New multi-commodity flow formulations for the pooling problem

Natashia Boland
Georgia Institute of Technology, Atlanta, U.S.A.

Thomas Kalinowski Fabian Rigterink
The University of Newcastle, Australia

June 11, 2015

Abstract

The pooling problem is a nonconvex nonlinear programming problem with numerous applications. The nonlinearities of the problem arise from bilinear constraints that capture the blending of raw materials. Bilinear constraints are well-studied and significant progress has been made in solving large instances of the pooling problem to global optimality. This is due in no small part to reformulations of the problem. Recently, Alfaki and Haugland proposed a multi-commodity flow formulation of the pooling problem based on input commodities. The authors proved that the new formulation has a stronger linear relaxation than previously known formulations. They also provided computational results which show that the new formulation outperforms previously known formulations when used in a global optimization solver. In this paper, we generalize their ideas and propose new multi-commodity flow formulations based on output, input and output and (input, output)-commodities. We prove the equivalence of formulations, and we study the partial order of formulations with respect to the strength of their LP relaxations. In an extensive computational study, we evaluate the performance of the new formulations. We study the trade-off between disaggregating commodities and therefore increasing the size of formulations versus strengthening the relaxed linear programs and improving the computational performance of the nonlinear programs. We provide computational results which show that output commodities often outperform input commodities, and that disaggregating commodities further only marginally strengthens the linear relaxations. In fact, smaller formulations often show a significantly better performance when used in a global optimization solver.

Keywords Pooling problem · Bilinear programming · Nonlinear programming · Linear relaxation · Global optimization · Blending

1 Introduction

The pooling problem has received significant research attention since its proposal by Haverly in 1978 [30]. With the advancement of optimization software, research focus shifted from local to global optimization methods, and from finding better local optima to finding global optima faster. Success in finding global optima fast largely depends on the problem formulation. This led to the proposal of numerous alternative formulations for the pooling problem, including the P-, Q-, PQ- and HYB-formulations, and most recently a multi-commodity flow formulation based on input commodities. All of these formulations are *equivalent* in the sense that there exists a one-to-one correspondence between the feasible sets that preserves the

objective function value. Despite their equivalence, the alternative formulations often differ significantly in strength of the LP relaxations and in solve times of the nonlinear programs. An important characteristic of a formulation is the degree of variable disaggregation, which comes at a cost: the higher the degree of variable disaggregation, the larger the problem (e.g. in terms of the number of bilinear terms, variables and constraints). Formulations with a low degree of variable disaggregation (e.g. the Q-formulation) often perform poorly, but so do formulations with a high degree of variable disaggregation (e.g. a multi-commodity flow formulation based on path commodities). This an interesting trade-off to study and one of the main contributions of this paper, which can be summarised as follows: (1) we introduce new multi-commodity flow formulations for the pooling problem, (2) we prove the equivalence and present an almost complete description of the partial order of formulations with respect to the strength of the LP relaxations, and (3) we evaluate the performance of the new formulations in an extensive computational study to find a good degree of variable disaggregation.

2 Literature review

The pooling problem is a strongly NP-hard [9] nonconvex nonlinear programming problem (NLP) with various applications. Informally, the problem can be stated as follows: Given a set of raw material suppliers (inputs) and qualities of the supplies, find a cost-minimizing way of blending these raw materials in intermediate pools and outputs so as to satisfy requirements on the final output qualities. The blending in pools and outputs introduces bilinear constraints. The problem can be described as a minimum cost network flow problem with additional bilinear constraints to capture the blending of raw materials. The most common applications of pooling problems are in the refining and petrochemical industries [49]. However, blending is also an important feature of numerous other manufacturing processes, including agriculture, mining, food manufacturing and pulp and paper production [49]. The pooling problem was first proposed by Haverly in 1978 [30]. Since then, an extensive literature has been published. Reviews on the pooling problem and its variations can be found in [13, 19, 28, 39]. We also refer to three PhD theses on this subject [7, 27, 38]. There are a number of solution techniques to solve pooling problems, which can broadly be classified as local and global optimization methods.

Local optimization methods

Noteworthy among the local optimization methods are *Successive Linear Programming* (SLP) algorithms and *Global Optimal Search* (GOS). SLP algorithms linearly approximate pooling problems using a first-order Taylor expansion, then solve the linear program to find a new feasible point, linearize the problem at the new point, and iterate [15, 41, 57]. SLP algorithms are often used to locally improve solutions [33]. GOS, on the other hand, decomposes the variable set into two sets of complicating and non-complicating variables, then alternates between solving a projection of the primal problem (the upper bounding problem) and a series of relaxed dual problems (the lower bounding problems) [21].

Global optimization methods

Global optimization methods can broadly be classified as decomposition-based algorithms and branch and bound algorithms. Noteworthy among the decomposition-based algorithms are the *Global Optimization Algorithm* (GOP) [22] and other Lagrangian-based approaches. Similar to GOS, GOP decomposes the problem into primal and relaxed dual subproblems by introducing new variables and partitioning the resulting variable set. The decomposition provides lower and upper bounds on the global optimum. GOP has been proven to attain finite ϵ-convergence and ϵ-global optimality [22, 50]. The development of GOP resulted in a series of publications by Floudas and Visweswaran [23, 51, 52, 53] and the implementation

of the cGOP solver [54]. Similar to GOS and GOP, other Lagrangian-based approaches generate a Lagrangian dual to the problem and then use a branch and bound algorithm to generate a converging sequence of upper bounds (solutions to the primal) and lower bounds (solutions to the dual) [2, 11, 16].

Finally, branch and bound algorithms relax the original nonconvex nonlinear problem to obtain a convex and/or linear relaxation which provides lower and upper bounds on the optimal objective function value. The quality of these bounds depends on the type of relaxation and the size of the domain. The branch and bound algorithm then divides the domain into smaller subdomains. Once these subdomains are sufficiently small, the lower and upper bounds converge to the optimal objective function value.

Linear relaxations. In [35], McCormick introduced a linear relaxation of the bilinear term $z = xy$ on $x \in [x^L, x^U]$ and $y \in [y^L, y^U]$ using the following four inequalities:

$$z \geq xy^L + x^L y - x^L y^L, \qquad z \geq xy^U + x^U y - x^U y^U,$$
$$z \leq xy^L + x^U y - x^U y^L, \qquad z \leq xy^U + x^L y - x^L y^U.$$

It was later shown by Al-Khayyal et al. [3, 4] that the former two of these four inequalities provide the convex envelope while the latter two provide the concave envelope of xy. In other words, the four inequalities describe the convex hull of the set $\{(x, y, z) \ : \ z = xy,\ x^L \leq x \leq x^U,\ y^L \leq y \leq y^U\}$.

Piecewise linear relaxations. It was shown in [12] that the maximum difference between the variable z and the bilinear term xy is $d_{\max} = \frac{1}{4}(x^U - x^L)(y^U - y^L)$, i.e., $d_{\max}$ is proportional to the size of the domain. Hence, the McCormick relaxation is tighter on smaller domains. By partitioning the domain *a priori* and constructing a series of sub-relaxations, one can construct a tighter relaxation in the same domain. But since only one of the segments is active for a given point in the domain, the problem is represented by a mixed-integer linear program (MILP) rather than a linear program (LP). A recent review on advanced MILP formulation techniques [47] studies the trade-off between smaller and stronger formulations. For a MILP encoding formulation framework, see [56]. Using univariate partitioning and equal segment lengths, Misener et al. [40] introduced two MILP formulations, one with a linear and the other one with a logarithmic number of binary variables. Inspired by a recent paper on mixed-integer models for nonseparable piecewise linear optimization [48], both formulations partition $[x^L, x^U]$ into N segments of equal length $a = \frac{1}{N}(x^U - x^L)$. *A priori* domain partitioning has been successfully applied to solve large pooling problem instances to global optimality [26, 37, 42, 55]. The literature on bivariate partitioning and unequal segment lengths, however, is relatively sparse [18, 29]. A description of a solver that globally optimizes pooling problems using piecewise linear relaxations, APOGEE, was published recently [40].

Rectangular versus triangular domains. While the aforementioned convex and linear relaxations are relaxations on *rectangular* domains, Linderoth [34] recently showed that on *triangular* domains a combination of convex (but nonlinear) underestimators and linear overestimators or linear underestimators and concave (but nonlinear) overestimators results in tighter relaxations. By using nonlinear relaxations, Linderoth reduced the maximum difference between the variable z and the bilinear term xy from $d^{\square}_{\max} = \frac{1}{4}(x^U - x^L)(y^U - y^L)$ for rectangular domains to $d^{\triangle}_{\max} = \frac{1}{16}(x^U - x^L)(y^U - y^L)$ for triangular domains. The trade-off is that one relaxes the nonconvex NLP by a convex NLP instead of an LP.

Relaxations may be tightened further by applying the *Reformulation-Linearization Technique* (RLT). The RLT reformulates an NLP by constructing a set of nonnegative variable factors using the problem constraints, and multiplies combinations of these factors with the original problem constraints to generate additional valid nonlinear constraints [43]. It was shown that when the problem is relaxed, the resulting relaxations are tighter than they would have been without the additional constraints [13, 14, 43, 44, 46]. A

recent paper by Gupte et al. [28] gives an excellent overview of topics that have been studied in the context of the pooling problem. The authors present four essential formulations, prove their equivalence and compare their corresponding strengths and sizes of formulations. They summarise the main complexity results for the pooling problem and compare different solution techniques. The principal focus of the paper is the discussion of variable discretization techniques. Motivated by the fact that MILP solvers are more mature than global optimization solvers, one may discretize some variables or, alternatively, some consistency requirements. Such MILPs provide inner approximations to the original problem. In an extensive computational study, the authors compare different discretization techniques on large-scale pooling problem instances to determine which strategy empirically performs best.

3 Problem definition

In the pooling problem, we send raw materials from inputs, blend them in pools, and finally send the blended raw materials from pools to outputs where they are blended again. We consider a directed graph $G = (V, A)$ where V is the set of vertices and A is the set of arcs. V is partitioned into three nonempty subsets $I, L, J \subset V$: I is the set of inputs, L the set of pools and J the set of outputs. Flows are blended in pools and outputs. We 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 ($A \cap (I \times I) = \emptyset$) or two outputs ($A \cap (J \times J) = \emptyset$) and no arcs from pools to inputs ($A \cap (L \times I) = \emptyset$) or outputs to pools ($A \cap (J \times L) = \emptyset$) or outputs to inputs ($A \cap (J \times I) = \emptyset$). Throughout this paper, we write $A = A_{IL} \cup A_{LL} \cup A_{LJ} \cup A_{IJ}$ where

$$A_{IL} = A \cap (I \times L), \qquad A_{LL} = A \cap (L \times L), \qquad A_{LJ} = A \cap (L \times J), \qquad A_{IJ} = A \cap (I \times J).$$

We further assume, without loss of generality, that there are no nodes that cannot contribute to any feasible solution. More precisely, we assume that every pool has at least one incoming arc and at least one outgoing arc, every input node has at least one outgoing arc, and every output node has at least one incoming arc.

We consider a set of qualities K whose quality values are tracked across the network. We assume linear blending, i.e., the quality value of pools and outputs $v \in L \cup J$ for quality $k \in K$ is a linear combination of the incoming quality values weighted by the corresponding incoming flows as fractions of the total incoming flow. Problem instances with $A_{LL} = \emptyset$ are referred to as *standard pooling problems* (SPPs), while instances with $A_{LL} \neq \emptyset$ are called *generalized pooling problems* (GPPs). Both SPPs and GPPs can be modelled as bilinear programs (BLPs), which is a special case of a quadratically constrained quadratic program (QCQP). A QCQP in turn is a special case of an NLP. Apart from the NLP formulation of the pooling problem, which assumes the arc set A to be fixed, there exist mixed-integer nonlinear programming (MINLP) extensions of the pooling problem in which A is variable [37]. In addition to the continuous flow and quality variables, we then have binary variables representing arc on-off switches. Last but not least, problem instances with $L = \emptyset$ are referred to as *blending problems*. Blending problems can be modelled as LPs.

For every pool and output $v \in L \cup J$, we denote the set of incoming arcs by $\delta^{\text{in}}(v)$, and for every input and pool $v \in I \cup L$, we denote the set of outgoing arcs by $\delta^{\text{out}}(v)$. Let y_a be the flow on arc $a \in A$ and let c_a be the cost of flow on arc $a \in A$. The total incoming/outgoing flow of vertex $v \in V$ and the flow on arc $a \in A$ are bounded below by zero and bounded above by C_v and u_a, respectively. For every input $i \in I$ and quality $k \in K$, the quality values of the incoming raw materials are given by λ_{ik}. Similarly, for every output $v \in J$ and quality $k \in K$, the lower and upper bounds on the quality values of the outgoing

Table 1: Notation for the pooling problem

Sets	
V	Set of vertices
I	Set of inputs
L	Set of pools
J	Set of outputs
K	Set of qualities
A	Set of arcs
A_{IL}	Set of input-to-pool arcs: $A_{IL} := A \cap (I \times L)$
A_{LL}	Set of pool-to-pool arcs: $A_{LL} := A \cap (L \times L)$
A_{LJ}	Set of pool-to-output arcs: $A_{LJ} := A \cap (L \times J)$
A_{IJ}	Set of input-to-output arcs: $A_{IJ} := A \cap (I \times J)$
$\delta^{\text{in}}(v)$	Set of incoming arcs of $v \in L \cup J$
$\delta^{\text{out}}(v)$	Set of outgoing arcs of $v \in I \cup L$

Parameters	
c_a	Cost of flow on arc $a \in A$
λ_{ik}	Quality value of input $i \in I$ for quality $k \in K$
$\mu_{vk}^{\min}, \mu_{vk}^{\max}$	Minimum (maximum) acceptable quality value of output $v \in J$ for quality $k \in K$
C_v	Upper bound on total incoming/outgoing flow of $v \in V$
u_a	Upper bound on flow y_a, $a \in A$

Variables	
y_a	Flow on arc $a \in A$
p_{vk}	Quality value of $v \in I \cup L$ for quality $k \in K$ ($p_{vk} \equiv \lambda_{vk}$, $v \in I$, $k \in K$)
p_{ak}	$p_{ak} \equiv p_{vk}$, $a = (v, w) \in A$, $k \in K$
z_{ak}	Auxiliary variable: $z_{ak} = p_{ak} y_a$, $a \in A$, $k \in K$
q_{iv}^I	Fraction of total incoming/outgoing flow of $v \in I \cup L$ that comes from input $i \in I$
q_{ia}^I	$q_{ia}^I \equiv q_{iv}^I$, $a = (v, w) \in A$, $i \in I$
q_{jv}^J	Fraction of total incoming/outgoing flow of $v \in L \cup J$ that goes to output $j \in J$
q_{ja}^J	$q_{ja}^J \equiv q_{jw}^J$, $a = (v, w) \in A$, $j \in J$
q_{ija}^{IJ}	Fraction of flow y_a, $a \in A$, that comes from input $i \in I$ and goes to output $j \in J$
x_{ia}^I	Flow in y_a, $a \in A$, that comes from input $i \in I$
x_{ja}^J	Flow in y_a, $a \in A$, that goes to output $j \in J$
x_{ija}^{IJ}	Flow in y_a, $a \in A$, that comes from input $i \in I$ and goes to output $j \in J$

blended materials are given by $\mu_{vk}^{\min}$ and $\mu_{vk}^{\max}$, respectively. Table 1 summarizes the notation for the pooling problem.

4 Formulations

In this section we discuss various formulations for the pooling problem. An overview of known formulations is provided in Section 4.1, and new formulations are proposed in Section 4.2. In Section 4.3 we present a proof of the equivalence of all formulations, and we conclude with a discussion of the strength of formulations in Section 4.4 where we characterize the partial order of formulations with respect to the strength of their LP relaxations (up to one relation).

We will switch between arc- and node-based constraints throughout this section. For notational convenience, we therefore also introduce arc- and node-based variables which can be transformed into one another. For example, in the P-formulation, we will introduce a variable p_{vk} for every node $v \in I \cup L$ and quality $k \in K$ which represents the quality value of the blended material in a particular input or

pool node. It will be convenient to have symbols p_{ak} for $a \in A$ and $k \in K$ to refer to the quality values of the material corresponding to the flow on arc a. For $a \in \delta^{\text{out}}(v)$, this is the material that has been blended in node v, and therefore we have $p_{ak} = p_{vk}$ for all $v \in I \cup L$, $a \in \delta^{\text{out}}(v)$ and $k \in K$. Note, however, that the p_{ak} do not actually appear in an implementation of the model, but are replaced by the corresponding p_{vk}. We do the same in the MCF-I-(P)Q-formulations with variables q^I_{iv} and q^I_{ia}, and in the MCF-J-(P)Q-formulations with variables q^J_{jv} and q^J_{ja}.

