A special case of the generalized pooling problem arising in the mining industry

Natashia Boland[*] Thomas Kalinowski[†] Fabian Rigterink[†] Martin Savelsbergh[*]

April 26, 2016

Abstract

Iron ore and coal are substantial contributors to Australia's export economy. Both are blended products that are made-to-order according to customers' desired product qualities. Mining companies have a great interest in meeting these target qualities since deviations generally result in contractually agreed penalties. This paper studies a variation of the generalized pooling problem (GPP) arising in this context. The GPP is a minimum cost network flow problem with additional bilinear constraints to capture the blending of raw materials. In the variation we study, costs are not associated with network flows but with deviations from target qualities. We propose a bilinear program (BLP) that we solve locally using nonlinear programming solvers to obtain upper bounds. We linearly relax the BLP using McCormick relaxations and solve the resulting linear program (LP) to obtain lower bounds. A computational study on 26 instances, representing a real-life industry setting and having quarterly, half-yearly, annual and triannual planning horizons, shows that even for large-scale BLPs, these bounds can be calculated efficiently.

Keywords Blending, generalized pooling problem, bilinear programming, nonlinear programming

1 Introduction and problem description

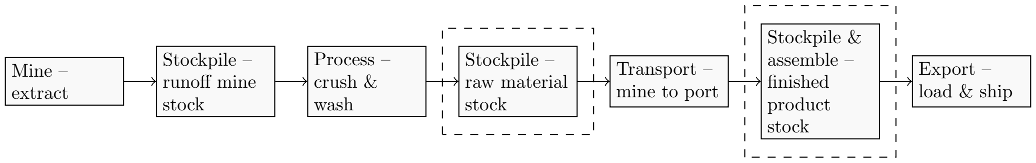
Mine – extract
Stockpile – runoff mine stock
Process – crush & wash
Stockpile – raw material stock
Transport – mine to port
Stockpile & assemble – finished product stock
Export – load & ship

Figure 1: Schematic of a mining supply chain

The aim of this paper is to model the raw material stock blending and finished product stock blending of a mining supply chain, shown in Figure 1. The notation that we use throughout this paper is given in Table 1. The key features of the mining supply chain blending model are as follows.

- After the raw material is mined and processed, it is stored on stockpiles for intermediate storage. We refer to the stockpiles as supply points, or inventories, denoted by $\mathcal{S} := \{1, \dots, S\}$.
- Important properties of the material to be blended are represented as a set of qualities $\mathcal{Q} := \{1, \dots, Q\}$. In the case of coal these may be ash, moisture, sulfur and volatile matter. We assume that for all qualities, customers prefer small quality values.
- For all $i \in \mathcal{S}$, supply comes in at known points in time $t \in \mathcal{T}_i^s := \{t_{i1}^s, \dots, t_{iT_i^s}^s\}$, where $t_{iT_i^s}^s > \dots > t_{i1}^s > 0$. The quantity and quality of incoming supply, s_{it} and q_{ikt} for all $k \in \mathcal{Q}$, respectively, is also known *a priori*. When a raw material is stored on a stockpile, the inventory's quantity and quality change. For

[*]Georgia Institute of Technology, Atlanta, GA, USA
[†]The University of Newcastle, Callaghan, Australia

all $i \in \mathcal{S}$ and $t \in \mathcal{T}_i^s$, let I_{it} capture the quantity stored on inventory i at time t, and for all $k \in \mathcal{Q}$, let x_{ikt}^s capture the value of quality k of that inventory at that time. We assume linear blending, i.e., the inventory's new quality is a weighted linear combination of the inventory's old quality and the quality of the incoming supply. I_{it} and x_{ikt}^s not only incorporate the preceding incoming supply's quantities, $s_{it'}$, and qualities, $q_{ikt'}$, for all $t' \in \mathcal{T}_i^s$, $t' \leq t$, but also the succeeding outgoing demand. It will become apparent in Section 3 why we choose this notation.

- Demand goes out at known points in time $t \in \mathcal{T}^d := \{t_1^d, \ldots, t_{\mathcal{T}^d}^d\}$, where $t_{\mathcal{T}^d}^d > \ldots > t_1^d > 0$. We use the raw material stored on the stockpiles to meet the demand d_t. Let y_{it} be the flow from supply point $i \in \mathcal{S}$ to the demand point at time $t \in \mathcal{T}^d$. Note that $\sum_{i \in \mathcal{S}} y_{it} = d_t$ must hold. For all $k \in \mathcal{Q}$, let x_{kt}^d be the quality after blending the y_{it} flows, i.e. the quality of d_t. Again, we assume linear blending. If demand is met with product that does not satisfy prescribed quality specifications, a penalty is incurred. The penalty is thus a function of x^d. The particular penalty function we use in this paper is described in the next section.

