J Glob Optim (2013) 56:917–937
DOI 10.1007/s10898-012-9890-7

A multi-commodity flow formulation for the generalized pooling problem

Mohammed Alfaki · Dag Haugland

Received: 15 February 2011 / Accepted: 9 March 2012 / Published online: 6 April 2012

Abstract The pooling problem is an extension of the minimum cost network flow problem where the composition of the flow depends on the sources from which it originates. At each source, the composition is known. In all other nodes, the proportion of any component is given as a weighted average of its proportions in entering flow streams. The weights in this average are simply the arc flow. At the terminals of the network, there are bounds on the relative content of the various components. Such problems have strong relevance in e.g. planning models for oil refining, and in gas transportation models with quality constraints at the reception side. Although the pooling problem has bilinear constraints, much progress in solving a class of instances to global optimality has recently been made. Most of the approaches are however restricted to networks where all directed paths have length at most three, which means that there is no connection between pools. In this work, we generalize one of the most successful formulations of the pooling problem, and propose a multi-commodity flow formulation that makes no assumptions on the network topology. We prove that our formulation has stronger linear relaxation than previously suggested formulations, and demonstrate experimentally that it enables faster computation of the global optimum.

Keywords Pooling problem · Multi-commodity flow · Linear relaxation · Bilinear constraint · Convex and concave envelopes

1 Motivation

The classic blending problem, which appears in many industrial settings, involves determination of the optimal blend of raw materials to produce a certain quantity of end products. The composition of the end products is subject to certain specifications, and the decision

M. Alfaki · D. Haugland (✉)
Department of Informatics, University of Bergen, P.O. Box 7803, 5020 Bergen, Norway
e-mail: dag@ii.uib.no

M. Alfaki
e-mail: mohammeda@ii.uib.no

maker must determine in what proportions the raw materials should be used in the various end products. The optimal blend means the minimum cost blend of inputs satisfying the given specifications. Such problems can be formulated as linear programs (LP), and are therefore very easy to solve.

A considerably more difficult problem occurs if the blended flow of raw materials is mixed in intermediate tanks (pools) and then blended again to form end products. The nodes are partitioned into sources, pools and terminals, and each arc goes either from a source to a pool, from a source to a terminal, or from a pool to a terminal. The corresponding flow problem in networks with this particular tripartite structure is referred to as the *pooling problem*. Tracking the quality from the sources to the terminals leads to bilinear constraints, and consequently the pooling problem can possibly have many local optima. Recently, it was shown (Alfaki and Haugland 2012) that the pooling problem is strongly NP-hard.

Most mathematical programming formulations for the pooling problem belong to either of two main classes. Belonging to the first class are formulations that, in addition to flow variables, make use of variables representing the quality of the flow. Noteworthy among these is the P-formulation, which was first introduced by Haverly (1978, 1979). In contrast, Ben-Tal et al. (1994) suggested the Q-formulation, where the quality variables are replaced by variables representing flow proportions. This model has later been shown to perform better when fed into generic branch-and-cut algorithms. By applying the reformulation-linearization technique (Sherali et al. 1998; Sherali and Adams 1999; Sherali 2002) to the Q-formulation, Tawarmalani and Sahinidis (2002) managed to derive a stronger model referred to as the PQ-formulation. The superiority in strength over the P-formulation was proved theoretically, and experiments with the global optimizer BARON (Sahinidis 1996) showed that standard test instances could be solved very quickly. Building on this work, Alfaki and Haugland (2012) managed to improve the PQ-formulation even further.

Practical applications, particularly in integrated water systems in chemical processes (Karuppiah and Grossmann 2006) and pipeline transportation of natural gas (Li et al. 2011), frequently reveal that flow streams leaving one pool may in their turn be blended in pools further downstream in the network. Single-period planning models based on networks with a single layer of pools may also have this property when extended to multi-period models with inventories. In such models, each pair of inventory and time period becomes a pool, and constraints that define the pool qualities are necessary if the input quality cannot be assumed constant over time. Contradicting the assumptions made in the works cited in the paragraph above, the applications in question imply a flow network with paths intersecting more than one pool.

The optimization problem arising when no longer assuming the tripartite network structure is referred to (Audet et al. 2004; Misener and Floudas 2009) as the *generalized pooling problem* (GPP). For the purpose of a clear distinction, the pooling problem is henceforth in the current work also referred to as the *standard* pooling problem.

According to Main (1993), paths of many pools tend to reduce the chance for local optimization methods to end up in the global optimum. This can be overcome by extending the P-formulation to GPP, and apply a global solver handling bilinear constraints to the resulting model. Such an extension is straightforward. Given the theoretical and computational superiority of the PQ-formulation for the standard pooling problem, it is however reasonable to expect that an extension of the PQ-formulation will be competitive with an extended P-formulation. This motivates the goal of the current work: Derive a computationally efficient model for GPP by extending the idea of the PQ-formulation to networks with arbitrary structure.

Some attempts to generalize pooling problem formulations based on proportion variables can be found in the literature. Audet et al. (2004) indicate a hybrid model, with both quality and proportion variables. Following the idea of Ben-Tal et al. (1994), they introduce proportion variables at pools that receive flow only from sources, and apply quality variables in the spirit of the P-formulation for other pools.

Recently, generalizations of the pooling problem that make no assumptions on the network structure have been developed. Meyer and Floudas (2006) and Misener et al. (2010) proposed a mixed integer nonlinear (MINLP) formulation for the design of waste-water treatment. The networks may contain arcs between pools, since the flow of water may go through several stages of refinement (reduction of contamination). Their model is based on quality variables with constraints generalizing those of the P-formulation. This is a very natural choice of variables, since it does not seem appropriate to model quality updates in terms of proportion variables. Because the authors also consider a fixed charge for opening the arcs, binary variables are introduced, and the model becomes an MINLP. To tighten the relaxation, Misener et al. (2010) also apply a novel piecewise underestimation for the nonconvex bilinear terms (Wicaksono and Karimi 2008; Gounaris et al. 2009). Another extension of the standard pooling problem has been proposed by Misener and Floudas (2010) to maximize the profit of blending reformulated gasoline subject to environmental standards, involving complex emission constraints. Other interesting applications of global optimization related to network flow problems are given in Chapter 6 in the textbook of Horst et al. (2000) and by Guisewite (1995).

The contribution from this paper is a new multi-commodity flow formulation for the GPP. Complying with the PQ-formulation, it is based uniquely on proportion variables in addition to the flow variables. In network instances where all paths intersect at most one pool, our formulation coincides with the PQ-formulation, and we demonstrate that the favorable theoretical and computational properties carry over to GPP.

The paper is organized as follows: In Sect. 2, we give some notations and definitions that will be used throughout the paper. Section 3 presents the P-formulation applied to GPP, and in Sect. 4, we propose and analyze the multi-commodity flow formulation. The analysis includes a comparison with the P-formulation and the hybrid model of Audet et al. (2004), both of which turn out to have weaker relaxations than our model. Section 5 presents computational experiments where the said formulations are compared, and Sect. 6 concludes the paper.

