Discrete flow pooling problems in coal supply chains

N. Boland [a], **T. Kalinowski** [b], **F. Rigterink** [b]

[a] *H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology*
[b] *School of Mathematical and Physical Sciences, The University of Newcastle, Australia*
Email: fabian.rigterink@uon.edu.au

Abstract: The pooling problem is a nonconvex nonlinear programming problem (NLP) with applications in the refining and petrochemical industries, but also the coal mining industry. The problem can be stated as follows: given a set of raw material suppliers (inputs) and qualities of the supplies, find a cost-minimising way of blending these raw materials in intermediate pools and outputs so as to satisfy requirements on the output qualities. The blending in two stages (in pools and outputs) introduces bilinear constraints. The pooling problem can alternatively be described as a minimum cost network flow problem with additional bilinear constraints to capture the blending of raw materials.

In this paper we study a variation of the pooling problem that arises naturally in the coal mining industry. Coal is made-to-order according to customers' desired product qualities. Deviations from these target qualities result in contractually agreed bonuses and penalties. In the pooling problem variation we study, costs are associated with these bonuses and penalties instead of network flows. While in the original pooling problem we have hard bounds on the qualities and unmet demand is penalised in the objective function, in our coal mining variation we have hard demand constraints and deviations from target qualities are penalised. This makes finding a feasible solution easy, while in the pooling problem finding a nontrivial feasible solution that satisfies the quality requirements is already hard. An implication of this is that we are able to solve larger problem instances than those typically studied in the pooling problem literature.

To model the coal blending process accurately, we define a time-expanded network where the intermediate pools represent coal stockpiles over time. Since coal is transported in large quantities, we study the trade-off between continuous and discretized flows in coal blending, i.e., solving a continuous flow problem where arbitrarily small flows are allowed versus solving a discretized flow problem where flows must be in multiples of some basic unit, e.g. trainloads. We also study two exact mixed-integer linear programming (MILP) linearizations of these mixed-integer nonlinear programs (MINLPs), which can be derived from unary and binary expansions of the flow integrality constraint. Such discretizations are typically studied as approximations to an originally continuous problem, however, in our application, a discretized formulation describes the original problem more accurately than a continuous formulation.

The paper is organized as follow: in Section 1.1, we introduce the pooling problem formally as it is commonly stated in the literature. In Section 1.2, we formulate the coal supply chain extension of the pooling problem as MINLPs and MILPs. We conclude the paper with Section 2 where we provide computational results for four different formulations which we run for a real-life industry setting.

Keywords: *Coal blending, pooling problem, bilinear programming, nonlinear programming, mixed-integer bilinear programming, mixed-integer nonlinear programming*

Table 1. Notation

(a) Pooling problem

Sets	
V	Set of vertices
I	Set of inputs
L	Set of pools
J	Set of outputs
A	Set of arcs
$\delta^-(v)$	Set of incoming arcs of $v \in L \cup J$
$\delta^+(v)$	Set of outgoing arcs of $v \in I \cup L$
K	Set of qualities

Parameters	
Λ	Adjacency matrix of $G = (V, A)$
c_a	Per unit cost of flow on arc $a \in A$
λ_{vk}	Quality value of input $v \in I$ for quality $k \in K$
L_v^V, U_v^V	Lower and upper bound on total flow through $v \in V$
L_a^A, U_a^A	Lower and upper bound on y_a, $a \in A$
L_{vk}^K, U_{vk}^K	Lower and upper bound on quality value of output $v \in J$ for quality $k \in K$

Variables	
y_a	Flow on arc $a \in A$
q_{iv}	Fraction of total flow through $v \in I \cup L$ that comes from input $i \in I$
q_{ia}	Fraction of y_a, $a \in A$, that comes from input $i \in I$: $q_{ia} = q_{iv}$, $a = (v, w) \in A$, $i \in I$
x_{ia}	Flow in y_a, $a \in A$, that comes from input $i \in I$

(b) Coal supply chain extension

Sets	
S	Set of run-of-mine stockpiles
T	Set of time points at which supply comes in and demand goes out

