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 in the blending problem 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 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, as noted by Bodington and Baker [12]. 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 Karuppiah et al. [29], 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 [cf. [11], [55], again analogous to the pooling problem.

Early efforts in solving the pooling problem were based on recursive LP [26] and successive LP [8] methods. An algorithm based on generalized Benders’ decomposition was proposed by [15]. Sensitivity of local optimal solutions with respect to problem parameters has been analyzed by [17], [20]. All these methods could not address the issue of convergence to a global optimal solution. More recently, many global optimization algorithms have been proposed. Visweswaran and Floudas [58] used duality theory and Lagrangian relaxations to develop a procedure called “GOP” (Global Optimization Algorithm), which is applicable to a wide range of nonconvex NLPs. Ben-Tal et al. [10] proposed another duality-related approach by defining an alternative formulation for the pooling problem. More studies in Lagrangian-based methods are found in [11], [5]. Foulds et al. [16] solved pooling problems using a branch-and-bound algorithm for general bilinear problems. Later, Quesada and Grossmann [44] extended this approach to general chemical process network problems with bilinear terms. Audet et al. [7] solved pooling problems using a branch-and-cut algorithm developed for nonconvex QCQPs.

The contributions of this chapter are two-fold: in the first half, we review and present some new results on formulations and relaxations for the pooling problem. The second half studies discretization strategies for a general BLP and we provide empirical evidence for the effectiveness of this approach in solving the pooling problem.

The remainder of this chapter is structured as follows. Section 2 presents various optimization models for the pooling problem. We prove the equivalence of alternate formulations and compare their problem sizes. Complexity status of the problem is discussed. Section 3 discusses relaxations for the pooling problem obtained using envelopes of bilinear terms and Lagrangian duality. The strengths of these relaxations are analytically investigated. We present slightly stronger versions of previously known results. In section 4, we consider a general BLP and discretize one variable in every bilinear term. 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 on the pooling problem, we test the viability of obtaining good feasible solutions to BLP by solving a MILP approximation. The chapter concludes with a discussion in section 5.
We close this introduction by commenting that although the study of pooling problems was motivated using the example of refinery processes, the problem also finds applications in other fields of chemical engineering such as wastewater treatment [28] and emissions regulation [15]. A detailed review of industrial applications can be found in Kallrath [27] and Visweswaran [57]. For previous surveys on pooling problems, refer to Audet et al. [7], Misener and Floudas [37], Tawarmalani and Sahinidis [53].

2 Problem Formulations

This section formally defines the pooling problem as a type of a bilinear network flow problem on an arbitrary directed graph. First let us define several parameters that will be useful in stating the pooling problem in mathematical terms. We adopt the following notation throughout this chapter: conv(·) denotes the convex hull of a set, cl(·) is its closure and cl(conv(·)) is the closure of its convex hull. e is a vector of ones, 0 is a vector of zeros, and e_i is the i^{th} unit vector. \( \mathbb{R} \) is the set of reals and \( \mathbb{Z} \) the set of integers. Proj\_x(·) is the projection operator onto the x-space. \( \mathcal{M}_i \), \( \mathcal{M}_i \) denotes the i^{th} row (column) of a matrix \( \mathcal{M} \).

2.1 Model parameters

Consider a acyclic directed graph \( G = (\mathcal{N}, A) \) where \( \mathcal{N} \) is the set of nodes and \( A \) the set of arcs. The set \( \mathcal{N} \) can be partitioned into three nonempty subsets \( I, L, J \subset \mathcal{N} \). Here \( I \) denotes the set of inputs, \( L \) the set of pools, and \( J \) the set of outputs. We also assume that \( A \subseteq (I \times L) \cup (L \times L) \cup (L \times J) \cup (I \times J) \), 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 the set \( A \). Traditionally, problem instances with \( \mathcal{A} \cap (L \times L) = \emptyset \) are referred to as \textit{standard pooling problems} and as \textit{generalized pooling problems}, otherwise. In this work, we do not differentiate between these two cases since our aim is to present a more unified treatment for all classes of pooling problems. If the need arises to treat these two cases separately, then we explicitly state so. Let \( K \) denote the set of specifications that are tracked across the problem.

For each arc \( (i, j) \in A \), let \( c_{ij} \) be the variable cost of sending a unit flow on this arc. For every node \( i \in \mathcal{N} \), let \( C_i \) be the capacity of this node, i.e. the maximum amount of incoming or outgoing flow from node \( i \). 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 \( j \in J \), \( C_j \) is the maximum demand. The upper bound on flow on arc \( (i, j) \in A \) is denoted as \( u_{ij} \). Typically, \( u_{ij} = \min\{C_i, C_j\} \). However we allow the arcs in \( G \) to carry arbitrary upper bounds. \( \lambda_{ik} \) denotes the level of spec \( k \) in raw material at input \( i \), for all \( i \in I \) and \( k \in K \). Likewise, \( \mu_{jk}^{\text{min}} \) and \( \mu_{jk}^{\text{max}} \) are the lower and upper bound requirements on level of spec \( k \) at output \( j \), for all \( j \in J \) and \( k \in K \). We assume that each of the values \( \sum_k \lambda_{ik}, \sum_k \mu_{jk}^{\text{min}}, \sum_k \mu_{jk}^{\text{max}} \) is between \([0, 1]\) for all \( i \in I, j \in J \). This assumption is without loss of generality since the given values for these parameters can always be normalized. This 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 \((i, j) \in A\). For notational simplicity, we will always write equations using the flow variables \( y_{ij} \) with the understanding that \( y_{ij} \) is defined only for \((i, j) \in A\). At each pool \( l \in L \), the total amount of incoming flow must equal the total amount of outgoing flow.

\[
\sum_{i \in I \cup L} y_{il} = \sum_{j \in L \cup J} y_{jl}, \quad l \in L. \tag{1}
\]

The capacity constraints at each node in \( G \) are stated as

\[
\sum_{j \in L \cup J} y_{ij} \leq C_i, \quad i \in I, \tag{2a}
\]

\[
\sum_{j \in L \cup J} y_{lj} \leq C_l, \quad l \in L, \tag{2b}
\]

\[
\sum_{i \in I \cup L} y_{ij} \leq C_j, \quad j \in J. \tag{2c}
\]
Finally, flows in $G$ are bounded by individual arc capacities.

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

Denote $F := \{ y \in \mathbb{R}^{|A|}_+ : (4) - (5) \}$ as the polyhedral set that defines feasible flows on $G$.

### 2.2 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 spec values whose concentration, denoted by $p_{jk}$ for $j \in L \cup J, k \in K$, can be determined as

$$p_{jk} = \left\{ \begin{array}{cl} \frac{\sum_{i \in I} \lambda_{ik}y_{ij} + \sum_{l \in L} p_{lk}y_{lj}}{\sum_{i \in I \cup L} y_{ij}}, & \sum_{i \in I \cup L} y_{ij} > 0 \\ 0, & \sum_{i \in I \cup L} y_{ij} = 0. \end{array} \right.$$ (4)

Since $0 \leq \sum_{i \in K} \lambda_{ik} \leq 1$, 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 the following bilinear form,

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

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

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 [37]. In this paper, we will make the assumption of linear blending at pools and outputs.

We are now ready to formally state the pooling problem.

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

$$\min_{y, p} \quad \sum_{(i, j) \in A} c_{ij}y_{ij}$$

s.t. \quad $y \in F$ \quad (Pooling)

$$\mu_{jk}^\text{min} \leq p_{jk} \leq \mu_{jk}^\text{max}, \quad j \in J, k \in K.$$

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

$$\sum_{i \in I} \lambda_{ik}y_{ij} + \sum_{l \in L} p_{lk}y_{lj} \leq \mu_{jk}^\text{max} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K$$ (5a)

$$\sum_{i \in I} \lambda_{ik}y_{ij} + \sum_{l \in L} p_{lk}y_{lj} \geq \mu_{jk}^\text{min} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K.$$ (5b)

Note that the spec tracking constraints corresponding to the pools are retained. Thus we have the following optimization model, introduced as the $p$-formulation by Haverly [26].

$$\min_{y, p} \quad \sum_{(i, j) \in A} c_{ij}y_{ij}$$

s.t. \quad $y \in F$

$$(4a), (5).$$
2.3 Complexity

As seen from (P), the pooling problem is a bilinear program which is a generalization of the strongly NP-hard linear maxmin problem [24]. Recently, a formal proof was provided for the NP-hardness of the pooling problem.

**Proposition 1** (Alfaki and Haugland [1]). *The pooling problem is NP-hard.*

**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 \(n = |V|\) is the number of nodes in the graph \(G' = (V,E)\) for the stable set problem. The set of arcs is \(A = (I \times L) \cup (L \times J)\) 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 \(i \in I\), \(\lambda_i = 1, \lambda_{i,n+i} = -1\); \(\mu_{jk}^{\min} = -1 \forall j, k\); and for any \(j \in J\), \(\mu_{j,\infty}^{\max} = -1/n, \mu_{jk}^{\max} = 1\) for \(k = 1, \ldots, n\) such that \((j,k) \notin E\), 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 \(V' := \{v = 1, \ldots, n: y'_{iv} > 0\}\) is a stable set in \(G'\) of cardinality at least \(\sum_{v=1}^{n} y'_{iv}\).

**Corollary 1** (Alfaki and Haugland [1]). *The pooling problem with a fixed number of pools is NP-hard.*

Alfaki and Haugland also gave a recursive algorithm that runs in polynomial time for fixed \(|K|\) and solves a single pool problem with no direct arcs from inputs to outputs.

Now observe that the sole purpose of having variables \(p_{lk}\) in (P) is to enforce that all the outgoing arcs from a pool carry the same concentration value for a spec. Consider a pooling problem where \(|j| \in \mathcal{N}: (l,j) \in A\) = 1 for \(l \in L\). Substitute a new variable \(w_{lkj}\) for bilinear terms \(p_{lk} y_{lj}\) in (4a) and (5). Since each pool has only one outgoing arc, we need not enforce the spec consistency constraints \(w_{lkj} = p_{lk} y_{lj}\). Thus the (P) formulation can be completely linearized in this special case and solved as a single LP in polynomial time.

2.4 Alternate formulations

2.4.1 Proportion model : \(q\)-formulation

The \(q\)-formulation for the standard pooling problems was proposed by Ben-Tal et al. [10]. In this formulation, Ben-Tal et al. modeled (5) using proportion variables \(q_{il}\), for \(l \in L, i \in I\), which denote the fraction of incoming flow to pool \(l\) that is contributed by input \(i\). Thus,

\[
\sum_{i \in I} q_{il} = 1 \quad l \in L
\]

\[
y_{il} = q_{il} \sum_{i' \in I} y_{i'l} = q_{il} \sum_{j \in L \cup J} y_{lj} \quad i \in I, l \in L.
\]

Then, in the case of standard pooling problems, the spec tracking constraints (4) imply that

\[
p_{lk} = \sum_{i \in I} \lambda_{ik} q_{il}, \quad l \in L, k \in K.
\]

For generalized pooling problems, perhaps a straightforward extension of this idea will be to define proportion variables \(q_{il}\) for \(l \in L, i \in I \cup L\). However this has two drawbacks: 1) Due to the presence of pool-pool arcs, there will be bilinear terms of the form \(q_{il} y_{lj}\). Although this still retains the bilinear structure of the problem, it does not allow partitioning of the variables into two sets as is the case when all bilinear terms are of the form \(q_{il} y_{lj}\). Hence, this formulation has to be treated as a QCQP, for which building good relaxations can sometimes prove more challenging than for a BLP. To remedy this issue, Audet et al. [7] proposed a formulation that involves both \(p\) and \(q\) variables, as described in [2.4.3]. 2) The number of proportion variables is \(O(|L|^2 + |I||L|)\), which can be significantly large.

Alfaki and Haugland [3] developed a \(q\)-formulation for generalized pooling problems that has bilinear terms of the form \(q_{il} y_{lj}\) with \(O(|I||L|)\) proportion variables. We sketch the basic ideas behind this formulation.
For every pool \( l \in L \), define \( I_l \) as the subset of inputs from which there exists a directed path to \( l \) in \( G \). Let \( q_{il} \) denote the fraction of incoming flow to pool \( l \in L \) that originated from input \( i \in I_l \). Note that in this definition of the proportion variable \( q_{il} \), we do not distinguish between flows that started at \( i \) and reached \( l \) along different paths. By definition, the \( q \)'s must sum to 1 across all inputs and hence we have

\[
q_l \in \Delta_{|I_l|} := \{ q_l \geq 0 : \sum_{i \in I_l} q_{il} = 1 \}, \quad l \in L,
\]

where \( q_l \) is the vector \( (q_{il})_{i \in I_l} \).

Since in the pooling problem, 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. The flow of commodity \( i \) on arc \((l, j)\) is given by \( v_{ij} = q_{il}y_{lj} \) 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

\[
y_{il} + \sum_{l' \in L, i' \in I_l} q_{il'}y_{l'j} = q_{il} \sum_{j \in L \cup J} y_{lj}, \quad l \in L, i \in I_l,
\]

(7)

In the context of the \( p \)-formulation, specifications are commodities and \( (4a) \) serves the role of commodity balance constraints. It can be verified that equations (6) and (7) render the flow balance constraints (1) redundant.

The spec level requirement constraints at the output are modeled as

\[
\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{i \in L \cap I_l} \lambda_{ik} q_{il} y_{ij} \leq \mu_{jk}^{\max} \sum_{i \in I_l \cup L} y_{ij}, \quad j \in J, k \in K \tag{8a}
\]

\[
\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{i \in L \cap I_l} \lambda_{ik} q_{il} y_{ij} \geq \mu_{jk}^{\min} \sum_{i \in I_l \cup L} y_{ij}, \quad j \in J, k \in K \tag{8b}
\]

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

\[
\begin{align*}
\min_{y,p} \quad & \sum_{(i,j) \in A} c_{ij} y_{ij} \\
\text{s.t.} \quad & y \in F \\
& (6) - (8).
\end{align*}
\]

(9)

It is easily observed that in the case of standard pooling problems, the above formulation reduces to the one proposed by Ben-Tal et al. [10]. Also note that using (6), we can rewrite (8) as

\[
\begin{align*}
\sum_{i \in I} (\lambda_{ik} - \mu_{jk}^{\max}) y_{ij} + \sum_{i \in L \cap I_l} (\lambda_{ik} - \mu_{jk}^{\max}) q_{il} y_{ij} & \leq 0, \quad j \in J, k \in K \tag{9a} \\
\sum_{i \in I} (\lambda_{ik} - \mu_{jk}^{\min}) y_{ij} + \sum_{i \in L \cap I_l} (\lambda_{ik} - \mu_{jk}^{\min}) q_{il} y_{ij} & \geq 0, \quad j \in J, k \in K. \tag{9b}
\end{align*}
\]

One may also wish to use proportions of flows traveling from a pool \( l \) to an output \( j \). Using these proportion variables and following the same steps as discussed in this section, Alfaki and Haugland [4] developed a new formulation (TP) for the standard pooling problem. They also proposed the (STP) formulation which uses the proportion variables from both (Q) and (TP) and consequently, has more variables and bilinear terms.

2.4.2 \( pq \)-formulation

The \( pq \) formulation, introduced by Tawarmalani and Sahinidis [53] for standard pooling problems, is obtained by appending some valid inequalities to the \( q \)-formulation. These inequalities are given by

\[
\begin{align*}
\sum_{i \in I_l} q_{il} y_{lj} & = y_{lj}, \quad l \in L, j \in L \cup J \tag{10a} \\
\sum_{j \in L \cup J} q_{il} y_{lj} & \leq C_{il} q_{il}, \quad l \in L, i \in I_l. \tag{10b}
\end{align*}
\]
and are derived via the Reformulation Linearization Technique (RLT) [50] by multiplying (6) with
\( y_j \) and \( \{2b\} \) with \( q_{il} \). These constraints were independently derived by Quesada and Grossmann [44]
for processing network problems.

\[
\min_{y,\varphi} \sum_{(i,j)\in A} c_{ij} y_{ij}
\]
\[
s.t. \quad y \in \mathcal{F}
\]
\[
(\mathcal{PQ})
\]

Although the addition of (10) retains the validity of the \( q \)-formulation by not cutting off any feasible points, it helps to obtain a significantly stronger polyhedral relaxation of the pooling problem, as explained in [3.2.2].

### 2.4.3 A 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 \( q_{ij}q_{ij}^t \) 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 the set of pools \( L_I := \{ l \in L : \exists \ell' \in L s.t. (\ell', l) \in \mathcal{A} \} \), i.e. pools with incoming arcs from some input nodes, and concentration variables are used for pools in \( L \setminus L_I \). We skip the details of this formulation since it can be easily obtained by combining the previous sections. Let this hybrid formulation be denoted by \((\mathcal{HYB})\).

### 2.4.4 Equivalence of formulations

We now formally prove the correctness of the foregoing formulations for the pooling problem. Two formulations are said to be 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 \((\mathcal{P}), (\mathcal{Q}), (\mathcal{PQ}), \) and \((\mathcal{HYB})\) are equivalent for \( G \).

