Comparison of Discrete and Continuous Models for the Pooling Problem*

Mohammed Alfaki[1] **and Dag Haugland**[1]

1 **Department of Informatics, University of Bergen,**
P.O. Box 7803, N-5020 Bergen, Norway.
`mohammeda@ii.uib.no` **and** `dag@ii.uib.no`

Abstract

The pooling problem is an important global optimization problem which is encountered in many industrial settings. It is traditionally modeled as a bilinear, nonconvex optimization problem, and solved by branch-and-bound algorithms where the subproblems are convex. In some industrial applications, for instance in pipeline transportation of natural gas, a different modeling approach is often made. Rather than defining it as a bilinear problem, the range of qualities is discretized, and the complicating constraints are replaced by linear ones involving integer variables. Consequently, the pooling problem is approximated by a mixed-integer programming problem. With a coarse discretization, this approach represents a saving in computational effort, but may also lead to less accurate modeling. Justified guidelines for choosing between a bilinear and a discrete model seem to be scarce in the pooling problem literature. In the present work, we study discretized versions of models that have been proved to work well when formulated as bilinear programs. Through extensive numerical experiments, we compare the discrete models to their continuous ancestors. In particular, we study how the level of discretization must be chosen if a discrete model is going to be competitive in both running time and accuracy.

1998 ACM Subject Classification G.1.6 Optimization

Keywords and phrases Global Optimization, Industrial Optimization, Graphs and Networks, Pooling Problem

Digital Object Identifier 10.4230/OASIcs.ATMOS.2011.112

1 Introduction

The pooling problem is an important industrial optimization problem that originates from the petroleum refineries. It can be considered as an extension of the minimum cost flow problem on networks of three sets of nodes, referred to as sources, pools and terminals. From each source, a raw material is supplied to the network. The qualities of the raw materials depend on the source from which they are supplied. At the pools, raw materials of possibly unequal qualities are mixed to form intermediate products. In their turn, the intermediate products are blended again to form end products at the terminals. The resulting qualities of end products thus depend on what sources they originate from, and in what proportions. Restrictions, which may vary between the terminals, apply to these qualities.

Earlier work on the optimization of the pooling problem can be traced back to Haverly [12] in 1978, and since then there has been a continuous interest in the problem. Mainly,

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

the literature focuses on formulations, solution methods, applications, and experimental evaluations.

The problem is often formulated as a non-convex, continuous optimization problem, and many solution methods have been proposed to solve it. The ambition of the earliest approaches was to find a good local optimum. This includes the popular method of [12], which solves a sequence of linear programs approximating the problem. Based on Benders decomposition, Floudas and Aggarwal [2] proposed an algorithm to search for the global solution. Building on this, Floudas and Visweswaran [10] developed an algorithm based on Lagrangian relaxation techniques. Other Lagrangian-based algorithms were proposed by Adhya et al. [1] and Almutairi and Elhedhli [6]. Foulds et al. [11] developed a branch-and-bound algorithm based on linear relaxations of bilinear programs as suggested by McCormick [14] and Al-Khayyal and Falk [3].

Several continuous formulations have been proposed for the pooling problem. In addition to traditional network flow variables, the models also need some representation of product quality. The most straightforward approach [12], is to introduce a decision variable for each pool, and to let the variable be defined as the quality of the product at the given pool. As an alternative, Ben-Tal et al. [8] proposed a formulation where the quality variables are replaced by variables representing the proportions in which the pool receives flow from various sources. Tawarmalani and Sahinidis [19] strengthen this formulation by the application of reformulation-linearization technique (RLT) suggested by Sherali and Adams [18]. Following the idea of proportion variables, Alfaki and Haugland [4] proposed two formulations: In the first model, the source proportions introduced in [8] are replaced by terminal proportions. By combining source and terminal proportions, the second model becomes stronger than the first and also stronger than the model in [19].

In its traditional form, the pooling problem is defined on tripartite networks where all arcs connect a source to a pool, a pool to a terminal or a source to a terminal. Contrary to formulations with quality variables, formulations based on proportion variables cannot easily be generalized to arbitrary networks. Audet et al. [7] considered the case where there are connections between pools, and suggested a hybrid formulation involving both quality and proportion variables. Using only flow and proportion variables, Alfaki and Haugland [5] proposed a multi-commodity flow formulation for arbitrary networks, and proved that it dominates the hybrid formulation and a quality based formulation.