We assume that every raw material flow (may it be as incoming supply, outgoing demand or as a flow between the supply points and the demand point) happens instantaneously. The quantities and qualities are also updated instantaneously. We do not consider any transfer times or costs between the supply points and the demand point. We assume that, at any time, the past, aggregated, incoming supply is larger than or equal to the past, aggregated outgoing demand: demand can always be met. The problem is to decide how much from each supply should be used to meet the demand at each demand time point, so as to minimize the total penalty. As we shall show in Section 3, this problem can be viewed as a *pooling problem*; we call it the *mining pooling problem* (MPP). The pooling problem was first proposed by Haverly [12] in 1978. Since then, an extensive literature has been published. Reviews on the pooling problem and its variations are found in [4, 8, 16]. We also refer the reader to two PhD theses on this subject [10, 15]. There are a number of solution techniques to solve pooling problems such as successive linear programming, the reformulation-linearization technique and linear relaxations. In this paper, we study the latter. Applications are found in chemical engineering, e.g. in petroleum refining [16]. A description of a computational tool that globally optimizes pooling problems, APOGEE, has been published recently [17]. The literature on pooling problems arising in the mining industry is, however, sparse [6, 9, 19].

2 Bilinear programming model

Recall that the objective is to minimize the total penalty that has to be paid by the mining company as a consequence of meeting demand with product having qualities that do not satisfy prescribed product specifications. Here, we use a penalty function that consists of two linear pieces for each quality $k \in \mathcal{Q}$ and for each demand time point $t \in \mathcal{T}^d$. Let u_{kt} be the contractually agreed soft upper bound on quality k for demand time point t. The left piece represents quality values $x_{kt}^d < u_{kt}$ for which no penalty has to be paid. The right piece represents quality values $x_{kt}^d \geq u_{kt}$, for which the mining company has to pay a nonnegative per unit penalty p_{kt}. We define the excess quality function $e_{kt}(x_{kt}^d) := \max\{0, x_{kt}^d - u_{kt}\}$ to be the deviation of x_{kt}^d from u_{kt} if $x_{kt}^d \geq u_{kt}$, and 0 otherwise. The penalty incurred from quality k in meeting demand d is then given by the per unit penalty p_{kt} multiplied by $e_{kt}(x_{kt}^d)$, the deviation of the quality value x_{kt}^d from its soft upper bound u_{kt}, multiplied by the demand d_t (hence a *per unit* penalty). This yields the piecewise affine convex objective function

$$g(x^d) = \sum_{k \in \mathcal{Q}} \sum_{t \in \mathcal{T}^d} p_{kt} e_{kt}(x_{kt}^d) d_t.$$

Penalty constraint: The objective function above is modelled linearly by the introduction of new variables, δ_{kt}^u, for each $k \in \mathcal{Q}$ and $t \in \mathcal{T}^d$, that are required to satisfy the following constraints:

$$\delta_{kt}^u \geq 0, \qquad \forall k \in \mathcal{Q},\ t \in \mathcal{T}^d, \tag{1a}$$

$$\delta_{kt}^u \geq x_{kt}^d - u_{kt}, \qquad \forall k \in \mathcal{Q},\ t \in \mathcal{T}^d. \tag{1b}$$

Table 1: Overview of the notation

Sets and indices

$i \in \mathcal{S} := \{1, \ldots, S\}$	Supply points
$k \in \mathcal{Q} := \{1, \ldots, Q\}$	Qualities
$t \in \mathcal{T}_i^s := \{t_{i1}^s, \ldots, t_{iT_i^s}^s\}$	Time points at which supply comes in at $i \in \mathcal{S}$
$t \in \mathcal{T}^d := \{t_1^d, \ldots, t_{T^d}^d\}$	Time points at which demand goes out

Variables

Inventory and flow variables	
I_{it}	Inventory at $i \in \mathcal{S}$ at $t \in \mathcal{T}_i^s$
y_{it}	Flow from $i \in \mathcal{S}$ to the demand point at $t \in \mathcal{T}^d$
Quality variables	
x_{ikt}^s	Quality value of $k \in \mathcal{Q}$ of I_{it}
x_{kt}^d	Quality value of $k \in \mathcal{Q}$ of d_t
δ_{kt}^u,	Deviation of x_{kt}^d from u_{kt} if $x_{kt}^d \geq u_{kt}$, else 0

Parameters

Inventory and flow parameters	
I_{i0}	Initial inventory at $i \in \mathcal{S}$
s_{it}	Incoming supply at $i \in \mathcal{S}$ at $t \in \mathcal{T}_i^s$
d_t	Outgoing demand at $t \in \mathcal{T}^d$
Quality parameters	
x_{ik0}^s	Quality value of $k \in \mathcal{Q}$ of I_{i0}
u_{kt}	Soft upper bound on x_{kt}^d
p_{kt}	Per unit penalty to be paid if $x_{kt}^d \geq u_{kt}$
q_{ikt}	Quality value of $k \in \mathcal{Q}$ of s_{it}