**Proof.** First let us show that for any feasible point \((q, y)\) in \((\mathcal{Q})\) there exists some \( p \) satisfying (4a) and (5). For any \( \ell', l \in L \) such that \((\ell', l) \in \mathcal{A}\), we have \( I_{\ell'} \subseteq I_l \) and hence \( I_{\ell'} \cap I_l = I_{\ell'} \). Choose a \( k \in K \) and multiply both sides of equation (7) with \( \lambda_{kl} \). Summing over \( i \in I_l \) produces

\[
\sum_{i \in I_l} \lambda_{ik} y_{il} + \sum_{\ell' \in L \setminus I_{\ell'}} \sum_{t' \in L \setminus I_{\ell'}} \lambda_{ik} q_{it'} y_{it'} = \sum_{i \in I_l} \lambda_{ik} q_{il} \sum_{j \in L \setminus I_l} y_{ij} \quad l \in L
\]

Thus, \( \sum_{i \in I_l} \lambda_{ik} q_{it} \) that satisfies (4a) and (5) and hence is feasible to \((\mathcal{P})\).

Now let us show that for any feasible point \((p, y)\) in \((\mathcal{P})\) there exists some \( q \) satisfying (6), (7), and \( p_{lk} = \sum_{i \in I_l} \lambda_{ik} q_{it} \) for \( l \in L, k \in K \). This implies that \((q, y)\) is feasible to \((\mathcal{Q})\) since (8) follows from (6) and \( p_{lk} = \sum_{i \in I_l} \lambda_{ik} q_{it} \) now choose some pool \( l \in L \).

**Case 1.** \( \sum_{i \in I_l \cup J} y_{ij} = \sum_{j \in I_l \cup J} y_{ij} = 0 \). Here equation (7) trivially holds true. By problem definition, all flows originate with specification values only at the input nodes and since equation (11) balances total flow at each pool in \( G \), it must be that the vector of specification values \( p_{il} \) at pool \( l \) is some convex combination of the input specifications \( \lambda_i, \forall i \). Thus, there exist \( q_{it} \forall i \in I_l \) satisfying (6) such that \( p_{lk} = \sum_{i \in I_l} \lambda_{ik} q_{it} \) \( \forall k \).

**Case 2.** \( \sum_{j \in I_l \cup J} y_{ij} > 0 \). For \( j \in L \cup J \), define \( \xi_{lj} \) to be the fraction of outgoing flow from \( l \) directed towards \( j \),

\[
\xi_{lj} := \frac{y_{lj}}{\sum_{j \in I_l \cup J} y_{lj}}
\]

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 \( \tau := \{i, \tau_1, \ldots, \tau_{m(\tau)}, l\} \in T_{il} \). Then the total flow from \( i \) that
reaches \( l \) along path \( \tau \) is

\[
\sigma_{il}^\tau = y_{i\tau} \xi_{m(\tau)} l \prod_{o=1}^{m(\tau)-1} \xi_{\tau o \tau + 1}.
\]

Construct the \( q \) variables as follows

\[
q_{il} = \sum_{\tau \in T_{il}} \sigma_{il}^\tau / \sum_{i' \in I_l} y_{i'l}.
\]

The flow balance equations (11) imply that there is no supply at pools and all the supply originates at inputs. Hence, the quantity \( \sum_{i \in I_l} \sum_{\tau \in T_{il}} \sigma_{il}^\tau \), which designates the total flow from all inputs to \( l \) must equal the total flow into \( l \) which is \( \sum_{i' \in I_l} y_{i'l} \). Hence \( \sum_{i \in I_l} q_{il} = 1 \). Similarly, the total quantity of spec \( k \) at pool \( l \) is given by \( \sum_{i \in I_l} \lambda_{ik} \sum_{\tau \in T_{il}} \sigma_{il}^\tau \) and hence \( p_{ik} = \sum_{i \in I_l} \lambda_{ik} q_{il} \). Now the left hand side of (7) is

\[
y_{il} + \sum_{i' \in I_l} \sum_{\tau \in T_{il}} \frac{\sigma_{i'\tau}^\tau y_{i'\tau}}{\sum_{i' \in I_l} y_{i'\tau}} = y_{il} + \sum_{i' \in I_l} \sum_{\tau \in T_{il}} \sigma_{i'\tau}^\tau \xi_{i'\tau} = \sum_{\tau \in T_{il}} \sigma_{il}^\tau = q_{il} \sum_{j \in J_l} y_{ij}
\]

where the second equality follows from the following two observations: 1) \( y_{il} \) is defined if and only if \( (i, l) \in A \), and 2) any non-direct path between \( i \) and \( l \) must pass through intermediate pools \( l' \) such that \( i \in I_{l'} \). Then by (11), the term \( \sigma_{i'\tau}^\tau \xi_{i'\tau} \) denotes the total flow from \( i \) to \( l \) along the path \( \tau' = \tau \cup (l', l) \in T_{il} \).

Thus, we have shown that (P) and (Q) are equivalent formulations of the pooling problem. The equivalence of (Q) and (PQ) follows by noting that (PQ) is obtained by appending valid inequalities (10) to (Q). Finally, the correctness of (HYB) can be shown using the steps of the above proof for pools in \( L_I \).

\[ \square \]

### 2.5 Problem sizes

The alternate formulations of P, Q, PQ 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 \( A \) and they only differ in the use of non-flow variables and additional constraints. Since bilinearities are what makes the pooling problem hard to solve, we mention the number of bilinear terms and bilinear constraints along with the number of non-flow variables in Table[1]

<table>
<thead>
<tr>
<th>Formulation</th>
<th>Non-flow variables</th>
<th>Bilinear terms</th>
<th>Bilinear constr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>( \sum_{i \in L}</td>
<td>I_i</td>
<td>)</td>
</tr>
<tr>
<td>Q</td>
<td>( \sum_{i \in L}</td>
<td>I_i</td>
<td>)</td>
</tr>
<tr>
<td>PQ</td>
<td>( \sum_{i \in L}</td>
<td>I_i</td>
<td>)</td>
</tr>
<tr>
<td>HYB</td>
<td>( \sum_{i \in L_I}</td>
<td>I_i</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 1: Comparing problem sizes for multiple formulations of the pooling problem.
2.6 Variants

We have already mentioned two types of pooling problems - standard and generalized, depending on the absence or presence of arcs between pools, respectively. A broader class of network flow problems with bilinear terms is described by Lee and Grossmann [31], Quesada and Grossmann [44]. Nonlinear blending rules have also been proposed, see Misener and Floudas [27] for a discussion and Reallaf et al. [13] for one specific example of nonlinear blending where the bilinear terms in the pooling problem are replaced by cubic terms. Ruiz et al. [48] studied a variant of the standard pooling problem where total flow into an output, given by \( \sum_{i \in I \cup L} y_{ij} \), is fixed to some nonzero constant, for each output \( j \in J \). An extended pooling problem that imposes upper bounds on emissions from outputs was introduced in Misener et al. [10]. Other examples of MINLP models can be found in D’Ambrosio et al. [13], Meyer and Floudas [36], Misener and Floudas [35], Nishi [42], Visweswaran [57]. 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.

3 Relaxations

The pooling problem is a nonconvex problem where nonconvexities arise due to the presence of bilinear terms. 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, such as BARON and COUENNE, use different bound tightening techniques that are updated at each node of the branch-and-bound tree. In this section, we discuss properties of the commonly used relaxations for the pooling problem. We divide our discussion into two parts: in the first part, we address the commonly used strategy of relaxing the entire problem by building relaxations of bilinear terms with respect to bounds on the variables participating in the product term; in the second part, we discuss the more general approach of relaxing suitable subsets of constraints in the problem. The results from the second half of this section will also prove useful in constructing relaxations when we discretize some of the variables in §4.

3.1 Envelopes of bilinear functions

One way of relaxing a nonconvex function is to obtain the convex underestimator and concave overestimator of the function over its domain. When the function consists of a single bilinear term, we wish to relax the set

\[ X := \{(\chi, \rho, \omega) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}_+ : \omega = \chi \rho, \chi \in [a_1, b_1], \rho \in [a_2, b_2]\}. \]  

McCormick [34] proposed the following four inequalities to relax \( X \)

\[ \omega \geq b_2 \chi + b_1 \rho - b_1 b_2, \quad \omega \geq a_2 \chi + a_1 \rho - a_1 a_2, \]

\[ \omega \leq b_2 \chi + a_1 \rho - a_1 b_2, \quad \omega \leq a_2 \chi + b_1 \rho - a_2 b_1. \]  

Here (13a) and (13b) define the convex and concave envelope of \( \omega = \chi \rho \) over the box \([a_1, a_2] \times [b_1, b_2]\), respectively, and are commonly referred to as the McCormick envelopes. Later, Al-Khayyal and Falk [2] proved that (13) in fact defines \( \text{conv}(X) \). These single term envelopes are a common choice for relaxations used in a branch-and-bound algorithm for solving pooling problems, dating back to Foulds et al. [16] to the best of our knowledge. They can be applied to both (P) and (Q). For brevity, we will use \((\chi, \rho, \omega) \in \mathcal{M}(X)\) or equivalently \((\chi, \rho, \omega) \in \mathcal{M}(\omega = \chi \rho)\) to denote the McCormick relaxation (13), where the bounding box \([a_1, b_1] \times [a_2, b_2]\) will be the natural bounds on the associated variables unless otherwise stated explicitly.

When the nonconvex function consists of sums of multiple bilinear terms, stronger relaxations can be obtained by developing under- and over-estimators for the entire function [cf. 54]. In general, it is not true that sum of convex (concave) envelopes of a finite family of functions \( \{f_r(\cdot)\}_{r \in R} \) gives the convex (concave) envelope of the sum \( \sum_{r \in R} f_r(\cdot) \). Luendetke et al. [53], Meyer and Floudas [35], Rikun [46], Tardella [52] develop some sufficient conditions when this holds true. As argued in Misener and Floudas [39] Property 3.1.3.1], the special structure of the pooling problem implies that the strongest relaxations of the individual bilinear functions can be obtained using envelopes of each bilinear term.
Proposition 3 (Sum decomposition rule, Misener and Floudas [39]). For the pooling problem, envelopes of bilinear functions, such as \( \sum_{k \in K} p_{ik} y_{ij} \) and \( \sum_{k \in K} p_{ik} y_{ij} - p_{ik} \sum_{j \in L \cup J} y_{ij} \) in \( \mathcal{P} \), taken over bounds on the associated variables are given by single term McCormick inequalities.

3.2 Relaxing feasible sets

Now we turn our attention to finding good relaxations for constraints in the pooling problem. First, we are interested in studying a relaxation of the feasible set that arises at each pool. For the two formulations \( \mathcal{P} \) and \( \mathcal{Q} \), relaxations of the feasible sets corresponding to pool \( l \) are given by \( \mathcal{P}_l \) and \( \mathcal{Q}_l \), respectively, defined as

\[
\mathcal{P}_l := \left\{ \left( \{ p_{ik} \}_{k \in K}, \{ y_{ij} \}_{j \in L \cup J}, \{ w_{lkj} \}_{j \in L \cup J} \right) : w_{lkj} = p_{ik} y_{ij}, \ k \in K, j \in L \cup J \right\}
\]

\[
\mathcal{Q}_l := \left\{ \left( \{ q_{ij} \}_{i \in I_l}, \{ y_{ij} \}_{j \in L \cup J}, \{ v_{ij} \}_{i \in I_l, j \in L \cup J} \right) : v_{ij} = q_{ij} y_{ij}, \ i \in I_l, j \in L \cup J \right\}
\]

where \( K \) is a \((m-1)\)-simplex and hence a polytope, it has finitely many extreme points given

The above single pool relaxations are constructed by dropping the incoming arcs at pool \( l \) along with their respective bounds and the commodity balance constraints (14a) and (7) for \( \mathcal{P} \) and \( \mathcal{Q} \), respectively. Observe that we have also included new variables \( w_{lkj} \) and \( v_{ij} \) for the bilinear terms in \( \mathcal{P}_l \) and \( \mathcal{Q}_l \), respectively. The lower and upper bounds on specs in the form \( p_{ik} \leq p_{ik} \leq p_{ik} \) for \( \mathcal{P} \), since all pools receive flows that only originated at some input, and are included here because tighter bounds on variables are known to lead to stronger relaxations, as evident for example in the McCormick inequalities of [13].

We first state some general results on sets defined by multiple bilinear terms that will be useful in comparing relaxations of \( \mathcal{P}_l \) and \( \mathcal{Q}_l \).

3.2.1 Convex hulls of bilinear terms

Let \( \mathcal{X}^+ \) be a general bilinear set defined as follows

\[
\mathcal{X}^+ := \{ (\chi, \rho, \omega) \in \mathbb{R}^m_+ \times \mathbb{R}^m \times \mathbb{R}^{m \times n} : \omega = \chi \rho^\top, \chi \in \Theta, \rho \in \Upsilon \},
\]

where \( \Theta = \{ \chi \in \mathbb{R}^m_+ : a^\top \chi = a_0 \} \) with \( a > 0, a_0 > 0 \) is a \((m-1)\)-simplex in \( \mathbb{R}^m \) and \( \Upsilon \) is some (possibly mixed integer) subset of \( \mathbb{R}^n \) such that \( \text{conv}(\Upsilon) \) is a polyhedron.

Theorem 1. Suppose that the convex hull of \( \Upsilon \) is a polyhedron \( \text{conv}(\Upsilon) = \{ \rho : B \rho \geq b_0 \} \). Let \( \omega_i \) denote the \( i \)-th row of \( \omega \). Then the closure convex hull of \( \mathcal{X}^+ \) is a polyhedron given by

\[
\text{cl conv}(\mathcal{X}^+) = \{ (\chi, \rho, \omega) : \chi \in \Theta, B \omega_i^\top \geq b_0 \chi_i, i = 1, \ldots, m, \sum_{i = 1}^m a_i \omega_i^\top = a_0 \rho \}
\]

If \( \text{conv}(\Upsilon) \) is a polytope, then \( \text{conv}(\mathcal{X}^+) \) is also a polytope.

Proof. Denote \( \mathcal{X}^{++} := \{ (\chi, \rho, \omega) : \omega = \chi \rho^\top, \chi \in \Theta, \rho \in \text{conv}(\Upsilon) \} \).

Claim 1. \( \text{conv}(\mathcal{X}^+) = \text{conv}(\mathcal{X}^{++}) \). The \( \subseteq \)-inclusion is trivial since \( \Upsilon \subseteq \text{conv}(\Upsilon) \), whereas the \( \supseteq \)-inclusion follows from Carathéodory’s theorem applied to \( \text{conv}(\Upsilon) \).

Since \( \Theta \) is a \((m-1)\)-simplex and hence a polytope, it has finitely many extreme points given by \( \bigcup_{i = 1}^m \chi^{(i)} \) where \( \chi^{(i)} = a_i e_i/a_i \) for \( i = 1, \ldots, m \). Define the following finite family of polyhedral sets,

\[
\Psi_i := \{ (\chi, \rho, \omega) : \chi = a_i e_i/a_i, B \rho \geq b_0, \omega = a_i e_i/a_i \rho^\top \} \quad i = 1, \ldots, m.
\]

10
Claim 2. \( \text{conv}(X^+) = \text{conv}(\bigcup_{i=1}^m \Psi_i) \). From Claim 1 it suffices to show that \( \text{conv}(X^+) = \text{conv}(\bigcup_{i=1}^m \Psi_i) \). The construction of \( \Psi_i \) implies that \( \Psi_i \subseteq X^+ \), for \( i = 1, \ldots, m \), and hence we have \( \text{conv}(X^+) \supseteq \text{conv}(\bigcup_{i=1}^m \Psi_i) \). Now consider some point \((\chi, \rho, \omega) \in X^+\). Since \( \chi \in \Theta = \text{conv}(\bigcup_{i=1}^m \chi^{(i)}) \), it follows that \((\chi, \rho, \omega) \in \text{conv}(\bigcup_{i=1}^m (\chi^{(i)}, \rho^{(i)}, \rho^{(i)}) \subseteq \text{conv}(\bigcup_{i=1}^m \Psi_i) \). Thus, \( X^+ \subseteq \text{conv}(\bigcup_{i=1}^m \Psi_i) \) and consequently, \( \text{conv}(X^+) \subseteq \text{conv}(\bigcup_{i=1}^m \Psi_i) \).

It follows from Claim 2 that \( \text{cl conv}(X^+) = \text{cl conv}(\bigcup_{i=1}^m \Psi_i) \). Balas' result [9] Theorem 2.1] on disjunctive programming gives us

\[
\text{cl conv}(X^+) = \text{Proj}_{\chi, \rho, \omega} \left\{ (\chi', \rho', \omega') \in \text{dom}[\chi, \rho, \omega, \lambda] : \chi = \sum_i \chi^i, \rho = \sum_i \rho^i, \omega = \sum_i \omega^i, B \rho^i \geq b_0 \lambda_i, \chi^i = \frac{a_0 e_i \lambda_i}{a_i}, \omega^i = \frac{a_0 e_i}{a_i} \rho^i \implies \lambda \geq 0, \sum_i \lambda_i = 1 \right\}.
\]

In order to obtain the projection, note that \( \chi_i = \chi^i = a_0 \lambda_i / a_i \), for all \( i \), and hence \( \lambda_i = a_i \chi_i / a_0 \). Also \( \omega_i = \omega^i = a_0 \rho^i / a_i \). Hence \( \rho^i = a_i \omega_i / a_0 \). Now \( \rho = \sum_i \rho^i \) implies that \( \rho = \sum_i a_i \omega_i / a_0 \).