4.1 Known formulations

4.1.1 P-formulation

Proposed by Haverly in 1978 [30], the P-formulation uses flow variables y, quality variables p (hence the name of the formulation) and their product z. The flow variable y_a is defined for all arcs $a \in A$, the quality variables p_{vk} for all nodes $v \in I \cup L$ and for all qualities $k \in K$. For notational convenience we set $p_{ak} = p_{vk}$ for every node $v \in I \cup L$, every $a \in \delta^{\text{out}}(v)$ and every quality $k \in K$. Finally, we define variables z_{ak} for all arcs $a \in A$ and all qualities $k \in K$ to represent the product $p_{ak}y_a$. The P-formulation can be stated as follows:

$$
\begin{aligned}
[\text{P}] \quad & \min_{p,y,z} \sum_{a \in A} c_a y_a \\
\text{s.t.} \quad & \sum_{a \in \delta^{\text{in}}(v)} y_a = \sum_{a \in \delta^{\text{out}}(v)} y_a, && v \in L, && (1)\\
& \sum_{a \in \delta^{\text{out}}(v)} y_a \le C_v, && v \in I \cup L, && (2)\\
& \sum_{a \in \delta^{\text{in}}(v)} y_a \le C_v, && v \in J, && (3)\\
& 0 \le y_a \le u_a, && a \in A, && (4)\\
& p_{vk} = \lambda_{vk}, && v \in I,\ k \in K, && (5)\\
& \sum_{a \in \delta^{\text{in}}(v)} z_{ak} = \sum_{a \in \delta^{\text{out}}(v)} z_{ak}, && v \in L,\ k \in K, && (6)\\
\mu^{\min}_{vk} \sum_{a \in \delta^{\text{in}}(v)} y_a \le & \sum_{a \in \delta^{\text{in}}(v)} z_{ak} \le \mu^{\max}_{vk} \sum_{a \in \delta^{\text{in}}(v)} y_a, && v \in J,\ k \in K, && (7)\\
& z_{ak} = p_{ak} y_a, && a \in A,\ k \in K. && (8)
\end{aligned}
$$

Constraint (1) is flow conservation which ensures that at every pool, the total incoming flow equals the total outgoing flow. Constraints (2) and (3) are vertex capacity constraints and (4) is an arc capacity constraint. Constraints (1)–(4) define a feasible flow. Constraint (5) ensures that the quality values for input node $v \in I$ are equal to the input data, and (6) is the pool blending constraint which ensures that the p variables track the quality values across the network. Finally, (7) is the output blending constraint and (8) outsources the py terms – which would otherwise appear in (6) and (7) – into a single constraint. Note that by substituting (8) into (6) and (7), the P-formulation may also be stated without the z variables.

4.1.2 Input commodities: MCF-I-(P)Q-formulation

The MCF-I-Q-formulation uses fraction variables q^I (hence the name of the formulation), flow variables y and disaggregated flow variables x^I, which are products of q^I and y. Since we will also propose a multi-commodity flow formulation based on output commodities, we use the superscript I for both q^I

and x^I so as to distinguish from q^J and x^J, which we will introduce later. For $v \in I \cup L$, let q^I_{iv} denote the fraction of the total flow through node v that comes from input node $i \in I$. In particular, for any input node $v \in I$ we have $q^I_{vv} = 1$ and $q^I_{iv} = 0$ for $i \neq v$. Since the blending is linear, we know that the fraction of the flow on an arc $a \in \delta^{\text{out}}(v)$ that comes from input $i \in I$ is equal to q^I_{iv}, and for convenience we introduce $q^I_{ia} = q^I_{iv}$ for all $v \in I \cup L$ and all $a \in \delta^{\text{out}}(v)$. Furthermore, let x^I_{ia} denote the flow in y_a, $a \in A$, that comes from input $i \in I$. The formulation has been first proposed for standard pooling problems by Ben-Tal et al. in 1994 [16] and has been extended to generalized pooling problems by Alfaki and Haugland in 2013 [8]. It can be stated as follows:

$$\begin{aligned}
\text{[MCF-I-Q]} \quad & \min_{q^I, x^I, y} \sum_{a \in A} c_a y_a \\
\text{s.t.} \quad & (1)\text{–}(4), \\
& q^I_{iv} \geq 0, && v \in I \cup L,\ i \in I, && (9)\\
& \sum_{i \in I} q^I_{iv} = 1, && v \in I \cup L, && (10)\\
& \sum_{a \in \delta^{\text{in}}(v)} x^I_{ia} = \sum_{a \in \delta^{\text{out}}(v)} x^I_{ia}, && v \in L,\ i \in I, && (11)\\
& \mu^{\min}_{jk} \sum_{a \in \delta^{\text{in}}(j)} y_a \leq \sum_{i \in I} \sum_{a \in \delta^{\text{in}}(j)} \lambda_{ik} x^I_{ia} \leq \mu^{\max}_{jk} \sum_{a \in \delta^{\text{in}}(j)} y_a, && j \in J,\ k \in K, && (12)\\
& x^I_{ia} = q^I_{ia} y_a, && a \in A,\ i \in I. && (13)
\end{aligned}$$

Constraints (9) and (10) ensure that all fraction variables are between zero and one and that for every input and pool, the fractions sum to one. (11) can be viewed as a disaggregated flow conservation constraint. Note that (1) ensures that at every pool, the total incoming flow must equal the total outgoing flow, regardless of the origin of the flows. Constraint (11), on the other hand, ensures that at every pool, the total incoming flow originating from an input must equal the total outgoing flow originating from the same input. In fact, using (10) and (13), we can show that (11) implies (1):

$$\begin{aligned}
\sum_{a \in \delta^{\text{in}}(v)} y_a &\overset{(10)}{=} \sum_{a \in \delta^{\text{in}}(v)} y_a \sum_{i \in I} q^I_{ia} \overset{(13)}{=} \sum_{a \in \delta^{\text{in}}(v)} \sum_{i \in I} x^I_{ia} = \sum_{i \in I} \sum_{a \in \delta^{\text{in}}(v)} x^I_{ia} \\
&\overset{(11)}{=} \sum_{i \in I} \sum_{a \in \delta^{\text{out}}(v)} x^I_{ia} = \sum_{a \in \delta^{\text{out}}(v)} \sum_{i \in I} x^I_{ia} \overset{(13)}{=} \sum_{a \in \delta^{\text{out}}(v)} y_a \sum_{i \in I} q^I_{ia} \overset{(10)}{=} \sum_{a \in \delta^{\text{out}}(v)} y_a.
\end{aligned}$$

However, we still include (1) to strengthen the linear relaxation. Constraint (12) is analogous to (7) in that it ensures that the quality restrictions are satisfied. (13) outsources the $q^I y$ terms – which would otherwise appear in (11) and (12) – into a single constraint. Inspired by the ideas of the Reformulation-Linearization Technique (RLT) [43], the MCF-I-PQ-formulation adds valid (but redundant) constraints to the MCF-I-Q-formulation. The formulation has been first proposed for standard pooling problems by Tawarmalani et al. in 2002 [45] and has been extended to generalized pooling problems by Alfaki and Haugland in 2013 [8]. It can be stated as follows:

$$\begin{aligned}
\text{[MCF-I-PQ]} \quad & \min_{q^I, x^I, y} \sum_{a \in A} c_a y_a \\
\text{s.t.} \quad & (1)\text{–}(4),\ (9)\text{–}(13), \\
& y_a = \sum_{i \in I} x^I_{ia}, && a \in A, && (14)\\
& \sum_{a \in \delta^{\text{out}}(v)} x^I_{ia} \leq C_v q^I_{iv}, && v \in L,\ i \in I. && (15)
\end{aligned}$$

(14) is obtained by multiplying (10) with y_a for $a \in A$, and using (13). (15) is obtained by multiplying (2) with q^I_{iv} for $i \in I$, $v \in L$, and again using (13). The latter two redundant constraints significantly strengthen the relaxation of the MCF-I-Q-formulation.

In addition to P, MCF-I-Q and MCF-I-PQ, the HYB-formulation has been proposed by Audet et al. in 2004 [13]. The formulation is a hybrid of P, MCF-I-Q and MCF-I-PQ, using both the P-formulation's quality variables and the MCF-I-(P)Q-formulation's fraction variables in addition to the flow variables. The formulation distinguishes between pools for which there are no incoming pool-to-pool arcs (i.e., all incoming arcs are input-to-pool arcs), $L_I = \{v \in L : \nexists\, w \in L \text{ s.t. } (w,v) \in A\}$, and all other pools $L \setminus L_I$. Then some of the P-formulation constraints are added for all $v \in L \setminus L_I$ and some of the MCF-I-(P)Q-formulation constraints are added for all $v \in L_I$. However, since the HYB-formulation's strength is often inferior to MCF-I-PQ [8], we omit the formulation in this paper.

Note that we may replace constraint (6) in the P-formulation by

$$\sum_{a \in \delta^{\text{in}}(v)} z_{ak} = \sum_{a \in \delta^{\text{in}}(v)} p_{vk} y_a, \quad v \in L,\ k \in K.$$

While this variant of the P-formulation is valid, when solving the linear relaxations of the nonlinear problems, we may obtain different lower bounds on the optimal objective function value. This is due to different bilinear terms, different bounds on the participating variables and therefore different strengths of formulations. However, compared to the above variant, we found that constraint (6) generally provides a stronger linear relaxation. In fact, we could not find any instances for which the above variant yields better lower bounds. Therefore, we do not study this variant any further.

4.2 New formulations

4.2.1 Output commodities: MCF-J-(P)Q-formulation

To the best of our knowledge, a multi-commodity flow formulation based on output commodities is not yet known for generalized pooling problems. For standard pooling problems, however, Alfaki and Haugland proposed a similar formulation in [9]. In addition to the flow variables, our new formulation uses the fraction variable q^J and the disaggregated flow variable x^J. For $v \in L \cup J$, let q^J_{jv} denote the fraction of the total flow through node v that goes to output node $j \in J$. In particular, for any output node $v \in J$ we have $q^J_{vv} = 1$ and $q^J_{jv} = 0$ for $j \neq v$. Since the blending is linear, we know that the fraction of the flow on an arc $a \in \delta^{\text{in}}(v)$ that goes to output $j \in J$ is equal to q^J_{jv}, and for convenience we introduce $q^J_{ja} = q^J_{jv}$ for all $v \in L \cup J$ and all $a \in \delta^{\text{in}}(v)$. With x^J_{ja} representing the flow in y_a, $a \in A$, that goes to output $j \in J$ the formulation can be stated as follows:

$$\begin{aligned}
\text{[MCF-J-Q]} \quad & \min_{q^J, x^J, y} \ \sum_{a \in A} c_a y_a \\
\text{s.t.} \quad & (1)\text{–}(4), \\
& q^J_{jv} \geq 0, && v \in L \cup J,\ j \in J, && (16)\\
& \sum_{j \in J} q^J_{jv} = 1, && v \in L \cup J, && (17)\\
& \sum_{a \in \delta^{\text{in}}(v)} x^J_{ja} = \sum_{a \in \delta^{\text{out}}(v)} x^J_{ja}, && v \in L,\ j \in J, && (18)\\
\mu^{\min}_{jk} \sum_{a \in \delta^{\text{in}}(j)} y_a \leq & \sum_{i \in I} \sum_{a \in \delta^{\text{out}}(i)} \lambda_{ik} x^J_{ja} \leq \mu^{\max}_{jk} \sum_{a \in \delta^{\text{in}}(j)} y_a, && j \in J,\ k \in K, && (19)\\
& x^J_{ja} = q^J_{ja} y_a, && a \in A,\ j \in J. && (20)
\end{aligned}$$

(16) and (17) ensure that all fraction variables are between zero and one and that for every pool and output, the fractions sum to one. (18) implies (1) using (17) and (20), (19) is analogous to (12) and (20) is analogous to (13), outsourcing the $q^J y$ terms into a single constraint. We again add valid (but redundant) constraints to the MCF-J-Q-formulation to strengthen its relaxation:

$$\begin{aligned}
\text{[MCF-J-PQ]} \quad & \min_{q^J, x^J, y} \sum_{a \in A} c_a y_a \\
\text{s.t.} \quad & (1)\text{–}(4),\ (16)\text{–}(20), \\
& y_a = \sum_{j \in J} x^J_{ja}, \quad a \in A, && (21) \\
& \sum_{a \in \delta^{\text{out}}(v)} x^J_{ja} \leq C_v q^J_{jv}, \quad v \in L,\ j \in J. && (22)
\end{aligned}$$

4.2.2 Input and output commodities: MCF-(I+J)-(P)Q-formulation

As we will show in Section 4.4, MCF-I-(P)Q and MCF-J-(P)Q are incomparable, i.e., there are pooling problem instances for which MCF-I-(P)Q is stronger than MCF-J-(P)Q and vice versa. But since the two formulations share the y variables, they may complement each other in a multi-commodity flow formulation based on input *and* output commodities. Including both the variables and constraints of MCF-I-Q and MCF-J-Q, we obtain the MCF-(I+J)-Q-formulation:

$$\begin{aligned}
\text{[MCF-(I+J)-Q]} \quad & \min_{q^I, q^J, x^I, x^J, y} \sum_{a \in A} c_a y_a \\
\text{s.t.} \quad & (1)\text{–}(4),\ (9)\text{–}(13),\ (16)\text{–}(20).
\end{aligned}$$

Analogously, including both the variables and constraints of MCF-I-PQ and MCF-J-PQ, we obtain the MCF-(I+J)-PQ-formulation:

$$\begin{aligned}
\text{[MCF-(I+J)-PQ]} \quad & \min_{q^I, q^J, x^I, x^J, y} \sum_{a \in A} c_a y_a \\
\text{s.t.} \quad & (1)\text{–}(4),\ (9)\text{–}(22).
\end{aligned}$$

4.2.3 (Input, output)-commodities: MCF-(I×J)-(P)Q-formulation

It is important to note that the q^I and q^J variables (resp. the x^I and x^J variables) in MCF-I-(P)Q, MCF-J-(P)Q and MCF-(I+J)-(P)Q are interdependent. However, in the MCF-(I+J)-(P)Q-formulations, the only link between the input and output commodities are the y variables. We can strengthen the link and better capture the interdependence by introducing (input, output)-commodities. Let q^{IJ}_{ija} denote the fraction of flow y_a, $a \in A$, that comes from input $i \in I$ and goes to output $j \in J$. Analogously, let x^{IJ}_{ija} denote the flow in y_a, $a \in A$, that comes from input $i \in I$ and goes to output $j \in J$. Using these additional variables, the MCF-(I×J)-Q-formulation can be stated as follows:

Table 2: Known and new formulations for standard pooling problems (SPP) and generalized pooling problems (GPP)

Formulation	SPP	GPP
P	known [30]	known [30]
MCF-I-(P)Q	known [16, 45]	known [8]
MCF-J-(P)Q	known [9]	new
MCF-(I+J)-(P)Q	known [9]	new
MCF-(I×J)-(P)Q	new	new

$$\text{[MCF-(I×J)-Q]} \quad \min_{q^I,q^J,q^{IJ},x^I,x^J,x^{IJ},y} \sum_{a\in A} c_a y_a$$

$$\text{s.t.} \quad (1)\text{–}(4),\ (9)\text{–}(13),\ (16)\text{–}(20),$$

$$q^{IJ}_{ija} \geq 0, \qquad a\in A,\ i\in I,\ j\in J, \tag{23}$$

$$\sum_{i\in I}\sum_{j\in J} q^{IJ}_{ija} = 1, \qquad a\in A, \tag{24}$$

$$q^{IJ}_{ija} = q^{I}_{ia} q^{J}_{ja}, \qquad a\in A,\ i\in I,\ j\in J, \tag{25}$$

$$\sum_{a\in\delta^{\text{in}}(v)} x^{IJ}_{ija} = \sum_{a\in\delta^{\text{out}}(v)} x^{IJ}_{ija}, \qquad v\in L,\ i\in I,\ j\in J, \tag{26}$$

$$\mu^{\min}_{jk} \sum_{a\in\delta^{\text{in}}(j)} y_a \leq \sum_{i\in I}\sum_{a\in\delta^{\text{in}}(j)} \lambda_{ik} x^{IJ}_{ija} \leq \mu^{\max}_{jk} \sum_{a\in\delta^{\text{in}}(j)} y_a, \quad j\in J,\ k\in K, \tag{27}$$

$$x^{IJ}_{ija} = q^{IJ}_{ija} y_a, \qquad a\in A,\ i\in I,\ j\in J. \tag{28}$$

As before, (23) and (24) ensure that the additional fraction variables are between zero and one and that for every arc, the fractions sum to one. Then (25) captures the interdependence between q^I and q^J. Using the new variables, we disaggregate the flow conservation constraints (11) and (18) in (26) and add the output blending constraint (27). Note that we could substitute (25) into (28), which would introduce trilinear terms. For a linear relaxation of the trilinear term $w = xyz$, see e.g. [36]. Adding valid (but redundant) constraints to the MCF-(I×J)-Q-formulation, we conclude with the MCF-(I×J)-PQ-formulation:

$$\text{[MCF-(I×J)-PQ]} \quad \min_{q^I,q^J,q^{IJ},x^I,x^J,x^{IJ},y} \sum_{a\in A} c_a y_a$$

$$\text{s.t.} \quad (1)\text{–}(4),\ (9)\text{–}(28),$$

$$y_a = \sum_{i\in I}\sum_{j\in J} x^{IJ}_{ija}, \quad a\in A, \tag{29}$$

$$x^{IJ}_{ija} \leq u_a q^{IJ}_{ija}, \qquad a\in A,\ i\in I,\ j\in J. \tag{30}$$

An overview of known and new formulations is shown in Table 2. For future reference, we define the following sets of nonlinear formulations:

$$\mathcal{F}_{\text{Q}} := \{\text{MCF-I-Q, MCF-J-Q, MCF-(I+J)-Q, MCF-(I×J)-Q}\}, \qquad \text{set of all Q-formulations,}$$

$$\mathcal{F}_{\text{PQ}} := \{\text{MCF-I-PQ, MCF-J-PQ, MCF-(I+J)-PQ, MCF-(I×J)-PQ}\}, \qquad \text{set of all PQ-formulations,}$$

$$\mathcal{F}_{\text{all}} := \{\text{P}\} \cup \mathcal{F}_{\text{Q}} \cup \mathcal{F}_{\text{PQ}}, \qquad \text{set of all formulations.}$$

4.3 Equivalence of formulations

It was proven in [27], Proposition 1.1, that the MCF-I-Q-formulation is equivalent to the P-formulation if the network is acyclic. The main result of this section is that the assumption of acyclicity is not necessary,

i.e., the formulations are equivalent even if the network contains directed cycles. Furthermore our proof easily extends to the equivalence of the MCF-J-Q-formulation and the P-formulation. Note that for every input-to-output arc in a graph representing a pooling problem, we may add a pool and split the arc into an input-to-pool and pool-to-output arc. Therefore, without loss of generality, we assume that $A_{IJ} = \emptyset$.