Parameters	
M	Flow multiple (trainload)
y_{st}^-	Incoming supply to stockpile $s \in S$ at time $t \in T$
y_t^+	Outgoing demand at time $t \in T$
c_{tk}^-, c_{tk}^+	Per unit negative cost (bonus) and per unit positive cost (penalty) of d_{tk}^- and d_{tk}^+, respectively, for $t \in T$, $k \in K$
λ_{stk}^-	Quality value of flow y_{st}^-, $s \in S$, $t \in T$, for quality $k \in K$
ℓ_{tk}, u_{tk}	Soft lower and upper bound on p_{tk}, $t \in T$, $k \in K$
L_s^S, U_s^S	Lower and upper bound on size of stockpile $s \in S$

Variables	
p_{tk}	Quality value of flow y_t^+, $t \in T$, for quality $k \in K$
d_{tk}^-, d_{tk}^+	Deviation of p_{tk} from ℓ_{tk} and u_{tk}, respectively, for $t \in T$, $k \in K$
z_{tk}^-, z_{tk}^+	Binary variable indicating that p_{tk} is smaller than ℓ_{tk} or larger than u_{tk}, respectively, for $t \in T$, $k \in K$
ζ_{ar}	Binary variable used to expand y_a, $a \in A$ (r depends on the type of expansion, unary or binary)
ξ_{iar}	Variable used to model the product $q_{ia}\zeta_{ar}$, $a \in A$, $i \in I$ (r depends on the type of expansion, unary or binary)

1 PROBLEM FORMULATION

1.1 Pooling problem

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 is the set of pools and J is 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 backward 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 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. Instances with $A_{LL} = \emptyset$ are referred to as *standard pooling problems* (SPPs), and instances with $A_{LL} \neq \emptyset$ are referred to as *generalized pooling problems* (GPPs). Both SPPs and GPPs can be modelled as bilinear programs, which are special cases of quadratically constrained quadratic programs, which in turn are special cases of nonlinear programs. Instances with $L = \emptyset$ are referred to as *blending problems*, which can be modelled as linear programs.

For every pool and output $v \in L \cup J$, we denote the set of incoming arcs of v by $\delta^-(v)$, and for every input and pool $v \in I \cup L$, we denote the set of outgoing arcs of v by $\delta^+(v)$. Let y_a be the flow on arc $a \in A$ and let c_a be the corresponding per unit cost. The total flow through vertex $v \in V$ (resp. the flow on arc $a \in A$) is bounded below by L_v^V (resp. L_a^A) and above by U_v^V (resp. U_a^A). For every input $i \in I$ and quality $k \in K$, the quality value of the incoming raw material is given by λ_{ik}. Similarly, for every output $v \in J$ and quality $k \in K$, the lower and upper bounds on the quality value of the outgoing blend are given by L_{vk}^K and U_{vk}^K, respectively. Table 1 (a) summarises the notation for the pooling problem.

Significant differences in solution quality (when solved locally) and solve time (when solved locally or globally) can be seen when reformulating the pooling problem. Such reformulations typically use different (aggregating or disaggregating) variables and/or additional valid (but redundant) constraints. Recently, Alfaki and Haugland (2013) proposed a multi-commodity flow formulation for the pooling problem based on input commodities. Boland et al. (2015) generalised these ideas and proposed new multi-commodity flow formulations based on output, input and output and (input, output)-commodities. Their computational results suggest that input and output commodities perform best, and since input commodities are more intuitive than output commodities, we now present the multi-commodity flow formulation based on input commodities, commonly referred to as the PQ-formulation.

The PQ-formulation uses fraction variables q, flow variables y and disaggregated flow variables x, which are products of q and y. Let q_{iv} denote the fraction of total flow through $v \in I \cup L$ that comes from input $i \in I$. In particular, for $v \in I$ we have $q_{vv} = 1$ and $q_{iv} = 0$ for $i \in I \setminus \{v\}$. Since we assume linear blending, the fraction of y_a, $a = (v, w) \in A$, that comes from input $i \in I$ is equal to q_{iv}. For convenience, we introduce $q_{ia} = q_{iv}$ for all $a = (v, w) \in A$ and $i \in I$. Introducing both arc- and node-based fraction variables allows us to formulate both arc- and node-based constraints. Note, however, that the q_{ia} do not appear in an implementation of the model, but are replaced by the corresponding q_{iv}. Lastly, let x_{ia} denote the flow in y_a, $a \in A$, that comes from input $i \in I$. The PQ-formulation can be stated as follows:

$$
\begin{aligned}
\text{[PQ]} \quad & \min_{q,x,y} \sum_{a \in A} c_a y_a & & & \\
\text{s.t.} \quad & \sum_{a \in \delta^-(v)} y_a = \sum_{a \in \delta^+(v)} y_a, & & v \in L, & (1)\\
& L_v^V \leq \sum_{a \in \delta^+(v)} y_a \leq U_v^V, & & v \in I \cup L, & (2)\\
& L_v^V \leq \sum_{a \in \delta^-(v)} y_a \leq U_v^V, & & v \in J, & (3)\\
& L_a^A \leq y_a \leq U_a^A, & & a \in A, & (4)\\
& q_{iv} \geq 0, & & v \in I \cup L,\ i \in I, & (5)\\
& \sum_{i \in I} q_{iv} = 1, & & v \in I \cup L, & (6)\\
& \sum_{a \in \delta^-(v)} x_{ia} = \sum_{a \in \delta^+(v)} x_{ia}, & & v \in L,\ i \in I, & (7)\\
& L_{vk}^K \sum_{a \in \delta^-(v)} y_a \leq \sum_{i \in I} \sum_{a \in \delta^-(v)} \lambda_{ik} x_{ia} \leq U_{vk}^K \sum_{a \in \delta^-(v)} y_a, & & v \in J,\ k \in K, & (8)\\
& y_a = \sum_{i \in I} x_{ia}, & & a \in A, & (9)\\
& L_v^V q_{iv} \leq \sum_{a \in \delta^+(v)} x_{ia} \leq U_v^V q_{iv}, & & v \in L,\ i \in I, & (10)\\
& x_{ia} = q_{ia} y_a, & & a \in A,\ i \in I. & (11)
\end{aligned}
$$

(1) is a flow conservation constraint which ensures that at every pool, the total incoming flow equals the total outgoing flow. (2) and (3) are vertex capacity constraints and (4) is an arc capacity constraint. Constraints (5) and (6) ensure that all fraction variables are between zero and one and that for every input and pool, the fraction variables sum to one. (7) can be interpreted as a disaggregated flow conservation constraint: while (1) ensures that at every pool, the total incoming flow must equal the total outgoing flow (regardless of the origin of the flows), (7) ensures that at every pool, the total incoming flow originating from a particular input must equal the total outgoing flow originating from the same input. It can be shown that (7) implies (1), i.e., that (1) is redundant. (8) is the output blending constraint. (9) and (10) are valid (but redundant) constraints. Adding such redundant constraints significantly improves the computational performance of a formulation. (11) outsources the qy terms – which would otherwise appear in (7)–(10) – into a single constraint. Note that by substituting (11) into (7)–(10), the PQ-formulation can also be stated without the x variables.

1.2 Coal supply chain extension

We consider the simplified coal supply chain shown in Figure 1. Coal is mined and loaded onto run-of-mine-stockpiles. It is then transported by rail to a *cargo assembly terminal* where it is offloaded onto stockpiles built for specific orders. The order-specific coal blends are then reclaimed and loaded onto ships. Contrary to

