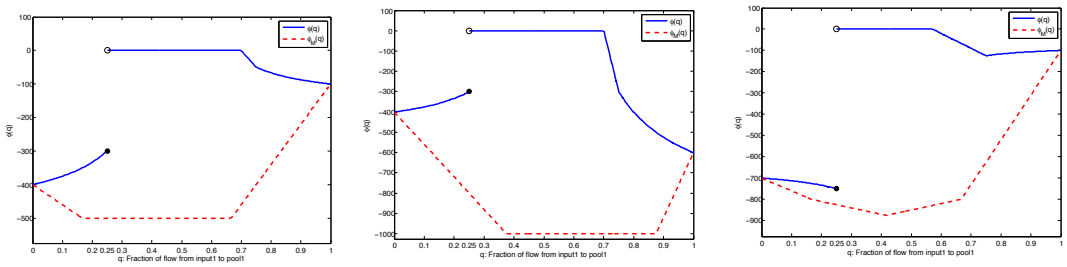
For the standard pooling problem, various Lagrangian relaxations have been proposed over the years. A Lagrangian dual of $\mathbb{Q}$ was used by [13] to generate a converging sequence of lower bounds in a branch-and-bound algorithm. For $\mathbb{P}$, all constraints except the bounds (5a) and (3) on p's and y's, respectively, were dualized by [1] whereas [6] went one step further by dualizing only the bilinear constraints in $\mathbb{P}$ and $\mathbb{PQ}$ and solving a big-M MILP formulation as a subproblem. We first present the value function of [13] as applied to a generalized pooling problem and use it to show that the pq-relaxation is equivalent to a specific Lagrangian dual of the pooling problem. This establishes a direct connection between the two and also extends [67, Proposition 9.9] to generalized pooling problems. We also present additional Lagrangian relaxations and discuss their strength.

Consider the bilinear formulation $\mathbb{PQ}$ and suppose that we treat q as a parameter to obtain an LP for every $q \in \prod_{l\in L} \Delta_{|I_l|}$. This LP is not decomposable over L. Let $\phi\colon \prod_{l\in L} \Delta_{|I_l|} \mapsto \Re_-$ denote the optimal value of this LP. This function is not only nonsmooth but also discontinuous on its domain since q appears on both left and right hand side of the constraints. The pooling problem can then be equivalently stated as the global optimization problem

$$z^* = \min_q \{\phi(q)\colon q_{\cdot l} \in \Delta_{|I_l|} \ \ \forall l \in L\} \tag{26}$$

Now suppose that we substitute every bilinear term $q_{il}y_{lj}$ in $\mathbb{PQ}$ with a new variable v_{ilj} and add the McCormick envelopes $0 \le v_{ilj} \le u_{lj}q_{il}$ and inequalities (20c), (20d). This gives us an LP and we denote its value function by $\phi_{\mathcal{M}}\colon \prod_{l\in L} \Delta_{|I_l|} \mapsto \Re_-$. Since the q and y variables are separable in this feasible and bounded LP, It follows that $\phi_{\mathcal{M}}(\cdot)$ is a polyhedral function. Also it is evident that $\phi_{\mathcal{M}}(\cdot) \le \phi(\cdot)$ and $z^{pq} = \min_q\{\phi_{\mathcal{M}}(q)\colon q_{\cdot l} \in \Delta_{|I_l|} \ \ \forall l \in L\}$.

Example 1 We illustrate the two functions $\phi_{\mathcal{M}}(\cdot)$ and $\phi(\cdot)$ on the Haverly [36] standard pooling problem with 3 inputs, 1 pool, 2 outputs, and 1 specification. The solitary pool accepts flows from the first two inputs, whereas the third input is connected directly to the two outputs. Hence we have $q_1 + q_2 = 1$. Figure 2 plots $\phi(q_1)$ and $\phi_{\mathcal{M}}(q_1)$.



(a) Haverly1 (b) Haverly2 (c) Haverly3

Fig. 2 Value functions $\phi(q_1)$ (solid line) and $\phi_{\mathcal{M}}(q_1)$ (dotted line) for Haverly instances. For all three instances, $\phi(q_1)$ is lower semicontinuous at $q_1 = 0.25$ and $z^{pq} < z^*$. Observe that Haverly3 has discontinuity of $\phi(q_1)$ at its optimal solution.

Since $\phi(\cdot)$ is defined by an LP as $\phi(q) = \min_{y,v}\{c^\top y\colon y \in \mathcal{F}, (20), v_{ilj} = q_{il}y_{lj} \ \ \forall i,l,j\}$, strong duality dictates that $\phi(q)$ is equal to the value of the Lagrangian bound obtained by dualizing constraints (20), $\sum_j y_{ij} \le C_i$ for all $i \in I$ and $\sum_i y_{ij} \le C_j$ for all $j \in J$. Observe that the constraints that haven't been dualized, namely $\sum_j y_{lj} \le C_l$ for all $l \in L$ and $0 \le y_{ij} \le u_{ij}$ for all $(i,j) \in \mathcal{A}$, are separable across pools. Hence this Lagrangian dual can be written as

$$\begin{aligned} \phi(q) = \max_{\rho\ge\mathbf{0},\tau} \ \min_{y,v} \ & \varphi(\rho,\tau,\{y_{ij}\}_{i\in I}) + \sum_{l\in L} \psi_l(\rho,\tau,q_{\cdot l},y_{l\cdot},v_{\cdot l\cdot}) \\ \text{s.t.} \quad & \sum_j y_{lj} \le C_l \ \ \forall l \in L, \ 0 \le y_{ij} \le u_{ij} \ \ \forall (i,j) \in \mathcal{A}, \ v_{ilj} = q_{il}y_{lj} \ \ \forall i,l,j \end{aligned} \tag{27}$$

where $\varphi(\rho,\tau,\cdot)$ and $\psi_l(\rho,\tau,\cdot,\cdot,\cdot)$ are affine functions for fixed multipliers ρ,τ. Substituting the representation of $\phi(q)$ from (27) into the global optimization in (26) and invoking saddle point

duality to interchange outermost min and max produces a lower bound z^{LAG1} on z^*:

$$z^* \geq z^{\mathrm{LAG1}} := \max_{\rho \geq \mathbf{0}, \tau} \min_{q,y,v} \varphi(\rho, \tau, \{y_{ij}\}_{i \in I}) + \sum_{l \in L} \psi_l(\rho, \tau, q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot})$$
$$\text{s.t.} \quad 0 \leq y_{ij} \leq u_{ij} \ \ \forall (i,j) \in \mathcal{A}, i \in I, \ (q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot}) \in \mathcal{Q}_l \ \ \forall l \in L.$$

Clearly z^{LAG1} is a Lagrangian lower bound obtained by dualizing all constraints, except the ones in Ω, in the $\mathbb{PQ}$ formulation. For every fixed ρ, τ, the two functions φ and ψ do not share any common variables and are hence separable. This along with separability of ψ_l's across $l \in L$ implies that the inner minimization problem for z^{LAG1} is sum-decomposable. Thus we obtain

$$z^{\mathrm{LAG1}} = \max_{\rho \geq \mathbf{0}, \tau} \Big[\min_{y} \varphi(\rho, \tau, \{y_{ij}\}_{i \in I}) \quad \text{s.t. } 0 \leq y_{ij} \leq u_{ij} \ \ \forall (i,j) \in \mathcal{A}, i \in I \Big] + \sum_{l \in L} \Big[\min_{q,y,v} \psi_l(\rho, \tau, q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot}) \quad \text{s.t. } (q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot}) \in \mathcal{Q}_l \Big]$$

Since ψ_l is a affine function for every fixed ρ and τ, the second minimization is equivalent to optimizing ψ_l over $\mathrm{conv}(\mathcal{Q}_l)$ and this convex hull was obtained in equation (22). Hence,

$$z^{\mathrm{LAG1}} = \max_{\rho \geq \mathbf{0}, \tau} \Big[\min_{y} \varphi(\rho, \tau, \{y_{ij}\}_{i \in I}) \quad \text{s.t. } 0 \leq y_{ij} \leq u_{ij} \ \ \forall (i,j) \in \mathcal{A}, i \in I \Big] + \sum_{l \in L} \Big[\min_{q,y,v} \psi_l(\rho, \tau, q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot}) \quad \text{s.t. Inequalities from (22).} \Big]$$

Finally, observe that the above problem is a Lagrangian dual of the LP associated with the *pq*-relaxation and hence by strong duality, its value must be equal to z^{pq}. Thus we have argued the following.

Proposition 6 *Let z^{LAG1} be the Lagrangian lower bound on z^* obtained by dualizing the following constraints in the $\mathbb{PQ}$ formulation: (i) $\sum_j y_{ij} \leq C_i$ for all $i \in I$, (ii) $\sum_i y_{ij} \leq C_j$ for all $j \in J$, and (iii) Equations* (20)*. Then $z^{\mathrm{LAG1}} = z^{pq}$.*

The above result is possible due to two key steps: the decomposition of the problem across pools after dualizing the appropriate constraints and the strength of the valid inequalities (20c), (20d) as proved in equation (22).