Let X be the set of flow vectors $y \in \mathbb{R}^A_{\geq 0}$ which satisfy the capacity constraints:

$$X = \left\{ y \in \mathbb{R}^A \ : \ (1),\ (2),\ (3),\ (4) \right\}.$$

For a vector $y \in X$ we write y_v for the total flow through node $v \in L$, i.e., $y_v = \sum_{a \in \delta^{\text{in}}(v)} y_a$. Furthermore, let $L_0 = L_0(y)$ denote the set of pool nodes v such that there is no flow through v, i.e., $L_0 = \{v \in L \ : \ y_v = 0\}$.

The basic idea behind the proof is that every vector $y \in X$ and every formulation in $\mathcal{F}_{\text{all}}$ there is a unique extension (by z variables for the P-formulation, and by q variables for the Q- and PQ-formulations) which satisfies all the blending constraints, i.e., all constraints except possibly the quality constraints at the output nodes (constraints (7), (12), (19)). Then it remains to be checked that this unique extension for y satisfies the quality constraints in one formulation if and only if it does in the other formulations.

For instance, for the P-formulation, given the flow vector $y \in X$, we claim that constraints (5), (6) and (8) uniquely determine the values z_{ak}. First note that constraints (5) and (8) imply $z_{ak} = \lambda_{vk} y_a$ for all $v \in I$ and $a \in \delta^{\text{out}}(v)$. Therefore we consider the values z_{ak} for $a \in A_{IL}$ as known, and the values for $a \in A \setminus A_{IL}$ as unknown. In the following, for a vector $\xi = (\xi_a)_{a \in A}$ and a subset $A' \subseteq A$, let $\xi|_{A'}$ denote the restriction of ξ to the coordinates indexed by A'. Using this notation, our claim is that $z_k|_{A_{LL} \cup A_{LJ}}$ is uniquely determined by y and $z_k|_{A_{IL}}$. Note that for arcs $a \in \delta^{\text{out}}(v)$ for some $v \in L_0$ we have $z_{ak} = 0$ for all k by (8). Now let $v \in L \setminus L_0$ and $a \in \delta^{\text{out}}(v)$. We can multiply (8) by y_v to obtain (recall that p_{ak} is just a placeholder for p_{vk})

$$y_v z_{ak} = y_v p_{vk} y_a = y_a p_{vk} \sum_{a' \in \delta^{\text{out}}(v)} y_{a'} = y_a \sum_{a' \in \delta^{\text{out}}(v)} p_{a'k} y_{a'} = y_a \sum_{a' \in \delta^{\text{out}}(v)} z_{a'k} \overset{(6)}{=} y_a \sum_{a' \in \delta^{\text{in}}(v)} z_{a'k}.$$

So constraints (6) and (8) imply

$$\begin{aligned} z_{ak} &= 0, && v \in L_0,\ a \in \delta^{\text{out}}(v), \\ y_v z_{ak} - y_a \sum_{a' \in \delta^{\text{in}}(v) \setminus A_{IL}} z_{a'k} &= y_a \sum_{a' = (i,v) \in \delta^{\text{in}}(v) \cap A_{IL}} \lambda_{ik} y_{a'}, && v \in L \setminus L_0,\ a \in \delta^{\text{out}}(v). \end{aligned}$$

Conversely, if z satisfies these constraints then we can find values of p_{ak} such that (5), (6) and (8) are satisfied (just put $p_{vk} = 0$ for $v \in L_0$ and $p_{vk} = z_{ak}/y_a$ for $v \in L \setminus L_0$ and $a \in \delta^{\text{out}}(v)$ with $y_a \neq 0$). In other words, constraints (5), (6) and (8) of the P-formulation are equivalent to the matrix equations $F z_k|_{A_{LL} \cup A_{LJ}} = b^P_k$ for all $k \in K$, where $F = F(y)$ and $b^P_k = b^P_k(y)$ are a square matrix of size $|A_{LL} \cup A_{LJ}|$ and a vector of dimension $|A_{LL} \cup A_{LJ}|$, respectively, defined by

$$\begin{aligned} F_{aa} &= y_v, && v \in L \setminus L_0,\ \ a \in \delta^{\text{out}}(v), \\ F_{aa} &= 1, && v \in L_0,\ \ a \in \delta^{\text{out}}(v), \\ F_{a,a'} &= -y_a, && v \in L,\ \ a \in \delta^{\text{out}}(v),\ \ a' \in \delta^{\text{in}}(v) \setminus A_{IL}, \\ b^P_{ak} &= y_a \sum_{a' = (i,v) \in \delta^{\text{in}}(v) \cap A_{IL}} \lambda_{ik} y_{a'}, && v \in L,\ \ a \in \delta^{\text{out}}(v). \end{aligned}$$

Similarly, constraints (10), (11) and (13) of the MCF-I-Q-formulation are equivalent to the matrix equa-

tions $Fx_i^I|_{A_{LL}\cup A_{LJ}} = b_i^I$ for all $i \in I$, where $b_i^I = (b_{ia}^I)_{a\in A_{LL}\cup A_{LJ}}$ for $i \in I$ is a vector (depending on $y \in X$), defined by

$$b_{ia}^I = y_a \sum_{a'\in\delta^{\rm in}(v)\cap A_{IL}} y_{a'}, \qquad v \in L,\ \ a \in \delta^{\rm out}(v),$$

and constraints (17), (18) and (20) are equivalent to the matrix equations $Bx_j^J|_{A_{IL}\cup A_{LL}} = b_j^J$ for all $j \in J$, where $B = B(y)$ is a square matrix of size $|A_{IL} \cup A_{LL}|$ and $b_j^J = b_j^J(y)$ is a $|A_{IL} \cup A_{LL}|$-dimensional vector, given by

$$\begin{aligned}
B_{aa} &= y_v, && v \in L \setminus L_0,\ \ a \in \delta^{\rm in}(v),\\
B_{aa} &= 1, && v \in L_0,\ \ a \in \delta^{\rm in}(v),\\
B_{a,a'} &= -y_a, && v \in L,\ \ a \in \delta^{\rm in}(v),\ \ a' \in \delta^{\rm out}(v) \setminus A_{LJ},\\
b_{ja}^J &= y_a \sum_{a'\in\delta^{\rm out}(v)\cap A_{LJ}} y_{a'}, && v \in L,\ \ a \in \delta^{\rm in}(v).
\end{aligned}$$

In order to prove our claim, we check that all of these systems have a unique solution.

Lemma 1. *For every $y \in X$, the matrices $F = F(y)$ and $B = B(y)$ are invertible.*

Proof. Note that $F(0)$ is the identity matrix, an therefore we can assume $y \neq 0$. Suppose F is not invertible, and let $x \neq 0$ be a vector with $Fx = 0$. We define a vector $q \in \mathbb{R}^L$ by $q_v = 0$ for $v \in L_0$ and

$$q_v = \frac{1}{y_v} \sum_{a\in\delta^{\rm in}(v)\setminus A_{IL}} x_a, \qquad v \in L \setminus L_0.$$

Note that $x \neq 0$ implies $\|q\|_\infty > 0$. Let $L_1 = \{v \in L \ :\ q_v = \|q\|_\infty\}$. Without loss of generality we assume $L_1 \neq \emptyset$. Let $a = (v', v)$ be an arc with $y_a \neq 0$, $v \in L_1$ and $v' \notin L_1$ (such an arc clearly exists). We put

$$\delta_1^{\rm in}(v) = \{a = (v'', v) \in A \ :\ v'' \in L \setminus L_0\}.$$

Case 1. $v' \in L$. In this case, $v' \notin L_1$ implies $q_{v'} < q_v$, and therefore

$$\begin{aligned}
q_v = \frac{1}{y_v} \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)} x_{a''} &= \frac{1}{y_v} \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)} \frac{y_{a''}}{y_{v''}} \sum_{b\in\delta^{\rm in}(v'')\setminus A_{IL}} x_b\\
&= \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)} \frac{y_{a''}}{y_v} q_{v''} = \frac{y_a}{y_v} q_{v'} + \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)\setminus\{a\}} \frac{y_{a''}}{y_v} q_{v''} < q_v \sum_{a''\in\delta^{\rm in}(v)} \frac{y_{a''}}{y_v} = q_v,
\end{aligned}$$

which is the required contradiction.

Case 2. $v' \in I$. In this case,

$$\begin{aligned}
q_v = \frac{1}{y_v} \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)} x_{a''} &= \frac{1}{y_v} \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)} \frac{y_{a''}}{y_{v''}} \sum_{b\in\delta^{\rm in}(v'')\setminus A_{IL}} x_b\\
&= \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)} \frac{y_{a''}}{y_v} q_{v''} \le q_v \sum_{a''=(v'',v)\in\delta_1^{\rm in}(v)} \frac{y_{a''}}{y_v} \le q_v \left(1 - \frac{y_a}{y_v}\right) < q_v,
\end{aligned}$$

which is the required contradiction.

The proof for matrix B is similar. □

From the remarks just before Lemma 1, the following lemmas are immediate.

Lemma 2. *Let $y \in X$, and let $z_k = (z_{ak})_{a \in A}$ be a vector with $z_{ak} = \lambda_{ik} y_a$ for all $a = (i, v) \in A_{IL}$. The following two statements are equivalent.*

1. *The pair (y, z) can be extended by a vector p such that constraints (5), (6), and (8) of the P-formulation are satisfied.*
2. *$z|_{A_{LL} \cup A_{LJ}} = F(y)^{-1} b_k^P$ for all $k \in K$.*

Lemma 3. *Let $y \in X$, and for each $i \in I$, let $x_i^I = (x_{ia}^I)_{a \in A}$ be the vector with $x_{ia}^I = y_a$ for all $a \in \delta^{\text{out}}(i)$ and $x_{ia}^I = 0$ for all $a \in \delta^{\text{out}}(I \setminus \{i\})$. The following two statements are equivalent.*

1. *The pair (y, x^I) can be extended by a vector q^I such that constraints (9), (10), (11) and (13) of the MCF-I-Q-formulation are satisfied.*
2. *$x_i^I|_{A_{LL} \cup A_{LJ}} = F(y)^{-1} b_i^I$ for all $i \in I$.*

Lemma 4. *Let $y \in X$, and for each $j \in J$, let $x_j^J = (x_{ja}^J)_{a \in A}$ be the vector with $x_{ja}^J = y_a$ for all $a \in \delta^{\text{in}}(j)$ and $x_{ja}^J = 0$ for all $a \in \delta^{\text{in}}(J \setminus \{j\})$. The following two statements are equivalent.*

1. *The pair (y, x^J) can be extended by a vector q^J such that constraints (16), (17), (18) and (20) of the MCF-J-Q-formulation are satisfied.*
2. *$x_j^J|_{A_{IL} \cup A_{LL}} = B(y)^{-1} b_j^J$ for all $j \in J$.*

We call a flow $y \in X$ *P-feasible* if there exists a vector $p \in \mathbb{R}^V$ such that (y, p, z) is a feasible solution for the P-formulation, where z is determined by (8). Similarly $y \in X$ is called *MCF-I-feasible* (resp. *MCF-J-feasible*) if there exist vectors $x^I \in \mathbb{R}^{I \times A}$ (resp. $x^J \in \mathbb{R}^{J \times A}$) and $q^I \in \mathbb{R}^{I \times V}$ (resp. $q^J \in \mathbb{R}^{J \times V}$) such that (x^I, q^I, y) (resp. (x^J, q^J, y)) is a feasible solution for the MCF-I-Q-formulation (MCF-J-Q-formulation). We are now prepared for the precise statement and the proof of the equivalence result.

Theorem 5. *For a flow vector $y \in X$ we have the equivalences*

$$y \text{ is P-feasible} \iff y \text{ is MCF-I-feasible} \iff y \text{ is MCF-J-feasible.}$$

Moreover, these equivalences are effective in the sense, that from a feasible solution for any of the formulations the corresponding solutions for the other formulations can be computed in polynomial time.

Proof. Let $y \in X$, set $F = F(y)$ and $B = B(y)$ and define vectors b_i^I for $i \in I$, b_j^J for $j \in J$ and b_k^P for $k \in K$ as described just before Lemma 1 above. We define vectors $z_k = (z_{ak})$ for $k \in K$, $x_i^I = (x_{ia}^I)$ for $i \in I$ and $x_j^J = (x_{ja}^J)$ for $j \in J$ by

$$z_k = F^{-1} b_k^P, \qquad x_i^I = F^{-1} b_i^I, \qquad x_j^J = B^{-1} b_j^J.$$

For $v \in L$ with $y_v = 0$, we put $p_{vk} = 0$ for all $k \in K$. For $v \in L$ with $y_v > 0$ we choose any $a \in \delta^{\text{out}}(v)$ with $y_a > 0$ and set $p_{vk} = z_{ak}/y_a$. Note that this does not depend on the choice of a because

$$y_v z_{ak} = y_a \sum_{a' \in \delta^{\text{in}}(v)} z_{a'k}, \qquad a \in \delta^{\text{out}}(v),$$

implies that $z_{ak} = z_{a'k}$ for all $a, a' \in \delta^{\text{out}}(v)$ with $y_a,\ y_{a'} > 0$. In view of Lemmas 2, 3 and 4, it is sufficient to check that (p, y) satisfies (7) if and only if (x^I, y) satisfies (12) if and only if (x^J, y) satisfies (19). The

first equivalence follows immediately from the observation that, for every $k \in K$, $b_k^P = \sum_{i \in I} \lambda_{ik} b_i^I$, and consequently

$$p_{ak} y_a = \sum_{i \in I} \lambda_{ik} x_{ia}^I, \qquad a \in A, \;\; k \in K.$$

For the last equivalence, we use that for all pairs $(i,j) \in I \times J$, we have

$$\sum_{a \in \delta^{\text{out}}(i)} x_{ja}^J = \sum_{a \in \delta^{\text{in}}(j)} x_{ia}^I,$$

which is proven in Lemma 6 below. This concludes the proof because we obtain

$$\sum_{i \in I} \sum_{a \in \delta^{\text{out}}(i)} \lambda_{ik} x_{ja}^J = \sum_{i \in I} \lambda_{ik} \sum_{a \in \delta^{\text{out}}(i)} x_{ja}^J = \sum_{i \in I} \lambda_{ik} \sum_{a \in \delta^{\text{in}}(j)} x_{ia}^I = \sum_{i \in I} \sum_{a \in \delta^{\text{in}}(j)} \lambda_{ik} x_{ia}^I. \qquad \square$$

Lemma 6. *Let the vectors x_i^I for $i \in I$ and x_j^J for $j \in J$ be defined as in the proof of Theorem 5. Then*

$$\sum_{a \in \delta^{\text{out}}(i)} x_{ja}^J = \sum_{a \in \delta^{\text{in}}(j)} x_{ia}^I$$

for every pair $(i,j) \in I \times J$ with $i \in I$.