2 Notations and definitions

We consider a directed graph $G = (N, A)$ with node set N and arc set A. For any node $i \in N$, let $N_i^+ = \{j \in N : (i, j) \in A\}$ and $N_i^- = \{j \in N : (j, i) \in A\}$ denote the set of out- and in-neighbors of i, respectively. We assume that G has non-empty sets $S, T \subseteq N$ of *sources* and *terminals*, respectively, where $N_s^- = \emptyset \, \forall s \in S$ and $N_t^+ = \emptyset \, \forall t \in T$. We refer to all nodes in $I = N \setminus (S \cup T)$ as *pools*. We define a finite set of *quality attributes* K. With each $i \in S \cup T$, we associate a real constant q_i^k for each $k \in K$. If $s \in S$, q_s^k is referred to as the *quality parameter* of attribute k at that source, and if $t \in T$, q_t^k is referred to as the *quality bound* of attribute k at terminal t. For each $i \in N$, we define the constant *flow capacity* b_i, and for each arc $(i, j) \in A$, we define the constant *unit cost* c_{ij}. This is slightly more general than defining costs and revenues only at the sources and the terminals, respectively, which is common practice in the pooling problem literature. For each $i \in N$, let S_i be the set of sources from which there exists a path to i in G (for all $s \in S$, we have $S_s = \{s\}$).

Define the *flow polytope* $\mathcal{F}(G, b)$ as the set of flow vectors $f \in \mathbb{R}_+^A$ satisfying

$$\sum_{j \in N_i^+} f_{ij} \leq b_i, \qquad i \in N \setminus T, \tag{1}$$

$$\sum_{j \in N_t^-} f_{jt} \leq b_t, \qquad t \in T, \tag{2}$$

$$\sum_{j \in N_i^+} f_{ij} - \sum_{j \in N_i^-} f_{ji} = 0, \quad i \in I. \tag{3}$$

Consider any $f \in \mathcal{F}(G, b)$. For all $i \in N$ and $k \in K$, define w_i^k as a product quality of attribute k at node i corresponding to flow f. That is, $w_i^k = q_i^k$ if $i \in S$, and for all $i \in N \setminus S$, we have

$$w_i^k \sum_{j \in N_i^-} f_{ji} = \sum_{j \in N_i^-} w_j^k f_{ji},$$

for all $k \in K$.

This means that the product qualities are arbitrary at pools and terminals with zero entering flow, and uniquely defined at all other nodes. The definition implies that we assume linear blending at pools and terminals, and that the flow on all arcs leaving node i has the unique quality w_i^k.

In this work, we consider the following extension of the minimum cost flow problem:

Problem 1 (*The generalized pooling problem*) Find $f \in \mathcal{F}(G, b)$ and a corresponding matrix of product qualities $w \in \mathbb{R}^{N \times K}$ satisfying $w_t^k \leq q_t^k \forall t \in T,\ k \in K$, such that $\sum_{(i,j) \in A} c_{ij} f_{ij}$ is minimized.

Without loss of generality, we have assumed that only upper quality bounds are imposed. Should a lower bound $w_t^k \geq \ell_t^k$ apply, we introduce a new quality attribute k^-, and define $q_s^{k^-} = -q_s^k$ for all $s \in S$, and let $q_t^{k^-} = -\ell_t^k$.

It follows directly from the strong NP-hardness of the standard pooling problem (Alfaki and Haugland 2012) that also Problem 1 is strongly NP-hard. The proof of this fact relies on a problem definition without lower bounds on the delivery to the terminals, or at least includes zero bounds as a possibility. For the purpose of a simple model, we have assumed the lower delivery bounds to be zero, but the main results in this work do not depend on this assumption.

3 A formulation based on quality variables

3.1 The P-formulation for the generalized pooling problem

The P-formulation of the problem is found by extending the minimum cost flow problem (with node capacities) in directed graphs. Its decision variables are exactly those indicated in the problem definition, namely the flow f_{ij} along arc (i, j), and w_i^k representing the quality in terms of attribute k of the flow leaving node $i \in N$. In addition, we introduce the decision variable v_{ij}^k (for all $(i, j) \in A,\ k \in K$), which is best understood by thinking of the attribute indexed by k as a contaminant. While the quality w_i^k is the relative content of the contaminant in the flow along arc (i, j), v_{ij}^k is the total amount of the contaminant in the same flow. Thus, $v_{ij}^k = w_i^k f_{ij}$ is a necessary relation.

For notational convenience, define the constants $w_s^k = q_s^k$ for all $s \in S$, $k \in K$. Problem 1 is then formulated as (recall that $N_s^- = \emptyset \, \forall s \in S$ and $N_t^+ = \emptyset \, \forall t \in T$):

$$[\mathrm{P}] \qquad \min_{f,w,v} \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{4}$$

$$\text{s.t.} \qquad f \in \mathcal{F}(G, b), \tag{5}$$

$$\sum_{j\in N_i^-} v_{ji}^k - \sum_{j\in N_i^+} v_{ij}^k = 0, \quad i \in I,\ k \in K, \tag{6}$$

$$\sum_{j\in N_t^-} v_{jt}^k - q_t^k \sum_{j\in N_t^-} f_{jt} \le 0, \quad t \in T,\ k \in K, \tag{7}$$

$$v_{ij}^k - w_i^k f_{ij} = 0, \qquad (i, j) \in A,\ k \in K. \tag{8}$$

Constraints (6) state that the amounts of contaminant k entering and leaving pool i are equal, and constraints (7) say that if any flow enters terminal t, then the relative content of k resulting from blending, $\frac{\sum_{j\in N_t^-} v_{jt}^k}{\sum_{j\in N_t^-} f_{jt}}$, must be below q_t^k.

The bilinear program (4)–(8) is referred to as the P-formulation for Problem 1. Alternative variants of the P-formulation can be derived, e.g. by substituting some or all occurrences of v_{ij}^k by $w_i^k f_{ij}$. We have introduced the v-variables for the sole purpose of collecting all bilinear terms in one set of constraints (8).

3.2 Linear relaxations of bilinear terms

For all formulations considered in this work, we apply a well established technique for relaxing bilinear constraints. It is illustrated here in the case of the P-formulation.

For all $k \in K$ and $(i, j) \in A$, let $\left[\underline{w}_i^k, \overline{w}_i^k\right]$ and $\left[\underline{f}_{ij}, \overline{f}_{ij}\right]$ be some intervals enclosing the feasible values of variables w_i^k and f_{ij}, respectively. Conservative, but quickly computed bounds are given as e.g. $\min_{s\in S_i} q_s^k \le w_i^k \le \max_{s\in S_i} q_s^k$ and $0 \le f_{ij} \le \min\left\{b_i, b_j\right\}$.

It is well known (McCormick 1976; Al-Khayyal and Falk 1983) that since (8) is bilinear, v_{ij}^k can be bounded between the convex and concave envelopes of $w_i^k f_{ij}$ on $\left[\underline{w}_i^k, \overline{w}_i^k\right] \times \left[\underline{f}_{ij}, \overline{f}_{ij}\right]$ by imposing four linear inequalities, henceforth referred to as the McCormick inequalities:

$$v_{ij}^k \ge \underline{w}_i^k f_{ij} + \underline{f}_{ij} w_i^k - \underline{w}_i^k \underline{f}_{ij}, \tag{9}$$

$$v_{ij}^k \ge \overline{w}_i^k f_{ij} + \overline{f}_{ij} w_i^k - \overline{w}_i^k \overline{f}_{ij}, \tag{10}$$

$$v_{ij}^k \le \underline{w}_i^k f_{ij} + \overline{f}_{ij} w_i^k - \underline{w}_i^k \overline{f}_{ij}, \tag{11}$$

$$v_{ij}^k \le \overline{w}_i^k f_{ij} + \underline{f}_{ij} w_i^k - \overline{w}_i^k \underline{f}_{ij}. \tag{12}$$

A linear relaxation of the P-formulation is thus obtained by replacing (8) by (9)–(12). The tightness of the relaxation obviously depends on the tightness of the variable bounds.