Now suppose that we dualize only the bilinear constraints (20) (recall that v_{ilj} replaces the bilinear term $q_{il}y_{lj}$ in (8) and (9)), as was proposed in [6]. Then the value of this Lagrangian lower bound, denoted by z^{LAG2}, is equal to

$$z^{\mathrm{LAG2}} = \min\{c^\top y \colon (q,y,v) \in \mathrm{conv}(\tilde{\mathcal{S}}), \ (20)\}, \tag{28}$$

where $\tilde{\mathcal{S}}$ was defined in (25). This explicit LP representation of z^{LAG2} is possible due to the polyhedrality of the convex hull of $\tilde{\mathcal{S}}$ and well-established results for Lagrangian duality [cf. 55, §II.3.6]. Also note that:

Observation 8 *If C_i and C_j are redundant for all $i \in I, j \in J$, then $z^{\mathrm{LAG2}} = z^{pq}$.*

It is not clear how to solve the LP for z^{LAG2} since a complete inequality description of $\mathrm{conv}(\tilde{\mathcal{S}})$ is unknown. Instead, one may have to resort to a subgradient algorithm that formulates the Lagrangian subproblem as a 0\1 MILP [6]. However it is worth noting we can possibly tighten the lower bound z^{pq} using any valid inequality for $\mathrm{conv}(\tilde{\mathcal{S}})$ that is not valid to $\prod_l \mathrm{conv}(\mathcal{Q}_l)$.

A third Lagrangian relaxation can be obtained by dualizing only the consistency constraints $v_{ilj} = q_{il}y_{lj} \ \forall i,l,j$ and (20d) in $\mathbb{PQ}$ so that for the remaining constraints, neither are there any product terms between q's and y's nor are q's and v's present in the same constraint (cf. equation (21)). This allows us to follow standard duality arguments and use polyhedrality of the convex hull of

$$\check{\mathcal{S}} = \{(q,y,v,\xi) \colon q_{\cdot l} \in \Delta_{|I_l|} \ \forall l \in L, y \in \mathcal{F}, (20\mathrm{a}) - (20\mathrm{c})\}. \tag{29}$$

to express the lower bound corresponding to this Lagrangian dual as

$$z^{\mathrm{LAG3}} = \min\{\tilde{c}^\top y \colon (q,y,v,\xi) \in \mathrm{conv}(\check{\mathcal{S}}), \ (20\mathrm{d}), \ \xi_{ilj} = v_{ilj} \ \forall l \in L, i \in I_l, j \in L \cup J\}. \tag{30}$$

By construction, it follows that

$$z^* \geq z^{\mathrm{LAG3}}, \quad \text{and} \quad z^* \geq z^{\mathrm{LAG2}} \geq z^{\mathscr{R}_3} \geq z^{\mathrm{LAG1}} = z^{pq}. \tag{31}$$

There is no dominance relation between z^{LAG3} and z^{LAG2} or z^{LAG1} since the third Lagrangian dualizes $v_{ilj} = q_{il}y_{lj}$ whereas the first two do not. In our computational experiments, we used disjunctive formulations for the second and third Lagrangian relaxations to test the quality of the lower bounds produced by z^{LAG2} and z^{LAG3}.

4 Variable Discretizations

The discretization strategies for the pooling problem can be broadly classified into two categories: (i) restrict some of the variables appearing in the problem to take one amongst a finite set of pre-specified values within their respective domains, or (ii) discretize the consistency requirements at each pool in the network. The two strategies result in MILP approximations of the pooling problem; the first strategy is applicable to any bilinear program whereas the second strategy, which was proposed by Dey and Gupte [21], specifically exploits the structure of the pooling problem to obtain a network flow MILP restriction. Other heuristics [3, 7, 9] for finding feasible solutions to the pooling problem have been proposed in literature. In §5, we empirically compare the performance of our variable discretizations against the feasible solutions obtained from other methods.

In this section, we focus on obtaining feasible solutions to the pooling problem by discretizing some of its variables. We illustrate our approach in the context of a (possibly mixed integer) bilinear program. Then we extend our ideas to the pooling problem by highlighting different choices for selecting a variable to discretize. Our motivation for studying discretization methods is based on the fact that MILP solvers are more mature in terms of branching strategies, cutting planes, heuristics etc. than global optimization solvers and hence it is more likely that we can solve a MILP faster than a BLP or MIBLP.

4.1 Overview

The general idea behind variable discretizations is as follows. Consider a bilinear program where each bilinear term can be represented by the set defined in (17): $\mathcal{W} = \{(x, y, \omega) \colon \omega = xy, x \in [0, u^x], y \in [0, u^y]\}$. For the sake of simplicity, we assume the lower bounds on x and y to be zero and u^x to be a positive integer. Although we assumed that both x and y are continuous variables, the presented ideas can be easily extended to the case when the original problem is a mixed integer bilinear program and one or both x and y are integer variables. Now suppose that we discretize y, i.e. restrict y to take only integer values within its bounds $[0, u^y]$. This produces an approximation of $\mathcal{W}$ denoted by $\mathcal{W}^y := \{(x, y, \omega) \in \mathcal{W} \colon y \in \mathbb{Z}\}$. Substituting $\mathcal{W}^y$ for every occurrence of $\mathcal{W}$ produces a MIBLP approximation of the BLP. Note that for $u^y \geq 2$ we have $\mathcal{W}^y \subsetneq \mathcal{M}(\mathcal{W}^y) = \text{conv}(\mathcal{W}^y)$ [cf. 33, Proposition 2.1], where $\mathcal{M}(\mathcal{W}^y)$ represents the McCormick relaxation of $\mathcal{W}^y$ obtained using (15). There are various approaches for modeling the requirement $y \in \{0, 1, \ldots, u^y\}$ using additional $0\backslash 1$ variables - two common methods are the unary and the binary formulation. The former, denoted by $\mathcal{U}(\mathcal{W}^y)$, adds $u^y + 1$ $0\backslash 1$ variables whereas the latter, denoted by $\mathcal{B}(\mathcal{W}^y)$, adds only $\ell(u^y) := \lfloor \log_2 u^y \rfloor + 1$ many $0\backslash 1$ variables.

$$\begin{aligned}
\mathcal{U}(\mathcal{W}^y) := \Big\{(x, y, \omega, \zeta, \nu) \colon \omega = \sum_{r=0}^{u^y} r\nu_r,\ y = \sum_{r=0}^{u^y} r\zeta_r,\ \sum_{r=0}^{u^y} \zeta_r = 1,\ x \in [0, u^x], \\
(x, \zeta_r, \nu_r) \in \mathcal{M}(\{\nu_r = x\zeta_r\})\ \ \forall r,\ \zeta_r \in \{0, 1\}\ \ \forall r\Big\} \\
\mathcal{B}(\mathcal{W}^y) := \Big\{(x, y, \omega, \zeta, \nu) \colon \omega = \sum_{r=1}^{\ell(u^y)} 2^{r-1}\nu_r,\ y = \sum_{r=1}^{\ell(u^y)} 2^{r-1}\zeta_r,\ \sum_{r=1}^{\ell(u^y)} 2^{r-1}\zeta_r \leq u^y,\ x \in [0, u^x], \\
(x, \zeta_r, \nu_r) \in \mathcal{M}(\{\nu_r = x\zeta_r\})\ \ \forall r,\ \zeta_r \in \{0, 1\}\ \ \forall r\Big\}
\end{aligned}$$

In the above, $\mathcal{M}(\{\nu_r = x\zeta_r\})$ denotes the McCormick relaxation (16) for $\nu_r = x\zeta_r$. Note that $\zeta_r \in \{0, 1\}, \forall r$ and $\sum_r \zeta_r = 1$ imply a SOS-1 constraint in $\mathcal{U}(\mathcal{W}^y)$, which can be reformulated using a logarithmic number of $0\backslash 1$ variables and constraints as shown by [68]. This *log unary formulation* $\mathcal{L}(\mathcal{W}^y)$ may sometimes exhibit faster computational performance in a branch-and-bound algorithm. The reformulation sizes of $\mathcal{U}(\cdot), \mathcal{L}(\cdot), \mathcal{B}(\cdot)$ can be compared as follows: the number of $0\backslash 1$ variables is $u^y + 1, \ell(u^y), \ell(u^y)$, respectively; $\mathcal{B}(\cdot)$ has the least number of continuous variables and constraints whereas $\mathcal{L}(\cdot)$ has the most of each. The LP relaxations of $\mathcal{U}(\cdot)$ and $\mathcal{B}(\cdot)$ were compared in [33] and it was proven that in general, neither dominates the other. Substituting any one of $\mathcal{U}(\mathcal{W}^y)$ or $\mathcal{B}(\mathcal{W}^y)$ for every occurrence of $\mathcal{W}^y$ produces a MILP approximation of BLP. In a recent study [33], facet-defining inequalities were proposed for $\mathcal{B}(\mathcal{W}^y)$ and it was empirically shown that the binary reformulation MILP is sometimes solved faster than the MIBLP corresponding to $\mathcal{W}^y$.