Proof. For every pair $(i,j) \in I \times J$ and every $a \in A$ we set $x_{ija} = q_{ia}^I q_{ja}^J y_a$. Then the vector $x_{ij} = (x_{ija})_{a \in A}$ satisfies flow conservation in all nodes $v \in L$:

$$\begin{aligned}
\sum_{a \in \delta^{\text{in}}(v)} x_{ija} &= \sum_{a \in \delta^{\text{in}}(v)} q_{ia}^I q_{ja}^J y_a = q_{jv}^J \sum_{a \in \delta^{\text{in}}(v)} q_{ia}^I y_a = q_{jv}^J \sum_{a \in \delta^{\text{in}}(v)} x_{ia}^I \\
&= q_{jv}^J \sum_{a \in \delta^{\text{out}}(v)} x_{ia}^I = q_{jv}^J \sum_{a \in \delta^{\text{out}}(v)} q_{ia}^I y_a = q_{jv}^J q_{iv}^I \sum_{a \in \delta^{\text{out}}(v)} y_a = q_{jv}^J q_{iv}^I y_v \\
\sum_{a \in \delta^{\text{out}}(v)} x_{ija} &= \sum_{a \in \delta^{\text{out}}(v)} q_{ia}^I q_{ja}^J y_a = q_{iv}^I \sum_{a \in \delta^{\text{out}}(v)} q_{ja}^J y_a = q_{iv}^I \sum_{a \in \delta^{\text{out}}(v)} x_{ja}^J \\
&= q_{iv}^I \sum_{a \in \delta^{\text{in}}(v)} x_{ja}^J = q_{iv}^I \sum_{a \in \delta^{\text{in}}(v)} q_{ja}^J y_a = q_{iv}^I q_{jv}^J \sum_{a \in \delta^{\text{in}}(v)} y_a = q_{iv}^I q_{jv}^J y_v.
\end{aligned}$$

Clearly, the value of this flow is on the one hand

$$\sum_{a \in \delta^{\text{out}}(i)} x_{ija} = \sum_{a \in \delta^{\text{out}}(i)} q_{ia}^I q_{ja}^J y_a = \sum_{a \in \delta^{\text{out}}(i)} q_{ja}^J y_a = \sum_{a \in \delta^{\text{out}}(i)} x_{ja}^J,$$

and on the other hand

$$\sum_{a \in \delta^{\text{in}}(j)} x_{ija} = \sum_{a \in \delta^{\text{in}}(j)} q_{ia}^I q_{ja}^J y_a = \sum_{a \in \delta^{\text{in}}(j)} q_{ia}^I y_a = \sum_{a \in \delta^{\text{in}}(j)} x_{ia}^I,$$

and this concludes the proof. $\square$

Corollary 7. *All formulations $f \in \mathcal{F}_{\text{all}}$ are equivalent to the P-formulation and are therefore valid formulations of the pooling problem.*

Proof. The equivalence of the MCF-I-PQ- and MCF-J-PQ-formulations to the P-formulation follows from Theorem 5 and the fact that (14)–(15) and (21)–(22) are additional valid constraints to the corresponding

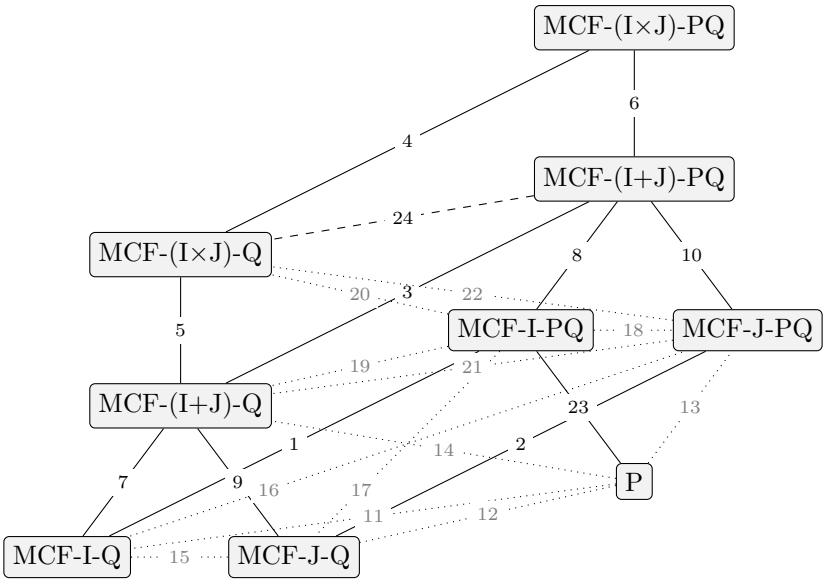


Figure 3: Hasse diagram of the partial order of formulations. All edges are directed upwards. A solid edge (f_1, f_2) indicates $f_2 \succeq f_1$. A dashed edge indicates a conjectured solid edge. Dotted edges indicate non-edges, i.e., two formulations are incomparable. The proof of Lemma 9 refers to the edge labels.

Q-formulations. The equivalence of the MCF-(I+J)-(P)Q- and the MCF-(I×J)-(P)Q-formulations to the P-formulation follows analogously. □

4.4 Strength of formulations

Note that all of the formulations in Sections 4.1–4.2 are special cases of a *bilinear program*. We can linearly relax such nonlinear problems by replacing every bilinear term with its McCormick relaxation, i.e., we replace every constraint in (8), (13), (20), (25) and (28) with the corresponding four McCormick inequalities. Given a formulation $f \in \mathcal{F}_{\text{all}}$ and an instance i, let $z[f, i]$ denote the optimal objective function value of problem $[f]$ for i and let $\underline{z}[f, i]$ denote the lower bound obtained by solving the linear relaxation of f for i. It is clear that $z[f, i]$ is identical for all nonlinear formulations $f \in \mathcal{F}_{\text{all}}$. However, this is not true for the linear relaxations of the formulations, as we show in the following.

Definition 8 (Partial order of formulations)**.** *Given two formulations f_1 and f_2, we write $f_1 \succeq f_2$ if and only if the lower bound obtained by solving the linear relaxation of f_1 is greater or equal to the lower bound obtained by solving the linear relaxation of f_2 for all instances i: $\underline{z}[f_1, i] \geq \underline{z}[f_2, i]$.*

Lemma 9. *Figure 3 shows the Hasse diagram of the partial order of formulations.*

Proof. The following item numbers refer to the edge labels in Figure 3.

1–10. Consider two formulations f_1 and f_2. As observed by other authors [8, 9, 27], if f_1 has all the constraints that f_2 has, and possibly additional constraints, then the lower bound obtained by solving the linear relaxation of f_1 is greater than or equal to the lower bound obtained by solving the linear relaxation of f_2: $f_1 \succeq f_2$. Therefore,

1–4. any PQ-formulation is at least as strong as its corresponding Q-formulation,

5–6. any (I×J)-formulation is at least as strong as its corresponding (I+J)-formulation,

7–10. any (I+J)-formulation is at least as strong as both its corresponding I- and J-formulations.

Table 4: Incomparable formulations. The following pairs of formulations (f_1, f_2) and altered gppL1 instances (i_1, i_2) prove that f_1 and f_2 are incomparable since we have $\underline{z}[f_1, i_1] > \underline{z}[f_2, i_1]$ and $\underline{z}[f_2, i_2] > \underline{z}[f_1, i_2]$ for every dotted edge in 3.

Edge	(f_1, f_2)	$c = (c_{a_1}, \ldots, c_{a_9})$ and $C = (C_1, \ldots, C_8)$ for (i_1, i_2)		$(\underline{z}[f_1, i_1] > \underline{z}[f_2, i_1])$, $(\underline{z}[f_2, i_2] > \underline{z}[f_1, i_2])$
11	f_1: P	i_1: $c = (0, -7, 4, 4, 6, 0, -7, -6, 0)$	$C = (5, 2, 6, 7, 1, 10, 8, 1)$	$-29.56 > -29.96$
	f_2: MCF-I-Q	i_2: $c = (-1, -7, -2, -9, 6, -5, -10, 4, -1)$	$C = (8, 1, 3, 8, 5, 4, 3, 4)$	$-48.00 > -48.60$
12	f_1: P	i_1: $c = (-7, 6, 7, 2, -6, -5, 1, -7, -9)$	$C = (3, 8, 6, 8, 5, 0, 7, 5)$	$-70.00 > -85.31$
	f_2: MCF-J-Q	i_2: $c = (-1, -7, -2, -9, 6, -5, -10, 4, -1)$	$C = (8, 1, 3, 8, 5, 4, 3, 4)$	$-42.60 > -48.60$
13	f_1: P	i_1: $c = (0, -7, 4, 4, 6, 0, -7, -6, 0)$	$C = (5, 2, 6, 7, 1, 10, 8, 1)$	$-29.56 > -30.20$
	f_2: MCF-J-PQ	i_2: $c = (-1, -7, -2, -9, 6, -5, -10, 4, -1)$	$C = (8, 1, 3, 8, 5, 4, 3, 4)$	$-42.60 > -48.60$
14	f_1: P	i_1: $c = (-9, -6, -7, -1, -3, -7, -3, -3, -9)$	$C = (6, 5, 5, 4, 5, 3, 2, 1)$	$-80.00 > -80.33$
	f_2: MCF-(I+J)-Q	i_2: $c = (-1, -7, -2, -9, 6, -5, -10, 4, -1)$	$C = (8, 1, 3, 8, 5, 4, 3, 4)$	$-42.60 > -48.60$
15	f_1: MCF-I-Q	i_1: $c = (-7, 6, 7, 2, -6, -5, 1, -7, -9)$	$C = (3, 8, 6, 8, 5, 0, 7, 5)$	$-70.00 > -85.31$
	f_2: MCF-J-Q	i_2: $c = (-1, -7, -2, -9, 6, -5, -10, 4, -1)$	$C = (8, 1, 3, 8, 5, 4, 3, 4)$	$-42.60 > -48.00$
16	f_1: MCF-I-Q	i_1: $c = (0, -7, 4, 4, 6, 0, -7, -6, 0)$	$C = (5, 2, 6, 7, 1, 10, 8, 1)$	$-29.96 > -30.20$
	f_2: MCF-J-PQ	i_2: $c = (-1, -7, -2, -9, 6, -5, -10, 4, -1)$	$C = (8, 1, 3, 8, 5, 4, 3, 4)$	$-42.60 > -48.00$
17	f_1: MCF-I-PQ	i_1: $c = (-7, 6, 7, 2, -6, -5, 1, -7, -9)$	$C = (3, 8, 6, 8, 5, 0, 7, 5)$	$-70.00 > -85.31$
	f_2: MCF-J-Q	i_2: $c = (-6, 9, -2, 7, -8, -3, -3, 5, -7)$	$C = (1, 5, 10, 9, 9, 8, 3, 5)$	$-85.15 > -85.75$
18	f_1: MCF-I-PQ	i_1: $c = (0, -7, 4, 4, 6, 0, -7, -6, 0)$	$C = (5, 2, 6, 7, 1, 10, 8, 1)$	$-29.56 > -30.20$
	f_2: MCF-J-PQ	i_2: $c = (5, -5, 10, -8, -5, -5, 2, 2, -3)$	$C = (1, 2, 1, 9, 8, 3, 4, 7)$	$-11.64 > -11.67$
19	f_1: MCF-I-PQ	i_1: $c = (-9, -6, -7, -1, -3, -7, -3, -3, -9)$	$C = (6, 5, 5, 4, 5, 3, 2, 1)$	$-80.00 > -80.33$
	f_2: MCF-(I+J)-Q	i_2: $c = (5, -5, 10, -8, -5, -5, 2, 2, -3)$	$C = (1, 2, 1, 9, 8, 3, 4, 7)$	$-11.64 > -11.67$
20	f_1: MCF-I-PQ	i_1: $c = (-9, -6, -7, -1, -3, -7, -3, -3, -9)$	$C = (6, 5, 5, 4, 5, 3, 2, 1)$	$-80.00 > -80.33$
	f_2: MCF-(I×J)-Q	i_2: $c = (5, -5, 10, -8, -5, -5, 2, 2, -3)$	$C = (1, 2, 1, 9, 8, 3, 4, 7)$	$-11.64 > -11.67$
21	f_1: MCF-J-PQ	i_1: $c = (-10, 8, 4, 1, -10, -2, 0, 6, 6)$	$C = (5, 1, 1, 2, 7, 2, 6, 6)$	$-12.00 > -12.83$
	f_2: MCF-(I+J)-Q	i_2: $c = (0, -7, 4, 4, 6, 0, -7, -6, 0)$	$C = (5, 2, 6, 7, 1, 10, 8, 1)$	$-29.56 > -30.20$
22	f_1: MCF-J-PQ	i_1: $c = (-10, 8, 4, 1, -10, -2, 0, 6, 6)$	$C = (5, 1, 1, 2, 7, 2, 6, 6)$	$-12.00 > -12.83$
	f_2: MCF-(I×J)-Q	i_2: $c = (0, -7, 4, 4, 6, 0, -7, -6, 0)$	$C = (5, 2, 6, 7, 1, 10, 8, 1)$	$-29.56 > -30.20$

11–22. Two formulations f_1 and f_2 are incomparable if and only if there are at least two instances i_1 and i_2 for which we have $\underline{z}[f_1, i_1] > \underline{z}[f_2, i_1]$ (i.e., f_1 is stronger than f_2 for i_1) and $\underline{z}[f_2, i_2] > \underline{z}[f_1, i_2]$ (i.e., f_2 is stronger than f_1 for i_2). Now consider the instance gppL1 [13] with the following characteristics:

$$
\begin{aligned}
&I = \{1, 2, 3\}, \;\; L = \{4, 5\}, \;\; J = \{6, 7, 8\}, \;\; K = \{1\}, \\
&A = \{\underbrace{(1,4)}_{=a_1}, \underbrace{(1,6)}_{=a_2}, \underbrace{(2,4)}_{=a_3}, \underbrace{(3,5)}_{=a_4}, \underbrace{(3,7)}_{=a_5}, \underbrace{(4,5)}_{=a_6}, \underbrace{(4,7)}_{=a_7}, \underbrace{(5,6)}_{=a_8}, \underbrace{(5,8)}_{=a_9}\}, \\
&\lambda_{11} = 3, \;\; \lambda_{21} = 1, \;\; \lambda_{31} = 2, \\
&\mu_{61}^{\min} = \mu_{71}^{\min} = \mu_{81}^{\min} = -\infty, \;\; \mu_{61}^{\max} = 2.5, \;\; \mu_{71}^{\max} = 1.75, \;\; \mu_{81}^{\max} = 1.5.
\end{aligned}
$$

By altering $c = (c_{a_1}, \ldots, c_{a_9})$ and $C = (C_1, \ldots, C_8)$, and letting $u_a = \min\{C_v, C_w\}$ for all $a = (v, w) \in A$, the instances shown in Table 4 prove that the formulations whose corresponding nodes in Figure 3 are connected by dotted edges are incomparable.

23. See [8], Proposition 2.

24. Our computational results suggest MCF-(I+J)-PQ $\succeq$ MCF-(I×J)-Q. □

5 Computational results

5.1 Instances

We tested our formulations on 34 standard and 40 generalized pooling problem instances from the literature. The instances can be downloaded from [5, 6] in GAMS [17] (gams.com) format. The SPP instances are composed of instances from different sources in the literature and randomly generated instances from [9, 10]. The literature instances are from [2] (Adhya1–4), [16] (Bental4–5), [24] (Foulds2–5), [30] (Haverly1–3), and [13] (RT2). The random instances (prefix *spp*) can be divided into three groups (infix *A*–*C*) where instances in the same group have the same number of inputs, pools, outputs and qualities, but a different number of arcs. A larger suffix number indicates a larger number of arcs. Details on the SPP instances are found in [9, 10]. The GPP instances (prefix *gpp*) are composed of extensions of literature SPP instances and randomly generated instances, both from [8]. Among the literature instances (infix *L*), gppL1 is from [13], gppL2–5 are extensions of Adhya1–4, gppL6–7 of Bental4–5, gppL8–11 of Foulds2–5, gppL12–14 of Haverly1–3, and gppL15 is an extension of RT2. The random instances can be divided into five groups (infix *A*–*E*) where again instances in the same group have the same number of inputs, pools, outputs and qualities, but a different number of arcs, and a larger suffix number indicates a larger number of arcs. Details on the GPP instances are found in [8]. For notational convenience, we define the following sets of pooling problem instances:

$$\begin{aligned}
\mathcal{I}_{\text{SPP,L}} &:= \{\text{Adhya1–4, Bental4–5, Foulds2–5, Haverly1–3, RT2}\}, && \text{set of literature SPP instances,}\\
\mathcal{I}_{\text{SPP,R}} &:= \{\text{sppA0–9, sppB0–5, sppC0–3}\}, && \text{set of random SPP instances,}\\
\mathcal{I}_{\text{SPP}} &:= \mathcal{I}_{\text{SPP,L}} \cup \mathcal{I}_{\text{SPP,R}}, && \text{set of SPP instances,}\\
\mathcal{I}_{\text{GPP,L}} &:= \{\text{gppL1–15}\}, && \text{set of literature GPP instances,}\\
\mathcal{I}_{\text{GPP,R}} &:= \{\text{gppA1–5, gppB1–5, gppC1–5, gppD1–5, gppE1–5}\}, && \text{set of random GPP instances,}\\
\mathcal{I}_{\text{GPP}} &:= \mathcal{I}_{\text{SPP,L}} \cup \mathcal{I}_{\text{SPP,R}}, && \text{set of GPP instances,}\\
\mathcal{I}_{\text{all}} &:= \mathcal{I}_{\text{SPP}} \cup \mathcal{I}_{\text{GPP}}, && \text{set of all instances.}
\end{aligned}$$

Some GPP instances from the literature violate our assumptions about non-contributing nodes. More specifically, gppA2 contains an isolated output (output 8 with $\delta^{\text{in}}(8) = \emptyset$) and gppC1 contains a pool which is a sink vertex (pool 14 with $\delta^{\text{out}}(14) = \emptyset$). We modify these instances from the literature as follows so that they satisfy our assumption. We remove all inputs and pools whose outdegree is zero and all pools and outputs whose indegree is zero. In the case of pools whose indegree or outdegree is zero, but not both, we also remove all outgoing or incoming arcs to such source or sink vertices. This process is iterated until no more vertices or arcs are removed. Tables 5–6 summarise characteristics of the SPP and GPP instances from the literature. Modified instances and characteristics are *emphasized*.

5.2 Implementation

We converted the 74 pooling problem instances in [5, 6] from GAMS to AMPL [25] (ampl.com) format. We used AMPL to model the different formulations and solved every instance for every formulation. We solved all relaxed linear programs with CPLEX 12.6.0.0 [31] (cplex.com) and all nonlinear programs with SCIP 3.0.0 [1] (scip.zib.de). We linked SCIP to CPLEX as the LP solver and to Ipopt 3.10 [32] (coin-or.org/Ipopt) as the NLP solver. It is worth noting that while Ipopt only guarantees local optimality,

Table 5: Characteristics of standard pooling problem instances from the literature

(a) Literature instances

	# vertices				# arcs		
Instance	$\|I\|$	$\|L\|$	$\|J\|$	$\|K\|$	$\|A_{IL}\|$	$\|A_{LJ}\|$	$\|A_{IJ}\|$
Adhya1	5	2	4	4	5	8	0
Adhya2	5	2	4	6	5	8	0
Adhya3	8	3	4	6	8	12	0
Adhya4	8	2	5	4	8	10	0
Bental4	4	1	2	1	3	2	2
Bental5	5	3	5	2	12	15	5
Foulds2	6	2	4	1	4	8	8
Foulds3	11	8	16	1	32	128	0
Foulds4	11	8	16	1	32	128	0
Foulds5	11	4	16	1	32	64	0
Haverly1	3	1	2	1	2	2	2
Haverly2	3	1	2	1	2	2	2
Haverly3	3	1	2	1	2	2	2
RT2	3	2	3	8	6	6	4

(b) Randomly generated instances

	# vertices				# arcs		
Instance	$\|I\|$	$\|L\|$	$\|J\|$	$\|K\|$	$\|A_{IL}\|$	$\|A_{LJ}\|$	$\|A_{IJ}\|$
sppA0	20	10	15	24	62	54	55
sppA1					78	48	53
sppA2					82	58	52
sppA3					83	64	71
sppA4					111	66	71
sppA5					110	86	81
sppA6					120	89	72
sppA7					135	111	79
sppA8					151	101	113
sppA9					164	122	121
sppB0	35	17	21	34	174	115	95
sppB1					250	126	139
sppB2					286	183	177
sppB3					378	203	209
sppB4					428	262	253
sppB5					481	281	282
sppC0	60	30	40	40	362	234	215
sppC1					461	312	297
sppC2					529	386	363
sppC3					639	429	383

SCIP guarantees global optimality. All our computations were carried out on a Dell PowerEdge R710 with dual hex core 3.06GHz Intel Xeon X5675 processors and 96GB RAM, running Red Hat Enterprise Linux 6 and using a single thread.