The above mentioned continuous formulations all have bilinear constraints. For the models with quality variables, this can be explained by the fact that the quality at a pool is defined as the weighted average of entering qualities, where the flow constitutes the weights. Reflecting the NP-hardness of the problem [4], bilinear constraints seem inescapable in continuous formulations, and represent a serious challenge to solution algorithms. Consequently, there is a need for easy-to-use and well studied solution strategies, such as mixed integer programs. This can be seen from the work of Faria and Bagajewicz [9], who discretized the quality variables of the wastewater treatment problem, which is closely related to the pooling problem, and replaced the bilinear constraints by "big M" constraints. Pushing in the same direction, Pham et al. [16] and Pham [15] eliminated the bilinear terms by discretizing the quality variables. Consequently, the pooling problem is approximated by a mixed-integer programming problem.

In this paper, we generalize the discretization approach proposed in Pham [16] to arbitrary networks. Through numerical experiments on large scale instances, we compare our discrete formulation with a continuous formulation. The purpose of this is to investigate whether discrete models are more suitable for finding good solutions when the global optimum is out of reach. By lower bounding techniques, we also aim to estimate the error introduced by discretizing the solution space.

The remainder of the paper is organized as follows: Section 2 introduces the pooling problem and one of its continuous formulations, and gives a brief description of the traditional solution methods. In Section 3, we present our discrete model and its extension to arbitrary networks. The numerical experiments are reported in Section 4, and major conclusions are summarized in Section 5.

2 Problem statement and formulation

We consider a directed graph (network) $G = (N, A)$ with node set N and arc set A. For each node $i \in N$, let $N_i^- = \{j \in N : (j,i) \in A\}$ and $N_i^+ = \{j \in N : (i,j) \in A\}$ denote the set of in- and out-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.

Define f_{ij} as the flow along the arc $(i,j) \in A$, and w_i^k $(k \in K)$ as the quality of flow leaving pool $i \in I$. The pooling problem is to assign flow values to all arcs in the pooling network such that each flow capacity b_i is respected at all nodes $i \in I$, and such that the total flow cost is minimized. Besides that, the quality of the flow leaving any pool is given as the weighted average of the quality of entering flow, where the flow values constitute the weights. More precisely, the matrix of qualities satisfies

$$\sum_{s \in N_i^- \cap S} q_s^k f_{si} + \sum_{j \in N_i^- \setminus S} w_j^k f_{ji} = w_i^k \sum_{j \in N_i^-} f_{ji}, \qquad i \in N \setminus S,\ k \in K, \tag{1}$$

if $\sum_{j \in N_i^-} f_{ji} > 0$. Otherwise, w_i^k is given an arbitrary value. In addition, the flow arriving at terminal $t \in T$ must for all attributes $k \in K$ satisfy the quality bounds q_t^k. Assuming that the qualities also at the terminals are given as weighted averages of entering flow, we arrive at the constraints:

$$\sum_{s \in N_t^- \cap S} q_s^k f_{st} + \sum_{j \in N_t^- \setminus S} w_j^k f_{jt} \le q_t^k \sum_{j \in N_t^-} f_{jt}, \qquad t \in T,\ k \in K. \tag{2}$$

Instead of defining quality variables, we associate a flow *commodity* with each source $s \in S$, where at most b_s units of the commodity can enter the network, and the commodity can leave the network at any $t \in T$. At all other nodes, the commodity neither enters nor leaves the network. Now, the variable f_{ij} defines the total flow of all commodities along arc $(i,j) \in A$. Relative to the total flow leaving node $i \in S \cup I$, let the variable y_i^s denote the proportion of commodity s (define $y_i^s = 0$ if $s \notin S_i$ and $y_s^s = 1$). Therefore, the quantity $y_i^s f_{ij}$ defines the flow of commodity s (meaning the commodity associated with source s, we simply refer to s as a commodity whenever convenient) along the arc (i,j). Based on the multi-commodity flow formulation, Alfaki and Haugland [5] proposed the following formulation to the pooling problem:

$$
\begin{aligned}
z^* = \min \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} && && (3)\\
\text{s.t.} \quad & \sum_{j\in N_i^+} f_{ij} \le b_i, && i \in S \cup I, && (4)\\
& \sum_{j\in N_t^-} f_{jt} \le b_t, && t \in T, && (5)\\
& \sum_{j\in N_i^-} y_j^s f_{ji} - \sum_{j\in N_i^+} y_i^s f_{ij} = 0, && s \in S_i,\ i \in I, && (6)\\
& \sum_{j\in N_t^-} \left(\sum_{s\in S_j} q_s^k y_j^s - q_t^k \right) f_{jt} \le 0, && t \in T,\ k \in K, && (7)\\
& \sum_{s\in S_i} y_i^s = 1, && i \in I, && (8)\\
& \sum_{s\in S_i} y_i^s f_{ij} = f_{ij}, && (i,j) \in A,\ i \in I, && (9)\\
& \sum_{j\in N_i^+} y_i^s f_{ij} \le y_i^s b_i, && s \in S_i,\ i \in I, && (10)\\
& f_{ij} \ge 0, && (i,j) \in A, && (11)\\
& 0 \le y_i^s \le 1, && s \in S_i,\ i \in I. && (12)
\end{aligned}
$$

The formulation (3)–(12) generalizes the PQ-formulation [19] for networks without directed paths connecting two pools. Constraints (4)–(5) impose the flow capacity constraints at all nodes, while (6) ensures that y_i^s is the proportion of the flow leaving pool i that originates from source s. The definition of y_i^s also implies (8). The desired quality at the terminals is achieved by (7). Constraints (9)–(10) are redundant RLT cuts [18] that contribute to stronger relaxations. They are derived respectively by multiplying (8) by f_{ij}, and by multiplying (4) by y_i^s.

2.1 Traditional solution methods

Because of the bilinear constraints (6)–(7) and (9)–(10), the feasible region of (3)–(12) is generally non-convex. Traditional solution approaches to such problems are typically based upon linear relaxation, which is embedded into a branch-and-bound procedure. Linear relaxations for the pooling problem are constructed by replacing each occurrence of the bilinear terms with its convex and concave envelopes [3, 14].

In the root node of a branch-and-bound algorithm, this relaxation is solved. In this way, the solution to the linear relaxation provides a lower bound on the global minimum. Convergence can then be attained through partitioning of the domain within a branch and bound framework.

3 Discrete formulation

To linearize the bilinear term $y_i^s f_{ij}$, we discretize the proportion variable y_i^s into $n+1$ known points, i.e. we divide the interval $[0,1]$ to $n \ge 1$ intervals. For simplicity, we assume that the number of discretization points is equal for all i and s, and that the discretization points are uniformly distributed on $[0,1]$. However, the methodology suggested in this work does not rely on these assumptions.

3.1 Computing a set of discretized proportion vectors

Consider any pool $i \in I$ and the corresponding set of sources S_i that can feed the pool. By the suggested discretization of y_i^s for all $s \in S_i$, we get $(n+1)^{|S_i|}$ different combinations of discretized proportions. However, many of these violate (8). Let $\Omega_i = \left\{Y \in \mathbb{R}^{S_i} : nY^s \in \{0,1,\ldots,n\}, \sum_{s \in S_i} Y^s = 1\right\}$ be the set of discrete values of y_i that satisfy (8). For the purpose of simple notation, let the sources in S_i be identified by the integers $1,\ldots,|S_i|$.

For any $Y \in \Omega_i$, the components of nY define a unique composition of n into $|S_i|$ parts. As demonstrated by Knuth [13, Section 7.2.1.3], there is hence a bijection between Ω_i and the set of $(|S_i|-1)$-combinations of $\{1,\ldots,n+|S_i|-1\}$. Let any such combination be denoted $\left(a_1,\ldots,a_{|S_i|-1}\right)$, where $1 \leq a_1 < \cdots < a_{|S_i|-1} \leq n+|S_i|-1$. It follows from [13, Section 7.2.1.3] that the corresponding $Y \in \Omega_i$ can be written $Y^s = (a_s - a_{s-1} - 1)/n$ $(s = 1,\ldots,|S_i|)$, where $a_0 = 0$ and $a_{|S_i|} = n+|S_i|$. The above reference also suggests an algorithm for enumerating all $(|S_i|-1)$-combinations of $\{1,\ldots,n+|S_i|-1\}$, and thereby also the set Ω_i. This is outlined in Algorithm 1.