After making these substitutions we get the desired result. If \( \text{conv}(\Psi) \) is a polytope, then \( \Psi \) is also a polytope for all \( i \), and hence the convex hull of the finite union \( \bigcup_{i=1}^m \Psi_i \) is a polytope. \( \square \)

Remark 1 (Connections to RLT in bounded case). A particular case of the set \( \Theta \) defined in Theorem 1 is the standard \((m-1)\)-simplex given by \( \Theta = \Delta_{m-1} := \{ \chi \geq 0 : \sum_i \chi_i = 1 \} \). In this case, since \( \text{conv}(X^+) = \text{conv}(\bigcup_{i=1}^m \Psi_i) \), we can rewrite as

\[
\text{conv}(X^+) = \text{conv}\{ (\chi, \rho, \omega) : \chi = \rho \chi^\top, \chi \in \Delta_{m-1}, \chi \in \{ 0, 1 \}^m, \rho \in \text{conv}(\Psi) \}.
\]

Further if \( \text{conv}(\Psi) \) is assumed to be a polytope, there exist finite lower and upper bounds on \( \rho_j \) for all \( j = 1, \ldots, n \). Since \( \chi \in \{ 0, 1 \}^m \), we can exactly reformulate the bilinear term \( \omega_{ij} \chi = \chi_i \rho_j \) using its McCormick envelopes, for \( i = 1, \ldots, m, j = 1, \ldots, n \). This implies that

\[
\text{conv}(X^+) = \text{conv}\{ (\chi, \rho, \omega) : \Theta \chi + B \rho + C \omega \geq b, \chi \in \Delta_{m-1}, \chi \in \{ 0, 1 \}^m \}
\]

for some matrices \((\Theta, B, C)\) and vector \(\eta\). Note that the set on the right hand side is the convex hull of a mixed integer linear set where the binary variables \( \chi \) are SOS1. The result of Sherali et al. [51] implies that \( \text{conv}(X^+) \) can be obtained via a RLT procedure that involves multiplying each linear constraint from the system \( \Theta \chi + B \rho + C \omega \geq b \chi_i \), for \( i = 1, \ldots, m \), and \( 1 - \sum_{i=1}^m \chi_i \). It was shown by Sherali et al. that \( \chi_i = 0 \), \( \forall \), \( i \), and since \( \chi_i \in \{ 0, 1 \} \) we can strengthen \( \chi_i^2 = \chi_i \). After carrying out the multiplication on the McCormick envelopes, we obtain \( \omega_{ij} \chi_k = 0 \), \( \forall \), \( i, j, k \). On substituting \( \omega_{ij} = \chi_i \rho_j \) we get exactly the linear description from the statement of Theorem 1. Our derivation using disjunctive programming [9] is more general because it accommodates the case when \( \text{conv}(\Psi) \) is a polyhedron and hence in the absence of lower/upper bounds on some \( \rho_j \), it is not possible to use McCormick envelopes to reformulate \( \omega_{ij} \chi_i \rho_j \), \( \forall \), \( i \), into a mixed integer linear set.

Remark 2 (Full dimensional simplex). The same steps used in the proof of Theorem 1 can also be used to address the case when \( \Theta \) is defined to be a \( m \)-simplex as \( \Theta = \{ \chi \in \mathbb{R}^m_+ : a \chi \leq a_0 \} \).

The inequalities describing \( \text{cl conv}(X^+) \) in Remark 2 can be construed as level-1 RLT inequalities obtained by multiplying \( B \rho \geq b_0 \) with each facet \( \chi_i \geq 0 \), \( \forall \), \( i \), \( a_0 - a_i \chi_i \geq 0 \) of \( \{ \chi \geq 0 : a \chi \leq a_0 \} \) and substituting \( \omega_{ij} = \chi_i \rho_j \). We next verify that this constructive procedure yields \( \text{cl conv}(X^+) \) also when \( \Theta \) is a general simplex in \( \mathbb{R}^m \) defined using \( m + 1 \) linear inequalities as \( \Theta = \{ \chi : \exists \chi \geq \chi \} \). Alternatively, we have \( \Theta = \text{conv}\{ \chi^{(1)}, \ldots, \chi^{(m+1)} \} \) for some affinely independent points \( \chi^{(1)}, \ldots, \chi^{(m+1)} \) in \( \mathbb{R}^m \). Let \( \mathcal{G} : \Delta_m \to \Theta \) be the affine map from the standard \( m \)-simplex \( \Delta_m := \{ \chi \in \mathbb{R}^m_+ : \sum_i \chi_i \leq 1 \} \) to \( \Theta \). In particular,

\[ \mathcal{G}(\chi) = \Theta \chi + \gamma, \quad \text{where } \Theta = [\chi^{(1)} - g \cdots \chi^{(m)} - g], \quad g = \chi^{(m+1)}. \]

Also let \( X^+ \) and \( X^+ \) be the bilinear sets from equation (15) corresponding to \( \chi \in \Delta_m \) and \( \chi \in \Theta \), respectively, and let \( \mathcal{A} : \hat{X}^+ \to X^+ \) be the affine map \( \mathcal{A}(\chi, \rho, \omega) = (\mathcal{G}(\chi), \rho, \Sigma \omega + \rho g \top) \).
Then we are interested in obtaining \( \text{cl\,conv}(X^+) = \text{cl\,conv}(\mathcal{A}(X^+)) = \mathcal{A}(\text{cl\,conv}(X^+)) \) where the last equality holds true because \( \text{cl\,conv}(X^+) \) is a polyhedron (cf. Remark 2) and \( \mathcal{A}(\cdot) \) is affine. Thus,

\[
\text{cl\,conv}(X^+) = \left\{ (\chi, \rho, \omega) : \chi \in \Theta, \ (\mathcal{G}^{-1}_i \cdot (\omega - g \rho^T)) B^T \geq b_0^T \mathcal{G}^{-1}_i (\chi - g), \ i = 1, \ldots, m \right\}. \quad (16a)
\]

\[
[\rho^T - e^T \mathcal{G}^{-1} (\omega - g \rho^T)] B^T \geq b_0^T \left[ 1 - e^T \mathcal{G}^{-1} (\chi - g) \right]. \quad (16b)
\]

Since at every extreme point of \( \Theta \) there must be exactly \( m \) active inequalities, we assume that the inequalities describing \( \Theta \) are sorted such that for \( i = 1, \ldots, m + 1, (\bar{\chi}_i, f_i) \) is inactive at \( \chi^{(i)} \) and active at all other vertices. Now for any \( i = 1, \ldots, m, m \mathcal{G}^{-1}_i \mathcal{G}_j \) for all \( j \neq i \) implies that \( \mathcal{G}^{-1}_i (\chi^{(i)} - g) = 0 \). Thus \( (\mathcal{G}^{-1}_i, \mathcal{G}^{-1}_i g) \) defines a hyperplane that is active at \( g = \chi^{(m+1)} \) and \( m - 1 \) other vertices \( \{\chi^{(j)}\}_{j \neq i} \). Since all the vertices are affinely independent in \( \mathbb{R}^m \), the collection \( \{\chi^{(m+1)}, \chi^{(j)} \forall j \neq i\} \) defines a unique (up to scaling) hyperplane and it must be that \( (\mathcal{G}^{-1}_i, \mathcal{G}^{-1}_i g) = \lambda_i (\bar{\chi}_i, f_i) \) for some positive scalar \( \lambda_i \). In particular, since \( \mathcal{G}^{-1}_i \mathcal{G}_i = 1 \) and \( \mathcal{G}_i \chi^{(i)} - f_i > 0 \), we have \( \lambda_i = 1/(\bar{\chi}_i, \chi^{(i)} - f_i) \). Then we can interpret (16a) as a (transposed) family of level-1 RLT inequalities obtained by multiplying \( B \rho \geq b_0 \) with \( \bar{\chi}_i, \chi^{(i)} - f_i \). Now since \( \chi = \sum_{i=1}^{m+1} \alpha_i \chi^{(i)} \) for some convex combination \( \alpha \) and since \( \mathcal{G}_i \chi^{(i)} - f_i \) for all \( i \neq j \), it is straightforward to verify that \( \alpha_i = (\bar{\chi}_i, \chi^{(i)} - f_i)/(\bar{\chi}_i, \chi^{(i)} - f_i) \) for \( i = 1, \ldots, m + 1 \).

Hence (16b) represents a family of level-1 RLT inequalities obtained by multiplying \( B \rho \geq b_0 \) with \( \mathcal{G}^{(m+1)} \chi^{(m+1)} - f_{m+1} \geq 0 \).

**Theorem 2.** Let \( \Theta = \{\chi : \bar{\chi} \chi \geq f\} \) be a simplex in \( \mathbb{R}^m \) and \( \text{conv}(\mathcal{T}) = \{\rho : B \rho \geq b_0\} \). Then,

\[
\text{cl\,conv}(X^+) = \{ (\chi, \rho, \omega) : \chi \in \Theta, \ B \omega^T \bar{\mathcal{G}}^T_i - f_i B \rho \geq b_0 (\bar{\chi}, \chi^{(i)} - f_i) \ (i = 1, \ldots, m + 1) \}.
\]

We now state another result that gives the convex hull of the intersection of finitely many general bilinear sets that share a common variable bounded within a simplex.

**Theorem 3.** Let \( \Theta \) be a simplex in \( \mathbb{R}^m \) and define \( \mathcal{X} : = \{(\chi, \{(\rho^t, \omega^t)\}) \in \mathcal{T} : (\chi, \rho^t, \omega^t) \in X^\mathcal{T} \forall t \in T\} \), where \( T \) is a finite set and for \( t \in T \), \( \mathcal{T}_t \) is a polytope and \( X^\mathcal{T}_t : = \{(\chi, \rho^t, \omega^t) : \chi \in \Theta, \rho^t \in \mathcal{T}_t, \omega^t = \chi \rho^T \} \). Then,

\[
\text{conv}(\mathcal{X}) = \{(\chi, \{(\rho^t, \omega^t)\}) s.t. : (\chi, \rho^t, \omega^t) \in \text{conv}(X^\mathcal{T}_t) \forall t \in T\}.
\]

**Proof.** Consider optimizing any arbitrary linear function over \( \mathcal{X} \).

\[
\theta^* = \min_{\mathcal{X}} \sum_t G_t \omega^t + \sum_t d_t^T \rho^t + e^T \chi \quad \text{s.t.} \quad (\chi, \{(\rho^t, \omega^t)\}) \in \mathcal{X}.
\]

It suffices to show that

\[
\theta^* = \min_{\mathcal{X}} \sum_t G_t \omega^t + \sum_t d_t^T \rho^t + e^T \chi \quad \text{s.t.} \quad (\chi, \{(\rho^t, \omega^t)\}) \in \text{conv}(X^\mathcal{T}_t), \ t \in T.
\]

By definition of \( \mathcal{X} \), \( \text{conv}(\mathcal{X}) \subseteq \cap_t \text{conv}(X^\mathcal{T}_t) \) and hence the \( \geq \) inequality is obvious. Now, since \( \omega^t = \chi \rho^T \) for any point \( (\chi, \{(\rho^t, \omega^t)\}) \in \mathcal{X} \), we rewrite \( \theta^* \) as

\[
\theta^* = \min_{\mathcal{X}} \gamma + \sum_t d_t^T \rho^t + e^T \chi \quad \text{s.t.} \quad (\chi, \{(\rho^t, \omega^t)\}) \in \text{conv}(\mathcal{X}), \ t \in T,
\]

where \( f_t(\chi, \rho^t) = \chi^T G_t \rho^t \), \( \text{cvx}(\sum_t f_t) \) is the convex envelope of \( \sum_t f_t \) taken over \( \Theta \times \prod_t \mathcal{T}_t \) and \( \text{epi}(\cdot) \) denotes the epigraph of a function. The above two problems are equivalent because (i) \( \Theta \times \prod_t \mathcal{T}_t \) is a compact convex set, and (ii) Tardella [52, Proposition 2] gives us

\[
\text{cvx} \left( \sum_t f_t(\chi, \rho^t) + \sum_t d_t^T \rho^t + e^T \chi \right) = \text{cvx} \left( \sum_t f_t(\chi, \rho^t) \right) + \sum_t d_t^T \rho^t + e^T \chi.
\]
Since Θ is a simplex and Υₜ is a polytope for all t, Rikun [46, Theorem 1.4] for the convex envelope of summation of functions gives us
\[
\text{cvx} \left( \sum_{t} f_t(\chi, \rho^t) \right) = \sum_{t} \text{cvx} \left( f_t(\chi, \rho^t) \right).
\]

Hence, it follows that
\[
\theta^* = \min_{\gamma} \gamma + \sum_{t} d^T_t \rho^t + c^T \chi \quad \text{s.t.} \quad \gamma \leq \sum_{t} \chi_t \in \text{epi cvx} \left( f_t(\chi, \rho^t) \right), \quad t \in T.
\]

Define \( F_t = \{(\chi, \rho^t, \gamma_t): f_t(\chi, \rho^t) = \gamma_t, \chi, \rho^t \in \Theta_t \} \). By definition,
\[
F_t = \text{Proj}_{\chi, \rho^t, \omega^t} \{(\chi, \rho^t, \gamma_t, \omega^t): \gamma_t = G_t \cdot \omega^t, (\chi, \rho^t, \omega^t) \in X^+_t \}.
\]

Hence, the convex hull of \( F_t \) is
\[
\text{conv}(F_t) = \text{Proj}_{\chi, \rho^t, \omega^t} \{(\chi, \rho^t, \gamma_t, \omega^t): \gamma_t = G_t \cdot \omega^t, (\chi, \rho^t, \omega^t) \in \text{conv}(X^+_t) \}.
\]

Also, since \( F_t \subseteq \text{epi} f_t \), then \( \text{conv}(F_t) \subseteq \text{epi cvx} f_t = \text{epi cvx} f_t \). Substituting this inclusion and the identity for \( \text{conv}(F_t) \) into \( \theta^* \) we get
\[
\theta^* \leq \min_{\gamma} \gamma + \sum_{t} d^T_t \rho^t + c^T \chi \quad \text{s.t.} \quad \sum_{t} G_t \cdot \omega^t \leq \gamma \quad \text{and} \quad (\chi, \rho^t, \omega^t) \in \text{conv}(X^+_t), \quad t \in T.
\]

\[\square\]

Remark 3 (Non-simplicial case). Note that Claim 2 in Theorem 1 only uses the fact that \( \Theta \) is a polytope and hence has finite number of extreme points. The stronger condition that \( \Theta \) is a simplex is used in projecting the extended formulation for \( \text{cl conv}(X^+) \). Indeed, if \( \Theta \) is not a simplex, then it may not be possible to perform the projection step in Theorem 1. Also, Theorem 3 is proven using Rikun’s result, which holds true only if \( \Theta \) is a simplex. To the best of our knowledge, a compact description in the \((\chi, \rho, \omega)-space is unknown for \( \text{cl conv}(X^+) \). The result of Günlü et al. [21] are able to describe \( \text{conv}(X^+) \) as a polytope in the \((\chi, \rho)-space after projecting the McCormick envelopes of the bilinear terms. The result of Günlü et al. does not extend to the case when \( \Theta \) is not a unit hypercube. Luedtke et al. [33] compare relative strengths of McCormick envelopes with respect to \( \text{conv}(X^+) \) for general hypercubes \( \Theta \) and \( \Upsilon \).

3.2.2 \( p \)- and \( pq \)-relaxations

First let us state the polyhedral relaxations of (F) and (PQ) obtained using single term McCormick inequalities [13] (The relaxation of (PQ) is stronger than \( \Theta \) since \( \Theta \) contains additional valid inequalities). For (F), we introduce an additional variable \( w_{lkj} \) to represent \( p_{lk} y_{lj} \) and relax this bilinear term using [13].

\[
\mathcal{M}(F) := \left\{ (p, y, w): y \in \mathcal{F}, p_{lk} \leq p_{lk} \leq p_{lk}, l \in L, k \in K \right\}
\]

\[
\sum_{i \in I} \lambda_{ik} y_{li} + \sum_{l' \in L} w_{l'kl} = \sum_{j \in L \cup J} w_{lkj}, \quad l \in L, k \in K
\]

\[
\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L} w_{lkj} \leq \mu^\text{max}_{ij} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K
\]

\[
\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L} w_{lkj} \geq \mu^\text{min}_{ij} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K
\]

\( (p_{lk}, y_{lj}, w_{lkj}) \in \mathcal{M}(\{w_{lkj} = p_{lk} y_{lj}\}), \quad l \in L, j \in L \cup J, k \in K \).
For (P,Q), denote \( v_{ilj} = q_{il}y_{lj} \), for \( l \in L, i \in I_l, j \in L \cup J \), and relax this bilinear term using (13).

\[
\mathcal{M}(PQ) := \{ (q, y, v) : y \in F, q_l \in \Delta_{|I_l|}, l \in L \\
y_{il} + \sum_{l' \in L, i' \in I_{l'}, l' \in l} v_{il'j} = \sum_{j \in L \cup J} v_{ilj}, \quad l \in L, i \in I_l \\
\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L, i \in I_l} \sum_{j \in L \cup J} \lambda_{ik} v_{ilj} \leq \mu_{jk}^{\max} \sum_{i \in I} y_{ij}, \quad j \in J, k \in K \\
\sum_{i \in I} \lambda_{ik} y_{ij} + \sum_{l \in L, i \in I_l} \sum_{j \in L \cup J} \lambda_{ik} v_{ilj} \geq \mu_{jk}^{\min} \sum_{i \in I} y_{ij}, \quad j \in J, k \in K \quad (pq\text{-relax}) \\
\sum_{j \in L \cup J} v_{ilj} = y_{lj}, \quad l \in L, j \in L \cup J \\
\sum_{i \in I_l} v_{ilj} \leq C_q y_{lj}, \quad l \in L, i \in I_l \\
(q_{il}, y_{lj}, v_{ilj}) \in \mathcal{M}(\{v_{ilj} = q_{il}y_{lj}\}) \quad l \in L, i \in I_l, j \in L \cup J \}
\]

We now use the results of the previous section to convexify \( Q_l \) for all \( l \).