Functions

$P(i,t) := \{t' \in \mathcal{T}_i^s : \; t' < t\}$ $p(i,t) := \begin{cases} \max\{P(i,t)\} & \text{if } \; P(i,t) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$	Predecessor(s) within $\mathcal{T}_i^s$, $\; i \in \mathcal{S}$: $P(i,t) \subset \mathcal{T}_i^s$ is the set of supply time points in $\mathcal{T}_i^s$ preceding $t \in \mathcal{T}_i^s$ and $p(i,t) \in \mathcal{T}_i^s$ is the immediate predecessor to t.
$S(i,t) := \{t' \in \mathcal{T}_i^s : \; t' > t\}$ $s(i,t) := \min\{S(i,t)\}$	Successor(s) within $\mathcal{T}_i^s$, $\; i \in \mathcal{S}$: $S(i,t) \subset \mathcal{T}_i^s$ is the set of supply time points in $\mathcal{T}_i^s$ succeeding $t \in \mathcal{T}_i^s$ and $s(i,t) \in \mathcal{T}_i^s$ is the immediate successor to t (provided that $t \neq \max\{\mathcal{T}_i^s\} \Leftrightarrow S(i,t) \neq \emptyset$).
$P_{\mathcal{T}_i^s}(t) := \{t' \in \mathcal{T}_i^s : \; t' \leq t\}$ $p_{\mathcal{T}_i^s}(t) := \max\{P_{\mathcal{T}_i^s}(t)\}$	Cross-predecessor(s): $P_{\mathcal{T}_i^s}(t) \subset \mathcal{T}_i^s$ is the set of supply time points in $\mathcal{T}_i^s$ preceding the demand time point $t \in \mathcal{T}^d$ and $p_{\mathcal{T}_i^s}(t) \in \mathcal{T}_i^s$ is the immediate predecessor to $t \in \mathcal{T}^d$.
$S_{\mathcal{T}^d}(i,t) := \begin{cases} \{t' \in \mathcal{T}^d : \; t \leq t' < s(i,t)\} \\ \quad \text{if } \; S(i,t) \neq \emptyset \\ \{t' \in \mathcal{T}^d : \; t \leq t'\} \\ \quad \text{otherwise} \end{cases}$ $s_{\mathcal{T}^d}(i,t) := \min\{S_{\mathcal{T}^d}(i,t)\}$	Cross-successor(s): $S_{\mathcal{T}^d}(i,t) \subset \mathcal{T}^d$ is the set of demand time points in $\mathcal{T}^d$ succeeding the supply time point $t \in \mathcal{T}_i^s$ and $s_{\mathcal{T}^d}(i,t) \in \mathcal{T}^d$ is the immediate successor to $t \in \mathcal{T}_i^s$ (provided that $t \not> \max\{\mathcal{T}^d\} \Leftrightarrow S_{\mathcal{T}^d}(i,t) \neq \emptyset$).

The optimization model requires the function

$$f(\delta^u) = \sum_{k\in\mathcal{Q}} \sum_{t\in\mathcal{T}^d} p_{kt}\delta^u_{kt} d_t$$

to be minimized. This objective, together with constraints (1a) and (1b), and nonnegativity of all p_{kt}, ensure that, in any optimal solution, $\delta^u_{kt} = \max\{0, x^d_{kt} - u_{kt}\}$, for each $k \in \mathcal{Q}$ and $t \in \mathcal{T}^d$.

Supply side inventory constraints: When supply comes in, the inventory's quantity and quality change. The following constraints capture the change in quantity:

$$I_{it} = I_{it'} + s_{it} - \sum_{t''\in S_{\mathcal{T}^d}(i,t)} y_{it''}, \qquad \forall i \in \mathcal{S},\ t \in \mathcal{T}^s_i,\ t' = p(i,t). \tag{2}$$

Supply side inventory blending constraints: The following constraints capture the change in quality. The inventory's new quality is a weighted linear combination of the inventory's old quality and the quality of the incoming supply:

$$x^s_{ikt} = \frac{x^s_{ikt'} I_{it'} + q_{ikt} s_{it}}{I_{it'} + s_{it}}, \qquad \forall i \in \mathcal{S},\ k \in \mathcal{Q},\ t \in \mathcal{T}^s_i,\ t' = p(i,t). \tag{3}$$

Since each stockpile i is assumed to be instantaneously, perfectly blended at each supply time point, for times t *between* consecutive supply time points, the quality values of a stockpile remain constant: all material taken off the stockpile to meet demand at such times have precisely the quality values at the time the last supply was added, $(x^s_{ikp_{\mathcal{T}^s_i}(t)})_{k\in\mathcal{Q}}$, and these are identical to the quality values of the material remaining in the stockpile after material has been take off to meet demand. Thus quality values need only be calculated at the supply time points as shown above, and these can be used to determine the quality values of material taken off to meet demand, as shown in the next constraint.