Algorithm 1 Discretization(i,n)

```
$\Omega_i \leftarrow \emptyset$, $a_s \leftarrow s \; \forall s = 0,1,\ldots,|S_i|-1$, $a_{|S_i|} \leftarrow n+|S_i|$
repeat
    $Y^s \leftarrow (a_s - a_{s-1} - 1)/n \; \forall s = 1,\ldots,|S_i|$
    $\Omega_i \leftarrow \Omega_i \cup \{Y\}$
    $s \leftarrow 1$
    while $a_s + 1 = a_{s+1}$ do
        $a_s \leftarrow s, s \leftarrow s+1$
    if $s < |S_i|$ then
        $a_s \leftarrow a_s + 1$
until $s = |S_i|$
return $\Omega_i$
```

It is shown in [13] that the while-loop of Algorithm 1 is executed $\frac{|S_i|-1}{n+1}|\Omega_i|$ times. The while-loop thus implies that enumerating $|\Omega_i|$ by use of Algorithm 1 does not run in $O(|\Omega_i|)$ time.

3.2 The discrete model defined in an extended graph

We introduce an extension of G where each pool i is replaced by a set I^i consisting of $|\Omega_i|$ duplications of i. Each new pool $j \in I^i$, corresponds to a unique $Y_j \in \Omega_i$ with components Y_j^s $s \in S_i$. We refer to these vectors as the discretized proportions. The set of pools in the extended network hence becomes $I_n = \cup_{i \in I} I^i$, and the extended network will be represented by the directed graph $G_n = (N_n, A_n)$, where $N_n = S \cup I_n \cup T$ and $A_n = A \cap (S \times T) \cup \{(j,l) : l \in I^i, (j,i) \in A\} \cup \{(l,j) : l \in I^i, (i,j) \in A\}$. For any $j \in I_n$, let $i(j)$ denote the parent pool in G. That is, $i(j)$ is the unique pool satisfying $j \in I^{i(j)}$. For completeness, let $i(j) = j$ for all $j \in S \cup T$.

For the selection of proportions at pool $i \in I$, define the binary variables p_j for each $j \in I^i$ such that,

$$p_j = \begin{cases} 1, & \text{if } y_i^s = Y_j^s \text{ for all } s \in S_i, \\ 0, & \text{otherwise,} \end{cases}$$

and impose the constraint $\sum_{j \in I^i} p_j = 1$ for each $i \in I$, to ensure compatibility with the original problem. In the extended network, the flow can pass through at most one $j \in I^i$, leading to the constraints $\sum_{l \in N_j^+} f_{jl} \leq b_i p_j$ for all $j \in I^i$, where f now denotes flow in the extended network. The number of pools in the extended graph G_n increases exponentially with n. To reduce $|I_n|$, we identify pairs of pools $i, i' \in I$ such that $N_i^+ = N_{i'}^+$ and $N_i^- = N_{i'}^-$. For all such pairs, we do not introduce $I^{i'}$.

The MILP formulation approximating the continuous formulation (3)–(12), can hence be stated as follows

$$z(n) = \min \quad \sum_{(j,l) \in A_n} c_{i(j),i(l)} f_{jl} \tag{13}$$

$$\text{s.t.} \quad \sum_{l \in N_s^+} f_{sl} \leq b_s, \qquad s \in S, \tag{14}$$

$$\sum_{l \in N_j^+} f_{jl} \leq b_{i(j)} p_j, \qquad j \in I_n, \tag{15}$$

$$\sum_{l \in N_t^-} f_{lt} \leq b_t, \qquad t \in T, \tag{16}$$

$$\sum_{l \in N_j^-} Y_l^s f_{lj} - \sum_{l \in N_j^+} Y_j^s f_{jl} = 0, \qquad s \in S_{i(j)},\ j \in I_n, \tag{17}$$