**Proposition 4.** The convex hull of \( Q_l \) is given by

\[
\text{conv}(Q_l) = \left\{ \left( \{q_{il}\}_{i \in I_l}, \{y_{lj}\}_{j \in L \cup J}, \{v_{ilj}\}_{j \in L \cup J} \right) : q_l \in \Delta_{|I_l|} \\
\sum_{j \in L \cup J} v_{ilj} \leq C_q y_{lj}, \quad i \in I_l \\
\sum_{i \in I_l} v_{ilj} = y_{lj}, \quad j \in L \cup J \\
0 \leq v_{ilj} \leq w_{lj} q_{il}, \quad i \in I_l, j \in L \cup J \right\}
\]

(17)

**Proof.** A direct application of Theorem \( \square \) with \( \Theta = \Delta_{|I_l|} \) and \( \Upsilon = \{y_{lj}\}_{j \in L \cup J} : \sum_{j \in L \cup J} y_{lj} \leq C_l, 0 \leq y_{lj} \leq u_{lj}, j \in L \cup J \} \)

Observe that the description of \( \text{conv}(Q_l) \) requires two McCormick inequalities for each bilinear term \( v_{ilj} = q_{il}y_{lj} \) and the two valid inequalities included in the \( pq\text{-formulation of Equation (13)} \) in the definition of \( Q_l \), we dropped the variables \( y_{il} \) for \( i \in I \cup L \). We could have retained these variables along with their bounds and still applied Theorem \( \square \) to obtain a tighter relaxation than the one presented in (17). However, this stronger relaxation comes at a cost of introducing McCormick inequalities for new bilinear terms of the form \( \bar{v}_{ijl} = q_{il}y_{lj} \), for \( i' \in I \cup L \), thereby increasing the size of this relaxation. Note that since the bilinear terms \( v_{ilj} = q_{il}y_{lj} \) also appear in the spec requirement constraints (8), the additional variables \( v_{ilj} \) introduced in (17) are no more than those necessary.

As noted in Remark \( \square \), Theorem \( \square \) greatly relied on the fact that the set \( \Theta \) is a simplex in order to be able to project out the auxiliary variables \( \lambda, \chi, \rho', \omega' \). This property has an important implication in our context. For the set \( P_l \), the variable \( \chi = \{p_{ik}\}_{k} \) and the set \( \Theta \) is a hypercube \([p_l, p_0] \). Hence Theorem \( \square \) cannot be applied to \( P_l \). The convex hull of \( P_l \) can be obtained using a sequential convexification/level-[K] RLT procedure of Sherali and Adams \[50\]. The relaxation of \( P_l \) using the McCormick envelopes of \( w_{lkj} = p_{lk}y_{lj} \), denoted as \( \mathcal{M}(P_l) \), is just one step of this RLT procedure and hence weaker than the convex hull of \( P_l \).

As seen in the proof of Proposition \( \square \), every point in \( Q_l \) can be mapped to a point in \( P_l \) using the linear mappings \( p_{lk} = \sum_{i \in I_L} \lambda_{ik} q_{il} \) and \( w_{lkj} = \sum_{i \in I_L} \lambda_{ik} v_{ilj} \). It follows that \( \text{conv}(Q_l) \) can be linearly mapped to \( \text{conv}(P_l) \). Hence, \( \text{conv}(Q_l) \) is a tighter relaxation than \( \mathcal{M}(P_l) \). Since this is true for every \( l \in L \), we have the next result.

**Proposition 5.** The \( pq\text{-relaxation } \mathcal{M}(PQ) \) is a stronger relaxation than the \( p\text{-relaxation } \mathcal{M}(P) \) in the sense that for any \( c \in \mathbb{R}^{|A|} \),

\[
\min\{c^\top y : (q, y, v) \in \mathcal{M}(PQ)\} \geq \min\{c^\top y : (p, y, w) \in \mathcal{M}(P)\}.
\]
Note that since we obtained the result of Proposition 5 by proving that \(\text{conv}(Q_1)\) is a tighter relaxation than \(M(P)\), our result is stronger than that of Tawarmalani and Sahinidis [53] for standard pooling problems and Alfaki and Haugland [5] for generalized pooling problems. The single pool argument that we adopted here extends to the hybrid formulation of [2.4.3] 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 \(M(\text{HYB})\) is between \(M(P)\) and \(M(Q)\). There is no relationship between the strengths of \(M(\text{P})\) and \(M(Q)\).

We close this section by remarking on the commodity balance constraints in the pooling problem. First, we address the issue of polyhedrality of the related feasible sets at each pool. Let \(\tilde{P}_l\) be the set containing constraints that define \(\tilde{P}_l\) and tracking \((4a)\) and incoming flow variables \(y_{il}, \forall i \in I\cup L\). Similarly for \(\tilde{Q}_l\). In the case of generalized pooling problems, it is shown in Gupte et al. [22] that the convex hulls of \(\tilde{P}_l\) and \(\tilde{Q}_l\) are unlikely to be polyhedral sets due to the presence of equations \((4a)\) and \((7)\), respectively. However, in the case of standard pooling problems, these complicating bilinear equality constraints are greatly simplified. For \(\tilde{Q}_l\), the commodity balance constraint becomes a defining identity for flows on incoming arcs as \(y_{il} = q_{il} \sum_{j \in L \cup J} y_{lj} \text{ for } i \in I \cup L\) (we dropped the bounds on \(y_{il}\) in the definition of \(\tilde{Q}_l\)). Then it is easy to show that the convex hull of \(\tilde{Q}_l\), and hence \(\tilde{P}_l\), is a polyhedral set. In fact, in this case, \(\text{conv}(Q_l)\) is given by \(\text{conv}(Q_l)\) and the defining identity \(y_{il} = \sum_{j \in L \cup J} y_{lj}, i \in I\).

Second, we discuss some additional relaxation techniques. Only the \(p\)-relaxation is considered since all the presented ideas can be extended to the \(pq\)-relaxation. For the tracking set defined by \((4a)\) along with bounded flows and bounded specifications, Ruiz and Grossmann [17] developed McCormick envelopes in a different space for this set. The corresponding relaxation may or may not be stronger than \(M(\tilde{P})\). For the same set, some additional valid inequalities in the \((p, y)\)-space were proposed by Gupte et al. [22]. Finally, observe that since \(\sum_{j \in L \cup J} p_{jk} y_{lj} = \sum_{j \in L \cup J} y_{lj}\), we can also add a new variable \(\tilde{w}_{lk}\) and further impose

\[
\sum_{j \in L \cup J} w_{lkj} = \tilde{w}_{lk}, \quad (p_{lk}, \sum_{j \in L \cup J} y_{lj}, \tilde{w}_{lk}) \in M\left(\{\tilde{w}_{lk} = p_{lk} \sum_{j \in L \cup J} y_{lj}\}\right)
\]

in the definition of \(M(\tilde{P})\). This relaxation was also devised by Liberti and Pantelides [32] using the Reduced RLT (RRLT) procedure. Due to the presence of nontrivial upper bound \(C_l\) on \(\sum_{j \in L \cup J} y_{lj}\), this new relaxation is not necessarily dominated by \(M(\tilde{P})\). However, it is important to observe that the result of Proposition 5 carries through even after tightening the \(p\)-relaxation with such a RRLT.

3.3 Other relaxations

3.3.1 Piecewise linear

The strength of the McCormick envelopes [13] for a single bilinear term, given by \(X\), depends on the bounds \([a_1, b_1]\) and \([a_2, b_2]\) on the variables \(x\) and \(\rho\), 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 [13] 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 MILP relaxation, referred to as the piecewise linear McCormick relaxation, of the set \(X\).

Piecewise linear McCormick relaxations were used by Meyer and Floudas [36] to solve some generalized pooling problems. Hasan and Karimi [25], Wicaksono and Karimi [59] proposed alternative MILP models for piecewise linear relaxations. An extensive computational study on small scale pooling problems was performed by Gounaris et al. [19] to investigate different partitioning levels and MILP models. Recently, Misener et al. [41] implemented a generic branch-and-bound based solver for pooling problems that uses piecewise linear MILP relaxations at each node of the branch-and-bound tree.

3.3.2 Lagrangian

For the standard pooling problem, various Lagrangian relaxations have been proposed over the years. Ben-Tal et al. [10] used a Lagrangian dual of the \(q\)-formulation to generate a converging
sequence of lower bounds in a branch-and-bound algorithm. Adhya et al. [4] dualized all constraints except the bounds on \( p_k \) and \( y_{ij} \) variables in the \( p \)-formulation. Almutairi and Elhedhli [6] went one step further by dualizing only the bilinear constraints in the \( p \)- and \( pp \)-formulations. A more general purpose global optimization algorithm (GOP) based on Lagrangian duals was applied to the Haverly test problems in Visweswaran and Floudas [58].

Here we show that the result of [53, Proposition 9.9] extends to generalized pooling problems.

**Proposition 6.** Consider the Lagrangian relaxation of \((PQ)\) obtained by dualizing all constraints except the ones in the set

\[
\Omega = \left\{ (q, y) : 0 \leq y_{ij} \leq u_{ij}, (i, j) \in A, \quad q_l \in \Delta_{i_l j_l}, i \in I, \quad \sum_{j \in I_l} y_{ij} \leq C_l, l \in L \right\}.
\]

Then the lower bound provided by this dual is equal to that due to \( M(PQ) \).

**Proof.** We follow the same steps as those in the proof of [53, Proposition 9.9]. Denote the proposed relaxation as \((R1)\). Consider the Lagrangian relaxation that dualizes all the constraints of \( M(PQ) \) except the ones used in the description of \( \text{conv}(Q_l) \) for \( l \in L \). Denote this relaxation as \((R2)\). By strong duality of linear programming, \((R2)\) attains the same optimal value as \( M(PQ) \). Once the bilinear constraints have been dualized, the Lagrangian subproblem in \((R1)\) becomes separable across pools. Observe that for any \( l \in L \), optimizing a bilinear function of the form \( \sum_{i,j} \alpha_{ij} q_l y_{ij} \) over \( \Omega \) is equivalent to optimizing \( \sum_{i,j} \alpha_{ij} v_{ilj} \) over \( \text{conv}(Q_l) \). Thus, for each dual solution, the Lagrangian subproblems for \((R1)\) and \((R2)\) attain the same value. Hence, the statement of the proposition. \( \square \)

### 4 Discretization strategies

In this section, we study a method of 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. We solve the different MILP models and compare the quality of the MILP solution value against the best feasible solution found by a global solver. 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 MILP approximations of BLP

Consider a bilinear program where each bilinear term can be represented by a set of the form \( \{ (\chi, \rho, \omega) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}_+ : \omega = \chi \rho, \chi \in [0, a], \rho \in [0, b] \} \). (18)

For the sake of simplicity, we have assumed the lower bounds on \( \chi \) and \( \rho \) to be zero. Although we assumed that both \( \chi \) and \( \rho \) are continuous variables, our ideas can be easily extended to the case when the original problem is a mixed integer bilinear program and one or both \( \chi \) and \( \rho \) are integer variables.

Now suppose that we discretize \( \rho \), i.e. restrict \( \rho \) to take only integer values within its bounds \([0, b]\). This gives us another set \( \mathcal{X}' \subset \mathcal{X} \) where

\[
\mathcal{X}' = \{(\chi, \rho, \omega) \in \mathbb{R}_+ \times \mathbb{Z} \times \mathbb{R}_+ : \omega = \chi \rho, \chi \in [0, a], \rho \in [0, b] \}.
\] (19)

Thus, \( \mathcal{X}' \) is a approximation of \( \mathcal{X} \). (If in \((18)\), \( \rho \in \mathbb{Z} \) is valid to \( \mathcal{X} \), then \( \mathcal{X}' = \mathcal{X} \).) Substituting \( \mathcal{X}' \) for every occurrence of \( \mathcal{X} \) gives a MIBLP approximation of BLP. It is important to note here that for \( b > 1 \)

\[
\mathcal{X}' \subset M(\mathcal{X}') = \text{conv}(\mathcal{X}').
\] (20)

The general integer restriction \( \rho \in \{0, 1, \ldots, b\} \) can be rewritten as a disjunction \( \rho \in \bigcup_{r=0}^{b} \{r\} \) and hence we have \( \mathcal{X}' = \bigcup_{r=0}^{b} \mathcal{X}'_r \), where \( \mathcal{X}'_r = \{(\chi, \rho, \omega) : \omega = r \chi, \rho = r, \chi \in [0, a] \} \). The
extended formulation for this disjunction of polytopes is given by

$$
U(\mathcal{X}^\rho) := \left\{ (\chi, \rho, \omega, \zeta, \nu) : \omega = \sum_{r=0}^b r \nu_r, \quad \rho = \sum_{r=0}^b r \zeta_r, \quad \sum_{r=0}^b \zeta_r = 1, \quad \chi \in \mathcal{M}(\nu_r = \chi \zeta_r), \quad r = 1, \ldots, b \right\}.
$$

Upon observing that \( \rho = \sum_{r=0}^b r \zeta_r \) can be interpreted as the base-1 expansion of the general integer variable \( \rho \), we refer to \( U(\mathcal{X}^\rho) \) as the \textit{unary reformulation} of \( \mathcal{X}^\rho \). Note that \( \zeta_r \in \{0, 1\}, \forall r \) and \( \sum_r \zeta_r = 1 \) imply a SOS-1 constraint, which can be reformulated using a logarithmic number of extra variables and constraints as shown by Vielma and Nemhauser [60].

$$
L(\mathcal{X}^\rho) := \left\{ (\chi, \rho, \omega, \zeta, \nu, \delta) : \omega = \sum_{r=0}^b r \nu_r, \quad \rho = \sum_{r=0}^b r \zeta_r, \quad \sum_{r=0}^b \zeta_r = 1, \quad \chi \in \mathcal{M}(\nu_r = \chi \zeta_r), \quad r = 1, \ldots, b \right\}.
$$

where \( \ell(b) := \lfloor \log_2 b \rfloor + 1 \) and \( \text{supp}(r) \) is the support of the binary encoding of \( r - 1 \) defined as

\[
\text{supp}(r) := \begin{cases} 
\{ t \in \{1, \ldots, \ell(b)\} : [(r - 1)/2^{t-1}] \mod 2 \equiv 1 \}, & r \geq 2 \\
\emptyset, & r = 1.
\end{cases}
\]

Although \( L(\mathcal{X}^\rho) \) has more variables and constraints than \( U(\mathcal{X}^\rho) \), it has fewer \( \{0, 1\} \) variables which can always be an advantage while trying to solve the problem with a branch-and-bound algorithm. We refer to \( L(\mathcal{X}^\rho) \) as the \textit{log unary reformulation} of \( \mathcal{X}^\rho \).

Using the base-2 expansion of \( \rho \) leads to the following \textit{binary reformulation} of \( \mathcal{X}^\rho \).

$$
B(\mathcal{X}^\rho) := \left\{ (\chi, \rho, \omega, \zeta, \nu) : \omega = \sum_{r=1}^{\ell(b)} 2^{r-1} \nu_r, \quad \rho = \sum_{r=1}^{\ell(b)} 2^{r-1} \zeta_r, \quad \sum_{r=1}^{\ell(b)} 2^{r-1} \zeta_r \leq b, \quad \chi \in \mathcal{M}(\nu_r = \chi \zeta_r), \quad r = 1, \ldots, \ell(b) \right\}.
$$

\[\text{supp}(r) := \begin{cases} 
\{ t \in \{1, \ldots, \ell(b)\} : [(r - 1)/2^{t-1}] \mod 2 \equiv 1 \}, & r \geq 2 \\
\emptyset, & r = 1.
\end{cases}\]

Remark 4. The reformulation sizes of \( U(\cdot), L(\cdot), B(\cdot) \) can be compared as follows.

\[
U(\cdot) \succ_{\text{bin}} L(\cdot) \equiv_{\text{bin}} B(\cdot)
\]

where the subscripts \( \text{bin}, \text{cont}, \text{eq} \) denote the number of binary variables, continuous variables, and constraints, respectively. The LP relaxations of \( U(\cdot) \) and \( B(\cdot) \) were compared in Gupte et al. [23] and it was proven that in general, neither dominates the other.
4.2 MILP approximations of the pooling problem

We now apply the models of the previous section to the pooling problem. Consider the formulation \((\mathcal{PQ})\). Each bilinear term is of the form \(v_{ij} = q_{il}y_{lj}\) for some \(l \in L, i \in I, j \in L \cup J\). Hence, the set corresponding to \(\mathcal{X}\) from §4.1 is

\[
Q_{ij} := \{(q_{il}, y_{lj}, v_{ij}) : v_{ij} = q_{il}y_{lj}, q_{il} \in [0, 1], y_{lj} \in [0, u_{lj}])\}, \quad l \in L, i \in I, j \in L \cup J. \tag{24a}
\]

We have two choices for discretization: either \(\rho = q_{il}\) or \(\rho = y_{lj}\). Similarly, the set representing a single bilinear term in \((\mathcal{P})\) is

\[
P_{ikj} := \{(p_{ik}, y_{lj}, w_{ik}) : w_{ik} = p_{ik}y_{lj}, p_{ik} \in [\bar{p}_{ik}, \bar{p}_{ik}], y_{lj} \in [0, u_{lj}])\}, \quad l \in L, k \in K, j \in L \cup J, \tag{24b}
\]

and we may discretize either \(p_{ik}\) or \(y_{lj}\).