Demand side blending constraints: To calculate the quality after blending the y_{it} flows, i.e. the quality of d_t, we linearly blend the preceding inventory qualities weighted by the y_{it} flows (where we use $\sum_{i\in\mathcal{S}} y_{it} = d_t$):

$$x^d_{kt} d_t = \sum_{\substack{i\in\mathcal{S} \\ t'=p_{\mathcal{T}^s_i}(t)}} x^s_{ikt'} y_{it}, \qquad \forall k \in \mathcal{Q},\ t \in \mathcal{T}^d. \tag{4}$$

Bounding constraints: Both I_{it} and y_{it} are nonnegative. I_{it} is bounded above by I^u_{it}, which is deduced from using as little raw material as possible from supply point i to meet demand. y_{it} is bounded above by d_t, which is deduced from using only raw material from supply point i to meet demand. x^s_{ikt} and x^d_{kt} are at least as good as the worst quality and at most as good as the best quality of all incoming supply. Thus, we have

$$\begin{aligned}
0 \le I_{it} &\le I^u_{it}, && \forall i \in \mathcal{S},\ t \in \mathcal{T}^s_i, && \text{(5a)}\\
0 \le y_{it} &\le d_t, && \forall i \in \mathcal{S},\ t \in \mathcal{T}^d, && \text{(5b)}\\
\min\{q_{ikt'} \mid t' \in \mathcal{T}^s_i,\ t' \le t\} \le x^s_{ikt} &\le \max\{q_{ikt'} \mid t' \in \mathcal{T}^s_i,\ t' \le t\}, && \forall i \in \mathcal{S},\ k \in \mathcal{Q},\ t \in \mathcal{T}^s_i, && \text{(5c)}\\
\min\{q_{ikt'} \mid i \in \mathcal{S},\ t' \in \mathcal{T}^s_i,\ t' \le t\} \le x^d_{kt} &\le \max\{q_{ikt'} \mid i \in \mathcal{S},\ t' \in \mathcal{T}^s_i,\ t' \le t\}, && \forall k \in \mathcal{Q},\ t \in \mathcal{T}^d, && \text{(5d)}
\end{aligned}$$

$$\text{where} \qquad I^u_{it} := \sum_{\substack{t'\in\mathcal{T}^s_i \\ t'\le t}} s_{it'} - \max\left\{0, \sum_{\substack{t'\in\mathcal{T}^d \\ t'\le t}} d_{t'} - \sum_{\substack{i'\in\mathcal{S} \\ i'\ne i}} \sum_{\substack{t'\in\mathcal{T}^s_{i'} \\ t'\le t}} s_{i't'}\right\}, \qquad \forall i \in \mathcal{S},\ t \in \mathcal{T}^s_i.$$

We are now ready to formulate the *mining pooling problem* (MPP):

$$\min_{\delta^u, I, y, x^s, x^d} f(\delta^u) \quad \text{s.t.} \quad (1) - (5). \tag{MPP}$$

(3) and (4) are *bilinear functions* with *bilinear terms* of the form Ix^s and yx^s, respectively. Such forms are not convex, thus MPP is a nonconvex, nonlinear program. It can be viewed as a special case of the *generalized pooling problem*, which we define next.

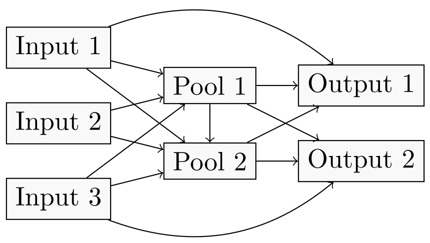

Figure 2: Example of a GPP graph

3 A special case of the generalized pooling problem

We consider an acyclic directed graph $G = (N, A)$, where N is the set of nodes and A the set of arcs. N is partitioned into three nonempty subsets $I, L, J \subset N$: 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 ($(I \times I) = \emptyset$) or two outputs ($(J \times J = \emptyset)$) and no arcs from pools to inputs ($(L \times I) = \emptyset$) or outputs to pools ($(J \times L) = \emptyset$) or outputs to inputs ($(J \times I) = \emptyset$). We further assume that every pool has in-degree and out-degree of at least 1. Similarly, every input (output) has out-degree (in-degree) of at least 1. Problem instances with $A \cap (L \times L) = \emptyset$ are referred to as *standard pooling problems* (SPPs) and as *generalized pooling problems* (GPPs), otherwise. An example of a GPP graph is shown in Figure 2. Both SPPs and GPPs can be modelled as bilinear programs. Problem instances with $L = \emptyset$ are referred to as *blending problems* (BPs). BPs can be modelled as linear programs.

It is worth noting that (2)–(4) constitute the main constraints of what is commonly referred to in the literature as the *p-formulation* (concentration model) of the GPP [10]. There are alternative formulations such as the *q-formulation* (proportion model), the *pq-formulation*, the *hybrid formulation*, and *multi-commodity flow formulations* [2, 3, 4, 7].