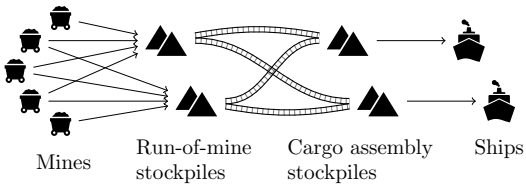


Figure 1. Simplified coal supply chain

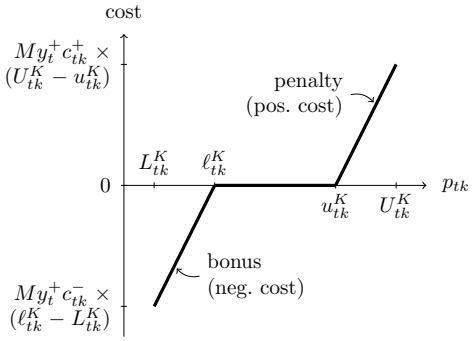
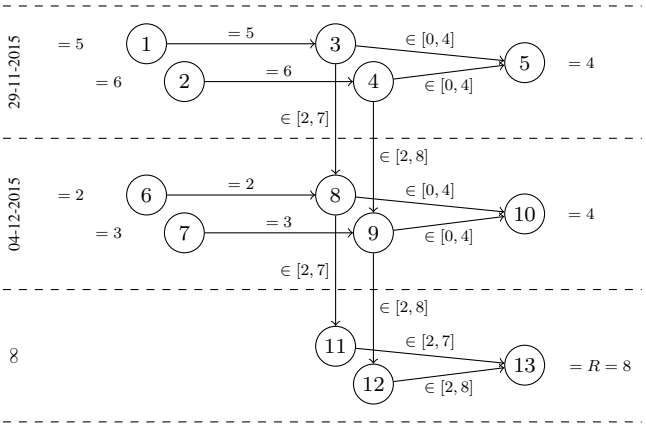


Figure 2. Bonus/penalty function

Trainload	8,000 [t]	
Stockpile	Minimum	Maximum
#1	16,000 [t]	56,000 [t]
#2	16,000 [t]	64,000 [t]
2015-11-29		
Supply	40,000 [t]	to #1
Supply	48,000 [t]	to #2
Demand	32,000 [t]	to ship #1
2015-12-04		
Supply	16,000 [t]	to #1
Supply	24,000 [t]	to #2
Demand	32,000 [t]	to ship #2

(a) Given data



(b) Corresponding pooling problem

Figure 3. Example of given data (a) and the corresponding pooling problem (b).

such a pull system, there is a trend towards push systems and *dedicated stockpile terminals* (see Boland et al. (2013)). However, in this paper, we only consider the former.

Let S denote the set of run-of-mine stockpiles whose sizes are bounded below by L_s^S and above by U_s^S for stockpile $s \in S$, and let T denote the set of time points at which orders/ships arrive. The preceding time point of $t \in T$ is $\tau(t) := \max\{t' \in T \cup \{-\infty\} : t' < t\}$. For every ship arriving at $t \in T$ (i.e., outgoing demand), we aggregate the coal that is mined and loaded onto run-of-mine stockpiles (i.e., incoming supply) during the time interval $(\tau(t), t]$. That is, for every $t \in T$, we have incoming supply to stockpile $s \in S$, denoted by y_{st}^-, and outgoing demand, denoted by y_t^+. Between the run-of-mine stockpiles and the cargo assembly terminal, coal is transported in multiples of trainloads of size M (e.g. $8,000\,[t]$). We assume that incoming supply y_{st}^- and outgoing demand y_t^+ are given in multiples of M. The same applies to arc and vertex capacities. Every order $t \in T$ has soft and hard lower and upper bounds $L_{tk}^K \leq \ell_{tk} \leq u_{tk} \leq U_{tk}^K$ on the coal blend quality values, which are denoted by p_{tk}. We assume that smaller quality values are preferred (consider e.g. an unwanted coal quality such as sulfur content). If $p_{tk} \in [L_{tk}^K, \ell_{tk}]$, then the mining company is paid a per unit bonus (a negative cost) of $c_{tk}^- < 0$ for the deviation $d_{tk}^- = \ell_{tk} - p_{tk} \geq 0$, i.e., we add $My_t^+ c_{tk}^- d_{tk}^-$ to the objective function. If $p_{tk} \in [u_{tk}, U_{tk}^K]$, then the mining company pays a per unit penalty (a positive cost) of $c_{tk}^+ > 0$ for the deviation $d_{tk}^+ = p_{tk} - u_{tk} \geq 0$, i.e., we add $My_t^+ c_{tk}^+ d_{tk}^+$ to the objective function. If $p_{tk} \in (\ell_{tk}, u_{tk})$, then neither a bonus nor a penalty is paid. Quality values $p_{tk} \in (-\infty, L_{tk}^K) \cup (U_{tk}^K, \infty)$ are infeasible. Table 1 (b) summarises the notation for the coal supply chain extension. The bonus/penalty function is shown in Figure 2.