$$\sum_{l \in N_t^-} \left(\sum_{s \in S_{i(l)}} q_s^k Y_l^s - q_t^k \right) f_{lt} \leq 0, \qquad t \in T,\ k \in K, \tag{18}$$

$$\sum_{j \in I^i} p_j = 1, \qquad i \in I, \tag{19}$$

$$p_j \in \{0,1\}, \qquad j \in I^i, i \in I, \tag{20}$$

$$f_{jl} \geq 0, \qquad (j,l) \in A_n. \tag{21}$$

Any feasible solution to (13)–(21) is a feasible solution to the original problem, and produces thereby an upper bound on z^*. The sequence $z(n)$ converges to z^* as $n \to \infty$, but even for instances of moderate size the computational burden represented by the MILP becomes prohibitively large for large values of n. However, with a coarse discretization, the optimal solution to (13)–(21) may be computable for instances where a global optimization algorithm based on a continuous formulation fails to converge within a reasonable time limit. In such instances, it is relevant to compare the optimal MILP-solution to the best solution obtained by an interrupted global optimization procedure.

3.3 Example

To illustrate the network extension outlined above, consider the first instance in Haverly [12], denoted Haverly1, depicted in Figure 1. Observe that node 4 is the unique pool in the network, and that $S_4 = \{1, 2\}$. Let $n = 2$, which implies that

$$(Y_j^s) = \begin{pmatrix} 0 & 1 \\ 1/2 & 1/2 \\ 1 & 0 \end{pmatrix}. \tag{22}$$

Each row of the matrix in (22) represents a possible combination of the flow proportions. Therefore, we replace pool 4 with 3 new pools ($I^4 = \{4, 5, 6\}$ and $I_2 = I^4$), and we change the numbering of the terminals accordingly.

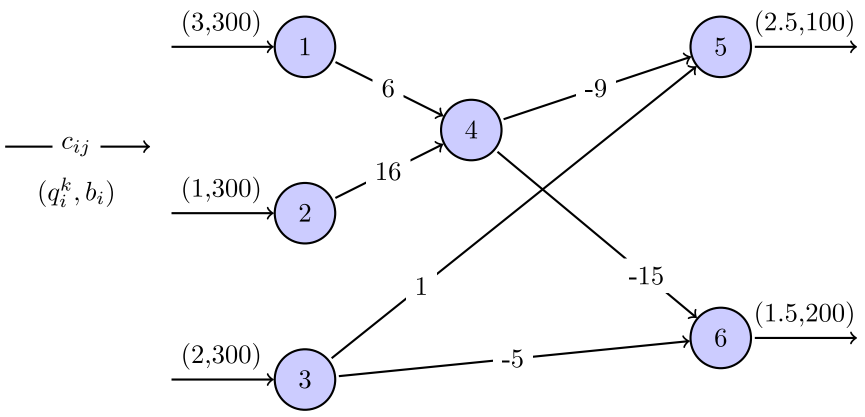

Figure 1 The Haverly1 pooling problem instance [12].

The set I^4 has the same set of neighbors as the original pool in Figure 1. The new network structure for Haverly1 instance is shown in Figure 2.

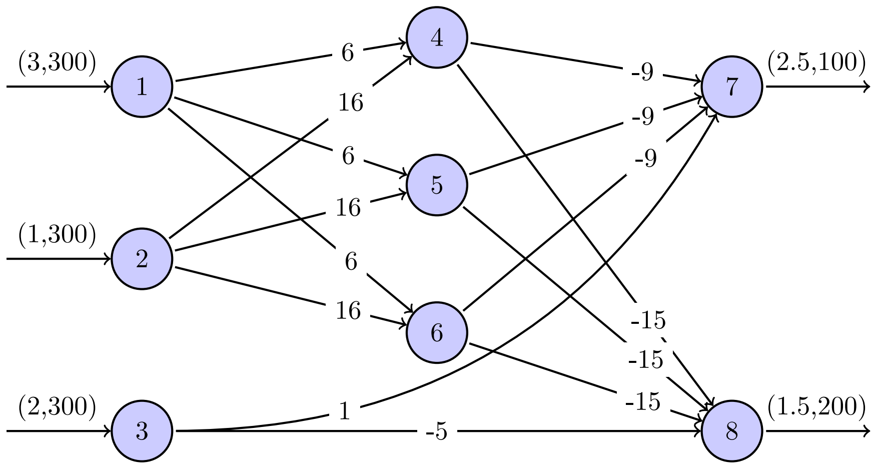

Figure 2 The discretized version of Haverly1 pooling problem instance with $n = 2$.

