Submitted to *Operations Research*
manuscript (Please, provide the manuscript number!)

Analysis of MILP Techniques for the Pooling Problem

Santanu S. Dey
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA santanu.dey@isye.gatech.edu

Akshay Gupte
Department of Mathematical Sciences, Clemson University, Clemson, SC agupte@clemson.edu

The *pq*-relaxation for the pooling problem can be constructed by applying McCormick envelopes for each of the bilinear terms appearing in the so-called *pq*-formulation of the pooling problem. This relaxation can be strengthened by using piecewise-linear functions that over- and under-estimate each bilinear term. While there is significant amount of empirical evidence to show that such piecewise-linear relaxations, which can be written as mixed integer linear programs (MILPs), yield good bounds for the pooling problem, to the best of our knowledge, no formal result regarding the quality of these relaxations is known. In this paper, we prove that the ratio of the upper bound obtained by solving piecewise-linear relaxations (objective function is maximization) to the optimal objective function value of the pooling problem is at most n, where n is the number of output nodes. Furthermore for any $\epsilon > 0$ and for any piecewise-linear relaxation, there exists a instance where the ratio of the relaxation value to the optimal value is at least $n - \epsilon$. This analysis naturally yields a polynomial-time n-approximation algorithm for the pooling problem. We also show that if there exists a polynomial-time approximation algorithm for the pooling problem with guarantee better than $n^{1-\epsilon}$ for any $\epsilon > 0$, then NP-complete problems have randomized polynomial time algorithms. Finally, motivated by the approximation algorithm, we design a heuristic which involves solving a MILP based restriction of the pooling problem. This heuristic is guaranteed to provide solutions within a factor of n. On large-scale test instances and in significantly lesser time, this heuristic provides solutions that are often orders of magnitude better than those given by commercial local and global optimization solvers.

Key words: Pooling Problem, Bilinear Programming, Approximation Algorithm, Mixed Integer Programming, Piecewise Linear Relaxations and Restrictions

Area of review: Optimization

OR/MS subject classification: Programming: integer, heuristic, nonlinear. Networks/graphs: multicommodity

History: Submitted April 2013; Revised June 2014, November 2014

1. Introduction: Pooling Problem and its MILP Based Relaxations and Restrictions

1.1. The Pooling Problem

The pooling problem, first proposed by Haverly (1978), is a type of network flow problem on a directed graph. There are three sets of nodes: input nodes, pool nodes and output nodes. Flows from input nodes are allowed either to pool nodes or directly to output nodes. The flows from pool nodes go to output nodes. There are no other arcs. The material that flows in this network possesses multiple specifications. For example, a specification could represent the concentration of a specific ingredient contained in the material of the flow. The 'raw' material at an input node is associated with a vector of values of different specifications. (Note that in general we do not assume that the specification values represent concentrations of ingredients, i.e. numbers between 0 and 1 only, but could be any real number). Specification values mix linearly, that is, the value of a given specification at a pool node is the weighed average of that specification for all the inflows from different input nodes. Similarly, the value of a given specification at an output node is the weighed average of that specification for all the inflows from different input and pool nodes. Each output node has an allowable range for each specification and the mixing process should occur in a way that the end products satisfy these lower and upper bounds for each specification. The objective is to maximize a linear weight function corresponding to the flows in arcs. Recently, Alfaki and Haugland (2013) proved that the pooling problem is NP-hard.

Formally, the input to the pooling problem consists of: (1) A network $(\mathcal{N}, \mathcal{A})$, where the nodes can be partitioned into three sets (I, J, L) where we call I as the set of input nodes, L as the set of pool nodes, and J as the set of output nodes; (2) The set of directed arcs $\mathcal{A}$ is a subset of the set $\{(I \times L) \cup (I \times J) \cup (L \times J)\}$. We consider the standard pooling problem where there are no arcs between pools (the general version allows arcs between pools). The in-degree and out-degree of every pool node is assumed to be at least 1, otherwise the pool together with the arcs incident to it can be removed from the instance; (3) Weight of per unit flow for each arc[1], that is a rational vector $f \in \mathbb{Q}^{|\mathcal{A}|}$; (4) A set of specifications K. A $|K|$-dimensional vector of specification values at each input node, that is $\lambda^i \in \mathbb{R}^{|K|}$ for all $i \in I$; (5) Two $|K|$-dimensional vectors of lower and upper bounds, respectively, on the specification values at each output node, that is $a^j, b^j \in \mathbb{R}^{|K|}$ for all $j \in J$; (6) Capacities on nodes and arcs[2].