4.2.1 Flow discretization

The flow discretized model is obtained by discretizing \(y_{lj}\) within \([0, u_{lj}]\), for all \(l \in L, j \in J\).

\[
\mathcal{FP}_{ij} := P_{ikj} \cap (\mathbb{R}_+ \times \mathbb{Z} \times \mathbb{R}_+), \quad l \in L, k \in K, j \in L \cup J \tag{25a}
\]

\[
\mathcal{FQ}_{ij} := Q_{ij} \cap (\mathbb{R}_+ \times \mathbb{Z} \times \mathbb{R}_+), \quad l \in L, i \in I, j \in L \cup J. \tag{25b}
\]

The flow discretized feasible set for \((\mathcal{P})\) is denoted by \(\mathcal{FP}\) and its binary MILP reformulation is \(\mathcal{B}(\mathcal{FP})\). Similarly, for the \(pq\)-formulation \((\mathcal{PQ})\). 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 \(C_I\) and \(u_{lj}\) are integers, for all \(l \in L, j \in L \cup J\), otherwise they can be replaced with \([\bar{C}_I]\) and \([u_{lj}]\), respectively.

Recall the set corresponding to pool \(l\) from (14b) and define \(\mathcal{FQ}_l\) to be its discretized counterpart.

\[
\mathcal{FQ}_l := \left\{ \left( \left\{ q_{il} \right\}_{i \in I_l}, \left\{ y_{lj} \right\}_{j \in L \cup J}, \left\{ v_{ilj} \right\}_{j \in L \cup J} \right) : \left( q_{il}, y_{lj}, v_{ilj} \right) \in \mathcal{FQ}_{ij}, i \in I_l, j \in L \cup J, \right. \left. q_{il} \in \Delta_{[I_l]}, \sum_{j \in L \cup J} y_{lj} \leq C_I \right\}.
\]

Proposition 7. For any \(l \in L\), \(\text{conv}(\mathcal{FQ}_l) = \text{conv}(Q_l)\).

Proof. Follows from Theorem 1 and integrality of the polytope \(\{y_l : \sum_{j \in L \cup J} y_{lj} \leq C_I, y_{lj} \in [0, [u_{lj}]], j \in L \cup J\}\).

For \(l \in L, j \in L \cup J\), define \(\mathcal{FQ}_{lj}\) as

\[
\mathcal{FQ}_{lj} := \left\{ \left( q_{ilj} \right)_{i \in I_l}, y_{lj}, v_{ilj} \right\} : \left( q_{ilj}, y_{lj}, v_{ilj} \right) \in \mathcal{FQ}_{lij}, i \in I_l, q_{ilj} \in \Delta_{[I_l]} \right\}.
\]

Thus by construction,

\[
\mathcal{FQ}_l = \{y_l : \sum_{j \in L \cup J} y_{lj} \leq C_I\} \cap \bigcap_{j \in L \cup J} \mathcal{FQ}_{lj}, \quad l \in L.
\]

A similar set \(\mathcal{FP}_{lj}\) is defined for the \(p\)-formulation. We now state a result for the convex hull of binary expansions of these sets.
Proposition 8. For any \( l \in L, j \in L \cup J \), let \( \mathcal{J}_{lj} \) denote the base-2 expansion of \( u_{lj} \) such that 
\[
u_{lj} = \sum_{r \in \mathcal{J}_{lj}} 2^{r-1}.
\]
Then,
\[
\text{conv}(\mathcal{B}(\mathcal{FQ}_{lj})) = \left\{ \left( q_l, y_{lj}, \{v_{lj}\}_{i \in I_l}, \{\zeta_{lj}\}_r, \{\nu_{lj}\}_r \right) : q_l \in \Delta_{I_l} \right\}
\]
\[
y_{lj} = \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{lj}, \quad v_{lj} = \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\nu_{lj}, \quad i \in I_l
\]
\[
0 \leq \nu_{lj} \leq q_{lj}, \quad r = 1, \ldots, \ell(u_{lj}), i \in I_l
\]
\[
\nu_{lj} + \sum_{r' \in \mathcal{J}_{lj}, r' > r} \zeta_{lj} \leq \{ r' \in \mathcal{J}_{lj}, r' > r \} \setminus \mathcal{J}_{lj}, \quad r = 1, \ldots, \ell(u_{lj})
\]

Similarly, the convex hull of \( \mathcal{B}(\mathcal{FP}_{lj}) \) can be derived. Furthermore,
\[
\text{conv} \left( \bigcap_{j \in L \cup J} \mathcal{B}(\mathcal{FQ}_{lj}) \right) = \bigcap_{j \in L \cup J} \text{conv}(\mathcal{B}(\mathcal{FQ}_{lj})),
\]

Proof. From Gupte et al. [23] Propositions 3.3 and 3.4 we obtain that the nontrivial facets describing the convex hull of the knapsack \( \{ \{ \zeta_{lj} \}_r : \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{lj} \leq u_{lj}, \zeta_{lj} \in \{0, 1\}, \forall r \} \) are
\[
\zeta_{lj} + \sum_{r' \in \mathcal{J}_{lj}, r' > r} \zeta_{lj} \leq \{ r' \in \mathcal{J}_{lj}, r' > r \} \setminus \mathcal{J}_{lj}, \quad r = 1, \ldots, \ell(u_{lj})
\]

Applying Theorem 1 then gives us the desired description for \( \text{conv}(\mathcal{B}(\mathcal{FQ}_{lj})) \). Since the sets \( \mathcal{B}(\mathcal{FQ}_{lj}), j \in L \cup J, \) share a common variable \( q_l \in \Delta_{I_l} \), the second result in the statement follows directly from Theorem 3.

The above result gives us the convex hull of the set \( \bigcap_{j \in L \cup J} \mathcal{B}(\mathcal{FQ}_{lj}) \) which does not include the pool capacity constraint. If additional valid inequalities are known for the set

\[
\left\{ \{ \zeta_{lj} \}_r, 2^{r-1}\zeta_{lj} \leq C_l, \sum_{r=1}^{\ell(u_{lj})} 2^{r-1}\zeta_{lj} \leq u_{lj}, j \in L \cup J, \zeta_{lj} \in \{0, 1\}, \forall r \right\}
\]

then Proposition 8 can be suitably strengthened.

4.2.2 Ratio discretization

Here we discretize the ratio variables \( q_{lj} \) for all \( l \in L, i \in I_l \). Although each ratio \( q_{lj} \) can be discretized into different intervals, for the ease of exposition, we assume that all the ratios are uniformly discretized into \( n \geq 1 \) intervals of equal length within \([0, 1]\). Note that unlike Section 4.2.1 where discretizing flows 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 \( n \).

\[
\mathcal{RQ}_{lj} := \{ (q_{lj}, y_{lj}, v_{lj}) : v_{lj} = q_{lj}y_{lj}, q_{lj} \in [0, n] \cap \mathbb{Z}, y_{lj} \in [0, u_{lj}] \}, \quad l \in L, i \in I_l, j \in L \cup J.
\]

The ratio discretized feasible set for the \( pq \)-formulation is denoted by \( \mathcal{RPQ} \). Applying the MILP reformulations of [4.1] to \( \mathcal{RQ}_{lj} \) gives us \( \mathcal{U}(\mathcal{RQ}_{lj}), \mathcal{L}(\mathcal{RQ}_{lj}), \) and \( \mathcal{B}(\mathcal{RQ}_{lj}) \), respectively. Consequently, the MILP models for \( \mathcal{RPQ} \) are \( \mathcal{U}(\mathcal{RPQ}), \mathcal{L}(\mathcal{RPQ}), \) and \( \mathcal{B}(\mathcal{RPQ}) \), respectively. For the binary reformulation \( \mathcal{B}(\mathcal{RPQ}) \), we can derive cutting planes similar to the ones presented in Proposition 8. However, we cannot obtain a similar result for the convex hull of \( \mathcal{B}(\mathcal{RQ}_{lj}) \) because the nondiscretized variables \( y_{lj} = \{ y_{lj} \}_{l \in L \cup J} \) belong to a polytope, given by \( \{ y_{lj} : \sum_{j \in L \cup J} y_{lj} \leq C_l, y_{lj} \in [0, u_{lj}] \} \), and not a simplex.

We now discuss a MILP reformulation of \( \mathcal{RPQ} \) due to Alfaki and Haugland [3]. In this formulation we do not include additional variables \( v_{lj} \) for the bilinear terms \( q_{lj}y_{lj} \) nor do we add
\( (\zeta_{ij}, \nu_{ij}) \forall i, r, l, j, \) of \( [1,1] \). Instead, for each pool \( l \in L \), we explicitly enumerate all the feasible points of the discretized simplex

\[
\tilde{\Delta}_{|I_l|} = \{ q_l \in Y_L^{|I_l|} : \sum_{i \in I_l} q_{il} = 1, nq_{il} \in [0, n] \cap \mathbb{Z}, i \in I_l \},
\]

and create duplicate nodes, one for each feasible point in \( \tilde{\Delta}_{|I_l|} \). These duplicate pools inherit properties, such as arc connectivity and node capacity, of its parent pool node. For each duplicate pool an additional binary variable is created which is equal to 1 if and only if the discretized ratio corresponding to that duplicate pool is selected. For each original pool, the binary variables corresponding to its duplicate pools are SOS-1 constrained. Note that the cardinality of \( \tilde{\Delta}_{|I_l|} \) is exponentially large and hence this model has exponentially many new \( \{0, 1\} \) variables and constraints.

Consider the pooling problem on directed graph \( G = (\mathcal{N}, \mathcal{A}) \). We will define the proposed MILP approximation for the pooling problem on an expanded directed graph \( \tilde{G} = (\tilde{\mathcal{N}}, \tilde{\mathcal{A}}) \). For each pool \( l \in L \), let \( L_l \) be a set of duplicate pools of cardinality \( |\tilde{T}_{|I_l|}| \). Thus, \( \tilde{L} = \bigcup_{l \in L} L_l \) is the set of duplicate pools and \( \tilde{\mathcal{N}} = \mathcal{N} \cup \tilde{\mathcal{L}} \). Denote \( \tilde{\Delta}_{|I_l|} = \{ \tilde{q}_l^1, \ldots, \tilde{q}_l^{|\tilde{T}_{|I_l|}|} \} \) where each \( \tilde{q}_l^i \) is a vector that can be computed apriori. For every \( l' \in L_l \), introduce an arc \((l', l)\). The new set of arcs is \( \tilde{\mathcal{A}} = \tilde{\mathcal{A}} \bigcup \bigcup_{l \in L} L_l \times \{l\} \).

Define \( y_{ij} \) as the flow on arc \((i, j)\) in \( \tilde{A} \). For \( l \in L, l' \in L_l \), let \( u_{rl} \in \{0, 1\} \) be a binary variable such that \( u_{rl} = 1 \) if and only if \( q_{il} = \tilde{q}_l^i, \forall i \in I_l \), i.e. the \( l' \)th discretized ratio is chosen at pool \( l \). Note that we do not need to add \( q \) variables in this MILP. There are two types of combinatorial constraints: the first one an SOS-1 constraint to ensure exactly one discretization is chosen for pool \( l \), and the second one a variable upper bound constraint such that \( y_{rl} = 0 \) if \( u_{rl} = 0 \).

\[
\sum_{l' \in L_l} u_{rl'} = 1, \quad l \in L \tag{28a}
\]
\[
y_{rl} \leq C_l u_{rl}, \quad l \in L, l' \in L_l. \tag{28b}
\]

The total incoming flow at \( l \) from its adjacent nodes in \( G \) must be equal to the total incoming flow from duplicate pools in \( \tilde{G} \).

\[
\sum_{i \in I_l \cap L} y_{il} = \sum_{l' \in L_l} y_{rl'}, \quad l \in L. \tag{29}
\]

The commodity balance constraints \((7)\) are discretized as

\[
y_{il} + \sum_{l' \in L_l} \sum_{l'' \in L_{l''}} d_{l''} q_{l''} y_{l''} = \tilde{q}_l^i \sum_{j \in L \cup J} y_{ij}, \quad l \in L, i \in I_l, l' \in L_l. \tag{30}
\]

The spec requirement constraint \((9)\) at the outputs is discretized as

\[
\sum_{i \in I} \lambda_i y_{ij} + \sum_{l' \in L} \sum_{l'' \in L_{l''}} \rho_j \tilde{q}_l^i y_{rl} \leq \rho_j^{\max} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K. \tag{31a}
\]
\[
\sum_{i \in I} \lambda_i y_{ij} + \sum_{l' \in L} \sum_{l'' \in L_{l''}} \rho_j \tilde{q}_l^i y_{rl} \geq \rho_j^{\min} \sum_{i \in I \cup L} y_{ij}, \quad j \in J, k \in K. \tag{31b}
\]

Since the values of \( \tilde{q}_l^i \) are known for all \( l \in L, l' \in L_l \), \((30)\) and \((31)\) are linear constraints. Thus, we have the following MILP approximation.

\[
\min_{y, u} \sum_{(i, j) \in \mathcal{A}} c_{ij} y_{ij}
\]

s.t. \( y \in \mathcal{F} \), \( u_{rl} \in \{0, 1\}, l \in L, l' \in L_l \)

\((28) - (31)\)

The minimization in \((EPQ)\) is over \((y, u)\) and there are exponentially many variables \( u \)'s and constraints \((28b), (30)\).
4.2.3 Spec discretization

The spec discretized model is obtained by discretizing \( p_{lk} \) within \([\underline{p}_{lk}, \bar{p}_{lk}]\) in (24b).

\[
SP \chi_{lkj} := \{ (p_{lk}, y_{lj}, w_{lkj}) : w_{lkj} = p_{lk} y_{lj}, y_{lj} \in [0, u_l], \\
np_{lk} \in \{ n\underline{p}_{lk} + (\bar{p}_{lk} - \underline{p}_{lk}) r \}^n_{r=0}, \ l \in L, k \in K, j \in L \cup J. \]  

Similar to ratio discretization, we will experiment with different values of \( n \). The spec discretized feasible set for the \( p \)-formulation is denoted by \( SP \) and the MILP models are \( U(SP) \), \( L(SP) \), and \( B(SP) \), respectively. For the binary reformulation \( B(SP) \), we can derive cutting planes similar to the ones presented in Proposition 8. The unary model for spec discretization was first studied by Pham et al. [43].

4.3 Computations

In this section we report computational results on several test instances of the pooling problem. Our purpose is to assess usefulness of the proposed discretization models. Towards this end, we solve each discretization model and the original (nondiscretized) pooling problem. Then we compare the best feasible solutions that are obtained after solving all these models. For ratio and spec discretization, we try different values for the level of discretization. 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. Kolodziej et al. [30] present one computational study of dynamically updating base-10 discretizations of mixed integer variants of the pooling problem.

4.3.1 Experimental setup

All experiments were run on a Linux machine with kernel 2.6.18 running on a 64-bit x86 processor and 32GB of RAM. We considered only the \( p \)- and \( pq \)-formulations and their discretized counterparts. We used Cplex 12.2 as the MILP solver and BARON 9.3.1 as the BLP and MIBLP solver. All formulations were modeled using GAMS 23.6. The time limit for solving each discretization was set to 1hr. The time limit for solving original problem formulation with BARON was set to 24hr. Since BARON [cf. 49] 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 sake of fair comparison, we gave BARON a much longer time limit of 24 hours versus 1 hour for the discretizations. We use the best feasible solution (if one exists) and the corresponding upper bound as returned by the solver at the end of its time limit. If no feasible solution is returned because the solver could not find one within 1hr, then we use the best upper bound value that might be reported by the solver.