More generally, let F be any bilinear formulation for Problem 1 and C a rectangular subset defined in the space of the bilinear variables such that it contains all their feasible values. We denote by $LP^F[C]$ the linear program obtained by replacing each bilinear term by a new variable constrained by the McCormick inequalities corresponding to C. Let $z^F[C]$ denote the optimal objective function value of $LP^F[C]$.

4 A multi-commodity flow formulation based on proportion variables

Tawarmalani and Sahinidis (2002) suggested the so-called PQ-formulation for the standard pooling problem, and proved that it is stronger than the P-formulation. In this section, we generalize these results to GPP.

4.1 The PQ-formulation for the standard pooling problem

Assume now that $A \subseteq (S \times I) \cup (I \times T) \cup (S \times T)$. Define the proportion variables y_i^s $(s \in S,\ i \in I)$ as the fraction of the flow through pool i that originates from source s. Combining the new variables with flow variables, the PQ-formulation for the standard pooling problem can in our notation be written as

$$\text{[PQ]} \quad \min_{f,y} \sum_{s \in S} \sum_{i \in I \cap N_s^+} c_{si} y_i^s \sum_{t \in N_i^+} f_{it} + \sum_{t \in T} \sum_{j \in N_t^-} c_{jt} f_{jt} \tag{13}$$

$$\text{s.t.} \quad \sum_{i \in I \cap N_s^+} y_i^s \sum_{t \in N_i^+} f_{it} + \sum_{t \in T \cap N_s^+} f_{st} \le b_s, \qquad s \in S, \tag{14}$$

$$\sum_{t \in N_i^+} f_{it} \le b_i, \qquad i \in I, \tag{15}$$

$$\sum_{j \in N_t^-} f_{jt} \le b_t, \qquad t \in T, \tag{16}$$

$$\sum_{i \in I \cap N_t^-} f_{it} \sum_{s \in S_i} q_s^k y_i^s + \sum_{s \in S \cap N_t^-} q_s^k f_{st} - q_t^k \sum_{j \in N_t^-} f_{jt} \le 0, \quad t \in T, k \in K, \tag{17}$$

$$\sum_{s \in S_i} y_i^s = 1, \qquad i \in I, \tag{18}$$

$$f_{it} - \sum_{s \in S_i} y_i^s f_{it} = 0, \qquad i \in I,\ t \in N_i^+, \tag{19}$$

$$\sum_{t \in N_i^+} y_i^s f_{it} - b_i y_i^s \le 0, \qquad i \in I,\ s \in S_i, \tag{20}$$

$$f_{jt} \ge 0, \qquad t \in T,\ j \in N_t^-,$$

$$0 \le y_i^s \le 1, \qquad i \in I,\ s \in S_i.$$

The model makes use of flow variables f_{it} only along arcs entering terminals. In contrast, the flow along arc $(s, i) \in A \cap (S \times I)$ is represented by $y_i^s \sum_{t \in N_i^+} f_{it}$. Multiplying y_i^s by the total flow through i gives the flow along (s, i), justifying the suggested flow representation.

Consequently, the first sum in the objective function covers cost of flow from sources to pools. Constraints (14)–(16) represent flow capacities at sources, pools and terminals, respectively, and (17) gives the quality constraint at the terminals. Constraint (18) follows directly from the definition of the proportion variables. Multiplying (18) by f_{it} yields (19), which hence is redundant. Similarly, (20) is obtained by multiplying (15) by y_i^s. Sahinidis and Tawarmalani (2005) have shown that adding the redundant inequalities largely improves the linear relaxation and accelerates search algorithms.

4.2 The MCF-formulation for general network instances

In order to generalize the PQ-formulation to arbitrary networks, we apply the set of variables defined for the multi-commodity network flow problem. We associate a flow commodity with each source $s \in S$, where at most b_s units of the commodity can enter the network. The commodity can leave the network at any $t \in T$, whereas at all other nodes, the commodity neither enters nor leaves the network. Since there is a bijection between the sets of commodities and sources, we will whenever appropriate refer to the commodity that enters at source s as commodity s. Hence, for all $(i, j) \in A$ and all $s \in S$, we let variable x^s_{ij} denote the flow of commodity s along arc (i, j).

We keep the variable f_{ij} denoting total flow of all commodities along arc $(i, j) \in A$, and let the variable y^s_i denote the proportion of the total flow leaving node $i \in S \cup I$ constituted by commodity s (define the constants $y^s_i = 0$ for $s \notin S_i$ and $y^s_s = 1$). To make the f-, y-, and x-variables consistent, we impose $x^s_{ij} = y^s_i f_{ij}$.

This results in the following formulation of the generalized pooling problem.

$$\text{[MCF]} \qquad \min_{f,y,x} \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{21}$$

$$\text{s.t.} \qquad (1)\text{–}(2),$$

$$\sum_{j\in N_i^-} x^s_{ji} - \sum_{j\in N_i^+} x^s_{ij} = 0, \qquad i \in I,\ s \in S_i, \tag{22}$$

$$\sum_{j\in N_t^-} \sum_{s\in S_j} \left(q^k_s - q^k_t\right) x^s_{jt} \le 0, \qquad t \in T,\ k \in K, \tag{23}$$

$$\sum_{s\in S_i} y^s_i = 1, \qquad i \in I, \tag{24}$$

$$\sum_{s\in S_i} x^s_{ij} - f_{ij} = 0, \qquad (i, j) \in A,\ i \in I, \tag{25}$$

$$\sum_{j\in N_i^+} x^s_{ij} - y^s_i b_i \le 0, \qquad i \in I,\ s \in S_i, \tag{26}$$

$$x^s_{ij} - y^s_i f_{ij} = 0, \qquad (i, j) \in A,\ s \in S_i, \tag{27}$$

$$f_{ij} \ge 0, \qquad (i, j) \in A, \tag{28}$$

$$0 \le y^s_i \le 1, \qquad i \in I,\ s \in S_i. \tag{29}$$

This is recognized as the multi-commodity minimum cost flow problem with the additional constraints (23) on the quality at the terminals, and the constraints (27) imposing the flow proportions y^s_i on all arcs with start node i. Bilinear terms occur exclusively in (27). Analogous to (19)–(20), constraints (25)–(26) are redundant, but are added for the same reason as (19)–(20) were added to the PQ-formulation.

We define two bilinear formulations to be *equivalent* if they have identical sets of bilinear terms, and, for all hyper-rectangles C, $z^{F_1}[C] = z^{F_2}[C]$. In the sense of this definition, the following result shows that the MCF-formulation generalizes the PQ-formulation.

Proposition 1 *If $A \subseteq (S \times I) \cup (I \times T) \cup (S \times T)$, then the PQ- and MCF-formulations are equivalent.*

Proof Because of the assumed network structure, (27) introduces a bilinear term $y_i^s f_{it}$ if and only if $(s, i, t) \in S \times I \times T$ such that $(s, i), (i, t) \in A$. It is easily verified that these are exactly the bilinear terms occurring in (13), (14), (17), (19) and (20). Hence, both formulations associate a bilinear term $y_i^s f_{it}$ with each path (s, i, t) in G.