5.2.1 Relaxed linear programs

We run the relaxed linear programs without a stopping criterion since the LPs generally solve fast and we are more interested in the strength of formulations rather than the solve times. We relax the bilinear constraints, shown in Table 7, and use the following lower and upper bounds on their participating variables p, q^I, q^J, q^{IJ} and y:

$$
\begin{aligned}
[\underline{p}_{ak}, \overline{p}_{ak}] &= [\min\{\lambda_{ik} \mid i \in I\}, \max\{\lambda_{ik} \mid i \in I\}], && a \in A,\ k \in K,\\
[0,1] &= \begin{cases} [\underline{q}^I_{ia}, \overline{q}^I_{ia}], & i \in I,\ a \in A,\\ [\underline{q}^J_{ja}, \overline{q}^J_{ja}], & j \in J,\ a \in A,\\ [\underline{q}^{IJ}_{ija}, \overline{q}^{IJ}_{ija}], & i \in I,\ j \in J,\ a \in A, \end{cases}\\
[\underline{y}_a, \overline{y}_a] &= [0, u_a], && a \in A.
\end{aligned}
$$

Note that z, x^I, x^J and x^{IJ} are used as auxiliary variables to outsource all bilinear terms into single constraints and are bounded by the corresponding four McCormick inequalities.

Table 6: Characteristics of generalized pooling problem instances from the literature

(a) Literature instances

	# vertices				# arcs			
Instance	$\|I\|$	$\|L\|$	$\|J\|$	$\|K\|$	$\|A_{IL}\|$	$\|A_{LL}\|$	$\|A_{LJ}\|$	$\|A_{IJ}\|$
gppL1	3	2	3	1	3	1	3	2
gppL2	5	2	4	4	5	2	8	0
gppL3	5	2	4	6	5	2	8	0
gppL4	8	3	4	6	8	6	12	0
gppL5	8	2	5	4	8	2	10	0
gppL6	4	2	2	1	4	2	4	0
gppL7	5	3	5	2	12	6	15	5
gppL8	6	2	4	1	4	2	8	8
gppL9	11	8	16	1	32	56	128	0
gppL10	11	8	16	1	32	56	128	0
gppL11	11	4	16	1	32	12	64	0
gppL12	3	2	2	1	3	2	4	0
gppL13	3	2	2	1	3	2	4	0
gppL14	3	2	2	1	3	2	4	0
gppL15	3	2	3	8	6	2	6	4

(b) Randomly generated instances

	# vertices				# arcs			
Instance	$\|I\|$	$\|L\|$	$\|J\|$	$\|K\|$	$\|A_{IL}\|$	$\|A_{LL}\|$	$\|A_{LJ}\|$	$\|A_{IJ}\|$
gppA1	3	2	3	2	6	1	5	3
gppA2			*2*		5	1	3	4
gppA3			3		6	1	3	2
gppA4			3		5	1	6	5
gppA5			3		5	1	6	6
gppB1	5	4	3	3	14	3	7	3
gppB2					15	4	9	9
gppB3					16	6	10	6
gppB4					19	6	9	6
gppB5					19	6	10	9
gppC1	8	*5*	6	4	*19*	*4*	15	13
gppC2		6			29	9	20	13
gppC3		6			27	11	25	12
gppC4		6			32	10	25	15
gppC5		6			28	13	27	14
gppD1	12	10	8	5	46	21	32	15
gppD2					66	21	33	22
gppD3					56	20	38	24
gppD4					64	26	43	33
gppD5					69	27	47	23
gppE1	10	10	15	12	40	39	67	35
gppE2					34	38	63	33
gppE3					56	56	66	41
gppE4					40	55	88	38
gppE5					61	62	86	39

Table 7: Problems, bilinear constraints and their participating variables

Problem	[P]	[MCF-I-(P)Q]	[MCF-J-(P)Q]	[MCF-(I+J)-(P)Q]	[MCF-(I×J)-(P)Q]
Constraints	(8)	(13)	(20)	(13), (20)	(13), (20), (25), (28)
Variables	(p, y)	(q^I, y)	(q^J, y)	$(q^I, y), (q^J, y)$	$(q^I, y), (q^J, y), (q^I, q^J), (q^{IJ}, y)$

5.2.2 Nonlinear programs

The nonlinear programs run with a time limit (`time_limit`) of 600 seconds and the default stopping criteria of SCIP (http://scip.zib.de/doc/html/PARAMETERS.php). All of the times that we report are total solve times representing the sum of system and user CPU seconds used by all solve commands (using AMPL's built-in timing parameter `_total_solve_time` and reading the SCIP log files). We do not report build times (i.e., the time it takes AMPL to build the model) since once a model is built, it is reused for different formulations and instances. If an instance cannot be solved to global optimality within `time_limit`, we report the lower and upper bounds that are found in the SCIP log files.

5.3 Measures

Throughout this section we will use different measures to compare the size and strength of formulations, as well as their computational performance. We can distinguish between *minimization* and *maximization measures*. For a minimization measure, we compare the value of a measure to the minimum of all values across all formulations. In this paper, we consider

$$\mathcal{M}_{\min} := \{\texttt{nbils}, \texttt{LP\_nvars}, \texttt{NLP\_nvars}, \texttt{LP\_ncons}, \texttt{NLP\_ncons}, \texttt{LP\_nnonzeros}, \texttt{NLP\_ub}, \texttt{NLP\_gap}, \texttt{LP\_time}, \texttt{NLP\_time}\},$$

where

- `nbils` is the number of bilinear terms,
- `LP_nvars`/`NLP_nvars` the number of variables in the LP/NLP,
- `LP_ncons`/`NLP_ncons` the number of constraints in the LP/NLP,
- `LP_nnonzeros` the number of nonzeros in the LP constraint matrix,
- `NLP_ub` the NLP upper bound,
- `NLP_gap` the difference between the NLP upper and lower bound,
- `LP_time`/`NLP_time` the total solve time of the LP/NLP.

Analogously, for a maximization measure, we compare the value of a measure to the maximum of all values across all formulations. We consider

$$\mathcal{M}_{\max} := \{\texttt{LP\_obj}, \texttt{NLP\_lb}\},$$

where

- `LP_obj` is the LP objective (an NLP lower bound),
- `NLP_lb` is the NLP lower bound.

A measure is specific to a formulation $f \in \mathcal{F}_{\text{all}}$ and an instance $i \in \mathcal{I}_{\text{all}}$, and we denote the value of a measure $m \in \mathcal{M}_{\text{all}} := \mathcal{M}_{\min} \cup \mathcal{M}_{\max}$ by $v_{m,f,i}$.

5.4 Criteria

For each of the following criteria, let $\mathcal{M} \subseteq \mathcal{M}_{\text{all}}$ be a subset of all measures, $\mathcal{F} \subseteq \mathcal{F}_{\text{all}}$ a subset of all formulations, and $\mathcal{I} \subseteq \mathcal{I}_{\text{all}}$ a subset of all instances. The mean of all values of a measure $m \in \mathcal{M}$ for a

Table 8: Size of formulations. $\bar{v}_{m,f}$ for $m \in \{$`nbils`, `LP_nvars`, `LP_ncons`, `LP_nnonzeros`, `NLP_nvars`, `NLP_ncons`$\}$ and $f \in \mathcal{F}_{\text{all}}$ across all instances $i \in \mathcal{I}_{\text{all}}$.

$\bar{v}_{m,f}$	f								
		MCF-I-		MCF-J-		MCF-(I+J)-		MCF-(I×J)-	
m	P	Q	PQ	Q	PQ	Q	PQ	Q	PQ
`nbils`	5617	1158		1127		2285		7918	
`LP_nvars`	11042	7006		6900		9490		23729	
`LP_ncons`	24084	6784	7084	6564	6844	12956	13536	48357	51960
`LP_nnonzeros`	26401	40155	42263	39865	41195	79440	82980	127230	100085
`NLP_nvars`	5425	5849		5773		7205		15811	
`NLP_ncons`	1616	2154	2453	2055	2336	3817	4397	16685	20288

formulation $f \in \mathcal{F}$ across all instances $i \in \mathcal{I}$ is

$$\bar{v}_{m,f} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} v_{m,f,i}.$$

In [20], Dolan and Moré proposed *performance profiles* as a tool for benchmarking and comparing optimization software. Traditionally, computing time is used as a performance measure to compare a set of solvers for a set of problems. However, the ideas in [20] can be used with other measures. As a baseline for comparisons, we use the *performance ratio*

$$r_{m,f,i} = \begin{cases} \frac{v_{m,f,i}}{\min\limits_{f' \in \mathcal{F}} v_{m,f',i}}, & \text{if } m \in \mathcal{M}_{\min} \text{ and } \min\limits_{f' \in \mathcal{F}} v_{m,f',i} > 0 \\ \frac{\min\limits_{f' \in \mathcal{F}} v_{m,f',i}}{v_{m,f,i}}, & \text{if } m \in \mathcal{M}_{\min} \text{ and } \max\limits_{f' \in \mathcal{F}} v_{m,f',i} < 0 \\ \frac{\max\limits_{f' \in \mathcal{F}} v_{m,f',i}}{v_{m,f,i}}, & \text{if } m \in \mathcal{M}_{\max} \text{ and } \min\limits_{f' \in \mathcal{F}} v_{m,f',i} > 0 \\ \frac{v_{m,f,i}}{\max\limits_{f' \in \mathcal{F}} v_{m,f',i}}, & \text{if } m \in \mathcal{M}_{\max} \text{ and } \max\limits_{f' \in \mathcal{F}} v_{m,f',i} < 0 \end{cases},$$

where $m \in \mathcal{M}$, $f \in \mathcal{F}$ and $i \in \mathcal{I}$. We can assume that either $\min_{f \in \mathcal{F}} v_{m,f,i} > 0$ or $\max_{f \in \mathcal{F}} v_{m,f,i} < 0$ for all $m \in \mathcal{M}_{\text{all}}$. The mean of a performance ratio is

$$\bar{r}_{m,f} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} r_{m,f,i},$$

where $m \in \mathcal{M}$ and $f \in \mathcal{F}$. The cumulative distribution function for a performance ratio is

$$\rho_{m,f}(\tau) = \begin{cases} \frac{1}{|\mathcal{I}|} |\{i \in \mathcal{I} : r_{m,f,i} \leq \tau\}|, & \text{if } m \in \mathcal{M}_{\min} \\ 1 - \frac{1}{|\mathcal{I}|} |\{i \in \mathcal{I} : r_{m,f,i} > \tau\}|, & \text{if } m \in \mathcal{M}_{\max} \end{cases},$$

where $m \in \mathcal{M}$, $f \in \mathcal{F}$ and $\tau \in \mathbb{R}$. If we draw an instance uniformly at random from $\mathcal{I}$ then $\rho_{m,f}(\tau)$ can be interpreted as the probability that $v_{m,f,i}$ is within a factor τ of the best possible value of the measure m across all formulations $f \in \mathcal{F}$. A performance profile for a triple $(m, \mathcal{F}, \mathcal{I})$ is a multiplot of the functions $\rho_{m,f}(\tau)$ for $f \in \mathcal{F}$.

Table 9: Size of formulations. $\bar{r}_{m,f}$ for $m \in \{$`nbils`, `LP_nvars`, `LP_ncons`, `LP_nnonzeros`, `NLP_nvars`, `NLP_ncons`$\}$ and $f \in \mathcal{F}_{\text{all}}$ across all instances $i \in \mathcal{I}_{\text{all}}$.

$\bar{r}_{m,f}$	f								
		MCF-I-		MCF-J-		MCF-(I+J)-		MCF-(I×J)-	
m	P	Q	PQ	Q	PQ	Q	PQ	Q	PQ
`nbils`	2.93	1.67		1.50		3.17		12.27	
`LP_nvars`	1.31	1.10		1.08		1.42		3.01	
`LP_ncons`	2.18	1.59	1.69	1.46	1.56	2.88	3.08	11.58	12.55
`LP_nnonzeros`	1.12	1.67	1.95	1.51	1.70	3.09	3.56	8.40	9.49
`NLP_nvars`	1.03	1.11		1.09		1.33		2.47	
`NLP_ncons`	1.05	1.78	2.07	1.69	1.98	3.02	3.60	13.63	16.78

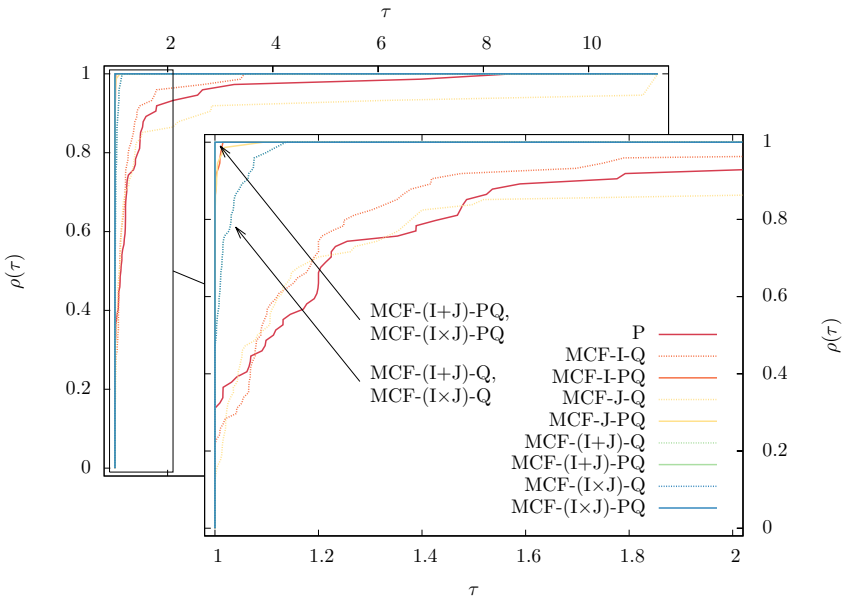


Figure 10: Performance profile ($\texttt{LP\_obj}, \mathcal{F}_{\text{all}}, \mathcal{I}_{\text{all}}$)

5.5 Size of formulations

Tables 8–9 show the size of formulations using the criteria $\bar{v}_{m,f}$ and $\bar{r}_{m,f}$ for $m \in \{$`nbils`, `LP_nvars`, `LP_ncons`, `LP_nnonzeros`, `NLP_nvars`, `NLP_ncons`$\}$ and $f \in \mathcal{F}_{\text{all}}$ across all instances $i \in \mathcal{I}_{\text{all}}$. Not surprisingly, the smallest formulations are MCF-J-(P)Q, MCF-I-(P)Q and P, followed by MCF-(I+J)-(P)Q. The largest formulations are MCF-(I×J)-(P)Q, in particular regarding the number of constraints and the number of bilinear terms.

5.6 Results on the relaxed linear programs

Figure 10 shows the performance profile ($\texttt{LP\_obj}, \mathcal{F}_{\text{all}}, \mathcal{I}_{\text{all}}$). Our computational results confirm the partial order of formulations in Lemma 9. The PQ-formulations are at least as strong as the corresponding Q-formulations. Recall that the Q-formulations are incomparable to the P-formulation. We also see that the cumulative distribution functions for the PQ-formulations mostly overlap, indicating that MCF-(I+J)-PQ and MCF-(I×J)-PQ only marginally improve the strength of MCF-I-PQ and MCF-J-PQ. Among the Q-formulations, MCF-(I+J)-Q and MCF-(I×J)-Q are equally strong, and MCF-I-Q and MCF-J-Q provide the smallest lower bounds (see the detailed computational results in Tables 15 and 17). While

Table 11: Number of instances that could be solved within `time_limit`. $|\mathcal{I}_<(\mathcal{I}, \{f\})|$ for all $\mathcal{I} \in \{\mathcal{I}_{\text{all}}, \mathcal{I}_{\text{SPP}}, \mathcal{I}_{\text{SPP,L}}, \mathcal{I}_{\text{SPP,R}}, \mathcal{I}_{\text{GPP}}, \mathcal{I}_{\text{GPP,L}}, \mathcal{I}_{\text{GPP,R}}\}$ and $f \in \mathcal{F}_{\text{all}}$.