As described in Boland et al. (2015), we construct a time-expanded pooling problem network where inputs and pools are (run-of-mine stockpile, time point)-pairs (s, t). The intermediate pools represent run-of-mine stockpiles over time. An output represents an order-specific cargo assembly stockpile. We model incoming supply

and outgoing demand by setting the arc and vertex capacities appropriately. Since we have flow conservation constraint (1), we add a supersink output to account for the stockpile surplus. An example of given data and the corresponding pooling problem is shown in Figure 3. More formally, we construct the graph $G = (V, A)$ and set the parameters as follows:

$$I = \{v^I_{st} : s \in S, t \in T\}, \quad L = \{v^L_{st} : s \in S, t \in T \cup \{\infty\}\}, \quad J = \{v^J_t : t \in T \cup \{\infty\}\},$$

$$\Lambda_{vw} = \begin{cases} 1 & \text{if } v = v^I_{st} \in I,\ w = v^L_{s't'} \in L:\ s = s',\ t = t' \\ 1 & \text{if } v = v^L_{st} \in L,\ w = v^L_{s't'} \in L:\ s = s',\ t = \tau(t') \\ 1 & \text{if } v = v^L_{st} \in L,\ w = v^J_{t'} \in J:\ t = t' \\ 0 & \text{otherwise} \end{cases},$$

$$\lambda_{vk} = \lambda^-_{stk}, \quad v = v^I_{st} \in I,\ k \in K,$$

$$[L^A_a, U^A_a] = \begin{cases} [y^-_{st}, y^-_{st}] & \text{if } v = v^I_{st} \in I,\ w \in L \\ [L^S_s, U^S_s] & \text{if } v = v^L_{st} \in L,\ w \in L \\ [0, y^+_t] & \text{if } v \in L,\ w = v^J_t \in J:\ t \neq \infty \\ [L^S_s, \min\{U^S_s, R\}] & \text{if } v = v^L_{st} \in L,\ w = v^J_{t'} \in J:\ t' = \infty \end{cases}, \quad a = (v, w) \in A,$$

$$[L^V_v, U^V_v] = \begin{cases} [\sum_{a \in \delta^+(v)} L^A_a, \sum_{a \in \delta^+(v)} U^A_a] & \text{if } v \in I \cup L \\ [y^+_t, y^+_t] & \text{if } v = v^J_t \in J:\ t \neq \infty \\ [R, R] & \text{if } v = v^J_t \in J:\ t = \infty \end{cases}, \quad v \in V,$$

$$[L^K_{vk}, U^K_{vk}] = \begin{cases} [L^K_{tk}, U^K_{tk}] & \text{if } t \neq \infty \\ [-\infty, \infty] & \text{if } t = \infty \end{cases}, \quad v = v^J_t \in J,\ k \in K,$$

where $R = \sum_{s \in S} \sum_{t \in T} y^-_{st} - \sum_{t \in T} y^+_t \geq 0$ is the stockpile surplus.

Continuous flow problem: In the continuous problem, arbitrarily small flows are allowed, i.e., we assume y to be continuous. Let z^-_{tk} and z^+_{tk} be binary variables used in constraints (12)–(16) to model the bonus/penalty function shown in Figure 2. The problem can be stated as follows:

$$\text{[C]} \quad \min_{d,p,q,x,y,z} \sum_{t \in T} \sum_{k \in K} M y^+_t (c^-_{tk} d^-_{tk} + c^+_{tk} d^+_{tk})$$

$$\text{s.t.} \quad (1)\text{–}(11)$$

$$z^-_{tk}, z^+_{tk} \in \{0, 1\}, \tag{12}$$

$$z^-_{tk} + z^+_{tk} \leq 1, \tag{13}$$

$$p_{tk} = \frac{1}{y^+_t} \sum_{i \in I} \sum_{a \in \delta^-(v^J_t)} \lambda_{ik} x_{ia}, \tag{14}$$

$$(\ell_{tk} - p_{tk})^+ \leq d^-_{tk} \leq \min\{z^-_{tk}(\ell_{tk} - L^K_{tk}),\ (\ell_{tk} - p_{tk}) + (1 - z^-_{tk})(U^K_{tk} - \ell_{tk})\}, \tag{15}$$