Given an instance \( I \) of the pooling problem, we solved \( P \) and \( PQ \) formulations using BARON for 24hr. We discuss the various discretization models solved for \( PQ \) and comment that a similar technique can be extrapolated to \( P \). For \( PQ \), we solved \( FPQ \) and \( RPQ \) as MIBLPs, and \( B(FPQ) \), \( B(RPQ) \), \( U(RPQ) \), \( L(RPQ) \), and \( EPQ \) as MILPs. For flow discretization, amongst all the MILPs, we considered only the binary expansion model \( B(FPQ) \) and discretized flows within their bounds, as explained in Section 4.2.1. In ratio discretization, we tested different values of \( n \). 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. For \( n \in \{1, 2, 4 \} \), we solved \( RPQ, U(RPQ) \), and \( EPQ \). For higher values \( n \in \{7, 15, 31 \} \), we solved \( RPQ, U(RPQ) \), \( L(RPQ) \), and \( B(RPQ) \). Since the number of variables and constraints in \( EPQ \) grows exponentially with \( n \), we did not consider this model for high values of \( n \). The effect of valid inequalities of Proposition 8 was tested with both \( B(FPQ) \) and \( B(FP) \).

To ensure numerical consistency among 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 = \( 10^{-3} \). Additionally, for Cplex,
we set Threads = 1 and MIPEmphasis = 1 (feasibility) and for solving generalized and large-scale standard instances with BARON, we set PreLPDo = 3 to preprocess only the original problem variables. We used SNOPT as the NLP solver with BARON. While solving PQ, FPQ, and RPQ with BARON, the valid inequalities [10] were added only at the root node by defining their corresponding .equclass suffix [cf. 19] to be equal to 1. For B(FPQ) and B(FP), the valid inequalities of Proposition 8 were added to the user cut pool of Cplex. 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.

4.3.2 Test instances

The pooling instances that are commonly used in literature mostly comprise the small-scale problems proposed many years ago [1, 10, 26]. Since these problems are solved in a matter of seconds by BARON, they are not of particular interest in testing our discretization methods. Our test set comprises of 30 medium- to large-scale instances of the pooling problem. 20 instances are standard pooling problems that were created by Alfaki and Haugland [4] and can be downloaded from the website [http://www.ii.uib.no/~mohammeda/spooling/]. These instances are named stdA0 - stdA9, stdB0 - stdB5, stdC0 - stdC3. Of the 10 generalized pooling instances, 3 are the ones used in Meyer and Floudas [36]. In these instances, besides the classical pooling problem of [2] there are additional binary decision variables related to the use of each arc or node in the graph. Also, the spec tracking constraints (4a) are formulated as

$$\eta_{lk} \left[ \sum_{i \in I} \lambda_{ik} y_{il} + \sum_{t \in T} P_{i,k} y_{lt} \right] = p_{lk} \sum_{j \in L \cup J} y_{lj} \quad l \in L, k \in K,$$

where $\eta_{lk}$ is an absorption coefficient of spec $k$ at pool $l$. Hence, to write the $pq$-formulation of this problem, we need to define ratio variables $q_{lt}^{\tau}$ along each path $\tau$ such that $q_{lt}^{\tau}$ denotes the ratio of incoming flow to $l$ along path $\tau$ starting from input $i$. The $p$ variables are given by

$$p_{lk} = \sum_{i} \sum_{\tau} \lambda_{ik} q_{lt}^{\tau} \quad l \in L, k \in K.$$

This makes the formulation larger in size due to its path dependency and hence we do not consider PQ and its discretizations for the 3 instances of Meyer and Floudas.

Some instances of the generalized pooling problem can also be found in Alfaki and Haugland [5]. However, in our experience, the $pq$-formulations of these instances were solved by BARON in less than 15 minutes and hence we chose not to include them in our test set due to their relative ease. In order to further expand our test set, we generated 7 random instances, Inst1 - Inst7, of the time indexed pooling problem, which we explain below.

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 $y_{i,j,t}$ on arc $(i, j) \in A$, 2) $s_{it}$ amounts of inventory to be held at a node $i \in N$, 3) $x_{i,l}^{in} = 1$ iff there is inflow at pool $l$, 4) $x_{i,l}^{out} = 1$ 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 $h_{lt}$ 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 $\tau$ and can only be used under contract. Typically, $\tau \leq |T|$ and the contracts are renewable. For a pool $l \in L_c$, the fixed cost $h_{lt}$ is associated with the entire contract.

We first state the $p$-formulation of this problem. $p_{kt}$ denotes the concentration value of spec $k$ at pool $l$ at time $t$. 
\[ \min \sum_{t \in T} \sum_{(i,j) \in A} c_{ij} y_{ijt} + \sum_{t \in T} \sum_{l \in L} h_l z_{lt}, \]

\[ a_{lt} + s_{i(t-1)} = \sum_{t \in L \cup J} y_{ilt} + s_{it}, \quad i \in I, t \in T, \]
\[ \sum_{i \in L \cup J} y_{ilt} + s_{j(t-1)} = s_{lt} + \sum_{j \in L \cup J} y_{jlt}, \quad l \in L, t \in T, \]
\[ \sum_{i \in L \cup J} y_{jlt} + s_{j(t-1)} = s_{jlt} + d_{jlt}, \quad j \in J, t \in T, \]
\[ \sum_{i \in I} \lambda_{ik} y_{ilt} + \sum_{l \in L} p_{kt} y_{ilt} + p_{k(t-1)} s_{l(t-1)}, \]
\[ = p_{kt} \left( \sum_{j \in L \cup J} y_{jlt} + s_{lt} \right), \quad l \in L, k \in K, t \in T, \]
\[ \sum_{i \in I} \lambda_{ik} y_{jlt} + \sum_{l \in L} p_{kt} y_{jlt} \leq \mu_{jk} \sum_{i \in I} y_{jlt}, \quad j \in J, k \in K, t \in T, \]
\[ \sum_{i \in I} \lambda_{ik} y_{jlt} + \sum_{l \in L} p_{kt} y_{jlt} \geq \mu_{jk} \sum_{i \in I} y_{jlt}, \quad j \in J, k \in K, t \in T, \]
\[ (y, s, x_{in}^t, x_{out}^t, z) \in Z, \]
\[ 0 \leq s_{lt} \leq C_l, \quad l \in L, t \in T, \]

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

\[ Z := \{(y, s, x_{in}^t, x_{out}^t, z) : \]
\[ y_{jlt} \in \{0\} \cup [\ell_j, u_{ij}], \quad (i, j) \in A, t \in T, x_{in}^t, x_{out}^t, z_{lt} \in \{0, 1\}, \quad l \in L, t \in T, \]  
\[ x_{in}^t + x_{out}^t \leq z_{lt}, \quad l \in L \setminus L_c, t \in T, \]  
\[ x_{in}^t + x_{out}^t \leq \min \left\{ 1, \sum_{l' = t - \tau_t^1 + 1}^t z_{lt'} \right\}, \quad l \in L_c, t \in T, \]  
\[ y_{ilt} \leq u_{ilt} x_{in}^t, \quad l \in L, i \in I \cup L, t \in T, \]  
\[ y_{jlt} \leq u_{jlt} x_{out}^t, \quad l \in L, j \in L \cup J, t \in T, \]  
\[ s_{lt} \leq C_l \sum_{l' = t - \tau_t^1 + 1}^t z_{lt'}, \quad l \in L_c, t \in T \} \]  

The combinatorial constraints can be explained as follows. Equation \([33]\) 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 \( t \). Equations \([34] \) and \([35] \) enforce either inflow or outflow at each pool and for contract pools, the constraint that there should be no flow if the contract has expired. The next two constraints \([36] \) and \([37] \) ensure consistency between incoming and outgoing binary variables and incoming and outgoing flows at each pool. The last constraint \([38] \) 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 \( I'_{[l, t]} = \{(i, t') \in I' : i \in I, t' \leq t\} \), i.e. all the input nodes in \( I \) that had a path to \( l \) and time index before \( t \). Thus the proportion variable \( q_{ilt} \) 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 \((l, j) \in A \) from pool \( l \), we have the bilinear terms \( v_{ilt} y_{jlt} = q_{ilt} y_{jlt} \) and \( v_{ilt}' s_{lt} = q_{ilt}' s_{lt} \). We can now formulate the time indexed pooling problem using the \( q \)- or \( pq \)-formulation for generalized pooling problems.
4.3.3 Results

Preliminary observations.

1. For \( n \leq 4 \), the performance of \( U(\cdot) \) and \( L(\cdot) \) was comparable and for \( B(\cdot) \), Cplex was able to find better quality feasible solutions and was closer to proving optimality as compared to \( U(\cdot) \) and \( L(\cdot) \). On the other hand for \( n > 4 \), \( L(\cdot) \) was a little better than \( U(\cdot) \) but worse than \( B(\cdot) \). Although we used the feasible solutions from \( L(\cdot) \) for sake of comparison in Table 4.3.3, we henceforth report \% gaps and detailed results for only the \( U(\cdot) \) and \( B(\cdot) \) MILPs since they seem to represent two extreme cases.

2. Note that \( y = 0 \in \mathcal{F} \) represents a feasible solution for the pooling problem defined in § 2. Hence, we provided \( y = 0 \) as a starting solution to Cplex and BARON, hoping that the solvers will be able to find nontrivial solutions. Since meyer*, Inst* have lower bounds, demands and/or combinatorial constraints on the flows, \( y = 0 \) does not represent a feasible solution for these instances.

3. While solving the unary discretizations of large-scale instances stdB*, stdC*, Inst5-7 with \( n > 4 \), we sometimes found it useful to set the following parameters in Cplex: (a) RootAlg = 4 for choosing barrier algorithm to solve root node relaxation, (b) VarSel = 4 for choosing the computationally inexpensive pseudo reduced cost branching scheme. Similarly for flow discretizations \( B(\text{FP}) \) and \( B(\text{FPQ}) \).

4. Valid inequalities of Proposition 8 were not significantly effective in strengthening the root node relaxations of \( B(\text{FP}) \) and \( B(\text{FPQ}) \); cf. Table 9 in Appendix A. They were slightly more effective on \( B(\text{FPQ}) \) than \( B(\text{FP}) \), as expected from the second statement of Proposition 8. In general, most of these cuts were separated deeper down the branch-and-cut tree than at the root node. Since our cuts did not seem to be very helpful on \( B(\text{FP}) \) and \( B(\text{FPQ}) \), we solved \( B(\text{SP}) \) and \( B(\text{RPQ}) \) for \( n = 7, 15, 31 \), knowing from [23, Proposition 2.2] that when \( n = 2^\gamma - 1 \) for some \( \gamma \in \mathbb{Z}_{+} \), then no additional cuts are needed to convexify the individual sets \( B(S^P \chi_{I_k}) \) and \( B(R^Q \chi_{I_k}) \), respectively.

The detailed results of our computational experiments are provided in Appendix A. In this section, we discuss important observations from our experiments.

We first present a summary of the results in Table 4.3.3. The quality of the solutions obtained by the different discretization strategies is compared using the percentage gap between upper bound and lower bound. This gives us an estimate of how well a particular discretization model might perform if implemented as a heuristic in a branch-and-cut algorithm. For each instance \( \mathcal{J} \) and method \( \mathcal{M} \), we record the best upper bound value \( \nu_{\mathcal{M}}(\mathcal{J}) \) returned upon termination of \( \mathcal{M} \). For every \( \mathcal{J} \), we record the best lower bound value \( \ell(\mathcal{J}) \) returned by BARON for \( P \) and \( \mathcal{P} \) after 24hr. Then \( \ell(\mathcal{J}) = \max\{\ell(\mathcal{J}) \text{ for } P, \ell(\mathcal{J}) \text{ for } \mathcal{P}\} \) and \% gap of method \( \mathcal{M} \) is

\[
\omega_{\mathcal{M}}(\mathcal{J}) = 100 \times \left| 1 - \frac{\nu_{\mathcal{M}}(\mathcal{J})}{\ell(\mathcal{J})} \right|
\]  

(39)

Note that for BARON solving \( P \) and \( \mathcal{P} \), \( \nu_{\mathcal{M}}(\mathcal{J}) \) and \( \omega_{\mathcal{M}}(\mathcal{J}) \) correspond to a 24hr. time limit. The table is to be read as follows: the first column is the name of the instance. The next three columns are for the \( p \)-formulation. Column 2 gives \% gap \( \omega_{\mathcal{M}}(\mathcal{J}) \) for \( P \) solved using BARON. Columns 3 and 4 are for the discretization approach on \( P \) with the smallest \% gap in column 3 and the corresponding discretization model in column 4. Columns 5 - 7 are for the \( pq \)-formulation. * indicates that unary and binary formulations for spec discretization produced the same value. † indicates that only a finite upper bound was returned by the solver without finding a feasible solution. A ~ 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.

From Table 4.3.3, we can observe that in the context of the \( p \)-formulation, the binary flow discretization model seems to be the best choice. For the standard pooling instances, the quality of solutions obtained by discretizing the \( pq \)-formulation is better than that due to \( p \)-formulation. Discretization of \( \mathcal{P} \) proved to be a more efficient approach than solving \( \mathcal{P} \) itself mostly for the large-scale standard instances stdB0 - stdC3. In the \( pq \)-formulation, there does not seem to be an obvious candidate for a good discretization model. Note that for \( n = 1 \), \( U(\mathcal{RPQ}) \) and
EPQ impose 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, the spec and ratio discretizations were mostly either provably infeasible or unable to find a solution within 1 hour. The \( p \)-formulation and its discretizations performed better than its \( pq \) 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 §4.3.2. \( B(\text{FP}) \) 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 \( P \) or \( PQ \) in 5 instances (marked with a \( \dagger \)). 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 \textit{meyer*} instances, solving \( P \) with BARON outperformed all discretization approaches.

<table>
<thead>
<tr>
<th>#</th>
<th>( P )</th>
<th>Best discretization of ( P )</th>
<th>( % ) gap</th>
<th>% gap Model</th>
<th>% gap</th>
<th>Best discretization of ( PQ )</th>
<th>% gap</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdA0</td>
<td>3.39</td>
<td>33.33 B(\text{FP}) cuts</td>
<td>3.16</td>
<td>3.29</td>
<td>B(\text{FP}) cuts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA1</td>
<td>42.21</td>
<td>41.10 B(\text{FP})</td>
<td>2.08</td>
<td>2.25</td>
<td>B(\text{FP})</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA2</td>
<td>70.61</td>
<td>32.38 B(\text{FP})</td>
<td>0.01 (744)</td>
<td>0.04</td>
<td>B(\text{FP})</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA3</td>
<td>15.88</td>
<td>15.88 B(\text{FP}) cuts</td>
<td>1.30</td>
<td>1.39</td>
<td>B(\text{FP})</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA4</td>
<td>7.79</td>
<td>49.20 B(\text{FP})</td>
<td>3.13</td>
<td>3.93</td>
<td>B(\text{RPQ}) n = 31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA5</td>
<td>22.38</td>
<td>61.55 B(\text{FP})</td>
<td>1.18</td>
<td>1.87</td>
<td>RPQ n = 31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA6</td>
<td>0.72</td>
<td>15.35 B(\text{FP})</td>
<td>0.67</td>
<td>0.71</td>
<td>RPQ n = 31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA7</td>
<td>10.55</td>
<td>40.82 B(\text{FP})</td>
<td>0.59</td>
<td>0.97</td>
<td>RPQ n = 31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA8</td>
<td>2.45</td>
<td>24.95 B(\text{FP})</td>
<td>0.19</td>
<td>0.36</td>
<td>B(\text{RPQ}) n = 7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdA9</td>
<td>32.08</td>
<td>34.41 B(\text{FP})</td>
<td>0.01 (10642)</td>
<td>0.08</td>
<td>B(\text{FP}) cuts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdB0</td>
<td>79.5</td>
<td>96.01 B(\text{FP})</td>
<td>5.16</td>
<td>5.96</td>
<td>B(\text{RPQ}) n = 15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdB1</td>
<td>35.44</td>
<td>71.53* B(\text{FP})</td>
<td>2.61</td>
<td>3.45</td>
<td>B(\text{RPQ}) n = 7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdB2</td>
<td>23.76</td>
<td>72.56* B(\text{FP})</td>
<td>14.37</td>
<td>4.26</td>
<td>U(\text{RPQ}) n = 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdB3</td>
<td>29.13</td>
<td>51.28* B(\text{FP})</td>
<td>0.19</td>
<td>0.58</td>
<td>EPQ n = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdB4</td>
<td>97.55</td>
<td>52.60* B(\text{FP})</td>
<td>0.01 (34860)</td>
<td>0.11</td>
<td>U(\text{RPQ}) n = 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdB5</td>
<td>19.16</td>
<td>50.25* B(\text{FP})</td>
<td>91.93</td>
<td>0.95</td>
<td>EPQ n = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdC0</td>
<td>97.01</td>
<td>91.04* B(\text{FP})</td>
<td>43.66</td>
<td>18.39</td>
<td>B(\text{RPQ}) n = 7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdC1</td>
<td>92.83</td>
<td>90.69* B(\text{FP})</td>
<td>35.08</td>
<td>18.90</td>
<td>EPQ n = 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdC2</td>
<td>96.09</td>
<td>84.62* B(\text{FP})</td>
<td>95.02</td>
<td>12.65</td>
<td>U(\text{RPQ}) n = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdC3</td>
<td>96.71</td>
<td>78.04* B(\text{FP})</td>
<td>95.93</td>
<td>6.25</td>
<td>U(\text{RPQ}) n = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| \textit{meyer4} | 0.01 (6562) | 2.38 B(\text{FP}) | Not applicable |
| \textit{meyer10} | 26.02       | 68.76 B(\text{FP}) | Not applicable |
| \textit{meyer15} | 36.52       | 51.68 B(\text{FP}) | Not applicable |

| Inst1 | 969.20\( ^{\dagger} \) | 9.68 B(\text{FP}) | 969.20\( ^{\dagger} \) | 9.68 B(\text{FP}) |
| Inst2 | 1242.26\( ^{\dagger} \) | 0.17 B(\text{FP}) | 1242.26\( ^{\dagger} \) | 3.14 B(\text{RPQ}) n = 7 |
| Inst3 | 0.01 (7585) | 0.01 (35) B(\text{FP}) | 9.33 | 0.01 (110) B(\text{FP}) |
| Inst4 | 0.01 (21904) | 0.01 (149) B(\text{FP}) | 0.76 | 0.01 (817) B(\text{FP}) |
| Inst5 | 462.58\( ^{\dagger} \) | 13.98 B(\text{FP}) | 462.58\( ^{\dagger} \) | – |
| Inst6 | 391.14\( ^{\dagger} \) | – | 391.14\( ^{\dagger} \) | – |
| Inst7 | 382.82\( ^{\dagger} \) | – | 382.82\( ^{\dagger} \) | – |

Table 2: Effect of discretization on the pooling problem. \% gap of best feasible solution and corresponding MILP model for each instance.