$\lvert\mathcal{I}_<(\mathcal{I}, \{f\})\rvert$	f									
	MCF-I-			MCF-J-		MCF-(I+J)-		MCF-(I×J)-	
$\mathcal{I}$	P	Q	PQ	Q	PQ	Q	PQ	Q	PQ
$\mathcal{I}_{\text{all}}$	14 (19%)	31 (42%)	45 (61%)	29 (39%)	52 (70%)	37 (50%)	41 (55%)	31 (42%)	35 (47%)
$\mathcal{I}_{\text{SPP}}$	10 (29%)	14 (41%)	14 (41%)	10 (29%)	15 (44%)	14 (41%)	14 (41%)	12 (35%)	14 (41%)
$\mathcal{I}_{\text{SPP,L}}$	10 (71%)	14 (100%)	14 (100%)	10 (71%)	14 (100%)	14 (100%)	13 (93%)	12 (86%)	14 (100%)
$\mathcal{I}_{\text{SPP,R}}$	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (5%)	0 (0%)	1 (5%)	0 (0%)	0 (0%)
$\mathcal{I}_{\text{GPP}}$	4 (10%)	17 (43%)	31 (78%)	19 (48%)	37 (93%)	23 (58%)	27 (68%)	19 (48%)	21 (53%)
$\mathcal{I}_{\text{GPP,L}}$	2 (13%)	11 (73%)	12 (80%)	10 (67%)	15 (100%)	10 (67%)	9 (60%)	9 (60%)	9 (60%)
$\mathcal{I}_{\text{GPP,R}}$	2 (8%)	6 (24%)	19 (76%)	9 (36%)	22 (88%)	13 (52%)	18 (72%)	10 (40%)	12 (48%)

these tables may suggest that the MCF-(I×J)-(P)Q-formulations do not strengthen the corresponding MCF-(I+J)-(P)Q-formulations at all, they do for a few instances. However, the difference in strength is marginal and we report any $r_{m,f,i} < 1.005$ as $r_{m,f,i} \approx 1$.

5.7 Results on the nonlinear programs

Let $\mathcal{I} \subseteq \mathcal{I}_{\text{all}}$ be a subset of all instances and $\mathcal{F} \subseteq \mathcal{F}_{\text{all}}$ a subset of all formulations. In this section, we distinguish between instances $i \in \mathcal{I}$ that could be solved within `time_limit` for all formulations $f \in \mathcal{F}$, and instances that could not:

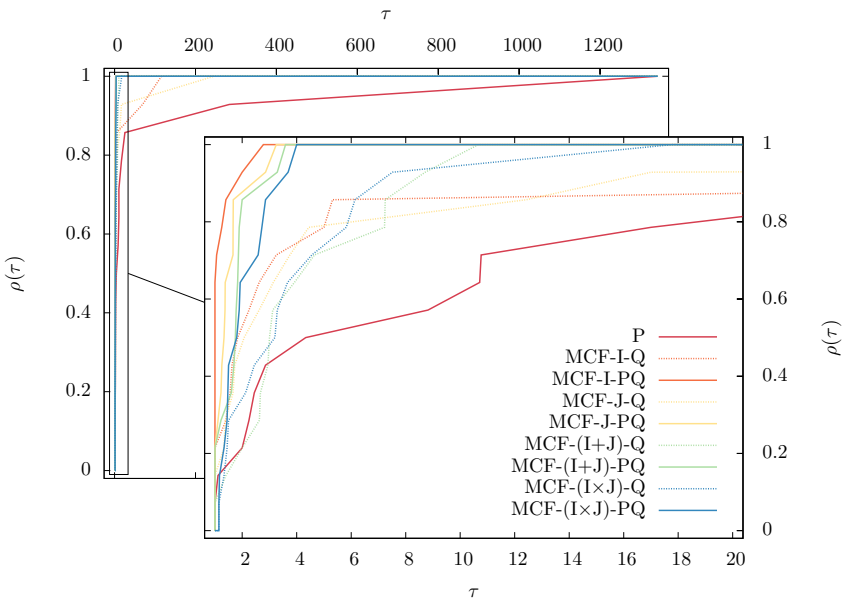
$$\mathcal{I}_<(\mathcal{I}, \mathcal{F}) := \{i \in \mathcal{I} : \max_{f \in \mathcal{F}} v_{\texttt{NLP\_time},f,i} < \texttt{time\_limit}\},$$

$$\mathcal{I}_\geq(\mathcal{I}, \mathcal{F}) := \{i \in \mathcal{I} : \min_{f \in \mathcal{F}} v_{\texttt{NLP\_time},f,i} \geq \texttt{time\_limit}\}.$$

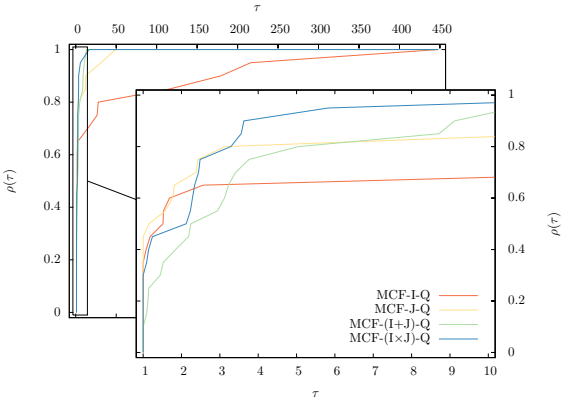
Note that, in general, the union of $\mathcal{I}_<(\mathcal{I}, \mathcal{F})$ and $\mathcal{I}_\geq(\mathcal{I}, \mathcal{F})$ is not equal to $\mathcal{I}$, since there may be instances $i \in \mathcal{I}$ that can be solved within `time_limit` for some, but not all formulations $f \in \mathcal{F}$.

Table 11 shows the number of instances that could be solved within `time_limit`. We see that MCF-J-PQ solves the most instances (52), followed by MCF-I-PQ (45) and MCF-(I+J)-PQ (41). When considering instances in $\mathcal{I}_<(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}})$ (i.e., instances that could be solved within `time_limit` across all formulations), we are interested in how `NLP_time` for a particular formulation compares to all other formulations. This is shown in the performance profiles in Figure 12. There are 14 instances that could be solved within `time_limit` across all formulations. For these instances, MCF-I-PQ has the best solve times, followed by the other PQ-formulations and finally the Q- and P-formulations. The P-formulation has the worst solve times. However, the sample size of only 14 instances provides little room for meaningful interpretation. If we consider only the instances that could be solved across all Q-formulations (excluding P), the sample size increases to 29. MCF-(I×J)-Q, MCF-(I+J)-Q and MCF-J-Q have similar solve times, only MCF-I-Q is considerably slower than the other Q-formulations. If we turn our attention to the instances that could be solved across all PQ-formulations (excluding P), the sample size increases to 35. MCF-J-PQ is faster than MCF-I-PQ, and MCF-(I×J)-PQ and MCF-(I+J)-PQ are the slowest formulations. We can conclude from the more meaningful last two performance profiles that MCF-J-PQ wins if we are interested in `NLP_time`.

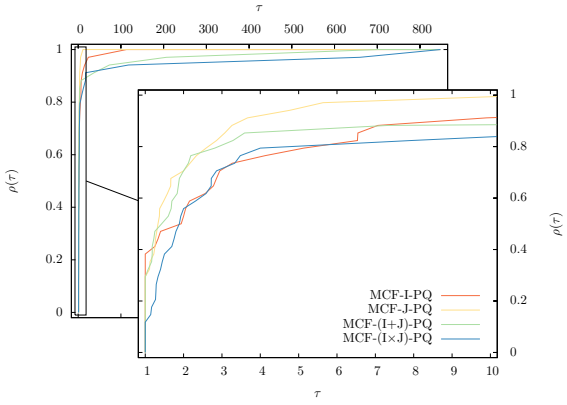
Finally, when considering instances in $\mathcal{I}_\geq(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}})$ (i.e., instances that could not be solved within `time_limit` across all formulations), we are interested in how `NLP_gap` for a particular formulation compares to all other formulations. This is shown in Figure 13, where the sample size is 22. The smallest gaps are provided by MCF-I-PQ and MCF-J-PQ, followed by their corresponding Q-formulations and the



(a) $(\texttt{NLP\_time}, \mathcal{F}_{\text{all}}, \mathcal{I}_<(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}}))$, $|\mathcal{I}_<(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}})| = 14$



(b) $(\texttt{NLP\_time}, \mathcal{F}_{\text{Q}}, \mathcal{I}_<(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{Q}}))$, $|\mathcal{I}_<(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{Q}})| = 29$



(c) $(\texttt{NLP\_time}, \mathcal{F}_{\text{PQ}}, \mathcal{I}_<(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{PQ}}))$, $|\mathcal{I}_<(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{PQ}})| = 35$

Figure 12: Performance profiles for `NLP_time`

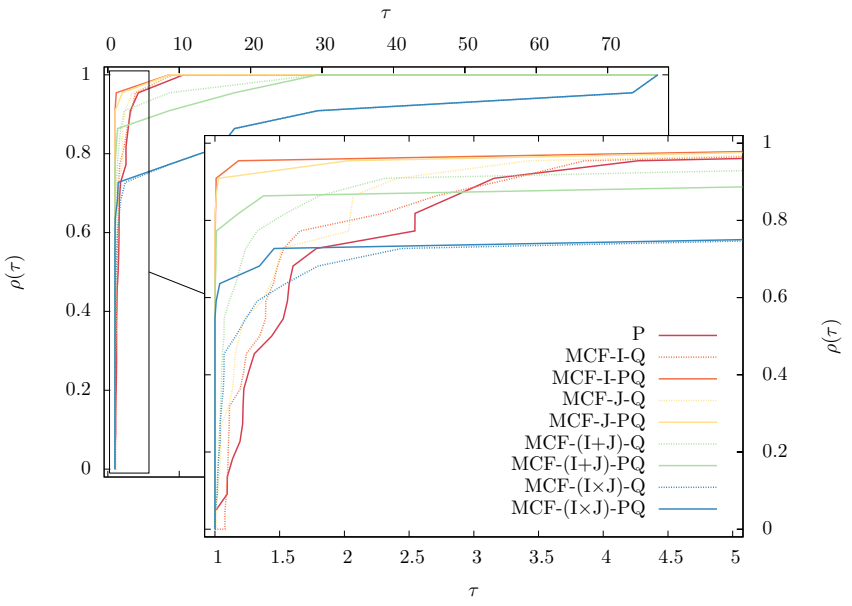


Figure 13: Performance profile $(\texttt{NLP\_gap}, \mathcal{F}_{\text{all}}, \mathcal{I}_{\geq}(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}}))$, $|\mathcal{I}_{\geq}(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}})| = 22$

P-formulation. It is well worth noting that the larger formulations MCF-(I+J)-(P)Q and MCF-(I×J)-(P)Q perform poorly, providing only large gaps. Especially for instances in $\mathcal{I}_{\text{SPP,R}}$ and $\mathcal{I}_{\text{GPP,R}}$, these formulations struggle to find upper bounds (see the detailed computational results in Tables 16, 18 and 19). The small gaps for MCF-J-PQ are more so due to the lower bounds than to the upper bounds.

6 Conclusion and outlook

6.1 Conclusion

In this paper, we introduced new multi-commodity flow formulations for the pooling problem. Inspired by [8], where Alfaki and Haugland introduced a formulation based on input commodities, we generalized their ideas and proposed new multi-commodity flow formulations based on output, input and output and (input, output)-commodities. We proved the equivalence of formulations and (almost) completely characterized the partial order of formulations with respect to the strength of their LP relaxations. In an extensive computational study on 34 standard and 40 generalized pooling problem instances, we evaluated the performance of the new formulations. We saw that there is a trade-off between disaggregating commodities and therefore increasing the size of formulations (e.g. in terms of the number of bilinear terms, variables and constraints) versus strengthening the lower bounds obtained by solving the relaxed linear programs and improving the solve times or gaps of the nonlinear programs. We saw that the PQ-formulations outperform the Q-formulations, with MCF-J-PQ being slightly better than MCF-I-PQ. But we also saw that MCF-(I+J)-PQ and MCF-(I×J)-PQ only marginally strengthen the lower bounds, often showing some of the worst solve times and gaps. These observations are summarised in Figure 14. Here we define the size of a formulation as

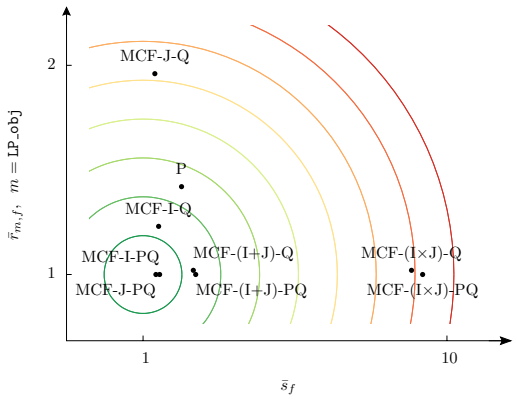
$$\bar{s}_f = \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \bar{r}_{m,f},$$

where $f \in \mathcal{F} \subseteq \mathcal{F}_{\text{all}}$, $\mathcal{M} = \mathcal{M}_{\text{size,LP}} := \{\texttt{nbils}, \texttt{LP\_nvars}, \texttt{LP\_ncons}\}$ if we consider the relaxed linear program and $\mathcal{M} = \mathcal{M}_{\text{size,NLP}} := \{\texttt{nbils}, \texttt{NLP\_nvars}, \texttt{NLP\_ncons}\}$ if we consider the nonlinear program. The closer a formulation is to $(1, 1)$ the better. Formulations in green circles performed well, and formulations in red circles performed poorly. We can conclude that MCF-J-PQ and MCF-I-PQ are the best formulations, with MCF-J-PQ being slightly better than MCF-I-PQ.

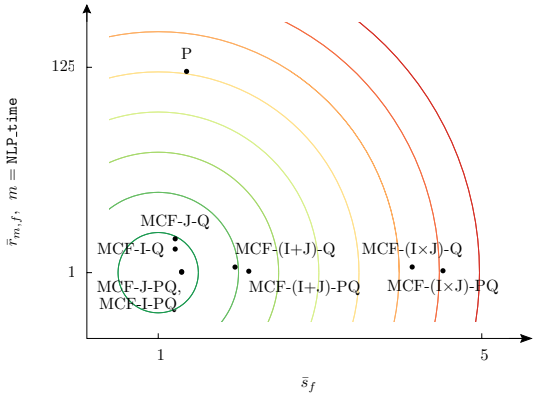
6.2 Outlook: Path commodities

Taking the approach of disaggregating variables to an extreme, we also implemented a multi-commodity flow formulation based on path commodities. However, we could not find any computational advantages of such a formulation. In fact, problem instances with a large number of vertices and arcs could not be modelled using this formulation since enumerating all simple directed paths is (in the worst case) exponential in the number of vertices. For this reason, we refrain from stating the formulation and discussing the formulation's computational results.

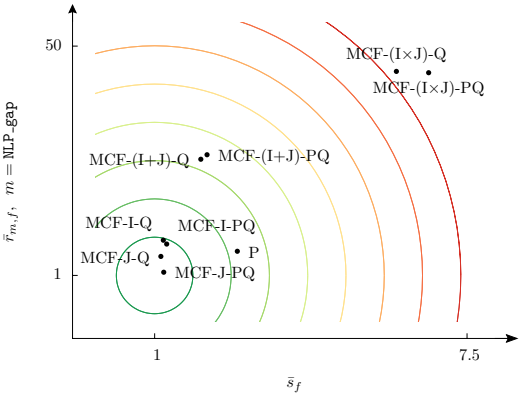
Acknowledgements This research was supported by the ARC Linkage Grant no. LP110200524, Hunter Valley Coal Chain Coordinator (hvccc.com.au) and Triple Point Technology (tpt.com). The authors would like to thank Dr Hamish Waterer for his contributions, both computationally and theoretically, to this research.



(a) `LP_obj`, $\mathcal{M} = \mathcal{M}_{\text{size,LP}}$, $\mathcal{I} = \mathcal{I}_{\text{all}}$



(b) `NLP_time`, $\mathcal{M} = \mathcal{M}_{\text{size,NLP}}$, $\mathcal{I} = \mathcal{I}_{<}(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}})$



(c) `NLP_gap`, $\mathcal{M} = \mathcal{M}_{\text{size,NLP}}$, $\mathcal{I} = \mathcal{I}_{\geq}(\mathcal{I}_{\text{all}}, \mathcal{F}_{\text{all}})$

Figure 14: Size of formulation versus mean of performance ratio for $m \in \{\texttt{LP\_obj}, \texttt{NLP\_time}, \texttt{NLP\_gap}\}$ across all formulations $f \in \mathcal{F} = \mathcal{F}_{\text{all}}$. The sets $\mathcal{M}$ and $\mathcal{I}$, which are required to calculate $\bar{s}$ and $\bar{r}$, are specified in the captions.

References

[1] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2009. `http://nbn-resolving.de/urn:nbn:de:0297-zib-11129`.

[2] N. Adhya, M. Tawarmalani, and N. V. Sahinidis. A Lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research*, 38(5):1956–1972, 1999.

[3] F. A. Al-Khayyal. Jointly constrained bilinear programs and related problems: An overview. *Computers & Mathematics with Applications*, 19(11):53–62, 1990.

[4] F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[5] M. Alfaki. Generalized pooling problem instances. `http://www.ii.uib.no/~mohammeda/gpooling/`. Accessed: June 11, 2015.

[6] M. Alfaki. Standard pooling problem instances. `http://www.ii.uib.no/~mohammeda/spooling/`. Accessed: June 11, 2015.

[7] M. Alfaki. *Models and Solution Methods for the Pooling Problem*. PhD thesis, The University of Bergen, 2012. `http://hdl.handle.net/1956/5847`.

[8] M. Alfaki and D. Haugland. A multi-commodity flow formulation for the generalized pooling problem. *Journal of Global Optimization*, 56(3):917–937, 2013.

[9] M. Alfaki and D. Haugland. Strong formulations for the pooling problem. *Journal of Global Optimization*, 56(3):897–916, 2013.

[10] M. Alfaki and D. Haugland. A cost minimization heuristic for the pooling problem. *Annals of Operations Research*, 222(1):73–87, 2014.