$$(p_{tk} - u_{tk})^+ \leq d^+_{tk} \leq \min\{z^+_{tk}(U^K_{tk} - u_{tk}),\ (p_{tk} - u_{tk}) + (1 - z^+_{tk})(u_{tk} - L^K_{tk})\}, \tag{16}$$

$$t \in T,\ k \in K.$$

Discretized flow problems: In the discretized problem, flows must be in multiples of trainloads. This can be modelled by adding an integrality constraint for y:

$$\text{[D]} \quad \min_{d,p,q,x,y,z} \sum_{t \in T} \sum_{k \in K} M y^+_t (c^-_{tk} d^-_{tk} + c^+_{tk} d^+_{tk})$$

$$\text{s.t.} \quad (1)\text{–}(16),$$

$$y_a \in \mathbb{Z}, \quad a \in A. \tag{17}$$

Modelling the unary and binary expansions of the integrality constraint (17), we can derive two exact linearizations of the nonlinear constraint (11), as described in Gupte et al. (2015). Let ζ_{ar} be the binary variables used in the expansion of y_a. We have $r \in \{0, \ldots, \overline{y}_a - \underline{y}_a\}$ for the unary and $r \in \{0, \ldots, \lfloor \log_2(\overline{y}_a - \underline{y}_a) \rfloor\}$ for the binary expansion of y_a. Finally, let ξ_{iar} be the continuous variable used to model the product $q_{ia}\zeta_{ar}$ for each r. The unary and binary expansions of [D] can be stated as follows:

$$\text{[D-U]} \quad \min_{d,p,q,x,y,z,\xi,\zeta} \sum_{t\in T}\sum_{k\in K} My_t^+(c_{tk}^-d_{tk}^- + c_{tk}^+d_{tk}^+)$$

$$\text{s.t.} \quad (1)\text{–}(10),\ (12)\text{–}(16),$$

$$\zeta_{ar} \in \{0,1\}, \qquad a\in A,\ r\in M_a, \tag{18}$$

$$x_{ia} = \sum_{r\in M_a}(\underline{y}_a + r)\xi_{iar}, \qquad a\in A,\ i\in I:\ M_a\neq\emptyset, \tag{19}$$

$$y_a = \underline{y}_a + \sum_{r\in M_a} r\zeta_{ar}, \qquad a\in A:\ M_a\neq\emptyset, \tag{20}$$

$$\sum_{r\in M_a}\zeta_{ar} = 1, \qquad a\in A:\ M_a\neq\emptyset, \tag{21}$$

$$q_{ia} = \sum_{r\in M_a}\xi_{iar}, \qquad a\in A,\ i\in I:\ M_a\neq\emptyset, \tag{22}$$

$$\underline{q}_{ia}\zeta_{ar} \le \xi_{iar} \le \overline{q}_{ia}\zeta_{ar}, \qquad a\in A,\ i\in I,\ r\in M_a, \tag{23}$$

$$\text{[D-B]} \quad \min_{d,p,q,x,y,z,\xi,\zeta} \sum_{t\in T}\sum_{k\in K} My_t^+(c_{tk}^-d_{tk}^- + c_{tk}^+d_{tk}^+)$$

$$\text{s.t.} \quad (1)\text{–}(10),\ (12)\text{–}(16),$$

$$\zeta_{ar} \in \{0,1\}, \qquad a\in A,\ r\in N_a, \tag{24}$$

$$x_{ia} = q_{ia}\underline{y}_a + \sum_{r\in N_a} 2^r\xi_{iar}, \qquad a\in A,\ i\in I:\ N_a\neq\emptyset, \tag{25}$$

$$y_a = \underline{y}_a + \sum_{r\in N_a} 2^r\zeta_{ar}, \qquad a\in A:\ N_a\neq\emptyset, \tag{26}$$

$$\underline{y}_a + \sum_{r\in N_a} 2^r\zeta_{ar} \le \overline{y}_a, \qquad a\in A:\ N_a\neq\emptyset, \tag{27}$$

$$\xi_{iar} \ge \max\{\underline{q}_{ia}\zeta_{ar},\ q_{ia} + \overline{q}_{ia}(\zeta_{ar}-1)\}, \qquad a\in A,\ i\in I,\ r\in N_a, \tag{28}$$