The relaxations $LP^{PQ}[C]$ and $LP^{MCF}[C]$ are now obtained by replacing all occurrences of $y_i^s f_{it}$ by some new variable $\hat{x}_{it}^s$, and by adding the corresponding McCormick inequalities. Hence, (27) becomes $\hat{x}_{it}^s = x_{it}^s$, and because of (25), we can substitute f_{it} by $\sum_{s\in S_i} \hat{x}_{it}^s$ everywhere in $LP^{MCF}[C]$ except from the McCormick inequalities. Likewise, (19) is transformed to $f_{it} = \sum_{s\in S_i} \hat{x}_{it}^s$, which allows the substitution of f_{it} everywhere but the McCormick inequalities in $LP^{PQ}[C]$. We observe that the two relaxations this way become identical, and $z^{PQ}[C] = z^{MCF}[C]$ follows. □

4.3 Strength of the MCF-formulation

In the remainder of the paper, we turn the attention to the general version of Problem 1, i.e. the assumption made in Proposition 1 will no longer be considered.

When comparing the strength of the P- and MCF-formulations, we face the challenge that the two formulations share only the f-variables. The assumptions about the bounds on respectively the w-variables (in [P]) and the y-variables (in [MCF]) must be consistent. To this end, we apply the bounds $\underline{w}_i^k = \min_{s\in S_i} q_s^k \le w_i^k \le \overline{w}_i^k = \max_{s\in S_i} q_s^k$ $(i \in I, k \in K)$, and $0 \le y_i^s \le 1$ $(i \in I, s \in S_i)$, respectively. The bounds on the flow variables are identical in the formulations: $\underline{f}_{ij} \le f_{ij} \le \overline{f}_{ij}$ $((i, j) \in A)$. We denote the corresponding hyper-rectangles C_P and C_{MCF}, respectively.

Proposition 2 below shows that the MCF-formulation is stronger than the P-formulation also in GPP. The result is a corollary of Proposition 9.1 in the book by Tawarmalani and Sahinidis (2002).

Proposition 2 $z^P[C_P] \le z^{MCF}[C_{MCF}]$.

Proof Assume (f, y, x) is a feasible solution to $LP^{MCF}[C_{MCF}]$. Following the idea of the proof of Proposition 9.1 by Tawarmalani and Sahinidis (2002), we show that letting $w_i^k = \sum_{s\in S_i} q_s^k y_i^s$ for all $i \in N$, $k \in K$, and $v_{ij}^k = \sum_{s\in S_i} q_s^k x_{ij}^s$ for all $(i, j) \in A$, $k \in K$, implies that (f, w, v) is feasible in $LP^P[C_P]$.

Obviously, (1)–(2) and $f \ge 0$ are satisfied. Further, (3) follows from (22) and (25), (6) follows by summing (22) multiplied by q_s^k over all $s \in S_i$, and (7) follows from (23) and (25). To show that the McCormick inequalities (9)–(12) hold, we first observe that $x_{ij}^s \ge \underline{f}_{ij} y_i^s + f_{ij} \underline{y}_i^s - \underline{f}_{ij} \underline{y}_i^s \ge \underline{f}_{ij} y_i^s$. Since $\underline{w}_i^k \le q_s^k$ for all $s \in S_i$ and $k \in K$, and $\sum_{s\in S_i} y_i^s = 1$, we have

$$\underline{w}_i^k f_{ij} + w_i^k \underline{f}_{ij} - \underline{w}_i^k \underline{f}_{ij} = \underline{w}_i^k \left(f_{ij} - \underline{f}_{ij}\right) + w_i^k \underline{f}_{ij}$$

$$= \underline{w}_i^k \sum_{s\in S_i} \left(x_{ij}^s - y_i^s \underline{f}_{ij}\right) + \underline{f}_{ij} \sum_{s\in S_i} q_s^k y_i^s \le \sum_{s\in S_i} q_s^k \left(x_{ij}^s - y_i^s \underline{f}_{ij}\right) + \underline{f}_{ij} \sum_{s\in S_i} q_s^k y_i^s = v_{ij}^k.$$

In an analogous manner, (10) follows from $x_{ij}^s \le \overline{f}_{ij} y_i^s$, $\overline{w}_i^k \ge q_s^k$, and (24), (11) follows from $x_{ij}^s \le \overline{f}_{ij} y_i^s$, $\underline{w}_i^k \le q_s^k$, and (24), and finally, (12) follows from $x_{ij}^s \ge \underline{f}_{ij} y_i^s$, $\overline{w}_i^k \ge q_s^k$, and (24).

The proof is complete by observing that the objective functions (4) and (21) are identical. $\square$

4.4 A hybrid model

Audet et al. (2004) suggested a model combining quality and proportion variables. In order to avoid terms where proportion variables are squared, which will be the result of a straightforward generalization of the PQ-formulation, they introduce proportion variables for pools that only have sources as in-neighbors, and quality variables for the remaining pools. Denote the former subset of pools $I_1 = \{i \in I : N_i^- = S_i\}$.

The authors formulate their model in terms of the instance depicted in Fig. 1, where both pools (nodes 4 and 5) have capacity 20. In agreement with the PQ-formulation, there is no explicit flow variable for arcs linking S to I_1, which in the given instance amounts to arcs (1,4) and (2,4), since respectively $y_4^1(f_{45} + f_{47})$ and $y_4^2(f_{45} + f_{47})$ represent flow along these arcs. For the purpose of keeping the number of bilinear terms low, Audet et al. (2004) also exploit the constraints (18) at node 4 in order to substitute one proportion variable. In our notation, without the suggested variable substitution, their formulation reads:

$$\begin{array}{rc}
\min\limits_{f,w,y} & -3f_{16} + 6y_4^1(f_{45} + f_{47}) \\
& +16y_4^2(f_{45} + f_{47}) - 3f_{37} \\
& +10f_{35} - 9f_{56} - 13f_{47} - 14f_{58} + 0f_{45}, \\
\text{supply at 1:} & f_{16} + y_4^1(f_{45} + f_{47}) \le 18, \\
\text{supply at 2:} & y_4^2(f_{45} + f_{47}) \le 18, \\
\text{supply at 3:} & f_{35} + f_{37} \le 18, \\
\text{capacity at 4:} & f_{45} + f_{47} \le 20, \\
\text{capacity at 5:} & f_{56} + f_{58} \le 20, \\
\text{demand at 6:} & f_{16} + f_{56} \le 10, \\
\text{demand at 7:} & f_{37} + f_{47} \le 15, \\
\text{demand at 8:} & f_{58} \le 20, \\
\text{quality balance at 5:} & 2f_{35} + (3y_4^1 + y_4^2)f_{45} = w_5^1(f_{56} + f_{58}) \\
\text{quality bound at 6:} & 3f_{16} + w_5^1 f_{56} \le 2.5(f_{16} + f_{56}), \\
\text{quality bound at 7:} & 2f_{37} + (3y_4^1 + y_4^2)f_{47} \le 1.75(f_{37} + f_{47}), \\
\text{quality bound at 8:} & w_5^1 \le 1.5, \\
\text{sum of proportions at 4:} & y_4^1 + y_4^2 = 1, \\
& y_4^1, y_4^2, f_{16}, f_{35}, f_{37}, f_{45}, f_{47}, f_{56}, f_{58} \ge 0.
\end{array}$$