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 \( y = 0 \), 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, Figure 2(a) plots the geometric average of the \% gap \( \omega(\mathcal{I}) \) 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. Thus, we see that on the 20 standard instances of Alfaki and Haugland \( \text{[T]} \), EPQ and \( U(\text{RPQ}) \) with \( n = 1 \) provide good solutions very quickly. The solutions from \( B(\text{RPQ}) \), although as good or better in \% gap, are found much later in the branch-and-cut tree. On the other hand, for the \( p \)-discretizations \( B(\text{FP}) \) and \( B(\text{SP}) \), Cplex seems to be unable

\( ^{\dagger} \)The legend for \( B(\text{FP}) \), \( B(\text{FP}) \) cuts, \( B(\text{FPQ}) \), \( B(\text{FPQ}) \) cuts corresponds to blue points with the marker symbols for \( n = 1, 2, 4, 7 \), respectively.
to improve upon the solution found close to the root node.

In Figure 2(b), we analyze the change in % gap from first to best solution by Cplex and the amount of time elapsed in branch-and-cut tree for this change to occur. On the vertical axis, we plot geometric average of the ratio of $\omega_M(I)$ for best solution to first solution (e.g. a ratio of 50% implies % gap is halved from first to best); on the horizontal axis, we plot geometric average of the difference in times at which first and best solutions were found. The largest change in $\omega_M(I)$ in shortest time happens for $U(RPQ)$ $n = 1$. For $B(RPQ)$ with $n = 7$, Cplex reduces the % gap more but over a longer period of time as compared to $n = 15, 31$.

Figure 2: Geometric mean % gaps versus geometric mean time.

---

If first solution is also the best solution, then this difference is set to 0.5.
Table 4.3.3 does 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 an 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. As before, let \( \nu_M(I) \) be the best solution value for instance \( I \) upon termination of method \( M \). We calculate the relative metric \( \eta_M(I) := \frac{\nu_M(I) - \nu_{\min}(I)}{\nu_{\max}(I) - \nu_{\min}(I)} \), where \( \nu_{\max}(I) = \max_M \nu_M(I) \) and \( \nu_{\min}(I) = \min_M \nu_M(I) \) are the worst and best known solutions for \( I \). Any point \((\kappa, \gamma)\) on the performance profile for method \( M \) indicates that \( \eta_M(I) \) is at most \( \kappa \) for a fraction \( \gamma \) of the 20 standard instances. Note that \( \eta_M(I) = 0 \) means that the best solution for \( I \) was obtained by \( M \). We see that \( B(RPQ) \) provides the most dominant profiles with \( n = 7, 31 \), followed by \( EPQ\ n = 2 \) and \( U(RPQ) \ n = 4 \). Although \( BARON \) produces best solutions on most number of instances by solving \( PQ \) over 24hr, its performance quickly deteriorates because it gives poor solutions on the large-scale standard instances.

![Figure 3: Performance profiles for the most significant discretization models with respect to the best feasible solutions obtained on standard instances.](image)

In Appendix A, we provide details on \% termination gaps, solution times, and number of branch-and-bound nodes for each discretization model solved in \textit{Cplex}.

5 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 of the pooling problem. In the second half, we discussed different discretization methods to approximate a bilinear program as a MILP. We computationally tested these ideas on random instances of the pooling problem. 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. Our experiments suggest that discretization seems to be a promising approach especially for large-scale and generalized pooling problems. An alternate discretization strategy, that does not depend on explicitly discretizing the variables in the pooling problem, was analytically and empirically studied by Dey and Gupte [14].

\[^3\text{Legend is same as Figure 2(a). Profiles for PQ, FPQ, RPQ, which are solved with BARON, are hashed lines.}\]
References


A Detailed computational results

In Tables 3 - 9 we present detailed outputs from our computational experiments. If a model is proved infeasible within 1 hour, then it is marked by $+\infty$, otherwise if neither feasibility nor infeasibility is proven, then it is marked by $-\infty$.

First, we present the lower ($\ell(I)$) and upper bound ($\nu_{\mu}(I)$) values from BARON for global optimization of $P$ and $PQ$. Upper bound values ($\ell(I)$) from solving discretized $PQ$ with BARON are also reported. For BARON, discretizations of $P$ are not reported since they were observed to perform worse than discretizations of $PQ$. Then, we present various results for solving discretizations as MILPs using Cplex. In Table 4, we present the best feasible solutions returned by Cplex in 1 hour when solving discretizations of $PQ$. In this table, the numbers in parenthesis correspond to the CPU time (in seconds) at which Cplex found this feasible solution during its branch-and-cut procedure. A similar output is provided for discretizations of $P$ in Table 5.

Next, we present % termination gaps for Cplex or solution times if Cplex terminates in 1 hour. This % gap is the value reported by Cplex upon termination and hence is different from the % gap $\omega_{\mu}(I)$ calculated in [59]. A $-\infty$ indicates the gap computation is not possible due to either lack of a lower bound (happens if MILP is too large for Cplex to finish processing root node) or lack of a upper bound (happens if model is not proven feasible). Finally, in Tables 8 and 9 we present the number of branch-and-bound nodes in Cplex and the % root LP gaps for flow discretizations. The root LP gap is given by $100 \times |1 - UB/LB|$ where $LB$ is the root LP relaxation value and $UB$ is the minimum value of $\nu_{\mu}(I)$ among the four flow discretizations - $B(FP)$ with and without user cuts, $B(FPQ)$ with and without user cuts.


<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>(v_{\mathcal{R}}(\mathcal{I}))</th>
<th>(v_{\mathcal{R}}(\mathcal{J}))</th>
<th>B(FPQ)</th>
<th>B(RPQ 1)</th>
<th>B(RPQ 2)</th>
<th>B(RPQ 4)</th>
<th>B(RPQ 7)</th>
<th>B(RPQ 15)</th>
<th>B(RPQ 31)</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdA0</td>
<td>-44549.64</td>
<td>-35727.11</td>
<td>-36979.79</td>
<td>-35812.33</td>
<td>-35732.18</td>
<td>-30432.46</td>
<td>-32891.83</td>
<td>-33578.04</td>
<td>-33978.51</td>
<td>-33530.09</td>
</tr>
<tr>
<td>stdA1</td>
<td>-42500.40</td>
<td>-17277.53</td>
<td>-39967.20</td>
<td>-36540.63</td>
<td>-38102.82</td>
<td>-36366.43</td>
<td>-34040.09</td>
<td>-36716.62</td>
<td>-37475.26</td>
<td></td>
</tr>
<tr>
<td>stdA2</td>
<td>-46162</td>
<td>-33224.20</td>
<td>-39446.54</td>
<td>-38573.73</td>
<td>-36991.7</td>
<td>-37583.52</td>
<td>-38720.03</td>
<td>-36371.62</td>
<td>-37426.92</td>
<td></td>
</tr>
<tr>
<td>stdA3</td>
<td>-49094.53</td>
<td>-39274.51</td>
<td>-42592.04</td>
<td>-41257.14</td>
<td>-40015.4</td>
<td>-36589.52</td>
<td>-37162.36</td>
<td>-37885.9</td>
<td>-38393.72</td>
<td></td>
</tr>
<tr>
<td>stdA4</td>
<td>-34610</td>
<td>-21930.92</td>
<td>-28255.63</td>
<td>-27982.07</td>
<td>-29918.3</td>
<td>-25819.8</td>
<td>-23289.17</td>
<td>-26161.52</td>
<td>-23696.56</td>
<td>-28784.94</td>
</tr>
<tr>
<td>stdA5</td>
<td>-43132</td>
<td>-42158.36</td>
<td>-42463.05</td>
<td>-42176.94</td>
<td>-40015.4</td>
<td>-36589.52</td>
<td>-37162.36</td>
<td>-22745.71</td>
<td>-22862.92</td>
<td></td>
</tr>
<tr>
<td>stdA6</td>
<td>-55083</td>
<td>-39968.73</td>
<td>-44682.25</td>
<td>-43567.79</td>
<td>-43495.77</td>
<td>-43046.86</td>
<td>-45380.11</td>
<td>-44251.04</td>
<td>-27726.04</td>
<td></td>
</tr>
<tr>
<td>stdA7</td>
<td>-42570</td>
<td>-29915.29</td>
<td>-30666.87</td>
<td>-30675.55</td>
<td>-30133.6</td>
<td>-30244.3</td>
<td>-30070.25</td>
<td>-30057.37</td>
<td>-21682.96</td>
<td></td>
</tr>
<tr>
<td>stdA8</td>
<td>-32438.61</td>
<td>-14896.96</td>
<td>-21932.63</td>
<td>-21359.11</td>
<td>-21825.28</td>
<td>-2271.45</td>
<td>-2271.45</td>
<td>-2271.45</td>
<td>-21876.23</td>
<td></td>
</tr>
<tr>
<td>stdB0</td>
<td>-61448.69</td>
<td>-9285.75</td>
<td>-45295.90</td>
<td>-42592.04</td>
<td>-42463.05</td>
<td>-42176.94</td>
<td>-40015.4</td>
<td>-36589.52</td>
<td>-37162.36</td>
<td></td>
</tr>
<tr>
<td>stdB1</td>
<td>-74131</td>
<td>-42026.36</td>
<td>-56161.21</td>
<td>-45367.79</td>
<td>-45367.79</td>
<td>-45367.79</td>
<td>-45367.79</td>
<td>-45367.79</td>
<td>-45367.79</td>
<td></td>
</tr>
<tr>
<td>stdB2</td>
<td>-62009</td>
<td>-42817.10</td>
<td>-63090.01</td>
<td>-62704.17</td>
<td>-62704.17</td>
<td>-62704.17</td>
<td>-62704.17</td>
<td>-62704.17</td>
<td>-62704.17</td>
<td></td>
</tr>
<tr>
<td>stdB3</td>
<td>-79116</td>
<td>-52481.55</td>
<td>-74050.47</td>
<td>-73911.21</td>
<td>-73911.21</td>
<td>-73911.21</td>
<td>-73911.21</td>
<td>-73911.21</td>
<td>-73911.21</td>
<td></td>
</tr>
<tr>
<td>stdB4</td>
<td>-119219</td>
<td>-2895.86</td>
<td>-96921.11</td>
<td>-54609.55</td>
<td>-66025.78</td>
<td>-3334.79</td>
<td>-3334.79</td>
<td>-3334.79</td>
<td>-3334.79</td>
<td></td>
</tr>
<tr>
<td>stdB5</td>
<td>-142041</td>
<td>-8466.31</td>
<td>-11817.99</td>
<td>-76800.45</td>
<td>-8177.79</td>
<td>-2459.31</td>
<td>-2459.31</td>
<td>-2459.31</td>
<td>-2459.31</td>
<td></td>
</tr>
<tr>
<td>stdB7</td>
<td>-139356</td>
<td>-4286.01</td>
<td>-130315.02</td>
<td>-5308.55</td>
<td>-1038.35</td>
<td>-3189.11</td>
<td>-3189.11</td>
<td>-3189.11</td>
<td>-3189.11</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Output from solving original formulations and discretizations with BARON. † denotes that only a finite upper bound was returned by BARON without finding a feasible solution.
<table>
<thead>
<tr>
<th>#</th>
<th>B(FFQ)</th>
<th>B(FFQ) cuts</th>
<th>U(RFFQ) 1</th>
<th>U(RFFQ) 2</th>
<th>U(RFFQ) 4</th>
<th>ERF 1</th>
<th>ERF 2</th>
<th>ERF 4</th>
<th>ERF 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdA0</td>
<td>-34975.63 (1048)</td>
<td>-35761.99 (3592)</td>
<td>-31990.52 (3)</td>
<td>-34175.71 (37)</td>
<td>-31990.52 (3)</td>
<td>-34175.71 (37)</td>
<td>-31990.52 (3)</td>
<td>-34175.71 (37)</td>
<td>-31990.52 (3)</td>
</tr>
<tr>
<td>stdA1</td>
<td>-29223.13 (3584)</td>
<td>-32392.31 (365)</td>
<td>-24900.16 (3)</td>
<td>-25179.84 (28)</td>
<td>-28402.34 (167)</td>
<td>-24900.16 (3)</td>
<td>-25179.84 (28)</td>
<td>-28402.34 (167)</td>
<td>-24900.16 (3)</td>
</tr>
<tr>
<td>stdA2</td>
<td>-23037.02 (343)</td>
<td>-23037.02 (343)</td>
<td>-19840.16 (3)</td>
<td>-20666.61 (159)</td>
<td>-21915.16 (1757)</td>
<td>-19840.16 (3)</td>
<td>-20666.61 (159)</td>
<td>-21915.16 (1757)</td>
<td>-19840.16 (3)</td>
</tr>
<tr>
<td>stdA3</td>
<td>-9412.73 (3574)</td>
<td>-9395.23 (358)</td>
<td>-36237.75 (9)</td>
<td>-37166.19 (98)</td>
<td>-36248.94 (88)</td>
<td>-36237.75 (9)</td>
<td>-37166.19 (98)</td>
<td>-36248.94 (88)</td>
<td>-36237.75 (9)</td>
</tr>
<tr>
<td>stdA4</td>
<td>-7012.4 (3582)</td>
<td>-8024.46 (1999)</td>
<td>-3812.61 (30)</td>
<td>-39288.42 (1220)</td>
<td>-39517.3 (3275)</td>
<td>-3812.61 (30)</td>
<td>-39288.42 (1220)</td>
<td>-39517.3 (3275)</td>
<td>-3812.61 (30)</td>
</tr>
<tr>
<td>stdA5</td>
<td>-5642.79 (3257)</td>
<td>-2698.32 (357)</td>
<td>-25447.07 (53)</td>
<td>-25770.49 (206)</td>
<td>-25826.71 (7)</td>
<td>-25447.07 (53)</td>
<td>-25770.49 (206)</td>
<td>-25826.71 (7)</td>
<td>-25447.07 (53)</td>
</tr>
<tr>
<td>stdA6</td>
<td>-80129.31 (359)</td>
<td>-11296 (357)</td>
<td>-41777 (224)</td>
<td>-42022.93 (331)</td>
<td>-42021.16 (1214)</td>
<td>-41777 (224)</td>
<td>-42022.93 (331)</td>
<td>-42021.16 (1214)</td>
<td>-41777 (224)</td>
</tr>
<tr>
<td>stdA8</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
<td>-30436.05 (356)</td>
</tr>
<tr>
<td>stdA9</td>
<td>-20147.46 (3540)</td>
<td>-21917.46 (3540)</td>
<td>-21917.46 (3540)</td>
<td>-21917.46 (3540)</td>
<td>-21917.46 (3540)</td>
<td>-21917.46 (3540)</td>
<td>-21917.46 (3540)</td>
<td>-21917.46 (3540)</td>
<td>-21917.46 (3540)</td>
</tr>
<tr>
<td>stdB0</td>
<td>-34078.05 (3562)</td>
<td>-34154.81 (3557)</td>
<td>-40171.43 (271)</td>
<td>-41092.33 (3254)</td>
<td>-41827.74 (3557)</td>
<td>-40171.43 (271)</td>
<td>-41092.33 (3254)</td>
<td>-41827.74 (3557)</td>
<td>-40171.43 (271)</td>
</tr>
<tr>
<td>stdB3</td>
<td>-53474.1 (3579)</td>
<td>-57117.93 (420)</td>
<td>-59181.04 (3544)</td>
<td>-59910.48 (3544)</td>
<td>-59910.48 (3544)</td>
<td>-59181.04 (3544)</td>
<td>-59910.48 (3544)</td>
<td>-59910.48 (3544)</td>
<td>-59181.04 (3544)</td>
</tr>
<tr>
<td>stdB5</td>
<td>-30196.22 (3579)</td>
<td>-30196.22 (3579)</td>
<td>-580.01 (121)</td>
<td>-580.01 (121)</td>
<td>-580.01 (121)</td>
<td>-580.01 (121)</td>
<td>-580.01 (121)</td>
<td>-580.01 (121)</td>
<td>-580.01 (121)</td>
</tr>
<tr>
<td>stdC0</td>
<td>-13009.08 (3534)</td>
<td>-52450.57 (3519)</td>
<td>-7350.04 (3516)</td>
<td>-7350.04 (3516)</td>
<td>-7350.04 (3516)</td>
<td>-7350.04 (3516)</td>
<td>-7350.04 (3516)</td>
<td>-7350.04 (3516)</td>
<td>-7350.04 (3516)</td>
</tr>
<tr>
<td>stdC1</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
<td>-53090.8 (3543)</td>
</tr>
<tr>
<td>stdC2</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
</tr>
<tr>
<td>stdC3</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
<td>-2880.99 (1)</td>
</tr>
</tbody>
</table>