[11] H. Almutairi and S. Elhedhli. A new Lagrangean approach to the pooling problem. *Journal of Global Optimization*, 45(2):237–257, 2009.

[12] I. P. Androulakis, C. D. Maranas, and C. A. Floudas. αBB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995.

[13] C. Audet, J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenović. Pooling problem: Alternate formulations and solution methods. *Management Science*, 50(6):761–776, 2004.

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

[15] T. E. Baker and L. S. Lasdon. Successive linear programming at Exxon. *Management Science*, 31(3):264–274, 1985.

[16] A. Ben-Tal, G. Eiger, and V. Gershovitz. Global minimization by reducing the duality gap. *Mathematical Programming*, 63(1–3):193–212, 1994.

[17] A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. *GAMS – A User's Guide*, 2015.

[18] P. M. Castro. Tightening piecewise McCormick relaxations for bilinear problems. *Computers & Chemical Engineering*, 72(0):300–311, 2015.

[19] S. S. Dey and A. Gupte. Analysis of MILP techniques for the pooling problem. *Operations Research*, 63(2):412–427, 2015.

[20] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[21] C. A. Floudas, A. Aggarwal, and A. R. Ciric. Global optimum search for nonconvex NLP and MINLP problems. *Computers & Chemical Engineering*, 13(10):1117–1132, 1989.

[22] C. A. Floudas and V. Visweswaran. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs – I. Theory. *Computers & Chemical Engineering*, 14(12):1397–1417, 1990.

[23] C. A. Floudas and V. Visweswaran. Primal-relaxed dual global optimization approach. *Journal of Optimization Theory and Applications*, 78(2):187–225, 1993.

[24] L. R. Foulds, D. Haugland, and K. Jörnsten. A bilinear approach to the pooling problem. *Optimization*, 24(1–2):165–180, 1992.

[25] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*, 2$^{\text{nd}}$ edition, 2003.

[26] C. E. Gounaris, R. Misener, and C. A. Floudas. Computational comparison of piecewise-linear relaxations for pooling problems. *Industrial & Engineering Chemistry Research*, 48(12):5742–5766, 2009.

[27] A. Gupte. *Mixed integer bilinear programming with applications to the pooling problem.* PhD thesis, Georgia Institute of Technology, 2012. `http://hdl.handle.net/1853/45761/`.

[28] A. Gupte, S. Ahmed, S. S. Dey, and M. S. Cheon. Relaxations and discretizations for the pooling problem. 2015. `http://www.optimization-online.org/DB_HTML/2015/04/4883.html`.

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

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

[31] IBM Corporation. *IBM ILOG CPLEX Optimization Studio: CPLEX User's Manual. Version 12 Release 6*, 2013.

[32] Y. Kawajir, C. Laird, and A. Wächter. *Introduction to Ipopt: A tutorial for downloading, installing, and using Ipopt. Revision 2020*, 2011.

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

[34] J. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Mathematical Programming*, 103(2):251–282, 2005.

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

[36] C. A. Meyer and C. A. Floudas. Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes. *Journal of Global Optimization*, 29(2):125–155, 2004.

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

[38] R. Misener. *Novel Global Optimization Methods: Theoretical and Computational Studies on Pooling Problems with Environmental Constraints.* PhD thesis, Princeton University, 2012. `http://arks.princeton.edu/ark:/88435/dsp015q47rn787`.

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

[40] R. Misener, J. P. Thompson, and C. A. Floudas. APOGEE: global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Computers & Chemical Engineering*, 35(5):876–892, 2011.

[41] F. Palacios-Gomez, L. Lasdon, and M. Engquist. Nonlinear optimization by successive linear programming. *Management Science*, 28(10):1106–1120, 1982.

[42] V. Pham, C. Laird, and M. El-Halwagi. Convex hull discretization approach to the global optimization of pooling problems. *Industrial & Engineering Chemistry Research*, 48(4):1973–1979, 2009.

[43] H. D. Sherali and W. P. Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31 of *Nonconvex Optimization and its Applications*. Springer US, 1999.

[44] H. D. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, 2(4):379–410, 1992.

[45] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, volume 65 of *Nonconvex Optimization and its Applications*. Springer US, 2002.

[46] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591, 2004.

[47] J. P. Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57:3–57, 2015.

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

[49] V. Visweswaran. MINLP: applications in blending and pooling problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, pages 1399–1405. Springer US, 2001.

[50] V. Visweswaran and C. A. Floudas. A global optimization algorithm (GOP) for certain classes of nonconvex

NLPs – II. Application of theory and test problems. *Computers & Chemical Engineering*, 14(12):1419–1434, 1990.

[51] V. Visweswaran and C. A. Floudas. New properties and computational improvement of the GOP algorithm for problems with quadratic objective functions and constraints. *Journal of Global Optimization*, 3(4):439–462, 1993.

[52] V. Visweswaran and C. A. Floudas. Computational results for an efficient implementation of the GOP algorithm and its variants. In *Global Optimization in Engineering Design*, volume 9 of *Nonconvex Optimization and its Applications*, pages 111–153. Springer US, 1996.

[53] V. Visweswaran and C. A. Floudas. New formulations and branching strategies for the GOP algorithm. In *Global Optimization in Engineering Design*, volume 9 of *Nonconvex Optimization and its Applications*, pages 75–109. Springer US, 1996.

[54] V. Visweswaran and C. A. Floudas. *cGOP: A deterministic global optimization package. User's guide. Version 1.1*, 1997.

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

[56] S. Yıldız and J. P. Vielma. Incremental and encoding formulations for mixed integer programming. *Operations Research Letters*, 41:654–658, 2013.

[57] J. Zhang, N.-H. Kim, and L. Lasdon. An improved successive linear programming algorithm. *Management Science*, 31(10):1312–1331, 1985.

A Detailed computational results

Table 15: Relative LP objectives and their baselines for all formulations and instances. More formally, let $m = \texttt{LP\_obj}$, $f \in \mathcal{F} = \mathcal{F}_{\text{all}}$ and $i \in \mathcal{I} = \mathcal{I}_{\text{all}}$. For a pair (f, i), we report $r_{m,f,i}$. The baselines are calculated as max obj $= \max_{f \in \mathcal{F}} v_{m,f,i}$.

		MCF-I-		MCF-J-		MCF-(I+J)-		MCF-(I×J)-		
Instance	P	Q	PQ	Q	PQ	Q	PQ	Q	PQ	max obj
Adhya1	1.18	1.09	1	1.06	1.02	1.01	1	1.01	1	-840.27
Adhya2	1.47	1.25	1	1.02	1	1.01	1	1.01	1	-574.78
Adhya3	1.52	1.35	1	1.02	1	1.01	1	1.01	1	-574.78
Adhya4	1.03	1.08	1	1.02	1.02	1.02	1	1.02	1	-961.93
Bental4	1.02	1.14	1.02	1.35	1	1	1	1	1	-541.67
Bental5	1	1	1	1.40	1	1	1	1	1	-3500.00
Foulds2	1	1.05	1	2.82	1	1.03	1	1.03	1	-1100.00
Foulds3	1	1	1	6.10	1	1	1	1	1	-8.00
Foulds4	1	1	1	11.03	1	1	1	1	1	-8.00
Foulds5	1	1	1	11.31	1	1	1	1	1	-8.00
Haverly1	1	1.10	1	1.40	1	1	1	1	1	-500.00
Haverly2	1	1.10	1	2.10	1	1	1	1	1	-1000.00
Haverly3	1	1.05	1	1.31	1.09	1.05	1	1.05	1	-800.00
RT2	1	1	1	1	1	1	1	1	1	-6697.66
sppA0	1.22	1.04	1.01	1.11	1	1.01	1	1.01	1	-37412.23
sppA1	1.43	1.15	1	1.12	1	1.03	1	1.03	1	-31438.51
sppA2	1.59	1.30	1.01	1.13	1	1.07	1	1.07	1	-23743.36
sppA3	1.11	1.07	1	1.01	1.01	1.01	1	1.01	1	-42032.79
sppA4	1.13	1.07	1	1.04	1	1.02	1	1.02	1	-43396.84
sppA5	1.22	1.18	1	1.02	1	1.01	1	1.01	1	-28257.75
sppA6	1.02	1.02	1	1	1	1	1	1	1	-42463.05
sppA7	1.23	1.16	1	1.14	1	1.07	1	1.07	1	-44682.25
sppA8	1.39	1.33	1	1.15	1	1.14	1	1.14	1	-30666.87
sppA9	1.48	1.21	1	1.13	1	1.07	1	1.07	1	-21933.99
sppB0	1.35	1.23	1	1.03	1	1.02	1	1.02	1	-45465.92
sppB1	1.13	1.12	1	1.04	1	1.04	1	1.04	1	-65523.34
sppB2	1.10	1.08	1	1.05	1	1.04	1	1.04	1	-56438.06
sppB3	1.07	1.07	1	1.01	1	1.01	1	1.01	1	-74050.47
sppB4	1.20	1.18	1	1.05	1	1.04	1	1.04	1	-59469.66
sppB5	1.20	1.19	1	1.11	1	1.08	1	1.08	1	-60696.36
sppC0	1.21	1.09	1.01	1.11	1.01	1.04	1	1.04	1	-98218.60
sppC1	1.20	1.11	1.01	1.12	1	1.05	1	1.05	1	-118673.48
sppC2	1.10	1.08	1	1.04	1	1.03	1	1.03	1	-135740.45
sppC3	1.07	1.07	1	1	1	1	1	1	1	-130315.02

Table 15 (continued)

Instance	P	MCF-I- Q	MCF-I- PQ	MCF-J- Q	MCF-J- PQ	MCF-(I+J)- Q	MCF-(I+J)- PQ	MCF-(I×J)- Q	MCF-(I×J)- PQ	max obj
gppL1	1	1.01	1	2.84	1	1	1	1	1	-43.00
gppL2	1.17	1.09	1	1.05	1	1.03	1	1.03	1	-853.47
gppL3	1.49	1.27	1	1.02	1	1.01	1	1.01	1	-574.78
gppL4	1.54	1.36	1	1.02	1	1.01	1	1.01	1	-574.78
gppL5	1.06	1.08	1	1.01	1	1.01	1	1.01	1	-972.44
gppL6	1.20	1.20	1.02	1.35	1	1	1	1	1	-541.67
gppL7	1	1	1	1.40	1	1	1	1	1	-3500.00
gppL8	1.09	1.06	1	2.82	1	1.06	1	1.06	1	-1100.00
gppL9	1	1	1	11.31	1	1	1	1	1	-8.00
gppL10	1	1	1	11.31	1	1	1	1	1	-8.00
gppL11	1	1	1	11.31	1	1	1	1	1	-8.00
gppL12	1.20	1.20	1	1.40	1	1	1	1	1	-500.00
gppL13	1.20	1.20	1	2.20	1	1	1	1	1	-1000.00
gppL14	1	1	1	1.20	1	1	1	1	1	-875.00
gppL15	1	1	1	1	1	1	1	1	1	-6697.66
gppA1	1	1	1	1	1	1	1	1	1	-1175.00
gppA2	1	1	1	1.50	1	1	1	1	1	-641.00
gppA3	1	1	1	1.11	1	1	1	1	1	-420.60
gppA4	1.12	1.12	1	1.18	1	1.12	1	1.12	1	-599.00
gppA5	2.57	2.68	1	1	1	1	1	1	1	-198.00
gppB1	1	1.38	1	1	1	1	1	1	1	-427.37
gppB2	1	1	1	1	1	1	1	1	1	-210.00
gppB3	1.07	1.07	1	1.01	1	1	1	1	1	-932.00
gppB4	1	1	1	1.27	1	1	1	1	1	-912.80
gppB5	1	1.02	1	1.08	1	1	1	1	1	-439.00
gppC1	1.15	1.07	1	1.15	1	1.01	1	1.01	1	-1352.72
gppC2	1.79	1.79	1	1.08	1	1	1	1	1	-682.14
gppC3	1.19	1.19	1	1.01	1	1	1	1	1	-1716.63
gppC4	1.02	1	1	1.09	1	1	1	1	1	-1512.10
gppC5	1.47	1.42	1	1.02	1	1	1	1	1	-1071.81
gppD1	1.26	1.25	1	1.05	1	1	1	1	1	-1994.00
gppD2	1.39	1.10	1	1.52	1	1.10	1	1.10	1	-1356.51
gppD3	1.01	1.01	1	1.03	1	1.01	1	1.01	1	-2071.00
gppD4	2.10	1.75	1	1.16	1	1	1	1	1	-637.86
gppD5	1.04	1.04	1	1.26	1	1.02	1	1.02	1	-1641.80
gppE1	3.26	1.47	1	1	1	1	1	1	1	-463.23
gppE2	2.66	1.70	1	1.04	1	1	1	1	1	-556.00
gppE3	6.81	3.46	1	1	1	1	1	1	1	-78.68
gppE4	1.78	1.42	1	1.05	1	1	1	1	1	-891.25
gppE5	8.55	3.38	1	1	1	1	1	1	1	-221.35

Table 16: Total solve times of the NLP, relative NLP lower and upper bounds and their baselines for all formulations and instances. If an instance could be solved within `time_limit`, we report the total solve time of the NLP as (time), otherwise we report the NLP lower and upper bounds as lb and ub. All times are in seconds. If no lower or upper bound is found, we write "-". More formally, let $m_1 =$ `NLP_time`, $m_2 =$ `NLP_lb`, $m_3 =$ `NLP_ub`, $f \in \mathcal{F} = \mathcal{F}_{\text{all}}$ and $i \in \mathcal{I} = \mathcal{I}_{\text{all}}$. If for a pair (f, i), we have $v_{m_1,f,i} <$ `time_limit`, then we report (time) $= (v_{m_1,f,i})$, otherwise we report lb $= r_{m_2,f,i}$ and ub $= r_{m_3,f,i}$. The baselines are calculated as max lb $= \max_{f\in\mathcal{F}} v_{m_2,f,i}$ and min ub $= \min_{f\in\mathcal{F}} v_{m_3,f,i}$.

	P		MCF-I-Q		MCF-I-PQ		MCF-J-Q		MCF-J-PQ		MCF-(I+J)-Q		MCF-(I+J)-PQ		MCF-(I×J)-Q		MCF-(I×J)-PQ			
Instance	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	max lb	min ub
Adhya1		(8.94)		(4.41)		(0.83)		(1.72)		(1.38)		(3.86)		(2.98)		(6.24)		(3.32)	-549.80	-549.80
Adhya2		(24.30)		(2.49)		(0.95)		(1.64)		(1.30)		(2.50)		(1.61)		(3.48)		(1.83)	-549.80	-549.80
Adhya3		(246.78)		(60.18)		(0.87)		(2.28)		(1.09)		(2.56)		(1.62)		(5.34)		(2.25)	-561.04	-561.04
Adhya4		(5.91)		(77.70)		(0.71)		(0.67)		(1.12)		(2.09)		(1.19)		(3.90)		(2.47)	-877.65	-877.65
Bental4		(0.08)		(0.26)		(0.16)		(0.30)		(0.26)		(0.58)		(0.14)		(0.17)		(0.12)	-450.00	-450.00
Bental5	1.02	1		(0.30)		(1.44)	1.15	1		(1.24)		(14.73)		(0.26)		(0.27)		(0.22)	-3500.00	-3500.00
Foulds2		(120.83)		(0.45)		(0.25)		(22.14)		(0.15)		(0.65)		(0.09)		(1.60)		(0.17)	-1100.00	-1100.00
Foulds3	4.49	1		(0.99)		(57.63)	5.01	1		(2.42)		(2.79)		(89.68)	1	1.01		(6.58)	-8.00	-8.00
Foulds4	6.26	1		(144.20)		(0.88)	6.98	1.01		(2.87)		(3.81)		(181.65)		(499.56)		(6.97)	-8.00	-8.00
Foulds5	2.75	1		(20.56)		(4.99)	7.70	1.04		(1.45)		(1.24)	1	2.09	1	1.05		(23.86)	-8.00	-8.00
Haverly1		(0.16)		(0.13)		(0.08)		(0.11)		(0.11)		(0.24)		(0.16)		(0.12)		(0.12)	-400.00	-400.00
Haverly2		(0.27)		(0.27)		(0.15)		(0.16)		(0.16)		(0.47)		(0.12)		(0.16)		(0.14)	-600.00	-600.00
Haverly3		(0.10)		(0.14)		(0.09)		(0.15)		(0.11)		(0.24)		(0.11)		(0.13)		(0.13)	-750.00	-750.00
RT2		(0.75)		(0.07)		(0.07)		(0.87)		(0.07)		(0.07)		(0.07)		(0.08)		(0.08)	-6697.66	-6697.66
sppA0	1.28	1	1.04	-	1	-	1.08	-	1	-	1.02	-	1	-	1.02	-	1.01	-	-36403.68	-23324.81
sppA1	1.46	1	1.13	-	1.01	-	1.11	-	1	1.03	1.06	-	1.02	-	1.06	-	1.02	-	-29724.08	-29232.62
sppA2	1.73	1.53	1.22	-	1	1	1.05	-		(300.56)	1.05	-		(456.25)	1.05	-	1.01	-	-23044.16	-23044.16
sppA3	1.17	1	1.09	-	1	-	1.05	-	1.05	1	1.04	-	1.01	-	1.04	-	1.02	-	-40154.60	-38243.30
sppA4	1.15	-	1.07	-	1	-	1.04	2.98	1.02	1	1.03	-	1	-	1.03	-	1.01	-	-42605.79	-37753.26
sppA5	1.22	-	1.18	-	1	-	1.02	1.98	1	1	1.01	-	1	-	1.01	-	1	-	-28257.75	-27155.64
sppA6	1.02	1	1.02	-	1	-	1.02	1.35	1	1.14	1	-	1	-	1	-	1	-	-42463.05	-41785.59
sppA7	1.23	-	1.16	-	1	-	1.17	1.39	1	1	1.07	-	1	-	1.07	-	1	-	-44682.25	-37034.99
sppA8	1.39	-	1.33	-	1	-	1.27	1.06	1	1	1.14	-	1	-	1.14	-	1	-	-30666.87	-24327.91
sppA9	1.68	-	1.21	-	1	-	1.16	1.15	1	1	1.07	-	1	-	1.07	-	1	-	-21933.99	-17343.25
sppB0	1.35	1	1.21	-	1	-	1.02	-	1	1.41	1.02	-	1	-	1.02	-	1	-	-45465.92	-28978.88
sppB1	1.13	-	1.12	-	1	-	1.05	1.55	1	1	1.04	-	1	-	1.04	-	1	-	-65523.34	-50504.45
sppB2	1.10	-	1.08	-	1	-	1.10	1.98	1	1	1.04	-	1	-	1.04	-	1	-	-56438.06	-30574.13
sppB3	1.07	-	1.07	-	1	-	1.01	1.24	1	1	1.01	-	1	-	1.01	-	1	-	-74050.47	-63915.74
sppB4	1.20	-	1.18	-	1	-	1.06	1.23	1	1	1.04	-	1	-	26.22	-	26.22	-	-59469.66	-48703.72
sppB5	1.20	-	1.19	-	1	-	1.14	1.42	1	1	1.08	-	1	-	29.04	-	29.04	-	-60696.36	-52727.23
sppC0	1.21	-	1.09	-	1.01	-	1.16	-	1.01	-	1.04	-	1.01	-	1.04	-	1	-	-98218.60	-
sppC1	1.19	-	1.10	-	1.01	-	1.15	-	1	-	1.05	-	1.01	-	15.80	-	15.80	-	-119031.69	-
sppC2	1.09	-	1.08	-	1	-	1.04	-	1	-	1.03	-	17.73	-	17.73	-	17.73	-	-136043.63	-
sppC3	1.07	-	1.07	-	1	-	1.01	1.32	1	1	21.62	-	21.62	-	21.62	-	21.62	-	-130315.02	-47226.66