The GPP is stated as a *static* problem, to be solved at one point in time, whereas MPP is a *dynamic* problem, deciding how to blend at multiple time points over a planning horizon. Nevertheless, the following construction enables us to represent the MPP using a GPP graph, and hence shows that the MPP can be viewed as a special case of the GPP.

1. **Nodes**
   (a) I **nodes:** For all $i \in \mathcal{S}$, $t \in \mathcal{T}_i^s$, create an input node it.
   (b) L **nodes:** For all $i \in \mathcal{S}$, $t \in \mathcal{T}_i^s$, create a pool node it.
   (c) J **nodes:** For all $t \in \mathcal{T}^d$, create an output node t.
2. **Arcs**
   (a) $(I \times L)$ **arcs:** For all $i \in \mathcal{S}$, $t \in \mathcal{T}_i^s$, create an arc from input node it to pool node it. Flows on these arcs represent s_{it}.
   (b) $(L \times L)$ **arcs:** For all $i \in \mathcal{S}$, $t \in \mathcal{T}_i^s$, $t' = p(i,t)$, create an arc from pool node it' to pool node it. Flows on these arcs represent $I_{it'}$.
   (c) $(L \times J)$ **arcs:** For all $i \in \mathcal{S}$, $t \in \mathcal{T}^d$, $t' = p_{\mathcal{T}_i^s}(t)$, create an arc from pool node it' to output node t. Flows on these arcs represent y_{it}.

4 Linear relaxation of the bilinear program

On the one hand, we can solve MPP as it is with a nonlinear programming solver able to handle nonconvex problems. However, state-of-the-art *global* solvers such as BARON [18] and Couenne [5] are relatively slow and struggle to solve large-scale nonconvex problems. On the other hand, *local* solvers such as Ipopt [20] are fast, but only find locally optimal solutions, which provide upper bounds on the MPP. In the GPP literature, a standard approach to finding lower bounds is to substitute the Ix^s and yx^s terms in constraints (3) and (4) by an auxiliary variable, z, so that (3) and (4) become linear, and the problem assumes the form of a *bilinear program* (BLP):

$$\begin{aligned} \min_{x,y,z} \quad & f(x,y,z) \\ \text{s.t.} \quad & g(x,y,z) \le 0, \\ & h(x,y,z) = 0, \\ & \quad z_{ij} = x_i y_j, \qquad \forall (i,j) \in B, \\ & \quad x^L \le x \le x^U, \\ & \quad y^L \le y \le y^U, \end{aligned} \tag{BLP}$$

where x (y) is a vector of I (J) continuous variables, z_{ij} is the bilinear term of x_i and y_j ($i \in \{1,\ldots,I\}$, $j \in \{1,\ldots,J\}$), $B = \{(i,j) \mid z_{ij} = x_i y_j\}$ is the set of bilinear terms, $f(x,y,z)$ is a linear function and $g(x,y,z)$ and $h(x,y,z)$ are linear vector functions [11].

Then one relaxes the bilinear terms $z_{ij} = x_i y_j$ for all $(i,j) \in B$ individually using so-called McCormick relaxations. Let xy be a bilinear term and let x^L, x^U, y^L and y^U be the lower and upper bounds on x and y, respectively. In [14], McCormick introduces a linear relaxation of xy using the following four inequalities:

$$\begin{aligned} & z \ge xy^L + x^L y - x^L y^L, && z \ge xy^U + x^U y - x^U y^U, \\ & z \le xy^L + x^U y - x^U y^L, && z \le xy^U + x^L y - x^L y^U. \end{aligned}$$

Al-Khayyal and Falk prove in [1] 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 form the convex hull of xy. The convex and concave envelopes (or under- and overestimators) of xy on $[-1,1] \times [-1,1]$ are shown in Figure 3. We will refer to the linear relaxation of MPP obtained in this way as MPP-L.

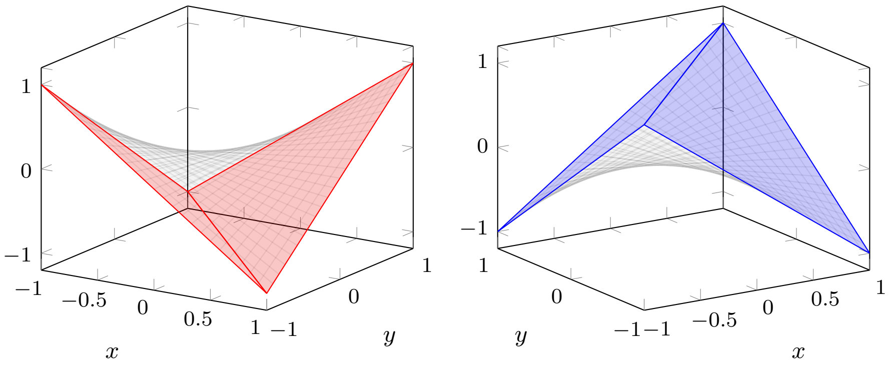

Figure 3: Under- (red) and overestimators (blue) of xy on $[-1,1] \times [-1,1]$

5 Computational study

Our industry partner provided us with an example data set representing supply and demand data (including quality specifications and contract penalties) of a real-life mining company for the time horizon 2009–2012. We split the data into problem instances of years, half-years and quarters. All problem instances share the following characteristics:

- There are two supply points, i.e. $S = 2$.
- There are four qualities: ash, moisture, sulfur and volatile matter, i.e. $Q = 4$.