4.2 Application to the pooling problem

The variable discretization approach for obtaining feasible solutions to the pooling problem was first studied in [57] wherein the authors presented the unary MILP model for discretizing the p variables in $\mathbb{P}$. Here we apply the discussion of the previous section and present a comprehensive list of discretized versions of the $\mathbb{P}$ and $\mathbb{PQ}$ formulations ($\mathcal{L}(\cdot)$ is not considered since it did not present any significant benefits over $\mathcal{B}(\cdot)$ in our computational experiments.). In $\mathbb{PQ}$, each bilinear term is of the form $v_{ilj} = q_{il}y_{lj}$ and the corresponding set for this bilinear term is

$$\mathcal{W}^{\mathbb{PQ}}_{ilj} := \{(q_{il}, y_{lj}, v_{ilj})\colon v_{ilj} = q_{il}y_{lj}, q_{il} \in [0,1], y_{lj} \in [0, u_{lj}]\} \quad l \in L, i \in I_l, j \in L \cup J.$$

We have two choices here: either discretize $y = q_{il}$ or $y = y_{lj}$. Similarly, the set representing a single bilinear term in $\mathbb{P}$ is

$$\mathcal{W}^{\mathbb{P}}_{lkj} := \{(p_{lk}, y_{lj}, w_{lkj})\colon w_{lkj} = p_{lk}y_{lj}, p_{lk} \in [p^{\min}_{lk}, p^{\max}_{lk}], y_{lj} \in [0, u_{lj}]\} \quad l \in L, k \in K, j \in L \cup J,$$

and we may discretize either p_{lk} or y_{lj}. We explain the discretized models for $\mathbb{PQ}$ and remark that suitable counterparts are defined for $\mathbb{P}$.

The flow discretized feasible set, which is obtained by replacing $\mathcal{W}^{\mathbb{PQ}}_{ilj}$ with $\mathcal{W}^{\mathbb{PQ}}_{ilj} \cap (\Re_+ \times \mathbb{Z} \times \Re_+)$ for every $l \in L, i \in I_l, j \in L \cup J$, is denoted by $\mathbb{FPQ}$ and its binary MILP reformulation is $\mathcal{B}(\mathbb{FPQ})$. Thus we only discretize *the outgoing flows from each pool*. Since the range $[0, u_{lj}]$ of y_{lj} is typically of high order, we only consider the binary expansion of y_{lj} in order to avoid adding too many extra 0\1 variables. We assume that C_l and u_{lj} are integers, for all $l \in L, j \in L \cup J$, otherwise they can be replaced with $\lfloor C_l \rfloor$ and $\lfloor u_{lj} \rfloor$, respectively. In case of the ratio variables, although the q_{il}'s can be discretized into different intervals, for the ease of exposition, we assume that they all are uniformly discretized into $n \geq 1$ intervals of equal length within $[0,1]$. Unlike the flow discretization where restricting the y_{lj}'s to integer values within their respective bounds seemed like a reasonable method, in this case there is no clear intuition behind a suitable choice of n. In our computations, we will experiment with different values of n. Given a positive integer n, for every $l \in L, i \in I_l, j \in L \cup J$, we have (note the dependence on n)

$$\{(q_{il}, y_{lj}, v_{ilj})\colon v_{ilj} = q_{il}y_{lj}, nq_{il} \in [0,n] \cap \mathbb{Z}, y_{lj} \in [0, u_{lj}]\}$$

as the ratio discretization of $\mathcal{W}^{\mathbb{PQ}}_{ilj}$. Substituting $\mathcal{W}^{\mathbb{PQ}}_{ilj}$ with the above set for each i, l, j gives us $\mathbb{RPQ}_n$ and its MILP reformulations $\mathcal{U}(\mathbb{RPQ}_n)$ and $\mathcal{B}(\mathbb{RPQ}_n)$.

4.2.1 Flow discretization

We derive some valid inequalities for $\mathcal{B}(\mathbb{FPQ})$ by exploiting the fact that $\mathbb{FPQ}$ does not discretize the ratio variables and hence the domain of $q_{\cdot l}$ is still a simplex $\Delta_{|I_l|}$. Recall the sets $\mathcal{Q}_l$ and $\mathcal{Y}_l$ from §3.2.1 and denote

$$\mathcal{FQ}_l := \big\{(q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot}) \in \mathcal{Q}_l\colon y_{lj} \in \mathbb{Z} \ \ \forall j \in L \cup J\big\}$$

as the flow discretized counterpart of $\mathcal{Q}_l$. From Theorem 1 and integrality of the polytope $\{y_{l\cdot} \in \mathcal{Y}_l\colon y_{lj} \in \mathbb{Z} \ \ \forall j \in L \cup J\}$, it follows that $\mathrm{conv}(\mathcal{FQ}_l) = \mathrm{conv}(\mathcal{Q}_l)$. The convex hull of $\mathcal{B}(\mathcal{FQ}_l)$ though is nontrivial. An explicit description of *all the facets* of $\mathrm{conv}(\mathcal{B}(\mathcal{Y}_l))$, where

$$\mathcal{B}(\mathcal{Y}_l) := \left\{\zeta_{\cdot l\cdot}\colon \sum_{j \in L\cup J} \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{rlj} \leq C_l, \ \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{rlj} \leq u_{lj} \ \ \forall j \in L \cup J, \ \zeta_{rlj} \in \{0,1\} \ \ \forall r, j\right\},$$

and Theorem 1 would imply the convex hull of $\mathcal{B}(\mathcal{FQ}_l)$. However, the convex hull of $\mathcal{B}(\mathcal{Y}_l)$ in the ζ-space is unknown in general. We use suitable relaxations of this set to derive valid inequalities for $\mathcal{B}(\mathcal{FQ}_l)$.

Proposition 7 *For every $l \in L, j \in L \cup J$, let $\mathfrak{T}_{lj}$ be a subset of $\{1, 2, \ldots, \ell(u_{lj})\}$ such that $u_{lj} = \sum_{r \in \mathfrak{T}_{lj}} 2^{r-1}$. Then for every $l \in L$, we have $\mathrm{conv}(\mathcal{B}(\mathcal{FQ}_l)) \subseteq \mathcal{T}_l$ where*

$$\begin{aligned}\mathcal{T}_l := \Big\{(q_{\cdot l}, y_{l\cdot}, v_{\cdot l\cdot}, \zeta_{\cdot l\cdot}, \nu_{\cdot l\cdot})\colon q_{\cdot l} \in \Delta_{|I_l|}, \ y_{lj} = \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{rlj} \ \ \forall j, \ v_{ilj} = \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\nu_{rilj} \ \ \forall i, j \\ 0 \leq \nu_{rilj} \leq q_{il} \ \ \forall i, j, r, \quad \sum_{i \in I_l} \nu_{irlj} = \zeta_{rlj} \ \ \forall j, r \\ \nu_{rilj} + \sum_{r' \in \mathfrak{T}_{lj}\colon r' > r} \nu_{r'ilj} \ \leq \ |\{r' \in \mathfrak{T}_{lj}\colon r' > r\}|\, q_{il} \quad \forall i \in I_l, r \notin \mathfrak{T}_{lj}\Big\}.\end{aligned}$$

Furthermore if $\sum_j u_{lj} \leq C_l$ *then* $\text{conv}(\mathcal{B}(\mathcal{FQ}_l)) = \mathcal{T}_l$.

Proof Dropping the capacity constraint $\sum_j y_{lj} \leq C_l$ from $\mathcal{Y}_l$ gives us $\mathcal{Y}_l \subseteq \prod_j [0, u_{lj}]$ and hence $\mathcal{B}(\mathcal{Y}_l) \subseteq \prod_j \mathcal{B}([0, u_{lj}])$ where $\mathcal{B}([0, u_{lj}]) := \{\zeta_{\cdot lj} \colon \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{rlj} \leq u_{lj},\, \zeta_{rlj} \in \{0,1\}\ \ \forall r\}$. Define $\widetilde{\mathcal{T}}_{lj} := \{(q_{\cdot l}, y_{lj}, v_{\cdot lj}) \colon q_{\cdot l} \in \Delta_{|I_l|}, y_{lj} \in [0, u_{lj}] \cap \mathbb{Z}, v_{ilj} = q_{il} y_{lj}\ \forall i \in I_l\}$ for all $l \in L, j \in L \cup J$. Clearly, $\mathcal{FQ}_l \subseteq \cap_j \widetilde{\mathcal{T}}_{lj}$ and hence $\text{conv}(\mathcal{B}(\mathcal{FQ}_l)) \subseteq \text{conv}(\cap_j \mathcal{B}(\widetilde{\mathcal{T}}_{lj}))$.