[1] Some variants of pooling problem have cost of per unit flow in arcs, profit per unit flow into output nodes and cost per unit flow out of input nodes. It is easy to construct $f \in \mathbb{Q}^{|\mathcal{A}|}$ such that the total weight of a given flow is equal to the net revenue, that is the profit obtained at the output nodes minus the total cost due to flows in various arcs and flows out of input nodes.

[2] We assume without loss of generality that the capacities of the nodes and arcs are consistent, that is, the capacity of a node is less than or equal to sum of capacities of all its out-arcs or all its in-arcs, the capacity of an arc is less than the minimum of the capacity of the two incident nodes, etc.

The *pq*-formulation of the pooling problem was proposed by Tawarmalani and Sahinidis (2002) (also see Quesada and Grossmann 1995) and it consists of appending some valid constraints derived via the Reformulation Linearization Technique of Sherali and Adams (1998) to the *q*-formulation due to Ben-Tal et al. (1994). The formulation involves the following variables: (1) y_{ij} - flow along arc (i,j) where $i \in I$ and $j \in J$; (2) y_{lj} - flow along arc (l,j) where $l \in L$ and $j \in J$; (3) q_{il} - fraction of flow in arc (i,l) to the total incoming flow to pool l; (4) v_{ilj} - the flow that reaches output node $j \in J$ starting from $i \in I$ via pool node $l \in L$. For notational simplicity, we will write the flow variables y_{uv} with the understanding that y_{uv} is defined only for $(u,v) \in \mathcal{A}$. Similarly, when we use v_{ilj}, it is with the understanding that $i \in I, l \in L, j \in J$ and $(i,l),(l,j) \in \mathcal{A}$. The *pq*-formulation is:

$$\max_{y,v,q} \sum_{i\in I, j\in J} f_{ij} y_{ij} + \sum_{i\in I, l\in L, j\in J} (f_{il} + f_{lj}) v_{ilj} \tag{1a}$$

$$\text{s.t. } v_{ilj} = q_{il} y_{lj} \ \forall i \in I, l \in L, j \in J \tag{1b}$$

$$\sum_{i\in I} q_{il} = 1 \ \forall l \in L \tag{1c}$$

$$\sum_{i\in I} \lambda^i_k y_{ij} + \sum_{i\in I, l\in L} \lambda^i_k v_{ilj} \le b^j_k \left(\sum_{i\in I} y_{ij} + \sum_{l\in L} y_{lj} \right) \ \forall k \in K, j \in J \tag{1d}$$

$$\sum_{i\in I} \lambda^i_k y_{ij} + \sum_{i\in I, l\in L} \lambda^i_k v_{ilj} \ge a^j_k \left(\sum_{i\in I} y_{ij} + \sum_{l\in L} y_{lj} \right) \ \forall k \in K, j \in J \tag{1e}$$

$$\sum_{j\in J} y_{ij} + \sum_{l\in L, j\in J} v_{ilj} \le c_i \ \forall i \in I, \qquad \sum_{j\in J} y_{lj} \le c_l \ \forall l \in L \tag{1f}$$

$$\sum_{i\in I} y_{ij} + \sum_{i\in I, l\in L} v_{ilj} \le c_j \ \forall j \in J, \qquad \sum_{j\in J} v_{ilj} \le c_{il} \ \forall i \in I, l \in L \tag{1g}$$

$$y_{lj} \le c_{lj} \ \forall l \in L, j \in J, \qquad y_{ij} \le c_{ij} \ \forall i \in I, j \in J \tag{1h}$$

$$y_{ij}, y_{lj}, v_{ilj}, q_{il} \ge 0 \ \forall i \in I, l \in L, j \in J \tag{1i}$$

$$\sum_{i\in I} v_{ilj} = y_{lj} \ \forall l \in L, j \in J \tag{1j}$$

$$\sum_{j\in J} v_{ilj} \le c_l q_{il} \ \forall i \in I, l \in L. \tag{1k}$$

In the above formulation, constraints (1b) enforce consistency of the specification values in the out-arcs of a pool node: The value of specification k in out-arc (l,j) of a pool l is: $\frac{\sum_{i\in I} \lambda^i_k v_{ilj}}{y_{lj}} = \frac{\sum_{i\in I} \lambda^i_k q_{il} y_{lj}}{y_{lj}} = \sum_{i\in I} \lambda^i_k q_{il}$, and thus independent of j, where the first equality is a consequence of (1b). Constraint (1c) enforces that the sum of ratios of flow from different input nodes i to a given pool node l must add up to one. Constraints (1d) and (1e) enforce that the final product in the output node must satisfy the upper and lower bounds on each specification. Constraints (1f) - (1h) are the various capacity constraints.