In the original data set, there are $T_1^s + T_2^s = 4132$ supplies and $T^d = 530$ demands. However, we pre-processed all problem instances to ensure feasibility and to decrease their size as follows:

- Consider a supply point $i \in \mathcal{S}$ and two time points at which supply comes in, $t, t' \in \mathcal{T}_i^s$, $t < t'$. If there exists no time point at which demand goes out in between t and t', i.e. there exists no $t'' \in \mathcal{T}^d$, $t \leq t'' < t'$, then s_{it} and $s_{it'}$ (and their respective quality values q_{ikt} and $q_{ikt'}$) can be equivalently represented as a single incoming supply at time point t'. The quality values of $s_{it'}^{\text{new}} = s_{it} + s_{it'}$ are then calculated as
$$q_{ikt'}^{\text{new}} = \frac{q_{ikt}s_{it} + q_{ikt'}s_{it'}}{s_{it} + s_{it'}}, \qquad \forall k \in \mathcal{Q}.$$
This is true for any number of incoming supplies for which there exists no outgoing demand in between. Applying this pre-processing step reduces T_1^s to 234 and T_2^s to 244.

- If at any time the cumulative incoming supply is smaller than the cumulative outgoing demand (i.e. demand cannot be met), we adjust the demand data. More precisely, we iteratively delete any demand d_t, $t \in \mathcal{T}^d$, where
$$\sum_{i \in \mathcal{S}} \sum_{\substack{t' \in \mathcal{T}_i^s \\ t' \leq t}} s_{it'} < \sum_{\substack{t' \in \mathcal{T}^d \\ t' \leq t}} d_{t'}.$$
This reduces T^d to 349.

Figure 4 shows the quantity of incoming (aggregated) supply and outgoing (aggregated) demand, Figure 5 the quality of incoming supply and soft upper bounds on outgoing demand over time.

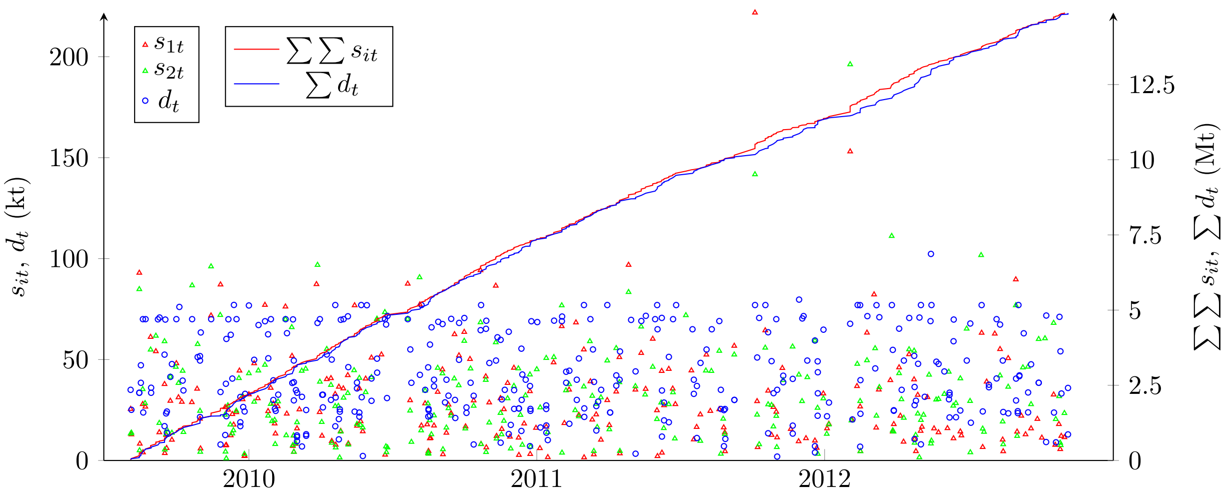

Figure 4: Quantity of incoming (aggregated) supply and outgoing (aggregated) demand over time

The linear problem MPP-L was solved with CPLEX 12.6.0.0 [13], and the nonlinear problem MPP with Ipopt 3.10.2 [20]. Table 2 shows the test results. Our computational study implies that, even for large-scale BLPs, both locally optimal upper bound solutions and McCormick relaxation lower bounds can be calculated efficiently. However, the gap between the lower and upper bounds is in some cases quite large, ranging from 1.7% to 41.9% on the instances tested, with an average of 19.4% (accurate to one decimal place). To close the gap, one may now consider partitioning I, y and x^s to tighten the McCormick relaxations, and combine solution of MPP-L with a branch-and-bound algorithm; this is the subject of future research.