The nontrivial facets of $\mathcal{B}([0, u_{lj}])$ are given in [31, 33] and are known to be the minimal cover inequalities $\zeta_{rlj} + \sum_{r' \in \mathfrak{T}_{lj} \colon r' > r} \zeta_{r'lj} \leq |\{r' \in \mathfrak{T}_{lj} \colon r' > r\}|$ for all $r \in \{1, \ldots, \ell(u_{lj})\} \setminus \mathfrak{T}_{lj}$. Theorem 1 then implies that the convex hull of $\mathcal{B}(\widetilde{\mathcal{T}}_{lj})$ is equal to its level-1 RLT relaxation. Using $\text{conv}(\mathcal{B}(\widetilde{\mathcal{T}}_{lj}))$ for all j and applying Corollary 1 to $\cap_j \mathcal{B}(\widetilde{\mathcal{T}}_{lj})$ leads to $\text{conv}(\cap_j \mathcal{B}(\widetilde{\mathcal{T}}_{lj})) = \cap_j \text{RLT1}(\mathcal{B}(\widetilde{\mathcal{T}}_{lj}))$, and the latter is exactly the proposed relaxation $\mathcal{T}_l$. Finally, if the value of C_l is trivial, then it follows from our derivation of $\mathcal{T}_l$ that $\text{conv}(\mathcal{B}(\mathcal{FQ}_l))$ is equal to $\mathcal{T}_l$. □

The relaxation in Proposition 7 can be strengthened using the following family of valid inequalities that take into account the pool capacity constraint.

Proposition 8 *Denote* $\beta_l := \max_j \ell(u_{lj})$. *The inequality*

$$\sum_{r=t}^{\beta_l} \sum_{j \colon r \leq \ell(u_{lj})} 2^{r-t} \nu_{rilj} \ \leq \ \left\lfloor \frac{C_l}{2^{t-1}} \right\rfloor q_{il}$$

is valid to $\text{conv}(\mathcal{B}(\mathcal{FQ}_l))$ *for all* $i \in I_l$ *and* $t = 1, 2, \ldots, \beta_l$.

Proof The capacity constraint $\sum_j \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{rlj} \leq C_l$ can be rearranged to

$$\sum_{r=1}^{\beta_l} \sum_{j \colon r \leq \ell(u_{lj})} 2^{r-1}\zeta_{rlj} \leq C_l. \tag{34}$$

Inequality (34) represents a divisible knapsack $\sum_{r=1}^{\beta_l} 2^{r-1}\zeta'_r \leq C_l$ upon the variable substitution $\zeta'_r = \sum_j \zeta_{rlj}$. Marcotte [47] presents the convex hull of such divisible knapsacks using β_l rounding inequalities:

$$\sum_{r=1}^{\beta_l} \left\lfloor \frac{2^{r-1}}{2^{t-1}} \right\rfloor \zeta'_r \ \leq \ \left\lfloor \frac{C_l}{2^{t-1}} \right\rfloor \quad \forall t = 1, 2, \ldots, \beta_l.$$

Back substituting for ζ'_r gives us valid inequalities for (34) (but not the convex hull since we relaxed the bound $\zeta'_r \leq |\{j \colon r \leq \ell(u_{lj})\}|$). A subsequent application of Theorem 1 with the simplex $\Delta_{|I_l|}$ yields the proposed inequalities. □

5 Computational Experiments

In this section we report computational results on several test instances of the pooling problem. The general approach is to solve the original pooling problem using a state-of-the-art global solver and compare the lower and upper bounds after a specified time limit with the lower bounds from §3 and upper bounds from §4. In our experiments, we do not implement our discretization strategies as part of a heuristic in solving the pooling problem. We simply want to determine which discretization strategy empirically seems to work best on the pooling problem. Once we have a good enough understanding of a suitable set of variables to discretize, then we can possibly use dynamic discretization, by iteratively refining the level of discretization, as a heuristic in a branch-and-bound algorithm. We leave this work for future research; see Kolodziej et al. [41] for one computational study of dynamically updating base-10 discretizations of mixed integer variants of the pooling problem.

For flow discretization, we discretized y_{lj} within its bounds $[0, u_{lj}]$ for all $l \in L, j \in L \cup J$ and considered only the binary reformulation MILP $\mathcal{B}(\cdot)$. Ratio and spec discretizations were tested for $n \in \{1, 2, 4, 7, 15, 31\}$. Clearly as n increases, there is a tradeoff between finding good feasible solutions versus being unable to solve the model to optimality due to its large size. In our preliminary computations, we did not observe any significant benefit with the $\mathcal{L}(\cdot)$ model. Based on the relative performance of the MILPs, we report $\mathcal{U}(\cdot)$ for $n \in \{1, 2, 4\}$, whereas for $n \in \{7, 15, 31\}$, we report $\mathcal{B}(\cdot)$.

We used `CPLEX 12.6` as the LP and MILP solver and `BARON 13.1` as the BLP and MIBLP solver. We used `SNOPT 7.2` as the NLP solver with `BARON`. `BARON` was run with a time limit of 6 hours on the NEOS server whereas `CPLEX` was run with a time limit of 1 hour on a Linux machine having a 64-bit x86 processor and 32GB of RAM. Since `BARON` is a branch-and-cut based global solver whose algorithm finds feasible solutions among many other things, such as tight bounds via node relaxations, variable bounding tightening, branching decisions etc., it is impossible to know exactly how much time was spent by `BARON` in finding feasible solutions. Hence, for the sake of fair comparison, we gave `BARON` a much longer time limit. To ensure numerical consistency between `BARON` and `CPLEX`, we used the following algorithmic parameters: `feasibility tolerance` $= 10^{-6}$, `integrality tolerance` $= 10^{-5}$, `relative optimality gap` $= 0.01\%$, and `absolute optimality gap` $= 10^{-3}$. For `CPLEX`, we also set `Threads` $= 1$ and `MIPEmphasis` $= 1$ (feasibility). The `MIPEmphasis` parameter is used to aid `CPLEX` in finding good feasible solutions at the expense of proof of optimality. We do not know of a similar parameter for `BARON`. Valid inequalities of §4.2.1 were added as user cuts to `CPLEX`.

Test instances. The pooling instances commonly used in literature mostly comprise the small-scale problems proposed many years ago [1, 13, 36]. Since these problems are solved in a matter of seconds by `BARON`, they are not of particular interest to us and are only used for demonstrating the strength of the Lagrangian relaxations in §5.1. Also, we test more extensively on the standard problem than the generalized problem since the former already seems to be a computationally hard problem to solve. We test on 70 randomly generated medium- to large-scale instances of the standard pooling problem - 20 of these were created in [5] and are labeled `std*` and the remaining 50 were created in [21] and are labeled `randstd*`. There are 10 generalized pooling instances[2] in our test set - 3 of these were used in [49] and we randomly generated 7 instances of the time indexed pooling problem described in §2.4.1. All 10 are formulated as MIBLPs. In the instances of [49], the spec tracking constraints (4a) are formulated as $\eta_{lk}\left[\sum_{i\in I}\lambda_{ik}y_{il}+\sum_{l'\in L}p_{l'k}y_{l'l}\right] = p_{lk}\sum_{j\in L\cup J}y_{lj}$ where η_{lk} is an absorption coefficient of spec k at pool l. Hence, to write the $\mathbb{PQ}$ formulation of this problem, we need to define ratio variables q^{τ}_{il} along each path τ such that q^{τ}_{il} denotes the ratio of incoming flow to l along path τ starting from input i. This makes the formulation extremely large in size due to its path dependency. Similar reasoning prevails for the time-indexed pooling problems. Indeed while experimenting on the generalized instances, the $\mathbb{PQ}$ formulation and its corresponding relaxations and discretizations exhibited a poor performance owing to their extremely large size. Hence we consider the $\mathbb{P}$ formulation for the 10 generalized instances.