Unfortunately, the constraint representing the quality bound at terminal 8 is too strict, and renders the formulation slightly incorrect. Although terminal 8 has pool 5 as its unique in-neighbor, it is not correct to transfer the quality bound to the pool. By doing so, all feasible solutions with zero flow along arc (5,8) and $w_5^1 > 1.5$ are incorrectly excluded. The error is corrected by replacing $w_5^1 \le 1.5$ by $w_5^1 f_{58} \le 1.5 f_{58}$. Such complementarity constraints

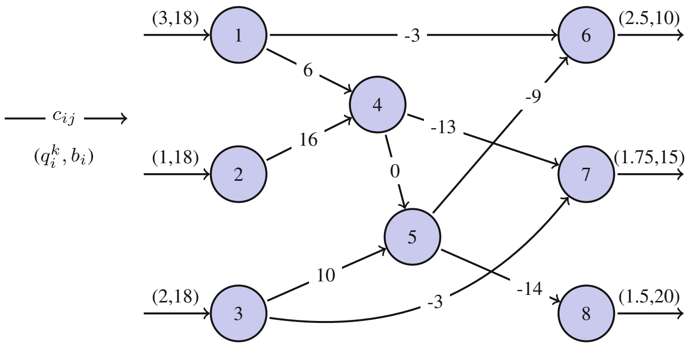

Fig. 1 Instance studied by Audet et al. (2004)

illustrate the binary nature of arcs linking terminals with only one neighbor: Either the arc must be closed or the quality at the pool must satisfy the bound at the terminal.

With the given data, it is however optimal to assign positive flow to arc (5,8), and the suggested formulation finds the optimal flow pattern $f_{16} = 20/3$, $f_{35} = 84/11$, $f_{37} = 114/11$, $f_{45} = 136/11$, $f_{47} = 51/11$, $f_{56} = 10/3$, $f_{58} = 50/3$, $y_4^1 = 13/136$, $y_4^2 = 123/136$, and $w_5^1 = 3/2$ (all other variables are zero at optimum). The corresponding cost is $-5621/132 \approx -42.58$, whereas Audet et al. (2004) incorrectly report the minimum cost to be -60.5.

The flaw (quality bound at 8) in the formulation of Audet et al. (2004) becomes active after a slight modification of the input data. If the supplies at sources 1 and 3 are increased to 30, the demands at terminals 6 and 7 are increased to 50 and 45, respectively, and the costs at arcs (1,6) and (5,6) are decreased to -6 and -12, respectively, we have $w_5^1 = 7/4$ and $f_{58} = 0$ in the optimal solution. The minimum cost is -220, whereas the formulation suggested by Audet et al. (2004) concludes with a solution of cost -210.

Below, we formulate the hybrid model in more general terms. For each pool which in the sense defined above is close to the sources, the hybrid model makes use of a proportion variable y_i^s for each neighboring source s. For other pools, a quality variable w_i^k for each attribute $k \in K$ is used. We repeat the principle of isolating the bilinear terms in dedicated constraints, which requires the introduction of variables $v_{ij}^k = w_i^k f_{ij}$ for all $k \in K$ and arcs (i, j) where $i \in I \setminus I_1$, and $x_{ij}^s = y_i^s f_{ij}$ for all $s \in S_i$ and arcs (i, j) where $i \in I_1$.

$$\text{[HYB]} \quad \min_{f,y,w,x,v} \sum_{s\in S} \left(\sum_{j\in N_s^+\setminus I_1} c_{sj} f_{sj} + \sum_{i\in N_s^+\cap I_1} \sum_{j\in N_i^+} c_{si} x_{ij}^s \right) + \sum_{i\in I} \sum_{j\in N_i^+} c_{ij} f_{ij} \tag{30}$$

$$\text{s.t.} \quad \sum_{j\in N_s^+\setminus I_1} f_{sj} + \sum_{i\in N_s^+\cap I_1} \sum_{j\in N_i^+} x_{ij}^s \le b_s, \quad s \in S, \tag{31}$$

$$\sum_{j\in N_i^+} f_{ij} \le b_i, \quad i \in I, \tag{32}$$

$$\sum_{i\in N_t^-} f_{it} \le b_t, \quad t \in T, \tag{33}$$

$$\sum_{j\in N_i^+} f_{ij} - \sum_{j\in N_i^-} f_{ji} = 0, \quad i \in I \setminus I_1, \tag{34}$$

$$\sum_{s \in N_i^- \cap S_i} q_s^k f_{si} + \sum_{j \in N_i^- \cap I_1} \sum_{s \in S_j} q_s^k x_{ji}^s + \sum_{j \in N_i^- \cap (I \setminus I_1)} v_{ji}^k$$
$$- \sum_{j \in N_i^+} v_{ij}^k = 0, \quad i \in I \setminus I_1,\ k \in K, \tag{35}$$
$$\sum_{s \in N_t^- \cap S_t} q_s^k f_{st} + \sum_{j \in N_t^- \cap I_1} \sum_{s \in S_j} q_s^k x_{jt}^s + \sum_{j \in N_t^- \cap (I \setminus I_1)} v_{jt}^k$$
$$-q_t^k \sum_{j \in N_t^-} f_{jt} \le 0, \quad t \in T,\ k \in K, \tag{36}$$
$$\sum_{s \in S_i} y_i^s = 1, \quad i \in I_1, \tag{37}$$
$$\sum_{s \in S_i} x_{ij}^s - f_{ij} = 0, \quad (i, j) \in A,\ i \in I_1, \tag{38}$$
$$\sum_{j \in N_i^+} x_{ij}^s - y_i^s b_i \le 0, \quad i \in I_1,\ s \in S_i, \tag{39}$$
$$x_{ij}^s - y_i^s f_{ij} = 0, \quad (i, j) \in A,\ i \in I_1,\ s \in S_i, \tag{40}$$
$$v_{ij}^k - w_i^k f_{ij} = 0, \quad (i, j) \in A,\ i \in I \setminus I_1,\ k \in K, \tag{41}$$
$$f_{ij} \ge 0, \quad (i, j) \in A,\ j \in N \setminus I_1, \tag{42}$$
$$0 \le y_i^s \le 1,\ i \in I_1,\ s \in S_i. \tag{43}$$

Let C_{HYB} be the hyper-rectangle defined by the bounds on all bilinear variables in (30)–(43) as defined in Proposition 2.

Proposition 3 $z^P [C_P] \le z^{HYB} [C_{HYB}] \le z^{MCF} [C_{MCF}]$.

Proof The proof is analogous to the proof of Proposition 2. □

5 Computational comparisons

The theoretical part of this work demonstrates that the MCF-formulation is stronger than the HYB-formulation, which in its turn is stronger than the P-formulation. Computational experiments are needed to quantify the differences between the strengths. Experiments are also useful when evaluating whether, and to what extent, stronger formulations lead to faster computation of a narrow interval enclosing the global optimum. To this end, we report in this section the lower bounds obtained by solving the linear relaxations of all three formulations applied to a number of instances. For a comparison of the global optimization capabilities, we have applied BARON 9.3.1 (Sahinidis 1996) as a global solver to the same set of formulations and instances. All experiments reported in this work were conducted on a computer equipped with quad-core 3.00 GHz processors, where each group of four cores share 8 GB of memory.