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).

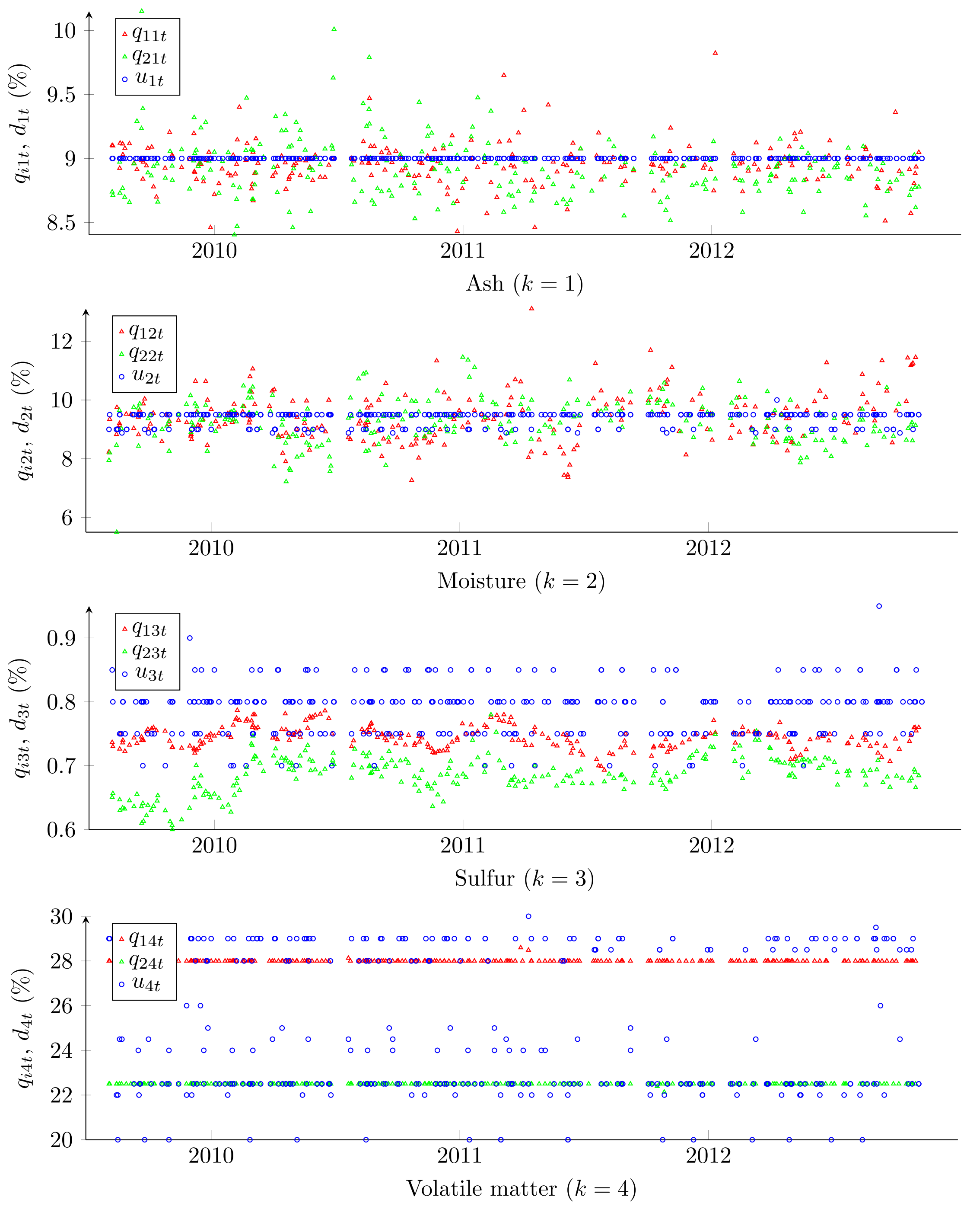

Figure 5: Quality of incoming supply and soft upper bounds on outgoing demand over time

Table 2: Number of variables (#Vars), number of constraints (#Cons), objective value (Obj) and solve time in seconds for all problem instances, MPP-L and MPP