4 Computational experiments

For computational comparison of the discrete and the continuous formulations, we have used the 35 large-scale instances, with 15 arbitrary instances from [5] and 20 standard instances taken from [4]. The instances are divided into six groups, three groups with arbitrary networks (arbC, arbD and arbE) and the other three groups (stdA, stdB and stdC) with standard instances. The instances in the former three groups can be downloaded from the web page `http://www.ii.uib.no/~mohammeda/gpooling/` and the other instances can be downloaded from `http://www.ii.uib.no/~mohammeda/spooling/`. Table 1 reports the network sizes and number of arcs range in the network for each group.

Table 1 Instance characteristics

Group	#instances	Size of node and quality sets				#arcs range
		$\lvert S\rvert$	$\lvert I\rvert$	$\lvert T\rvert$	$\lvert K\rvert$	
arbC	5	8	6	6	4	57 – 82
arbD	5	12	10	8	5	114 – 166
arbE	5	10	10	15	12	181 – 248
stdA	10	20	10	15	24	171 – 407
stdB	6	35	17	21	34	384 – 1044
stdC	4	60	15	50	40	811 – 1451

Computational experiments were conducted by submitting these instances using the formulation (3)–(12) to the global solver BARON [17] version 1.8.5. The same instances were submitted to ILOG CPLEX version 10.2 using the discretized formulation (13)–(21) with $n = 1, 2, 4$. For both strategies, we set the time limit of each run to one CPU-hour, and set the relative optimality tolerance to 10^{-3}. Experiments reported here are conducted on a computer equipped with quad-core 3.00GHz processors where each group of four cores share 8GB of memory.

The results of the computational experiments are reported in Table 2. The first column gives the instance name, columns 2–3 report the lower and the upper bound provided by BARON with the continuous formulation. Column 4–5 give, for each value of n, the best feasible solution to the discrete model that CPLEX could find within the time limit. In instances where CPLEX could not prove optimality within the time limit, the best solution is written in parentheses. A stroke (—) in the table means that no feasible solution was found. For each instance, unless both of the formulations give the same solution, the best solution found is written in bold.

BARON computed the global optima for 14 instances. In the other hand, the feasible solutions for 9 instances with the discretized formulation are the true optimal solutions. Eight instances were solved to optimality by both of the formulations. Comparing the upper bounds (the feasible solutions) provided by both the continuous and the discretized formulations, we observe that the discrete formulation found the best upper bound in 21 instances out of 35. Even for $n = 1$, which means that all pools receive flow from at most one source, the best solution from the discrete model tends to outperform the best solution obtained by the continuous one. However, increasing the number of discretization points beyond 2 seems appropriate only in the smaller instances, and failed to produce feasible solutions in the remaining ones. For the more complicated instances, no better results are obtained by extending the search from solutions with no blending at the pools ($n = 1$) to solutions allowing blending of at most two streams in equal proportions ($n = 2$).

Table 2 Comparison between continuous and discrete models for the pooling problem.