5.1 Instances

We have tested the formulations in question on 40 instances divided into two sets. The first set consists of standard pooling problem instances from the literature, which have been extended

Table 1 Characteristics of extended instances from the literature

Instance	Number of nodes, arcs and attributes				
	$\|S\|$	$\|I\|$	$\|T\|$	$\|A\|$	$\|K\|$
L1	3	2	3	9	1
L2	5	2	4	15	4
L3	5	2	4	15	6
L4	8	3	4	26	6
L5	8	2	5	20	4
L6	4	2	2	10	1
L7	5	3	5	38	2
L8	6	2	4	22	1
L9	11	8	16	216	1
L10	11	8	16	216	1
L11	11	4	16	108	1
L12	3	2	2	9	1
L13	3	2	2	9	1
L14	3	2	2	9	1
L15	3	2	3	18	8

such that all networks contain directed paths intersecting more than one pool. The instances in the second set are generated randomly.

5.1.1 Extended instances from the literature

The instance depicted in Fig. 1, henceforth denoted L1, already has an arc between pools, and is therefore not extended. In addition to L1, the first instance set consists of extensions of Examples 1–4 introduced by Adhya et al. (1999) (denoted L2–5), extensions of Problems 4–5 introduced by Ben-Tal et al. (1994) (denoted L6–7), extensions of Examples 2–5 introduced by Foulds et al. (1992) (denoted L8–11), extensions of the instances introduced by Haverly (1978) (denoted L12–14), and an extension of RT2 introduced by Audet et al. (2004) (denoted L15).

The extensions are made as follows. For each pair of pools $\{i, j\} \subseteq I$, where $i \neq j$, we add the arcs (i, j) and (j, i). This does however not produce any extension of instances with only one pool, that is, the instances of Haverly (1978) and Problem 4 (Ben-Tal et al. 1994). In such instances, we therefore first add a new pool i_s for each source s from which there is an arc to some terminal. We then add the arc (s, i_s), and each arc $(s, t) \in A$ where $t \in T$ is replaced by (i_s, t). Finally, a directed clique in the new set of pools is constructed as explained above. Table 1 reports the resulting node set cardinalities, the number of arcs, and the number of quality attributes in each new instance.

5.1.2 Random instances

The set of randomly generated instances is divided into 5 groups (denoted A, B, C, D and E, respectively), consisting of 5 instances each. All instances within each group have identical number of sources, pools, terminals, and quality attributes.

Table 2 Characteristics of randomly generated instances

Group	#instances	Number of nodes and attributes				Range of expected density
		$\|S\|$	$\|I\|$	$\|T\|$	$\|K\|$	
A	5	3	2	3	2	0.70–0.90
B	5	5	4	3	3	0.70–0.90
C	5	8	6	6	4	0.50–0.70
D	5	12	10	8	5	0.40–0.60
E	5	10	10	15	12	0.40–0.60

Arcs are introduced randomly in all instances in groups A–D in such a way that the network becomes acyclic. To this end, the pools are ordered $i_1, \ldots, i_{|I|}$, and arcs from i_k $(k = 2, \ldots, |I|)$ to any of $i_1, \ldots, i_{k-1}$ are avoided. For all other pairs of nodes (i, j), an arc from node i to node j may be introduced if $i \in S \cup I$ and $j \in I \cup T$. The probability of doing so is given by an instance specific parameter referred to as the *expected network density*. In the instances in group E, arcs are introduced in an analogous way, with the only difference that directed cycles of pools are allowed.

The arc costs are defined as $c_{ij} = d_i - d_j$ for all $(i, j) \in A$, where $d_i = 0$ for all pools $i \in I$. For sources and terminals, we let d_i be a randomly generated integer in the domains $\{0, \ldots, 5\}$ and $\{5, \ldots, 14\}$, respectively. All outcomes in a domain have equal probabilities. Similarly, the flow capacities, source qualities and quality bounds, are generated randomly from the domains $\{20, \ldots, 59\}$, $\{0, \ldots, 9\}$, and $\{2, \ldots, 6\}$, respectively. Table 2 reports the node set cardinalities, the number of quality attributes and the range of the expected network densities for each group of instances.

5.2 Comparing the strength of the relaxations

Since the extension procedure suggested in Sect. 5.1.1 implies that every pool gets another pool as its in-neighbor, we get $I_1 = \emptyset$ in all extended instances. It then follows from (30)–(43) that no proportion variables are defined in the HYB-formulation, and that the HYB-formulation thus degenerates to the P-formulation. Consequently, we report results for these formulations jointly for instances L2–15 and separately for all other instances.

In Tables 3 and 4, where the first column contains instance identifiers, we report the size of each test instance with the P-, HYB-, and MCF-formulations. The size is measured in terms of the number of variables (vars), the number of distinct bilinear terms (nlts), and the number of linear constraints (lcs). We have not provided the number of nonlinear constraints, because in all formulations in question, we have replaced each occurrence of a bilinear term by a new variable, and added a constraint equating the variable with the bilinear term. Hence, the number of nonlinear constraints equals the number of distinct bilinear terms.

For each formulation F, Tables 3 and 4 also report the optimal objective function value $z^F\,[C_F]$ of the relaxation referred to in Proposition 3. This number is given in bold if it is superior to the corresponding values obtained by the other formulations.

It is easily verified that when $|I||K| < \sum_{i \in I} |S_i|$, which is the case in L1–2, L4–14 and all instances in groups A–D, the number of bilinear terms is larger in model [MCF] than in its two competitors. When $|I||K| > \sum_{i \in I} |S_i|$, which is true for L3, L15 and in all instances in group E, the MCF-formulation introduces fewer bilinear terms than the formulations involving quality variables.

Table 3 Instance size and relaxed optimal objective function value of the P-, HYB-, and MCF-formulations for extended instances from the literature

Instance	P- and HYB-formulations				MCF-formulation			
	vars	nlts	lcs	z^P/z^{HYB}	vars	nlts	lcs	z^{MCF}
L2	63	40	37	−999.32	75	50	59	**−853.47**
L3	87	60	49	−854.10	75	50	67	**−574.78**
L4	152	108	60	−882.84	194	144	108	**−574.78**
L5	76	48	45	−1032.50	132	96	81	**−972.44**
L6	18	6	14	−650.00	42	24	34	**−550.00**
L7	86	42	32	−3500.00	134	84	71	−3500.00
L8	34	10	20	−1200.00	70	40	44	**−1100.00**
L9	408	184	67	−8.00	2328	2024	419	−8.00
L10	408	184	67	−8.00	2328	2024	419	−8.00
L11	188	76	55	−8.00	988	836	215	−8.00
L12	17	6	13	−600.00	33	18	29	**−500.00**
L13	17	6	13	−1200.00	33	18	29	**−1000.00**
L14	17	6	13	−875.00	33	18	29	−875.00
L15	98	64	53	−6331.73	48	24	57	**−6034.87**

All the test instances and the models studied in this paper are available in GAMS-format at http://www.ii.uib.no/~mohammeda/gpooling.

The columns of Tables 3 and 4 labeled z^F show that the MCF-relaxation dominates the P- and HYB-relaxations in 26 out of 40 instances. In all other instances, it gives the same lower bound as provided by the other two formulations. The results do not only confirm the theoretical results of Propositions 2 and 3, but also prove the existence of an instance (C1) in which both inequalities of Proposition 3 are strict. Furthermore, in some test instances (A5, D4, E1–3 and E5), $\left|z^{MCF}\right|$ is less than the half of both $\left|z^P\right|$ and $\left|z^{HYB}\right|$.