5.1 Results for relaxations

As expected from Proposition 4, the lower bound from $\mathscr{PQ}$ is far superior than that due to $\mathscr{P}$. First, we tested the two new Lagrangian relaxations proposed in §3.4. Since our goal is to simply test the quality of these lower bounds, we formulated an exponential-sized LP for each of these Lagrangians instead of obtaining the lower bounds using an iterative method such as the subgradient algorithm. These LP formulations arise from disjunctive programming after observing that $\text{conv}(\check{\mathcal{S}})$ and $\text{conv}(\check{\mathcal{S}})$ can each be written as the convex hull of the union of finitely many polytopes, where each polytope is obtained by fixing q to an extreme point of $\prod_{l\in L}\Delta_{|I_l|}$. Hence each LP has $\mathcal{O}(|I|^{|L|})$ many variables, which means that we can computationally test these disjunctive representations for only small-scale instances. Table 2 reports the performance of z^{LAG3}; we observed that z^{LAG2} was always equal to z^{pq}. Here the % gap closed by z^{LAG3} is equal to $100\times\left(\frac{z^{\text{LAG3}}-z^{pq}}{z^{*}-z^{pq}}\right)$. We note that our third Lagrangian relaxation provides a significant improvement over $\mathscr{PQ}$ on most of the instances. Hence we expect that it will perform quite well in practice, if some strong valid inequalities for $\text{conv}(\check{\mathcal{S}})$ can be separated in polynomial time. For `RT2`, we have $z^{\text{LAG3}} < z^{pq}$ which can happen as mentioned towards the end of §3.4.

For the medium- and large-scaled standard instances, we tested the three RLT relaxations – $\mathscr{R}_1, \mathscr{R}_2, \mathscr{R}_3$, that were proposed in §3.3. $\mathscr{R}_1$ did not improve upon the lower bound of $\mathscr{PQ}$ (quite possibly due to the fact that on these instances, the variable bounds u_{il}'s are mostly dominated by the capacity constraints) whereas $\mathscr{R}_3$ was too large in size and did not work well in practice. $\mathscr{R}_2$ did increase the lower bound on some of the instances as noted below, but also took considerably longer to solve owing to its large size. Table 3 reports these lower bounds, the % gap closed by them and the amount of CPU time in comparison to the pq-relaxation. To compute % gap closed,

[2] Some generalized instances can also be found in [4] but in our experience the pq-formulations of these instances were solved by `BARON` in less than 15 minutes and hence seem to be relatively ease.

#	z^{pq}	z^{LAG3}	% gap closed to z^*
Haverly1	-500	-400	100
Haverly2	-1000	-600	100
Haverly3	-800	-793.75	12.50
BenTal4	-550	-450	100
Bental5	-3500	-3500	-
Adhya1	-840.27	-688.56	52.23
Adhya2	-574.78	-565.85	35.74
Adhya3	-574.78	-568.55	45.38
Adhya4	-961.93	-900.62	72.75
RT2	-6030.34	-6691.88	-

Table 2 Lower bounds from Lagrangian relaxation.

we use the best-known upper bound on z^* (including results of next section) in case z^* itself is unknown.

#	$\mathscr{PQ}$		$\mathscr{R}_2$		% gap closed
	z^{pq}	Time (sec.)	$z^{\mathscr{R}_2}$	Time (sec.)	to z^*
stdA0	-37772.75	0.19	-37760.08	0.20	0.65
stdA1	-31516.93	0.11	-31503.61	0.14	0.59
stdA2	-23898.81	0.25	-23884.36	0.38	1.69
stdA3	-42066.64	0.22	-42027.12	1	1.50
randstd12	-58120.52	2	-57970.40	57	24.73
randstd16	-65639.73	4	-65517.76	16	19.01
randstd25	-75952.80	17	-75918.04	31	2.91
randstd27	-57084.07	7	-56994.45	38	5.62
randstd31	-104796.77	15	-104773.07	39	1.68
randstd32	-98374.73	15	-98249.31	110	7.77
randstd37	-94255.66	35	-93903.92	54	16.40
randstd41	-89315.91	10	-89276.38	860	0.63
randstd42	-99160.20	11	-98997.69	1089	2.03
randstd43	-108040.19	15	-107567.58	329	10.57
randstd47	-108611.61	16	-108512.79	912	1.63
randstd50	-143113.27	2	-142725.99	924	5.28
randstd54	-88157.35	15	-87767.20	510	36.82
randstd59	-159035.34	3	-159000.87	401	1.63

Table 3 Lower bounds from RLT relaxation.

5.2 Results for discretizations

We first compare the quality of the best feasible solution from discretization against those obtained from global solve with BARON (for 6hr), local solve with SNOPT (for 1hr) and the flow augmentation heuristic of Alfaki and Haugland [3]. This gives us an estimate of how well discretization methods might perform if implemented as a heuristic in a branch-and-cut algorithm. For each instance $\mathscr{I}$ and method $\mathscr{M}$, we report the percentage gap calculated as $\omega_{\mathscr{M}}(\mathscr{I}) = 100 \times |1 - \frac{\nu_{\mathscr{M}}(\mathscr{I})}{\ell(\mathscr{I})}|$, where $\ell(\mathscr{I})$ is the lower bound obtained from BARON (after 6hr) and $\nu_{\mathscr{M}}(\mathscr{I})$ is the upper bound returned upon termination of $\mathscr{M}$. We observed that on standard instances, the performance of $\mathbb{PQ}$ formulation, for both discretized and non-discretized models, was far superior than that of $\mathbb{P}$[3]. However for the generalized instances, this dominance did not hold. Accordingly, the results are summarized in Tables 4 and 5. In Table 5, a † indicates that only a finite upper bound was returned by the solver without finding a feasible solution whereas $-$ means that neither any feasible solution nor any finite upper bound was found within the time limit. If a method produces a feasible solution that is provably optimal, i.e. has 0.01% gap, then the total solution time in seconds for this method is noted in parenthesis.

[3] Amongst the various choices for discretizing $\mathbb{P}$, flow discretization was by far the best choice but the solutions from solving $\mathcal{B}(\mathbb{FP})$ were still very poor in comparison to discretizing $\mathbb{PQ}$.

In Table 4, discretization of $\mathbb{PQ}$ provided better solutions than solving $\mathbb{PQ}$ itself mostly for the large-scale instances. There does not seem to be an obvious candidate for a good discretization model. Note that $\mathcal{U}(\mathbb{RPQ}_1)$ imposes the restriction that there is no mixing at pools. Hence if these MILPs yielded good solutions, then it may well be an artifact of the specific instance and may not work well in general. For the generalized instances (Table 5), the spec and ratio discretizations were mostly either provably infeasible or unable to find a solution within 1 hour. The $\mathbb{P}$ formulation and its discretizations performed better than its $\mathbb{PQ}$ counterparts for our randomly generated `Inst*` instances. This is perhaps to be expected because the *pq*-formulation of these time indexed problems is much larger in size than the *p*-formulation, as explained in § 2.4.1. $\mathcal{B}(\mathbb{FP})$ was able to find good quality feasible solutions in a shorter time on 5 out of 7 of these instances. `BARON` was unable to find feasible solutions while solving $\mathbb{P}$ or $\mathbb{PQ}$ in 5 instances (markd with a †). However we note that on `Inst6` and `Inst7`, none of our discretization models yielded a feasible solution nor did `CPLEX` return any finite upper bound. On the `meyer*` instances, solving $\mathbb{P}$ with `BARON` outperformed all discretization approaches.