$$\xi_{iar} \le \min\{\overline{q}_{ia}\zeta_{ar},\ q_{ia} + \underline{q}_{ia}(\zeta_{ar}-1)\}, \qquad a\in A,\ i\in I,\ r\in N_a, \tag{29}$$

where $\underline{q}_{ia}$, $\overline{q}_{ia}$, $\underline{y}_a$ and $\overline{y}_a$ are the lower and upper bounds on q_{ia} and y_a, respectively, and

$$M_a = \begin{cases} \{0,\ldots,\overline{y}_a - \underline{y}_a\} & \text{if } \overline{y}_a - \underline{y}_a \ge 1 \\ \emptyset & \text{otherwise} \end{cases}, \quad a\in A,$$

$$N_a = \begin{cases} \{0,\ldots,\lfloor\log_2(\overline{y}_a - \underline{y}_a)\rfloor\} & \text{if } \overline{y}_a - \underline{y}_a \ge 1 \\ \emptyset & \text{otherwise} \end{cases}, \quad a\in A.$$

A trivial choice of the lower and upper bounds is $[\underline{q}_{ia}, \overline{q}_{ia}] = [0,1]$ and $[\underline{y}_a, \overline{y}_a] = [L_a^A, U_a^A]$, however, these bounds may be tightened in a preprocessing step. The importance of such a preprocessing step is evident in the unary and binary expansions [D-U] and [D-B] since the tightness of bounds determines the number of binary variables ζ_{ar} that need to be introduced. Note that [C] and [D] are MINLPs, while [D-U] and [D-B] are MILPs.

2 COMPUTATIONAL RESULTS AND CONCLUSION

Our industry partner provided us with a data set representing supply and demand data (including quality specifications and contractual bonuses and penalties) of a real life mining company for the time horizon of two years. We split the data into problem instances of years, half-years and quarters. There are two run-of-mine stockpiles (i.e., $|S| = 2$) with lower and upper bounds of $L_1^S = L_2^S = 2\,[M]$, $U_1^S = 7\,[M]$, $U_2^S = 8\,[M]$, and there are four qualities: ash, moisture, sulfur and volatile matter (i.e., $|K| = 4$). Problem sizes for the 14 instances vary between $|T| \in \{10,\ldots,22\}$ for quarterly, $|T| \in \{24,\ldots,42\}$ for half-yearly and $|T| \in \{54, 80\}$ for yearly instances. We used AMPL to model the different formulations and solved every instance for every formulation. The MINLPs [C] and [D] were solved with SCIP 3.0.0 (scip.zib.de) which we linked to CPLEX 12.6.0.0 (cplex.com) as the LP solver and to Ipopt 3.10 (coin-or.org/Ipopt) as the NLP solver. We solved the MILPs [D-U] and [D-B] with CPLEX 12.6.0.0. Yearly instances had a time limit of 60 minutes, half-yearly instances had 30 minutes, and quarterly instances had 15 minutes. All 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. Let $z[f,i]$ denote the optimal objective function value of problem $[f]$, $f \in \{\text{C}, \text{D}, \text{D-U}, \text{D-B}\}$, for instance i. It is clear that $z[\text{C},i] \le z[\text{D},i] = z[\text{D-U},i] = z[\text{D-B},i]$ for all i. Comparing the best known objective function values for the continuous and the

Table 2. Computational results for $M \in \{8,000\,[t]; 4,000\,[t]; 2,000\,[t]\}$. If an instance could be solved within its time limit, we report the total solve time in [s], otherwise we report the relative gap between the upper and lower bound in [%] in (brackets).