5.3 Global optimization performance

In order to compare the capabilities of the P-, HYB-, and MCF-formulations to solve the test instances to optimality, we have implemented them in the modeling language GAMS and used the optimizer BARON as global solver. We set the time limit for all formulations to one CPU-hour, and set both the relative and absolute optimality tolerances to 10^{-3}.

For each formulation, Tables 5 and 6 report in the first column (#nodes) the size of the search tree. The second column (lb (time)) gives the lower bound if BARON failed to solve the instance within the time limit, and the CPU-time in parentheses, otherwise. The third column (ub) within each formulation shows the best upper bound found by BARON.

The tables reveal that the MCF-formulation solves all but 2 (L4 and D1) out of 40 instances within the time limit, whereas both the P- and HYB-formulations are successful in this respect in only 22 instances (L1–2, L5–15, A1–5, B1–2 and B4–5). Among these, instances L1, L6–8, L12–14, A1–3, B1–2 and B4–5 could also be solved by all of the formulations in virtually no time (less than a second). This was accomplished without branching in instances L1, L7, A1–3, B1–2 and B4–5. In addition to the instances solved without branching by all

Table 4 Instance size and relaxed optimal objective function value of the P-, HYB-, and MCF-formulations for the randomly generated instances

Instance	P-formulation				HYB-formulation				MCF-formulation			
	vars	nlts	lcs	z^P	vars	nlts	lcs	z^{HYB}	vars	nlts	lcs	z^{MCF}
L1	15	4	15	−43.00	16	6	19	−43.00	24	10	27	−43.00
A1	31	12	20	−1175.00	32	15	25	−1175.00	39	18	34	−1175.00
A2	25	8	17	−641.00	23	8	20	−641.00	28	10	27	−641.00
A3	24	8	19	−420.60	24	10	23	−420.60	30	12	31	−420.60
A4	35	14	20	−669.40	37	18	26	−669.40	44	21	35	**−599.00**
A5	36	14	20	−508.80	34	14	25	−508.80	40	17	33	**−198.00**
B1	69	30	37	−427.37	63	30	46	−427.37	77	35	65	−427.37
B2	88	39	37	−210.00	78	35	45	−210.00	99	47	68	−210.00
B3	98	48	37	−993.33	107	60	46	−993.33	138	80	81	**−932.00**
B4	97	45	37	−912.80	104	55	45	−912.80	135	75	80	−912.80
B5	104	48	37	−439.00	106	53	44	−439.00	138	75	79	−439.00
C1	161	84	74	−1549.56	157	84	83	−1538.41	197	105	141	**−1352.72**
C2	211	116	74	−1222.67	221	134	90	−1222.67	306	194	161	**−682.14**
C3	243	144	74	−2045.75	253	162	95	−2045.75	355	238	170	**−1716.62**
C4	246	140	74	−1535.24	238	140	94	−1535.24	330	208	165	**−1512.10**
C5	266	160	74	−1573.63	280	178	86	−1573.63	414	288	178	**−1071.81**
D1	429	265	130	−2503.40	414	265	151	−2503.40	672	467	315	**−1994.00**
D2	462	270	130	−1883.70	494	317	161	−1883.70	784	538	342	**−1356.51**
D3	478	290	130	−2090.67	479	301	145	−2090.67	795	559	334	**−2071.00**
D4	561	345	130	−1341.70	576	365	143	−1341.70	951	682	355	**−637.86**
D5	586	370	130	−1711.00	586	380	153	−1711.00	1006	736	362	**−1641.80**
E1	1573	1272	345	−1512.10	1573	1272	345	−1512.10	1341	1060	531	**−463.23**

Table 4 continued

Instance	P-formulation				HYB-formulation				MCF-formulation			
	vars	nlts	lcs	z^{P}	vars	nlts	lcs	z^{HYB}	vars	nlts	lcs	z^{MCF}
E2	1500	1212	345	−1480.91	1380	1104	349	−1480.91	1187	926	512	**−556.00**
E3	1803	1464	345	−536.12	1803	1464	345	−536.12	1539	1220	547	**−78.68**
E4	2057	1716	345	−1583.42	2057	1716	345	−1583.42	1751	1430	568	**−891.25**
E5	2144	1776	345	−1892.80	2144	1776	345	−1892.80	1828	1480	573	**−221.35**

Table 5 Computational results from BARON applied to the P/HYB-, and MCF-formulations for extended instances from the literature

Instance	P/HYB-formulation			MCF-formulation		
	#nodes	lb (time)	ub	#nodes	lb (time)	ub
L2	172163	(1489.88)	−549.80	15157	(176.30)	−549.80
L3	189998	−556.38	−549.80	15805	(165.71)	−549.80
L4	56409	−882.84	−561.04	31270	−572.19	−561.04
L5	4320	(61.83)	−877.65	195	(5.04)	−877.65
L6	45	(0.07)	−450.00	25	(0.13)	−450.00
L7	1	(0.05)	−3500.00	1	(0.58)	−3500.00
L8	71	(0.11)	−1100.00	9	(0.10)	−1100.00
L9	0	(1.94)	−8.00	0	(28.41)	−8.00
L10	0	(0.74)	−8.00	11	(157.96)	−8.00
L11	0	(0.60)	−8.00	316	(738.62)	−8.00
L12	73	(0.08)	−400.00	19	(0.06)	−400.00
L13	101	(0.10)	−600.00	21	(0.11)	−600.00
L14	79	(0.08)	−750.00	47	(0.19)	−750.00
L15	2693	(41.46)	−4391.83	11	(0.14)	−4391.83

formulations, the MCF-formulation solved A4–5, B3 and C1–2 without branching in less than one CPU-second. It also solved instances L9, E1, E3 and E5 without branching in less than one CPU-minute.

Three of the instances (L9–11), all of which are extensions of those provided by Foulds et al. (1992), are solved significantly faster if the P-formulation rather than the MCF-formulation is used. For all formulations, the initial LP-bound is tight in these instances, and therefore optimality can be proved by any heuristic method that happens to output the global optimum. It seems as if the heuristic search provided by BARON is more efficient in the case of the P-formulation, at least in instances L10–11. Here, the optimal solution was hit early in the search, whereas the MCF-formulation required branching and several CPU-minutes in order to conclude.

Among instances solved by all formulations, L2, L5, L15, and A4–5 are solved significantly faster by the MCF-formulation. Only in instances L11 and D5 did it need more than 3 CPU-minutes, but the running time was close to 39 CPU-minutes in instance D5. A comparison between the P- and the HYB-formulations shows only small differences. They could solve exactly the same set of instances, and only in instances A4–5 a significantly faster convergence can be achieved by application of the HYB-formulation. Concerning the unsolved instances, the HYB-formulation provides better lower bounds in C1–2, and otherwise the bounds are identical.

Instances L4 and D1 could not be solved by any formulation. The superior lower bounding capabilities of the MCF-formulation (Proposition 3) are illustrated in both of these. In instance D1, we are left with an optimality gap less than 1.1 %, while the gap in instance L4 is close to 2.0 %. Owing to weaker lower bounds, the gaps are significantly larger for the other two formulations.