#	Best of `BARON`, `SNOPT`, [3]	Best discretization of $\mathbb{PQ}$		#	Best of `BARON`, `SNOPT`, [3]	Best discretization of $\mathbb{PQ}$	
	% gap	% gap	MILP		% gap	% gap	MILP
stdA0	0.60	0.69	$\mathcal{B}(\mathbb{FPQ})$	randstd26	0.00	0.08	$\mathcal{B}(\mathbb{RPQ}_{15})$
stdA1	2.72	2.74	$\mathcal{B}(\mathbb{FPQ})$	randstd27	4.69	1.62	$\mathcal{B}(\mathbb{FPQ})$
stdA2	0.00	0.03	$\mathcal{B}(\mathbb{FPQ})$	randstd28	0.08	0.51	$\mathcal{B}(\mathbb{RPQ}_7)$
stdA3	0.96	0.64	$\mathcal{B}(\mathbb{FPQ})$	randstd29	1.30	1.59	$\mathcal{B}(\mathbb{FPQ})$
stdA4	3.44	4.22	$\mathcal{B}(\mathbb{FPQ})$	randstd30	5.73	0.79	$\mathcal{B}(\mathbb{RPQ}_{31})$
stdA5	1.26	2.39	$\mathcal{B}(\mathbb{RPQ}_{15})$	randstd31	2.26	1.62	$\mathcal{B}(\mathbb{FPQ})$
stdA6	0.67	1.01	$\mathcal{B}(\mathbb{FPQ})$	randstd32	13.00	2.75	$\mathcal{B}(\mathbb{RPQ}_7)$
stdA7	0.73	1.08	$\mathcal{B}(\mathbb{RPQ}_{15})$	randstd33	2.85	2.99	$\mathcal{B}(\mathbb{FPQ})$
stdA8	0.32	0.20	$\mathcal{B}(\mathbb{FPQ})$	randstd34	2.33	1.59	$\mathcal{B}(\mathbb{FPQ})$
stdA9	0.00	0.13	$\mathcal{B}(\mathbb{FPQ})$	randstd35	1.53	1.67	$\mathcal{B}(\mathbb{FPQ})$
stdB0	6.08	6.29	$\mathcal{B}(\mathbb{RPQ}_{15})$	randstd36	0.26	0.34	$\mathcal{B}(\mathbb{RPQ}_{15})$
stdB1	3.19	4.00	$\mathcal{B}(\mathbb{RPQ}_7)$	randstd37	2.10	2.30	$\mathcal{B}(\mathbb{RPQ}_7)$
stdB2	3.94	4.75	$\mathcal{B}(\mathbb{RPQ}_2)$	randstd38	8.71	3.86	$\mathcal{B}(\mathbb{FPQ})$
stdB3	5.18	0.72	$\mathcal{B}(\mathbb{RPQ}_1)$	randstd39	15.00	2.45	$\mathcal{B}(\mathbb{RPQ}_7)$
stdB4	0.10	0.11	$\mathcal{B}(\mathbb{RPQ}_2)$	randstd40	13.00	10.00	$\mathcal{B}(\mathbb{RPQ}_{15})$
stdB5	0.62	1.19	$\mathcal{B}(\mathbb{RPQ}_1)$	randstd41	15.00	13.00	$\mathcal{B}(\mathbb{FPQ})$
stdC0	29.00	20.00	$\mathcal{B}(\mathbb{RPQ}_7)$	randstd42	22.00	16.00	$\mathcal{B}(\mathbb{RPQ}_1)$
stdC1	40.00	22.00	$\mathcal{B}(\mathbb{RPQ}_1)$	randstd43	22.00	24.00	$\mathcal{B}(\mathbb{RPQ}_7)$
stdC2	31.00	13.00	$\mathcal{B}(\mathbb{RPQ}_1)$	randstd44	6.25	6.00	$\mathcal{B}(\mathbb{RPQ}_{15})$
stdC3	18.00	6.25	$\mathcal{B}(\mathbb{RPQ}_1)$	randstd45	9.40	2.83	$\mathcal{B}(\mathbb{RPQ}_1)$
randstd11	13.00	13.00	$\mathcal{B}(\mathbb{FPQ})$	randstd46	22.00	33.00	$\mathcal{B}(\mathbb{RPQ}_7)$
randstd12	0.58	2.26	$\mathcal{B}(\mathbb{FPQ})$	randstd47	34.00	25.00	$\mathcal{B}(\mathbb{RPQ}_7)$
randstd13	1.56	2.51	$\mathcal{B}(\mathbb{FPQ})$	randstd48	15.00	15.00	$\mathcal{B}(\mathbb{RPQ}_1)$
randstd14	0.23	0.36	$\mathcal{B}(\mathbb{RPQ}_{15})$	randstd49	40.00	28.00	$\mathcal{B}(\mathbb{RPQ}_1)$
randstd15	0.57	2.41	$\mathcal{B}(\mathbb{RPQ}_{15})$	randstd50	18.00	36.00	$\mathcal{B}(\mathbb{RPQ}_{15})$
randstd16	0.12	0.14	$\mathcal{B}(\mathbb{FPQ})$	randstd51	13.00	6.21	$\mathcal{B}(\mathbb{RPQ}_1)$
randstd17	1.19	1.30	$\mathcal{B}(\mathbb{RPQ}_7)$	randstd52	5.45	13.00	$\mathcal{B}(\mathbb{RPQ}_7)$
randstd18	0.78	1.23	$\mathcal{B}(\mathbb{FPQ})$	randstd53	7.04	5.73	$\mathcal{B}(\mathbb{RPQ}_7)$
randstd19	0.56	10.00	$\mathcal{B}(\mathbb{FPQ})$	randstd54	1.96	1.28	$\mathcal{B}(\mathbb{RPQ}_7)$
randstd20	0.12	0.19	$\mathcal{B}(\mathbb{FPQ})$	randstd55	6.49	18.00	$\mathcal{B}(\mathbb{RPQ}_1)$
randstd21	3.94	4.00	$\mathcal{B}(\mathbb{FPQ})$	randstd56	5.57	2.17	$\mathcal{B}(\mathbb{FPQ})$
randstd22	0.01	0.04	$\mathcal{B}(\mathbb{RPQ}_{31})$	randstd57	24.00	9.51	$\mathcal{B}(\mathbb{FPQ})$
randstd23	1.64	2.96	$\mathcal{B}(\mathbb{RPQ}_7)$	randstd58	4.89	3.82	$\mathcal{B}(\mathbb{FPQ})$
randstd24	0.12	0.76	$\mathcal{B}(\mathbb{FPQ})$	randstd59	12.00	3.07	$\mathcal{B}(\mathbb{FPQ})$
randstd25	0.88	2.35	$\mathcal{B}(\mathbb{RPQ}_7)$	randstd60	11.00	4.47	$\mathcal{B}(\mathbb{FPQ})$

Table 4 Discretizations for standard pooling problems. % gap of best feasible solution and corresponding MILP model for $\mathbb{PQ}$ formulation of each instance.

Tables 4 and 5 do not provide any *a priori* information on which discretization to choose given a new instance, especially if the new instance is of the standard type where there does not seem to be a overwhelmingly best model. To compare the overall quality of the solutions returned by the different discretizations of standard instances, we plot performance profiles for the different MILPs in Figure 3. We see that $\mathcal{B}(\mathbb{RPQ}_7)$ provides the most dominant profile followed by $\mathcal{B}(\mathbb{FPQ})$. Although `BARON` produces best solutions on most number of instances (recall that global solve was given 6 hours), its performance quickly deteriorates because it gives poor solutions on the large-scale standard instances.

#	$\mathbb{P}$	Best discretization of $\mathbb{P}$		$\mathbb{PQ}$	Best discretization of $\mathbb{PQ}$	
	% gap	% gap	MILP	% gap	% gap	MILP
meyer4	0.01 (6562)	2.38	$\mathcal{B}(\mathbb{FP})$	Not	applicable	
meyer10	26.02	68.76	$\mathcal{B}(\mathbb{FP})$	Not	applicable	
meyer15	36.52	51.68	$\mathcal{B}(\mathbb{FP})$	Not	applicable	
Inst1	969.20†	9.68	$\mathcal{B}(\mathbb{FP})$	969.20†	9.68	$\mathcal{B}(\mathbb{FPQ})$
Inst2	1242.26†	0.17	$\mathcal{B}(\mathbb{FP})$	1242.26†	3.14	$\mathcal{B}(\mathbb{RPQ}_7)$
Inst3	0.01 (7585)	0.01 (35)	$\mathcal{B}(\mathbb{FP})$	9.33	0.01 (110)	$\mathcal{B}(\mathbb{FPQ})$
Inst4	0.01 (21594)	0.01 (149)	$\mathcal{B}(\mathbb{FP})$	0.76	0.01 (817)	$\mathcal{B}(\mathbb{FPQ})$
Inst5	462.58†	13.98	$\mathcal{B}(\mathbb{FP})$	462.58†	–	–
Inst6	391.14†	–	–	391.14†	–	–
Inst7	382.82†	–	–	382.82†	–	–

Table 5 Discretizations for generalized pooling problems. % gap of best feasible solution and corresponding MILP model for each instance.

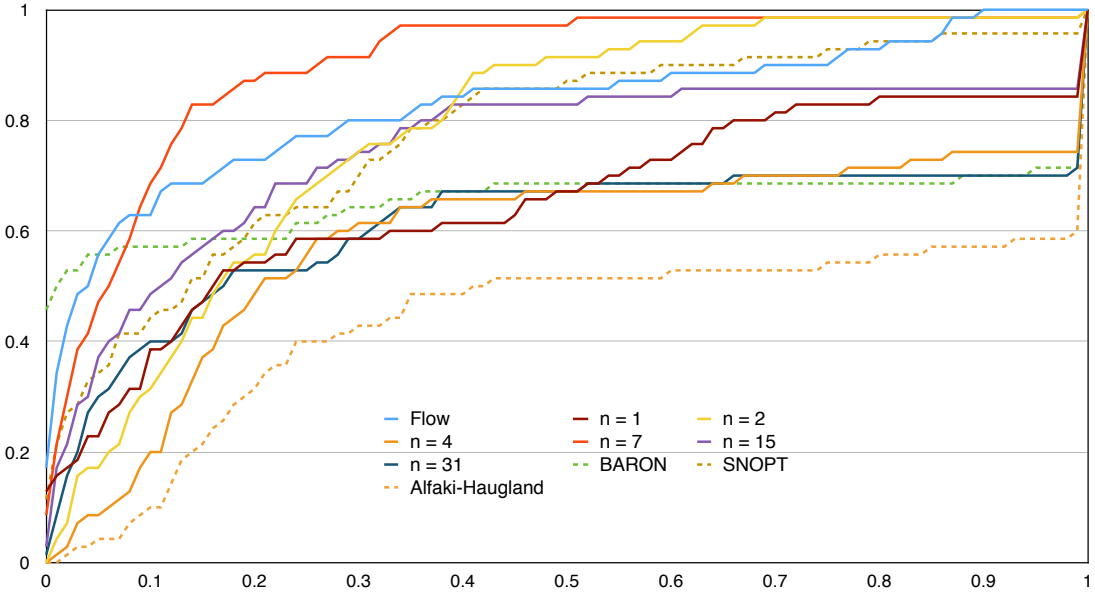


Fig. 3 Performance profiles of the best feasible solutions for the standard instances. Only most significant discretization models shown; values of n denote discretization level for $\mathbb{RPQ}_n$.