					MPP-L				MPP				
Instance		T_1^s	T_2^s	T^d	#Vars	#Cons	Obj	Time	#Vars	#Cons	Obj	Time	**Gap**
All (2009–2012)		234	244	349	12,480	31,931	35,381,254	7.03	5880	5531	47,319,100	0.02	25%
2009	Year	38	37	49	1841	4720	2,609,254	0.01	865	816	4,141,872	0.01	37%
	Half-year 2	38	37	49	1841	4720	2,609,254	0.01	865	816	4,141,872	0.01	37%
	Quarter 3	17	16	21	791	2018	1,611,342	0.00	375	354	2,281,114	0.00	29%
	Quarter 4	20	20	26	972	2482	1,022,164	0.00	460	434	1,757,963	0.01	42%
2010	Year	83	89	122	4416	11,302	8,747,566	1.05	2080	1958	11,822,217	0.01	26%
	Half-year 1	42	46	63	2262	5775	4,860,413	0.02	1070	1007	7,192,454	0.01	32%
	Half-year 2	42	44	58	2146	5496	3,612,735	0.37	1010	952	4,242,027	0.01	15%
	Quarter 1	20	21	28	1021	2601	2,378,455	0.00	485	457	2,963,570	0.00	20%
	Quarter 2	21	24	30	1109	2831	1,938,588	0.01	525	495	2,999,773	0.00	35%
	Quarter 3	20	21	24	949	2437	1,956,998	0.00	445	421	2,456,252	0.00	20%
	Quarter 4	22	23	31	1127	2872	427,871	0.00	535	504	599,408	0.01	29%
2011	Year	62	61	94	3275	8341	16,672,957	0.63	1555	1461	20,659,190	0.01	19%
	Half-year 1	35	34	49	1763	4498	9,563,052	0.01	835	786	10,517,145	0.01	9%
	Half-year 2	27	28	42	1455	3693	8,507,098	0.01	695	653	10,935,728	0.01	22%
	Quarter 1	18	19	26	933	2371	5,680,948	0.00	445	419	6,348,302	0.00	11%
	Quarter 2	17	15	21	778	1981	3,057,259	0.00	370	349	3,185,907	0.01	4%
	Quarter 3	11	10	14	509	1287	2,037,150	0.00	245	231	2,264,541	0.01	10%
	Quarter 4	15	15	22	770	1948	4,219,145	0.00	370	348	5,028,387	0.01	16%
2012	Year	51	56	78	2779	7093	10,392,413	0.02	1315	1237	11,320,438	0.01	8%
	Half-year 1	30	36	45	1652	4223	7,250,497	0.01	780	735	7,630,918	0.01	5%
	Half-year 2	21	20	31	1075	2724	3,040,012	0.00	515	484	3,385,814	0.01	10%
	Quarter 1	14	14	16	636	1628	1,400,109	0.00	300	284	1,626,699	0.00	14%
	Quarter 2	16	18	22	822	2096	2,917,539	0.00	390	368	2,967,360	0.00	2%
	Quarter 3	14	14	22	744	1874	2,254,115	0.00	360	338	2,395,868	0.00	6%
	Quarter 4	8	8	9	354	897	396,687	0.00	170	161	534,669	0.00	26%

References

[1] F. A. Al-Khayyal and J. E. Falk. Jointly Constrained Biconvex Programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

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

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

[4] 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.

[5] P. Belotti. *COUENNE: a user's manual.* coin-or.org/Couenne/couenne-user-manual.pdf.

[6] A. Bley, N. Boland, G. Froyland, and M. Zuckerberg. Solving mixed integer nonlinear programming problems for mine production planning with stockpiling. Optimization Online e-prints, November 2012. optimization-online.org/DB_HTML/2012/11/3674.html.

[7] N. Boland, T. Kalinowski, and F. Rigterink. New multi-commodity flow formulations for the pooling problem. *Journal of Global Optimization*, 2016. Advance online publication, 42 pages. DOI: 10.1007/s10898-016-0404-x.

[8] S. S. Dey and A. Gupte. Analysis of MILP Techniques for the Pooling Problem. *Operations Research*, 63(2):412–427, 2015.

[9] J. E. Everett. Iron ore production scheduling to improve product quality. *European Journal of Operational Research*, 129(2):355–361, 2001.

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

[11] M. M. F. Hasan and I. A. Karimi. Piecewise Linear Relaxation of Bilinear Programs Using Bivariate Partitioning. *AIChE Journal*, 56(7):1880–1893, 2010.

[12] C. A. Haverly. Studies of the Behavior of Recursion for the Pooling Problem. *SIGMAP Bulletin*, 25:19–28, 1978.

[13] IBM. *IBM ILOG CPLEX Optimization Studio: CPLEX User's Manual, Version 12 Release 6*, 2015.

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

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

[16] 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.

[17] 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.

[18] N. V. Sahinidis. BARON: A General Purpose Global Optimization Software Package. *Journal of Global Optimization*, 8(2):201–205, 1996.

[19] G. Singh, R. García-Flores, A. T. Ernst, P. Welgama, M. Zhang, and K. Munday. Medium-Term Rail Scheduling for an Iron Ore Mining Company. *Interfaces*, 44(2):222–240, 2014.

[20] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2005.