	$M = 8,000\,[t]$				$M = 4,000\,[t]$				$M = 2,000\,[t]$			
Instance	[C]	[D]	[D-U]	[D-B]	[C]	[D]	[D-U]	[D-B]	[C]	[D]	[D-U]	[D-B]
Year 1	(∞)	(8.44)	**(1.74)**	(3.82)	(∞)	(9.61)	**(0.00)**	(6.21)	**(2.40)**	(4.85)	(∞)	(∞)
Half-year 1	(1.92)	(0.60)	1072.02	**973.89**	(3.35)	(4.33)	**(0.76)**	(2.17)	(1.44)	(2.28)	**(1.27)**	(1.44)
Quarter 1	(0.15)	7.70	**3.91**	4.64	(0.63)	77.22	**27.72**	29.70	(0.47)	365.04	**189.79**	376.52
Quarter 2	(0.12)	22.81	14.44	**12.60**	(0.42)	579.48	**361.19**	679.32	(0.34)	(0.26)	(0.06)	**(0.03)**
Half-year 2	(∞)	415.43	**52.51**	231.83	(1.17)	(1.04)	**(0.97)**	(1.48)	(∞)	(1.77)	**(1.92)**	(2.36)
Quarter 3	(0.01)	**2.19**	4.38	3.93	(0.05)	10.49	17.10	**10.25**	(∞)	119.29	272.60	**47.00**
Quarter 4	688.62	6.04	**1.82**	3.88	(0.22)	101.37	**31.73**	52.99	(∞)	357.00	**152.21**	350.21
Year 2	(∞)	(∞)	**1752.94**	3556.30	(1.14)	(∞)	**(0.62)**	(1.37)	(∞)	(1.81)	**(0.74)**	(1.54)
Half-year 1	(0.66)	72.54	**16.30**	29.13	(0.89)	1243.82	**174.17**	727.81	(0.59)	(0.91)	**1766.16**	(0.32)
Quarter 1	(0.04)	2.87	**2.85**	3.54	(0.14)	**2.92**	10.22	14.16	(0.24)	(0.00)	**43.31**	189.39
Quarter 2	61.61	**0.33**	0.49	0.71	(0.02)	2.18	**2.06**	2.33	71.24	**1.17**	5.90	7.87
Half-year 2	(0.11)	7.59	**6.05**	6.20	(0.21)	69.25	**27.18**	98.54	(0.13)	**606.69**	1752.79	679.21
Quarter 3	10.36	**0.32**	0.54	0.69	407.20	**0.49**	1.28	0.71	137.01	**0.55**	1.81	1.57
Quarter 4	(0.01)	3.00	**1.44**	1.50	(0.04)	6.07	**5.16**	8.14	(∞)	**15.28**	24.53	89.99

discrete flow problems, we found that the gap between the two, $(z[\mathrm{D}, i] - z[\mathrm{C}, i])/z[\mathrm{C}, i]$, is in $[1.60\%, 9.94\%]$ for yearly, in $[0.13\%, 4.57\%]$ for half-yearly and in $[0.04\%, 4.25\%]$ for quarterly instances i. Computational results for $M \in \{8,000\,[t]; 4,000\,[t]; 2,000\,[t]\}$ are shown in Table 2. We also solved instances for half and quarter trainloads to see if [D-B] performs better than [D-U] if the y variables can take more discrete values. We see that the discretized problems [D], [D-U] and [D-B] perform much better than the continuous problem [C]. In 27 out of 42 runs, [D-U] outperforms [D] and [D-B]. There are only 9 runs in which [D] performs best, and only 5 runs in which [D-B] performs best – despite the fact that the formulation has far fewer binary variables than [D-U]. [C] only wins once, and that is due to [D-U] and [D-B] not finding any upper bounds. We can conclude that it is highly advantageous to discretize flow variables (if the application allows it), and that the exact MILP linearizations perform significantly better than the MINLPs. However, simply having fewer binary variables is no indicator for better performance, as demonstrated by the unary and binary encodings [D-U] and [D-B].

ACKNOWLEDGEMENT

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).

REFERENCES

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

Boland, N., T. Kalinowski, and F. Rigterink (2015). New multi-commodity flow formulations for the pooling problem. `http://www.optimization-online.org/DB_HTML/2015/06/4959.html`.

Boland, N., T. Kalinowski, F. Rigterink, and M. Savelsbergh (2015). A special case of the generalized pooling problem arising in the mining industry. `http://www.optimization-online.org/DB_HTML/2015/07/5025.html`.

Boland, N., M. Savelsbergh, and H. Waterer (2013). Shipping data generation for the Hunter Valley Coal Chain. `http://www.optimization-online.org/DB_HTML/2013/02/3755.html`.

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