Table 16 (continued)

Instance	P		MCF-I-Q		MCF-I-PQ		MCF-J-Q		MCF-J-PQ		MCF-(I+J)-Q		MCF-(I+J)-PQ		MCF-(I×J)-Q		MCF-(I×J)-PQ			
	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	lb	(time) ub	max lb	min ub
gppL1		(0.78)		(0.33)		(0.18)		(0.80)		(0.20)		(0.36)		(0.33)		(0.82)		(0.49)	-42.58	-42.58
gppL2	1.01	1	1.02	1	1	1	1	1		(3.15)	1	1	1	1	1	1	1	1	-549.80	-549.80
gppL3	1.07	1	1.01	1	1	1		(8.82)		(3.72)	1	1	1	1	1	1	1	1	-549.80	-549.80
gppL4	2.11	1	1.15	1.02	1.02	1		(8.25)		(7.20)		(72.55)	1	1	1.01	-	1.01	16.83	-561.04	-561.04
gppL5	1.08	1	1.06	1		(31.89)		(16.91)		(5.43)		(7.40)		(2.10)		(118.62)		(36.25)	-877.65	-877.65
gppL6	1.12	1		(37.67)		(0.77)		(0.50)		(0.30)		(0.79)		(0.15)		(0.21)		(0.50)	-450.00	-450.00
gppL7	1.55	1		(0.94)		(6.83)	1.25	1		(0.69)		(1.52)		(484.28)		(83.84)		(584.97)	-3500.00	-3500.00
gppL8	2.05	1		(2.65)		(0.93)		(82.87)		(0.71)		(3.04)		(1.35)		(3.29)		(0.48)	-1100.00	-1100.00
gppL9	12.50	1		(165.23)		(4.31)	10.49	1		(11.66)	1	-	1	-	1	-	1	-	-8.00	-8.00
gppL10	12.50	1		(32.39)		(10.54)	10.16	1		(4.64)	1	-	1	-	1	-	1	-	-8.00	-8.00
gppL11	12.50	1		(1.39)		(4.23)	10.92	1.07		(53.82)	1	-	1	-	1	-	1	-	-8.00	-8.00
gppL12	1.13	1		(5.86)		(0.30)		(0.33)		(0.19)		(1.06)		(0.15)		(0.21)		(0.19)	-400.00	-400.00
gppL13	1.08	1		(58.40)		(0.41)		(0.49)		(0.18)		(0.87)		(0.20)		(0.27)		(0.16)	-600.00	-600.00
gppL14	1.03	1		(1.93)		(0.98)		(0.13)		(0.20)		(1.74)		(0.15)		(0.29)		(0.21)	-750.00	-750.00
gppL15		(0.85)		(0.07)		(0.07)		(0.85)		(0.05)		(0.07)		(0.08)		(0.16)		(0.09)	-6697.66	-6697.66
gppA1		(0.20)		(0.07)		(0.07)		(0.22)		(0.20)		(0.61)		(0.23)		(0.23)		(0.20)	-1175.00	-1175.00
gppA2	1	1		(0.16)		(0.33)	1	1		(0.08)		(0.16)		(0.17)		(0.14)		(0.14)	-641.00	-641.00
gppA3	1	1		(0.36)		(0.28)	1	1		(0.13)		(0.41)		(0.14)		(0.41)		(0.22)	-420.60	-420.60
gppA4	1	1		(555.53)		(0.53)		(1.24)		(0.42)		(11.31)		(0.18)		(20.29)		(0.23)	-599.00	-599.00
gppA5	2.36	1	1.01	1		(0.09)		(0.36)		(0.33)		(0.60)		(0.66)		(0.74)		(0.18)	-198.00	-198.00
gppB1	1.04	1	1	1		(0.20)		(3.54)		(0.07)		(0.87)		(0.10)		(0.34)		(0.09)	-427.37	-427.37
gppB2		(0.22)		(0.09)		(0.09)		(0.09)		(0.09)		(0.96)		(0.17)		(0.22)		(0.12)	-210.00	-210.00
gppB3	1.45	1	1.07	1		(0.55)	1	1		(1.69)		(0.93)		(0.65)		(6.64)		(1.27)	-932.00	-932.00
gppB4	1.16	1		(0.73)		(0.85)	1.01	1		(1.24)		(4.98)		(0.26)		(3.81)		(4.45)	-912.80	-912.80
gppB5	1.06	1	1.02	1		(0.39)	1.01	1		(2.12)		(1.01)		(0.19)		(3.19)		(2.31)	-439.00	-439.00
gppC1	1.31	1	1.02	1		(2.19)	1.06	1		(0.31)	1	1		(22.69)	1.01	1.03		(204.37)	-1352.72	-1352.72
gppC2	2.51	1	1.51	1		(387.03)		(86.47)		(3.42)	1	1		(7.48)	1	1		(402.46)	-673.86	-673.86
gppC3	1.26	1	1.19	1.04		(3.07)	1.01	1		(20.51)	1	1.04		(320.72)	1	-	1	-	-1716.63	-1716.63
gppC4	1.09	1	1	1		(4.89)	1.07	1.01		(9.00)	1	-	1	-	1	-	1	-	-1512.10	-1512.10
gppC5	1.51	1	1.42	1		(27.08)	1	1		(8.09)	1	1	1	-	1	-	1	1.02	-1071.81	-1071.81
gppD1	1.31	1	1.25	1.05	1	-	1.05	1.03	1	-	1	1.12	1	-	1	-	1	-	-1994.00	-1988.32
gppD2	2.08	1	1.08	1.01	1	1	1.14	1.01		(98.71)	1.10	1.16	1	-	1.10	-	1	-	-1356.51	-1356.51
gppD3	1.03	1	1.01	1.02	1	1.01	1.03	1.01	1	1	1.01	-	1	-	1.01	-	1	-	-2071.00	-2071.00
gppD4	4.17	1	1.75	1.03	1	-	1.13	1		(116.26)	1	-	1	-	1	-	1	-	-637.86	-637.86
gppD5	1.55	1	1.04	1.28	1	1.24	1.23	1	1	1	1.02	-	1	-	1.02	-	1	-	-1641.80	-1641.80
gppE1	6.26	1	1.47	1		(104.92)		(24.06)		(578.03)		(196.71)		(264.15)	1	-	1	-	-463.23	-463.23
gppE2	4.36	1.03	1.70	1		(319.61)	1.01	1		(330.76)	1	-		(2.96)	1	-	1	-	-556.00	-556.00
gppE3	37.79	-	3.46	1.11		(126.98)		(30.86)		(524.04)		(594.93)		(4.42)	1	-	1	-	-78.68	-78.68
gppE4	2.93	1.03	1.42	1.02		(81.01)	1.05	1		(87.82)	1	-		(156.13)	1	-	1	-	-891.25	-891.25
gppE5	12.04	1	3.38	-	1	1		(44.91)		(92.66)		(321.66)		(134.00)	1	-	1	-	-221.35	-221.35

Table 17: Average relative LP objectives and number of times a formulation found the maximum LP objective for a set of instances. More formally, let $m = \texttt{LP\_obj}$ and $f \in \mathcal{F} = \mathcal{F}_{\text{all}}$. For a pair $(f, \mathcal{I})$, we report $\varnothing\,\text{obj} = \bar{r}_{m,f}$ and $\#\max = |\{i \in \mathcal{I} : r_{m,f,i} = 1\}|$ (as a percentage of $|\mathcal{I}|$).

	P			MCF-I-Q			MCF-I-PQ			MCF-J-Q			MCF-J-PQ			MCF-(I+J)-Q			MCF-(I+J)-PQ			MCF-(I×J)-Q			MCF-(I×J)-PQ		
$\mathcal{I}$	Ø obj	# max		Ø obj	# max		Ø obj	# max		Ø obj	# max		Ø obj	# max		Ø obj	# max		Ø obj	# max		Ø obj	# max		Ø obj	# max	
$\mathcal{I}_{\text{all}}$	1.42	23	(31%)	1.23	17	(23%)	1	68	(92%)	1.96	11	(15%)	1	69	(93%)	1.02	40	(54%)	1	74	(100%)	1.02	40	(54%)	1	74	(100%)
$\mathcal{I}_{\text{SPP}}$	1.17	9	(26%)	1.12	5	(15%)	1	29	(85%)	1.92	3	(9%)	1	29	(85%)	1.03	10	(29%)	1	34	(100%)	1.03	10	(29%)	1	34	(100%)
$\mathcal{I}_{\text{SPP,L}}$	1.09	9	(64%)	1.09	5	(36%)	1	13	(93%)	3.14	1	(7%)	1.01	11	(79%)	1.01	8	(57%)	1	14	(100%)	1.01	8	(57%)	1	14	(100%)
$\mathcal{I}_{\text{SPP,R}}$	1.22	0	(0%)	1.14	0	(0%)	1	16	(80%)	1.07	2	(10%)	1	18	(90%)	1.04	2	(10%)	1	20	(100%)	1.04	2	(10%)	1	20	(100%)
$\mathcal{I}_{\text{GPP}}$	1.63	14	(35%)	1.32	12	(30%)	1	39	(98%)	2.00	8	(20%)	1	40	(100%)	1.01	30	(75%)	1	40	(100%)	1.01	30	(75%)	1	40	(100%)
$\mathcal{I}_{\text{GPP,L}}$	1.13	7	(47%)	1.10	6	(40%)	1	14	(93%)	3.48	1	(7%)	1	15	(100%)	1.01	10	(67%)	1	15	(100%)	1.01	10	(67%)	1	15	(100%)
$\mathcal{I}_{\text{GPP,R}}$	1.93	7	(28%)	1.45	6	(24%)	1	25	(100%)	1.10	7	(28%)	1	25	(100%)	1.01	20	(80%)	1	25	(100%)	1.01	20	(80%)	1	25	(100%)

Table 18: Average relative NLP lower bounds and number of times a formulation found the maximum NLP lower bound for a set of instances. More formally, let $m = \texttt{NLP\_lb}$ and $f \in \mathcal{F} = \mathcal{F}_{\text{all}}$. For a pair $(f, \mathcal{I})$, we report $\varnothing\,\text{lb} = \bar{r}_{m,f}$ and $\#\max = |\{i \in \mathcal{I} : r_{m,f,i} = 1\}|$ (as a percentage of $|\mathcal{I}|$).

	P			MCF-I-Q			MCF-I-PQ			MCF-J-Q			MCF-J-PQ			MCF-(I+J)-Q			MCF-(I+J)-PQ			MCF-(I×J)-Q			MCF-(I×J)-PQ		
$\mathcal{I}$	Ø lb	# max		Ø lb	# max		Ø lb	# max		Ø lb	# max		Ø lb	# max		Ø lb	# max		Ø lb	# max		Ø lb	# max		Ø lb	# max	
$\mathcal{I}_{\text{all}}$	2.64	17	(23%)	1.16	33	(45%)	1	70	(95%)	1.65	34	(46%)	1	71	(96%)	1.29	52	(70%)	1.51	68	(92%)	2.44	50	(68%)	2.43	63	(85%)
$\mathcal{I}_{\text{SPP}}$	1.46	10	(29%)	1.08	14	(41%)	1	31	(91%)	1.55	10	(29%)	1	31	(91%)	1.63	15	(44%)	2.10	28	(82%)	4.12	15	(44%)	4.10	24	(71%)
$\mathcal{I}_{\text{SPP,L}}$	1.75	10	(71%)	1	14	(100%)	1	14	(100%)	2.20	10	(71%)	1	14	(100%)	1	14	(100%)	1	14	(100%)	1	14	(100%)	1	14	(100%)
$\mathcal{I}_{\text{SPP,R}}$	1.25	0	(0%)	1.13	0	(0%)	1	17	(85%)	1.09	0	(0%)	1	17	(85%)	2.07	1	(5%)	2.87	14	(70%)	6.30	1	(5%)	6.27	10	(50%)
$\mathcal{I}_{\text{GPP}}$	3.65	7	(18%)	1.23	19	(48%)	1	39	(98%)	1.74	24	(60%)	1	40	(100%)	1	37	(93%)	1	40	(100%)	1	35	(88%)	1	39	(98%)
$\mathcal{I}_{\text{GPP,L}}$	3.52	2	(13%)	1.02	11	(73%)	1	14	(93%)	2.92	11	(73%)	1	15	(100%)	1	15	(100%)	1	15	(100%)	1	14	(93%)	1	14	(93%)
$\mathcal{I}_{\text{GPP,R}}$	3.73	5	(20%)	1.35	8	(32%)	1	25	(100%)	1.03	13	(52%)	1	25	(100%)	1.01	22	(88%)	1	25	(100%)	1.01	21	(84%)	1	25	(100%)

Table 19: Average relative NLP upper bounds and number of times a formulation found the minimum NLP upper bound for a set of instances. More formally, let $m = \texttt{NLP\_ub}$ and $f \in \mathcal{F} = \mathcal{F}_{\text{all}}$. For a pair $(f, \mathcal{I})$, we report $\varnothing\,\text{ub} = \bar{r}_{m,f}$ and $\#\min = |\{i \in \mathcal{I} : r_{m,f,i} = 1\}|$ (as a percentage of $|\mathcal{I}|$).

	P			MCF-I-Q			MCF-I-PQ			MCF-J-Q			MCF-J-PQ			MCF-(I+J)-Q			MCF-(I+J)-PQ			MCF-(I×J)-Q			MCF-(I×J)-PQ		
$\mathcal{I}$	Ø ub	# min		Ø ub	# min		Ø ub	# min		Ø ub	# min		Ø ub	# min		Ø ub	# min		Ø ub	# min		Ø ub	# min		Ø ub	# min	
$\mathcal{I}_{\text{all}}$	1.01	56	(76%)	1.01	44	(59%)	1	51	(69%)	1.10	47	(64%)	1.01	66	(89%)	1.01	42	(57%)	1.02	44	(59%)	1	34	(46%)	1.41	37	(50%)
$\mathcal{I}_{\text{SPP}}$	1.03	19	(56%)	1	14	(41%)	1	15	(44%)	1.26	12	(35%)	1.02	27	(79%)	1	14	(41%)	1.07	14	(41%)	1	12	(35%)	1	14	(41%)
$\mathcal{I}_{\text{SPP,L}}$	1	14	(100%)	1	14	(100%)	1	14	(100%)	1	12	(86%)	1	14	(100%)	1	14	(100%)	1.08	13	(93%)	1	12	(86%)	1	14	(100%)
$\mathcal{I}_{\text{SPP,R}}$	1.09	5	(25%)	-	0	(0%)	1	1	(5%)	1.55	0	(0%)	1.04	13	(65%)	-	0	(0%)	1	1	(5%)	-	0	(0%)	-	0	(0%)
$\mathcal{I}_{\text{GPP}}$	1	37	(93%)	1.02	30	(75%)	1.01	36	(90%)	1	35	(88%)	1	39	(98%)	1.01	28	(70%)	1	30	(75%)	1	22	(55%)	1.63	23	(58%)
$\mathcal{I}_{\text{GPP,L}}$	1	15	(100%)	1	14	(93%)	1	15	(100%)	1	14	(93%)	1	15	(100%)	1	12	(80%)	1	12	(80%)	1	11	(73%)	2.32	11	(73%)
$\mathcal{I}_{\text{GPP,R}}$	1	22	(88%)	1.02	16	(64%)	1.01	21	(84%)	1	21	(84%)	1	24	(96%)	1.02	16	(64%)	1	18	(72%)	1	11	(44%)	1	12	(48%)