Table 4: \( \nu_{ij} \) and corresponding times from discretizations of PQ. Inst5-7 are omitted since none of the discretizations found a feasible solution nor did they terminate with provable infeasibility.
Table 5: \( \nu_n(\mathcal{I}) \) and corresponding times from discretizations of \( P \). For \( n > 4 \), no feasible solution was found nor was infeasibility proven for meyer10,15 and Inst1-2,5-7.
<table>
<thead>
<tr>
<th>#</th>
<th>B(FPQ)</th>
<th>B(RPQ)</th>
<th>cuts</th>
<th>U(RPQ) 1</th>
<th>U(RPQ) 2</th>
<th>U(RPQ) 4</th>
<th>EPQ 1</th>
<th>EPQ 2</th>
<th>EPQ 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdA0</td>
<td>4.85%</td>
<td>2.9%</td>
<td>11</td>
<td>2485</td>
<td>5.16%</td>
<td>6</td>
<td>125</td>
<td>5.59%</td>
<td></td>
</tr>
<tr>
<td>stdA1</td>
<td>3.57%</td>
<td>3.35%</td>
<td>3</td>
<td>97</td>
<td>495</td>
<td>1</td>
<td>434</td>
<td>1.1%</td>
<td></td>
</tr>
<tr>
<td>stdA2</td>
<td>1.76%</td>
<td>1.69%</td>
<td>4</td>
<td>248</td>
<td>2597</td>
<td>1</td>
<td>230</td>
<td>7.06%</td>
<td></td>
</tr>
<tr>
<td>stdA3</td>
<td>4.15%</td>
<td>4.85%</td>
<td>12</td>
<td>510</td>
<td>2437</td>
<td>4</td>
<td>1340</td>
<td>10.17%</td>
<td></td>
</tr>
<tr>
<td>stdA4</td>
<td>17.24%</td>
<td>5.7%</td>
<td>49</td>
<td>0.95%</td>
<td>3.84%</td>
<td>15</td>
<td>0.77%</td>
<td>9.47%</td>
<td></td>
</tr>
<tr>
<td>stdA5</td>
<td>6.92%</td>
<td>4.73%</td>
<td>112</td>
<td>2.76%</td>
<td>7.94%</td>
<td>38</td>
<td>3.1%</td>
<td>6.79%</td>
<td></td>
</tr>
<tr>
<td>stdA6</td>
<td>3.87%</td>
<td>2.82%</td>
<td>297</td>
<td>0.69%</td>
<td>1.05%</td>
<td>60</td>
<td>0.87%</td>
<td>1.22%</td>
<td></td>
</tr>
<tr>
<td>stdA7</td>
<td>1.5%</td>
<td>2.63%</td>
<td>474</td>
<td>2.65%</td>
<td>2.17%</td>
<td>118</td>
<td>3.08%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdA8</td>
<td>0.76%</td>
<td>0.62%</td>
<td>594</td>
<td>0.76%</td>
<td>0.81%</td>
<td>261</td>
<td>0.71%</td>
<td>56.91%</td>
<td></td>
</tr>
<tr>
<td>stdA9</td>
<td>0.53%</td>
<td>0.08%</td>
<td>49</td>
<td>0.21%</td>
<td>0.76%</td>
<td>44</td>
<td>0.21%</td>
<td>273.14%</td>
<td></td>
</tr>
<tr>
<td>stdB0</td>
<td>33.42%</td>
<td>33.11%</td>
<td>1830</td>
<td>6.56%</td>
<td>8.17%</td>
<td>1320</td>
<td>6.44%</td>
<td>16.95%</td>
<td></td>
</tr>
<tr>
<td>stdB1</td>
<td>10.38%</td>
<td>18.9%</td>
<td>420</td>
<td>2.42%</td>
<td>4.53%</td>
<td>179</td>
<td>2.94%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdB2</td>
<td>13.02%</td>
<td>15.82%</td>
<td>2.65%</td>
<td>3.88%</td>
<td>9.31%</td>
<td>2.44%</td>
<td>4.82%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdB3</td>
<td>105.25%</td>
<td>36.12%</td>
<td>0.72%</td>
<td>0.97%</td>
<td>3.27%</td>
<td>0.59%</td>
<td>0.83%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdB4</td>
<td>3.7%</td>
<td>4.12%</td>
<td>0.14%</td>
<td>0.11%</td>
<td>0.49%</td>
<td>0.12%</td>
<td>0.47%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdB5</td>
<td>182.15%</td>
<td>15.41%</td>
<td>25.22%</td>
<td>30.39%</td>
<td>14.09%</td>
<td>26.03%</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>stdC0</td>
<td>182.15%</td>
<td>87.84%</td>
<td>15.41%</td>
<td>25.22%</td>
<td>30.39%</td>
<td>14.09%</td>
<td>26.03%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdC1</td>
<td>989.97%</td>
<td>988.74%</td>
<td>21.45%</td>
<td>38.92%</td>
<td>45.21%</td>
<td>34.94%</td>
<td>26.03%</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdC2</td>
<td>552.42%</td>
<td>12.91%</td>
<td>20.31%</td>
<td>29.85%</td>
<td>14.64%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stdC3</td>
<td>355.33%</td>
<td>6.59%</td>
<td>12.1%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: % termination gaps or solution times in seconds (if MILP terminates) for discretizations of PQ. † denotes that the corresponding discretization was proved infeasible after indicated time.
<table>
<thead>
<tr>
<th>#</th>
<th>B(FF)</th>
<th>B(FF) cuts</th>
<th>U(SP) 1</th>
<th>U(SP) 2</th>
<th>U(SP) 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdA0</td>
<td>377.1%</td>
<td>85.16%</td>
<td>92.86%</td>
<td>353.27%</td>
<td>388.76%</td>
</tr>
<tr>
<td>stdA1</td>
<td>150.6%</td>
<td>225.96%</td>
<td>59.84%</td>
<td>322.66%</td>
<td>433.66%</td>
</tr>
<tr>
<td>stdA2</td>
<td>141%</td>
<td>214.94%</td>
<td>213E7%</td>
<td>1.94E7%</td>
<td>4.76E7%</td>
</tr>
<tr>
<td>stdA3</td>
<td>151.82%</td>
<td>97.96%</td>
<td>39.85%</td>
<td>190.15%</td>
<td>193.82%</td>
</tr>
<tr>
<td>stdA4</td>
<td>126.97%</td>
<td>414.04%</td>
<td>110.77%</td>
<td>414.04%</td>
<td>414.04%</td>
</tr>
<tr>
<td>stdA5</td>
<td>218.54%</td>
<td>507.41%</td>
<td>301.04%</td>
<td>507.41%</td>
<td>507.41%</td>
</tr>
<tr>
<td>stdA6</td>
<td>142.4%</td>
<td>19.99%</td>
<td>64.01%</td>
<td>142.4%</td>
<td>142.4%</td>
</tr>
<tr>
<td>stdA7</td>
<td>196.58%</td>
<td>108.31%</td>
<td>143.84%</td>
<td>196.58%</td>
<td>196.58%</td>
</tr>
<tr>
<td>stdA8</td>
<td>121.85%</td>
<td>121.85%</td>
<td>48.21%</td>
<td>121.85%</td>
<td>121.85%</td>
</tr>
<tr>
<td>stdA9</td>
<td>125.5%</td>
<td>137.36%</td>
<td>295.23%</td>
<td>471.99%</td>
<td>471.99%</td>
</tr>
<tr>
<td>stdB0</td>
<td>3298.71%</td>
<td>3298.71%</td>
<td>3113.76%</td>
<td>3303.01%</td>
<td>3303.01%</td>
</tr>
<tr>
<td>stdB1</td>
<td>299.93%</td>
<td>299.93%</td>
<td>297.49%</td>
<td>299.93%</td>
<td>299.93%</td>
</tr>
<tr>
<td>stdB2</td>
<td>302.41%</td>
<td>302.41%</td>
<td>296.34%</td>
<td>302.41%</td>
<td>302.41%</td>
</tr>
<tr>
<td>stdB3</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
</tr>
<tr>
<td>stdB4</td>
<td>152.94%</td>
<td>152.94%</td>
<td>152.94%</td>
<td>152.94%</td>
<td>152.94%</td>
</tr>
<tr>
<td>stdB5</td>
<td>142.03%</td>
<td>142.03%</td>
<td>142.03%</td>
<td>142.03%</td>
<td>142.03%</td>
</tr>
<tr>
<td>stdB6</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
</tr>
<tr>
<td>stdB7</td>
<td>153.22%</td>
<td>153.22%</td>
<td>153.22%</td>
<td>153.22%</td>
<td>153.22%</td>
</tr>
<tr>
<td>stdB8</td>
<td>142.03%</td>
<td>142.03%</td>
<td>142.03%</td>
<td>142.03%</td>
<td>142.03%</td>
</tr>
<tr>
<td>stdB9</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
<td>119.29%</td>
</tr>
<tr>
<td>stdC0</td>
<td>1272.54%</td>
<td>1272.54%</td>
<td>1257.96%</td>
<td>1272.54%</td>
<td>1272.54%</td>
</tr>
<tr>
<td>stdC1</td>
<td>1191.39%</td>
<td>1191.39%</td>
<td>1191.39%</td>
<td>1191.39%</td>
<td>1191.39%</td>
</tr>
<tr>
<td>stdC2</td>
<td>613.37%</td>
<td>613.37%</td>
<td>613.37%</td>
<td>613.37%</td>
<td>613.37%</td>
</tr>
<tr>
<td>stdC3</td>
<td>386.91%</td>
<td>386.91%</td>
<td>386.91%</td>
<td>386.91%</td>
<td>386.91%</td>
</tr>
</tbody>
</table>

Table 7: % termination gaps or solution times in seconds (if MILP terminates) for discretizations of P. † denotes that the corresponding discretization was proved infeasible after indicated time.
### Table 8: Number of branch-and-bound nodes for discretizations of \( \text{PQ} \) and \( \text{P} \).

<table>
<thead>
<tr>
<th>Inst1</th>
<th>34724</th>
<th>7421</th>
<th>9097</th>
<th>10826</th>
<th>15553</th>
<th>9520</th>
<th>10532</th>
<th>13719</th>
<th>5821</th>
<th>2353</th>
<th>584</th>
<th>19371</th>
<th>9539</th>
<th>9639</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inst2</td>
<td>5878</td>
<td>5723</td>
<td>16011</td>
<td>27002</td>
<td>11703</td>
<td>16120</td>
<td>26711</td>
<td>9361</td>
<td>4290</td>
<td>561</td>
<td>547</td>
<td>12128</td>
<td>8497</td>
<td>8194</td>
</tr>
<tr>
<td>Inst3</td>
<td>8703</td>
<td>14137</td>
<td>1134</td>
<td>133</td>
<td>2260</td>
<td>1148</td>
<td>20</td>
<td>474</td>
<td>48458</td>
<td>9272</td>
<td>2819</td>
<td>6891</td>
<td>17172</td>
<td>31014</td>
</tr>
<tr>
<td>Inst4</td>
<td>15138</td>
<td>6106</td>
<td>42266</td>
<td>24571</td>
<td>19215</td>
<td>42516</td>
<td>24019</td>
<td>17305</td>
<td>4393</td>
<td>1896</td>
<td>620</td>
<td>18501</td>
<td>8099</td>
<td>8620</td>
</tr>
<tr>
<td>Inst5</td>
<td>0</td>
<td>0</td>
<td>505</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inst9</td>
<td>0</td>
<td>0</td>
<td>505</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inst10</td>
<td>0</td>
<td>0</td>
<td>505</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The table contains data for various instances, each identifying the number of branch-and-bound nodes for different discretizations of \( \text{PQ} \) and \( \text{P} \). The columns represent the instance numbers, followed by the number of nodes for each discretization. The rows include columns for different standards (A, B, C, etc.) and instances (1, 2, 3, etc.).
Table 9: % gaps of root LP relaxations for $B(\text{FPQ})$ and $B(\text{FP})$.

<table>
<thead>
<tr>
<th>#</th>
<th>$B(\text{FPQ})$</th>
<th>$B(\text{FPQ})$ cuts</th>
<th>$B(\text{FP})$</th>
<th>$B(\text{FP})$ cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>stdA0</td>
<td>5.23%</td>
<td>4.85%</td>
<td>21.94%</td>
<td>21.94%</td>
</tr>
<tr>
<td>stdA1</td>
<td>7.28%</td>
<td>7.26%</td>
<td>34.86%</td>
<td>34.86%</td>
</tr>
<tr>
<td>stdA2</td>
<td>3.58%</td>
<td>3.57%</td>
<td>38.95%</td>
<td>38.94%</td>
</tr>
<tr>
<td>stdA3</td>
<td>6.64%</td>
<td>6.37%</td>
<td>15.72%</td>
<td>15.71%</td>
</tr>
<tr>
<td>stdA4</td>
<td>6.09%</td>
<td>6.05%</td>
<td>16.87%</td>
<td>16.87%</td>
</tr>
<tr>
<td>stdA5</td>
<td>4.52%</td>
<td>4.52%</td>
<td>22.04%</td>
<td>22.04%</td>
</tr>
<tr>
<td>stdA6</td>
<td>2.74%</td>
<td>2.74%</td>
<td>4.25%</td>
<td>4.25%</td>
</tr>
<tr>
<td>stdA7</td>
<td>1.48%</td>
<td>1.48%</td>
<td>20.08%</td>
<td>20.08%</td>
</tr>
<tr>
<td>stdA8</td>
<td>0.62%</td>
<td>0.62%</td>
<td>28.4%</td>
<td>28.4%</td>
</tr>
<tr>
<td>stdA9</td>
<td>0.08%</td>
<td>0.08%</td>
<td>32.44%</td>
<td>32.44%</td>
</tr>
<tr>
<td>stdB0</td>
<td>24.88%</td>
<td>24.88%</td>
<td>44.46%</td>
<td>44.46%</td>
</tr>
<tr>
<td>stdB1</td>
<td>9.4%</td>
<td>9.41%</td>
<td>19.93%</td>
<td>19.93%</td>
</tr>
<tr>
<td>stdB2</td>
<td>11.52%</td>
<td>11.51%</td>
<td>19.34%</td>
<td>19.34%</td>
</tr>
<tr>
<td>stdB3</td>
<td>26.54%</td>
<td>26.54%</td>
<td>31.24%</td>
<td>31.24%</td>
</tr>
<tr>
<td>stdB4</td>
<td>3.57%</td>
<td>3.57%</td>
<td>19.57%</td>
<td>19.57%</td>
</tr>
<tr>
<td>stdB5</td>
<td>50.25%</td>
<td>50.25%</td>
<td>58.68%</td>
<td>58.68%</td>
</tr>
<tr>
<td>stdC0</td>
<td>47.35%</td>
<td>47.03%</td>
<td>56%</td>
<td>56%</td>
</tr>
<tr>
<td>stdC1</td>
<td>90.83%</td>
<td>90.82%</td>
<td>92.26%</td>
<td>92.26%</td>
</tr>
<tr>
<td>stdC2</td>
<td>84.67%</td>
<td>84.67%</td>
<td>85.98%</td>
<td>85.98%</td>
</tr>
<tr>
<td>stdC3</td>
<td>78.04%</td>
<td>78.04%</td>
<td>79.46%</td>
<td>79.46%</td>
</tr>
<tr>
<td>meyer4</td>
<td>75.9%</td>
<td>75.05%</td>
<td>143.95%</td>
<td>143.81%</td>
</tr>
<tr>
<td>meyer10</td>
<td>75.47%</td>
<td>73.99%</td>
<td>143.95%</td>
<td>143.81%</td>
</tr>
<tr>
<td>Inst1</td>
<td>9.02%</td>
<td>8.99%</td>
<td>99.64%</td>
<td>99.64%</td>
</tr>
<tr>
<td>Inst2</td>
<td>11.74%</td>
<td>11.75%</td>
<td>99.59%</td>
<td>99.59%</td>
</tr>
<tr>
<td>Inst3</td>
<td>0.52%</td>
<td>0.52%</td>
<td>99.55%</td>
<td>99.55%</td>
</tr>
<tr>
<td>Inst4</td>
<td>1.92%</td>
<td>2.43%</td>
<td>9.26%</td>
<td>9.43%</td>
</tr>
<tr>
<td>Inst5</td>
<td>81.95%</td>
<td>81.95%</td>
<td>6.48%</td>
<td>6.57%</td>
</tr>
<tr>
<td>Inst9</td>
<td>80.2%</td>
<td>80.41%</td>
<td>0.53%</td>
<td>0.57%</td>
</tr>
<tr>
<td>Inst10</td>
<td>82.84%</td>
<td>82.82%</td>
<td>2.11%</td>
<td>2.17%</td>
</tr>
</tbody>
</table>