Inst.	Continuous model		Discrete model		
	lb	ub	$z(n=1)$	$z(n=2)$	$z(n=4)$
arbC0	-1352.72	**-1352.72**	-1262.38	-1348.83	-1350.30
arbC1	-673.86	**-673.86**	-508.00	-615.50	-655.62
arbC2	-1716.62	**-1716.62**	-1688.69	-1705.81	(-1710.76)
arbC3	-1512.10	**-1512.10**	-1489.70	-1505.43	(-1508.92)
arbC4	-1071.81	-1071.81	-1071.81	-1071.81	-1071.81
arbD0	-1994.00	-1571.11	-1833.33	**-1911.35**	—
arbD1	-1356.51	-1356.51	-1346.54	-1356.51	—
arbD2	-2071.00	-2065.85	-2069.06	**-2070.16**	—
arbD3	-637.86	-637.86	-637.86	-637.86	—
arbD4	-1641.80	-1641.80	-1641.43	-1641.80	—
arbE0	-463.23	-463.23	-463.23	-462.23	—
arbE1	-556.00	-556.00	-556.00	-556.00	—
arbE2	-78.68	-78.68	-78.68	-78.68	—
arbE3	-891.25	-891.25	-891.25	-891.25	—
arbE4	-221.35	-221.35	-221.35	-221.35	—
stdA0	-37402.74	-5383.70	-31990.52	-34175.71	(**-34853.43**)
stdA1	-30362.74	-29276.56	-24590.16	-25179.84	(**-28389.31**)
stdA2	-23044.16	**-23044.16**	-19846.94	-20666.60	(-21795.71)
stdA3	-41113.10	-31258.05	-36233.75	-37116.64	(**-38624.98**)
stdA4	-42999.89	-8770.94	-38126.91	(**-39331.58**)	(-39345.90)
stdA5	-28257.75	-6369.59	-26447.07	(**-27008.30**)	(-26729.51)
stdA6	-42463.05	-9555.82	-41777.00	(**-42022.93**)	(-41829.91)
stdA7	-44682.25	-5762.08	-42582.29	(**-43309.48**)	(-42227.89)
stdA8	-30666.87	-6576.76	-30341.61	(**-30435.00**)	(-30265.99)
stdA9	-21933.99	-14059.98	-21887.77	(**-21891.96**)	(-21527.08)
stdB0	-45441.79	-9075.24	-40171.43	(**-41036.54**)	(-40600.32)
stdB1	-65468.81	-34069.43	-60720.54	(**-62445.97**)	(-61858.06)
stdB2	-56512.64	-11149.29	-53261.82	(**-53355.55**)	—
stdB3	-74050.47	-11469.84	(**-73572.52**)	(-73469.63)	—
stdB4	-59469.66	-13145.64	(**-59399.63**)	(-59233.59)	—
stdB5	-60696.36	-10313.90	(**-60080.85**)	(-59486.56)	—
stdC0	-98792.76	-2400.00	(-77517.74)	(**-79384.25**)	—
stdC1	-119006.17	-12114.75	(**-97290.27**)	(-91215.32)	—
stdC2	-135916.19	-6342.08	(**-117024.36**)	(-115594.77)	—
stdC3	-130315.02	-8770.86	(**-122570.51**)	(-114675.85)	—

5 Conclusion

In this paper, we have given a mixed integer programming model serving as an approximation to the pooling problem. The model makes no assumption about the network structure, and admits for example directed paths intersecting more than one pool. Computational experiments on a set of large-scale instances show that a discrete model is superior to its continuous ancestor, even when a very coarse discretization is applied. With a fine discretization, the model implies a large computational effort. To cope with this, a topic for future research is to develop an adaptive discretization rule. Computations can be saved if the number of discretization points can be kept small, while gradually focusing the search on solution sets of decreasing size.

References

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

2 A. Aggarwal and C. Floudas. A decomposition strategy for global optimization search in the pooling problem. *OSRA Journal on Computing*, 2(3):225–235, 1990.

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

4 M. Alfaki and D. Haugland. Strong formulations for the pooling problem. *Journal of Global Optimization*, 2010. Submitted for publication.

5 M. Alfaki and D. Haugland. A multi-commodity flow formulation for the pooling problem in arbitrary networks. *Journal of Global Optimization*, 2011. Submitted for publication.

6 H. Almutairi and S. Elhedhli. A new Lagrangian approach to the pooling problem. *Journal of Global Optimization*, 45:237–257, 2009.

7 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.

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

9 D.C. Faria and M.J. Bagajewicz. A new approach for the design of multicomponent water/wastewater networks. *Computer Aided Chemical Engineering*, 25:43–48, 2008.

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

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

12 C. Haverly. Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, 25:19–28, 1978.

13 D. E. Knuth. *The Art of Computer Programming*, Volume 4A: Combinatorial Algorithms, Part 1. Addison-Wesley, Reading, Massachusetts, 2011.

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

15 V. Pham. A Global Optimization Approach to Pooling Problems in Refineries. Master's thesis, Department of Chemical Engineering, Texas A&M University, Texas, USA, 2007.

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

17 N. Sahinidis. BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996.

18 H.D. Sherali and W.P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

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