Next, we analyze the relative effort with which `CPLEX` is able to find feasible solutions of the different MILPs. We focus on the standard instances since on generalized instances, ratio and spec discretizations did not perform very well. Since we provided a starting solution $y = \mathbf{0}$, `CPLEX` was able to obtain a improved feasible solution after spending a small time solving its root node heuristic. Hence the first nontrivial solution was found by `CPLEX` normally within a few seconds. For each discretization, Table 6 presents the geometric average of the % gap $\omega_{\mathscr{M}}(\mathscr{I})$ for the best solution versus the geometric average of the CPU time at which `CPLEX` found this best solution. A good discretization model will be one that produces solutions with smallest % gap in shortest amount of time. First observe that the solutions from the discretization of $\mathbb{P}$ formulations are found

Discretization	Time (sec.)	Gap (%)	Discretization	Time (sec.)	Gap (%)
$\mathcal{B}(\mathbb{FPQ})$	511	3.29	$\mathcal{B}(\mathbb{FP})$	7.27	57.23
$\mathcal{U}(\mathbb{RPQ}_1)$	161	4.37	$\mathcal{U}(\mathbb{SP}_1)$	0.51	69.54
$\mathcal{U}(\mathbb{RPQ}_2)$	902	4.30	$\mathcal{U}(\mathbb{SP}_2)$	0.83	69.54
$\mathcal{U}(\mathbb{RPQ}_4)$	965	5.57	$\mathcal{U}(\mathbb{SP}_4)$	1.19	69.54
$\mathcal{B}(\mathbb{RPQ}_7)$	1462	3.42	$\mathcal{B}(\mathbb{SP}_7)$	1.02	69.54
$\mathcal{B}(\mathbb{RPQ}_{15})$	1202	4.83	$\mathcal{B}(\mathbb{SP}_{15})$	2.53	68.00
$\mathcal{B}(\mathbb{RPQ}_{31})$	1411	4.26	$\mathcal{B}(\mathbb{SP}_{31})$	1.25	69.54

Table 6 Geometric averages for optimality gap of best solution and time at which the solution was found.

very quickly but CPLEX is unable to improve upon them and hence the MILP solutions for $\mathbb{P}$ are extremely poor in quality, as was mentioned before. For $\mathbb{PQ}$, $\mathcal{U}(\mathbb{RPQ}_1)$ provides good solutions very quickly. Since $\mathcal{U}(\mathbb{RPQ}_1)$ discretizes each $q_{il} \in \{0, 1\}$, the short time required for finding these solutions is perhaps to be expected. However, it is surprising that such a naive discretization gives fairly good solutions on random instances. The next best MILP in terms of finding solutions quickly is $\mathcal{B}(\mathbb{FPQ})$ and this model also produces the best quality solutions on average (albeit $\mathcal{B}(\mathbb{RPQ}_7)$ is a close second).

Finally, we remark that an alternate discretization strategy, which does not depend on explicitly discretizing the variables in the pooling problem, was analytically studied by Dey and Gupte [21] recently and these MILP approximations produced very good feasible solutions on the 70 standard test instances. Our discretization approaches improve the previous best known upper bounds [21, Appendix D] on 6 out of these 70 instances, as noted in Table 7.

#	Lower Bound	Upper Bound		
	BARON	Previous Best [21]	New	Discretization
stdA3	-39681.80	-39301.98	-39429.60	$\mathcal{B}(\mathbb{FPQ})$
stdA8	-30666.87	-30569.03	-30604.28	$\mathcal{B}(\mathbb{FPQ})$
randstd27	-56406.56	-55213.20	-55490.76	$\mathcal{B}(\mathbb{FPQ})$
randstd30	-81110.45	-78505.72	-80472.19	$\mathcal{B}(\mathbb{RPQ}_{31})$
randstd34	-90621.44	-88506.19	-89178.30	$\mathcal{B}(\mathbb{FPQ})$
randstd51	-137423.00	-126741.65	-128894.46	$\mathcal{U}(\mathbb{RPQ}_1)$

Table 7 Improved upper bounds.

6 Conclusions

In this work, we first described the pooling problem, presented alternate formulations for it along with their properties, and gave some new results on the various polyhedral relaxations for this problem. We discussed different discretization methods to obtain inner approximations and presented some valid inequalities for the MILPs. These ideas were computationally tested on random instances. On the lower bounding side, the Lagrangian relaxations seem to be more promising than the RLT relaxations; however there still remains the significant hurdle of obtaining tractable polyhedral formulations or separation algorithms for the former. Many of our discretization models were able to find good quality feasible solutions in a relatively short amount of time. One can possibly further improve the performance of these discretizations by fine tuning the heuristics in a MILP solver or by using dynamic discretization strategies. Our experiments suggest that discretization seems to be a promising approach especially for large-scale standard or generalized pooling problems.

References

1. Adhya, N., Tawarmalani, M., Sahinidis, N.: A Lagrangian approach to the pooling problem. Industrial and Engineering Chemistry Research **38**(5), 1956–1972 (1999)
2. Al-Khayyal, F., Falk, J.: Jointly constrained biconvex programming. Mathematics of Operations Research **8**(2), 273–286 (1983)
3. Alfaki, M., Haugland, D.: A cost minimization heuristic for the pooling problem. Annals of Operations Research pp. 1–15 (2013). DOI 10.1007/s10479-013-1433-1
4. Alfaki, M., Haugland, D.: A multi-commodity flow formulation for the generalized pooling problem. Journal of Global Optimization **56**(3), 917–937 (2013)
5. Alfaki, M., Haugland, D.: Strong formulations for the pooling problem. Journal of Global Optimization **56**(3), 897–916 (2013)
6. Almutairi, H., Elhedhli, S.: A new Lagrangean approach to the pooling problem. Journal of Global Optimization **45**(2), 237–257 (2009)
7. Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., Mladenović, N.: Pooling problem: Alternate formulations and solution methods. Management Science **50**(6), 761–776 (2004)
8. Audet, C., Hansen, P., Jaumard, B., Savard, G.: A symmetrical linear maxmin approach to disjoint bilinear programming. Mathematical Programming **85**(3), 573–592 (1999)