We have also compared the results in the last column of Table 5 to the results reported in e.g. Adhya et al. (1999). In all of L2–15, the best solution found turns out to have the

Table 6 Computational results from BARON applied to the P-, HYB-, and MCF-formulations for randomly generated instances

Instance	P-formulation			HYB-formulation			MCF-formulation		
	#nodes	lb (time)	ub	#nodes	lb (time)	ub	#nodes	lb (time)	ub
L1	1	(0.02)	−42.58	1	(0.02)	−42.58	1	(0.02)	−42.58
A1	0	(0.02)	−1175.00	0	(0.02)	−1175.00	0	(0.02)	−1175.00
A2	0	(0.02)	−641.00	0	(0.02)	−641.00	0	(0.02)	−641.00
A3	0	(0.02)	−420.60	0	(0.02)	−420.60	0	(0.02)	−420.60
A4	3039	(6.31)	−599.00	619	(1.09)	−599.00	0	(0.02)	−599.00
A5	10577	(25.23)	−198.00	247	(0.57)	−198.00	0	(0.02)	−198.00
B1	1	(0.05)	−427.37	1	(0.04)	−427.37	0	(0.02)	−427.37
B2	0	(0.02)	−210.00	0	(0.03)	−210.00	0	(0.02)	−210.00
B3	168072	−993.33	−932.00	129968	−993.33	−932.00	0	(0.05)	−932.00
B4	0	(0.05)	−912.80	0	(0.03)	−912.80	0	(0.52)	−912.80
B5	1	(0.06)	−439.00	1	(0.08)	−439.00	0	(0.03)	−439.00
C1	51515	−1521.91	−1352.72	49204	−1479.52	−1352.72	0	(0.92)	−1352.70
C2	49226	−1222.67	−673.86	41380	−1220.27	−673.86	1	(0.62)	−673.86
C3	32914	−2045.75	−1716.63	25093	−2045.75	−1716.63	37	(22.00)	−1716.63
C4	21907	−1535.24	−1512.10	15490	−1535.24	−1512.10	109	(35.79)	−1512.10
C5	26661	−1573.63	−1071.81	24931	−1573.63	−1071.81	19	(22.81)	−1071.81
D1	12940	−2503.40	−1932.97	12277	−2503.40	−1911.16	5824	−1994.00	−1973.06
D2	11883	−1883.70	−1356.51	11368	−1883.70	−1356.51	55	(54.74)	−1356.51
D3	10621	−2090.67	−2071.00	10333	−2090.67	−2068.98	118	(97.86)	−2070.33
D4	8686	−1341.70	−637.86	7501	−1341.70	−637.86	127	(94.92)	−637.86
D5	6139	−1711.00	−1641.80	5518	−1711.00	−1641.80	1954	(2324.40)	−1641.80

Table 6 continued

Instance	P-formulation			HYB-formulation			MCF-formulation		
	#nodes	lb (time)	ub	#nodes	lb (time)	ub	#nodes	lb (time)	ub
E1	1483	−1512.10	−461.74	610	−1512.10	−461.74	0	(2.32)	−463.23
E2	762	−1480.91	−550.51	707	−1480.91	−550.51	64	(111.94)	−556.00
E3	766	−536.12	−0.00	1188	−536.12	−0.00	0	(13.91)	−78.68
E4	845	−1583.42	−665.00	755	−1583.42	−219.36	28	(115.52)	−891.25
E5	703	−1892.80	−0.00	858	−1892.80	−0.00	0	(42.59)	−221.35

same cost as the minimum cost in the instances from which they are extended. Note that for L4, the comparison is based upon the solution giving the sharpest upper bound, of which optimality is not proved. The observation shows that the extension described in Sect. 5.1.1 in none of the reported instances, possibly with the exception of L4, introduced arcs that enable cost reductions. The computational burden is nevertheless increased (L2–5, L9–11), as the PQ-formulation solves the original instances in virtually no time.

The experiments reported in this section indicate that in comparison with the two alternative formulations, the MCF-formulation represents a significantly stronger tool for locating the global optimal solution to instances of the pooling problem with multiple layers of pools. Tighter variable bounds provided in the branching process seem to have modest effect when weaker formulations are applied, whereas our formulation translates this into tighter lower bounds on the optimal objective function value, resulting in faster convergence to optimum.

6 Conclusions

In this paper, we have developed a multi-commodity flow formulation for the generalized pooling problem. The proposed model is an extension of the PQ-formulation for the standard pooling problem, which applies only to networks where all directed paths intersect at most one pool.

We have proved that our multi-commodity flow formulation has stronger relaxation than two other formulations from the literature on the generalized pooling problem. We have presented computational experiments with the proposed formulation and its two competitors applied to 15 extensions of instances from the literature and to 25 randomly generated instances with up to 35 nodes and 12 quality attributes. Experiments confirm that the suggested formulation performs better. By submitting our formulation to the global optimizer BARON, all but two instances could be solved to optimality within one CPU hour, whereas in the case of the other formulations, BARON missed the global optimum in 18 instances.

Acknowledgments This research was sponsored by the Norwegian Research Council, Gassco, and Statoil under contract 175967/S30.

References

Adhya, N., Sahinidis, N.V., Tawarmalani, M.: A Lagrangian approach to the pooling problem. Ind. Eng. Chem. Res. **38**(5), 1956–1972 (1999)

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

Alfaki, M., Haugland, D.: Strong formulations for the pooling problem. J. Glob. Optim. (The current issue). doi:10.1007/s10898-012-9875-6 (2012)

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

Ben-Tal, A., Eiger, G., Gershovitz, V.: Global minimization by reducing the duality gap. Math. Program. **63**(1), 193–212 (1994)

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

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

Guisewite, G.M.: Network problems. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization, pp. 609–648. Kluwer, Dordrecht (1995)

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

Haverly, C.A.: Behavior of recursion model-more studies. ACM SIGMAP Bull. **26**, 22–28 (1979)

Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Kluwer, Dordrecht, The Netherlands (2000)

Karuppiah, R., Grossmann, I.E.: Global optimization for the synthesis of integrated water systems in chemical processes. J. Comput. Chem. Eng. **30**(4), 650–673 (2006)

Li, X., Armagan, E., Tomasgard, A., Barton, P.I.: Stochastic pooling problem for natural gas production network design and operation under uncertainty. AIChE J. **57**(8), 2120–2135 (2011)

Main, R.A.: Large recursion models: practical aspects of recursion techniques. In: Ciriani, T., Leachman, R. (eds.) Optimization in Industry: Mathematical Programming and Modeling Techniques in Practice, pp. 241–249. Wiley, New York (1993)

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

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

Misener, R., Floudas, C.A.: Advances for the pooling problem: modeling, global optimization, and computational studies. Appl. Comput. Math. **8**(1), 3–22 (2009)

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

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

Sahinidis, N.V.: BARON: a general purpose global optimization software package. J. Glob. Optim. **8**(2), 201–205 (1996)

Sahinidis, N.V., Tawarmalani, M.: Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. J. Glob. Optim. **32**(2), 259–280 (2005)

Sherali, H.D.: Tight relaxations for nonconvex optimization problems using the reformulation-linearization/convexification technique (RLT). In: Pardalos, P.M., Romeijn, E. (eds.) Handbook of Global Optimization, vol. 2, pp. 1–164. Kluwer, Dordrecht, The Netherlands (2002)

Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer, Dordrecht, The Netherlands (1999)

Sherali, H.D., Adams, W.P., Driscoll, P.J.: Exploiting special structures in constructing a hierarchy of relaxations for 0–1 mixed integer problems. Oper. Res. **46**(3), 396–405 (1998)

Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Kluwer, Dordrecht, The Netherlands (2002)

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