9. Baker, T., Lasdon, L.: Successive linear programming at Exxon. Management Science pp. 264–274 (1985)
10. Bao, X., Sahinidis, N., Tawarmalani, M.: Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. Optimization Methods and Software, 24 **4**(5), 485–504 (2009)
11. Bao, X., Sahinidis, N., Tawarmalani, M.: Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. Mathematical Programming **129**(1), 129–157 (2011)
12. Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., Mahajan, A.: Mixed-integer nonlinear optimization. Acta Numerica **22**, 1–131 (2013). DOI 10.1017/S0962492913000032
13. Ben-Tal, A., Eiger, G., Gershovitz, V.: Global minimization by reducing the duality gap. Mathematical Programming **63**(1), 193–212 (1994)
14. Biegler, L., Grossmann, I., Westerberg, A.: Systematic methods for chemical process design. International Series in the Physical and Chemical Engineering Sciences. Prentice Hall (1997)
15. Bley, A., Boland, N., Froyland, G., Zuckerberg, M.: Solving mixed integer nonlinear programming problems for mine production planning with stockpiling (2012). Preprint at `http://www.optimization-online.org/DB_HTML/2012/11/3674.html`
16. Bodington, C., Baker, T.: A history of mathematical programming in the petroleum industry. Interfaces **20**(4), 117–127 (1990)
17. Burer, S., Letchford, A.N.: Non-convex mixed-integer nonlinear programming: a survey. Surveys in Operations Research and Management Science **17**(2), 97–106 (2012)
18. Burer, S., Saxena, A.: The MILP road to MIQCP. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, *IMA Volumes in Mathematics and its Applications*, vol. 154, pp. 373–405. Springer (2012)
19. Crama, Y.: Concave extensions for nonlinear 0–1 maximization problems. Mathematical Programming **61**(1-3), 53–60 (1993)
20. D'Ambrosio, C., Linderoth, J., Luedtke, J.: Valid inequalities for the pooling problem with binary variables. In: Günlük, O., Woeginger, G. (eds.) Integer Programming and Combinatorial Optimization, *Lecture Notes in Computer Science*, vol. 6655, pp. 117–129. Springer (2011)
21. Dey, S., Gupte, A.: Analysis of MILP techniques for the pooling problem. Operations Research ***to appear*** (2015). Preprint at `http://people.clemson.edu/~agupte/PoolingApproxIP.pdf`
22. Floudas, C., Aggarwal, A.: A decomposition strategy for global optimum search in the pooling problem. ORSA Journal on Computing **2**(3), 225–235 (1990)
23. Foulds, L., Haugland, D., Jörnsten, K.: A bilinear approach to the pooling problem. Optimization **24**(1), 165–180 (1992)
24. Frimannslund, L., El Ghami, M., Alfaki, M., Haugland, D.: Solving the pooling problem with lmi relaxations. In: TOGO10 – GLOBAL OPTIMIZATION WORKSHOP, pp. 51–54 (2010)
25. Frimannslund, L., Gundersen, G., Haugland, D.: Sensitivity analysis applied to the pooling problem. Tech. Rep. 380, University of Bergen (2008)
26. Furman, K., Androulakis, I.: A novel MINLP-based representation of the original complex model for predicting gasoline emissions. Computers & Chemical Engineering **32**(12), 2857–2876 (2008)
27. Gounaris, C., Misener, R., Floudas, C.: Computational comparison of piecewise-linear relaxations for pooling problems. Industrial & Engineering Chemistry Research **48**(12), 5742–5766 (2009)
28. Greenberg, H.: Analyzing the pooling problem. ORSA Journal on Computing **7**(2), 205–217 (1995)
29. Günlük, O., Lee, J., Leung, J.: A polytope for a product of real linear functions in 0/1 variables. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming, *IMA Volumes in Mathematics and its Applications*, vol. 154, pp. 513–529. Springer (2012)
30. Gupte, A.: Mixed integer bilinear programming with applications to the pooling problem. Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA (2012). URL `https://smartech.gatech.edu/handle/1853/45761`
31. Gupte, A.: Convex hull of superincreasing knapsacks and lexicographic orderings (2013). *under review*, Preprint at Optimization Online and arXiv
32. Gupte, A.: Bilinear programming with simplicial constraints (2015). URL `http://people.clemson.edu/~agupte/BilinSimpl.pdf`. Working paper
33. Gupte, A., Ahmed, S., Cheon, M., Dey, S.: Solving mixed integer bilinear problems using MILP formulations. SIAM Journal on Optimization **23**(2), 721–744 (2013)
34. Hasan, M., Karimi, I.: Piecewise linear relaxation of bilinear programs using bivariate partitioning. AIChE Journal **56**(7), 1880–1893 (2010)

35. Haugland, D.: The hardness of the pooling problem. In: Casado, L., García, I., Hendrix, E. (eds.) Proceedings of the XII GLOBAL OPTIMIZATION WORKSHOP: Mathematical and Applied Global Optimization (MAGO), pp. 29–32. Malaga (2014)
36. Haverly, C.: Studies of the behavior of recursion for the pooling problem. ACM SIGMAP Bulletin **25**, 19–28 (1978)
37. Kallrath, J.: Solving planning and design problems in the process industry using mixed integer and global optimization. Annals of Operations Research **140**(1), 339–373 (2005)
38. Karuppiah, R., Furman, K., Grossmann, I.: Global optimization for scheduling refinery crude oil operations. Computers & Chemical Engineering **32**(11), 2745–2766 (2008)
39. Karuppiah, R., Grossmann, I.: Global optimization for the synthesis of integrated water systems in chemical processes. Computers and Chemical Engineering **30**(4), 650–673 (2006)
40. Kim, S., Kojima, M.: Second order cone programming relaxation of nonconvex quadratic optimization problems. Optimization Methods and Software **15**(3-4), 201–224 (2001)
41. Kolodziej, S.P., Grossmann, I.E., Furman, K.C., Sawaya, N.W.: A discretization-based approach for the optimization of the multiperiod blend scheduling problem. Computers & Chemical Engineering **53**, 122–142 (2013)
42. Lee, S., Grossmann, I.: Global optimization of nonlinear generalized disjunctive programming with bilinear equality constraints: applications to process networks. Computers & chemical engineering **27**(11), 1557–1575 (2003)
43. Li, X., Armagan, E., Tomasgard, A., Barton, P.I.: Stochastic pooling problem for natural gas production network design and operation under uncertainty. AIChE Journal **57**(8), 2120–2135 (2011)
44. Li, X., Tomasgard, A., Barton, P.I.: Decomposition strategy for the stochastic pooling problem. Journal of Global Optimization **54**(4), 765–790 (2012)
45. Liberti, L., Pantelides, C.: An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. Journal of Global Optimization **36**(2), 161–189 (2006)
46. Luedtke, J., Namazifar, M., Linderoth, J.: Some results on the strength of relaxations of multilinear functions. Mathematical Programming **136**(2), 325–351 (2012)
47. Marcotte, O.: The cutting stock problem and integer rounding. Mathematical Programming **33**(1), 82–92 (1985)
48. McCormick, G.: Computability of global solutions to factorable nonconvex programs: Part I. convex underestimating problems. Mathematical Programming **10**(1), 147–175 (1976)
49. Meyer, C., Floudas, C.: Global optimization of a combinatorially complex generalized pooling problem. AIChE journal **52**(3), 1027–1037 (2006)
50. Misener, R., Floudas, C.: Advances for the pooling problem: Modeling, global optimization, and computational studies. Appl. Comput. Math **8**(1), 3–22 (2009)
51. Misener, R., Floudas, C.: Global optimization of large-scale generalized pooling problems: Quadratically constrained MINLP models. Industrial & Engineering Chemistry Research **49**(11), 5424–5438 (2010)
52. Misener, R., Gounaris, C., Floudas, C.: Mathematical modeling and global optimization of large-scale extended pooling problems with the (EPA) complex emissions constraints. Computers & Chemical Engineering **34**(9), 1432–1456 (2010)
53. Misener, R., Smadbeck, J.B., Floudas, C.A.: Dynamically generated cutting planes for mixed-integer quadratically constrained quadratic programs and their incorporation into GloMIQO 2. Optimization Methods and Software **30**(1), 215–249 (2015)
54. Misener, R., Thompson, J., Floudas, C.: APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. Computers & Chemical Engineering **35**, 876–892 (2011)
55. Nemhauser, G., Wolsey, L.: Integer and combinatorial optimization, *Discrete Mathematics and Optimization*, vol. 18. Wiley-Interscience (1988)
56. Nishi, T.: A semidefinite programming relaxation approach for the pooling problem. Master's thesis, Department of Applied Mathematics and Physics, Kyoto University (2010). URL `http://www-optima.amp.i.kyoto-u.ac.jp/result/masterdoc/21nishi.pdf`
57. Pham, V., Laird, C., El-Halwagi, M.: Convex hull discretization approach to the global optimization of pooling problems. Industrial and Engineering Chemistry Research **48**(4), 1973–1979 (2009)
58. Quesada, I., Grossmann, I.: Global optimization of bilinear process networks with multicomponent flows. Computers and Chemical Engineering **19**(12), 1219–1242 (1995)
59. Realff, M., Ahmed, S., Inacio, H., Norwood, K.: Heuristics and upper bounds for a pooling problem with cubic constraints. In: Foundations of Computer-Aided Process Operations. Savannah, GA (2012). URL `http://focapo.cheme.cmu.edu/2012/proceedings/data/papers/056.pdf`

60. Rikun, A.: A convex envelope formula for multilinear functions. Journal of Global Optimization **10**(4), 425–437 (1997)
61. Ruiz, J., Grossmann, I.: Exploiting vector space properties to strengthen the relaxation of bilinear programs arising in the global optimization of process networks. Optimization Letters **5**(1), 1–11 (2011)
62. Ruiz, M., Briant, O., Clochard, J., Penz, B.: Large-scale standard pooling problems with constrained pools and fixed demands. Journal of Global Optimization pp. 1–18 (2012)
63. Sherali, H., Adams, W.: A reformulation-linearization technique for solving discrete and continuous nonconvex problems, *Nonconvex Optimization and its Applications*, vol. 31. Kluwer Academic Publishers (1998)
64. Sherali, H.D.: Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. Acta mathematica vietnamica **22**(1), 245–270 (1997)
65. Smith, E.M., Pantelides, C.C.: A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex minlps. Computers & Chemical Engineering **23**(4), 457–478 (1999)
66. Tardella, F.: Existence and sum decomposition of vertex polyhedral convex envelopes. Optimization Letters **2**(3), 363–375 (2008)
67. Tawarmalani, M., Sahinidis, N.: Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications. Kluwer Academic Publishers (2002)
68. Vielma, J., Nemhauser, G.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Mathematical Programming **128**, 49–72 (2011)
69. Visweswaran, V.: MINLP: Applications in blending and pooling problems. In: Floudas, C., Pardalos, P. (eds.) Encyclopedia of Optimization, pp. 2114–2121. Springer (2009)