Problem (1) is a bilinear program, and hence nonconvex, because of the bilinear terms in (1b). Applying McCormick (1976) envelopes of (1b) yields the following constraints:

$$v_{ilj} \geq 0,\ v_{ilj} \geq y_{lj} + c_{lj}q_{il} - c_{lj},\ v_{ilj} \leq y_{lj},\ v_{ilj} \leq c_{lj}q_{il}. \tag{2}$$

The linear program (LP) defined by the objective function (1a) together with the constraints (1c) - (1k) and the constraints (2) is the *pq-relaxation*.

$$\max_{y,v,q} \{(1a)\colon \text{s.t. } (1c) - (1k),\ (2)\}. \tag{pq-relaxation}$$

We note here that constraints (1j) and (1k) are implied by constraints (1b) - (1i). These redundant constraints in (1) guarantee that the *pq*-relaxation is stronger than the LP relaxation obtained by the application of McCormick envelopes to the bilinear terms appearing in the so-called *p*-formulation; see Tawarmalani and Sahinidis (2002). A review of pooling problems can be found in Gupte et al. (2013), Misener and Floudas (2009).

1.2. MILP Based Relaxations

It is possible to strengthen the *pq*-relaxation, by improving the approximation of the bilinear constraints (1b) using piecewise-linear envelopes of the bilinear terms (1b). We formally define these next.

Definition 1 (Piecewise-Linear Relaxation Scheme).

1. Consider the bilinear set $\mathcal{B}^\theta := \{(\alpha,\beta,\chi)\,|\,\chi = \alpha\beta,\ 0 \leq \alpha \leq 1,\ 0 \leq \beta \leq \theta\}$. Let $g^\theta(\alpha,\beta) : [0,1]\times[0,\theta] \to \mathbb{R}$ and $h^\theta(\alpha,\beta) : [0,1]\times[0,\theta] \to \mathbb{R}$ be piecewise-linear continuous functions such that $g^\theta(\alpha,\beta) \geq \alpha\beta \geq h^\theta(\alpha,\beta)$ for all $(\alpha,\beta) \in [0,1]\times[0,\theta]$. Define the piecewise-linear relaxation S^θ of $\mathcal{B}^\theta$ as $S^\theta := \{(\alpha,\beta,\chi)\,|\,g^\theta(\alpha,\beta) \geq \chi \geq h^\theta(\alpha,\beta), (\alpha,\beta) \in [0,1]\times[0,\theta]\}$.

2. Suppose for every $\theta > 0$, we have decided a-priori a piecewise-linear relaxation $S(\theta)$[3]. We call this a *piecewise-linear relaxation scheme*.

3. Given a piecewise-linear scheme and a pooling problem defined by (1a) - (1k) together with (2), we construct a *piecewise-linear relaxation of the pooling problem* by replacing each of the bilinear equalities (1b) with the appropriate piecewise-linear relaxation using the scheme.

Notice that the above definition implies that the *pq*-relaxation, which is a linear relaxation obtained from McCormick envelopes (2), is the simplest piecewise-linear relaxation of the pooling problem. Any other piecewise-linear relaxation of the pooling problem can be modeled via the

[3] An example of a scheme: Divide the interval $[0,1]$ into a predetermined number of equally spaced strips and then construct the McCormick envelopes over each strip times the interval $[0,\theta]$. See Gounaris et al. (2009) for more examples.

addition of integer variables (cf. Nemhauser and Wolsey 1988, chap. 1), thus resulting in a *mixed integer linear programming (MILP) relaxation*. Gounaris et al. (2009), Meyer and Floudas (2006), Wicaksono and Karimi (2008) present computational studies of various MILP models and piecewise-linear relaxation schemes for bilinear programs.

1.3. MILP Based Restrictions

In order to construct a MILP based restriction, the main idea is to restrict the domain of a subsets of the variables to a finite set in such a way that the resulting set of feasible solutions in MILP representable. As an example, consider the constraint (1b). Suppose we restrict q_{il} to take values from the set $\left\{0, \frac{1}{M}, \frac{2}{M}, \ldots, 1\right\}$ for some $M \geq 1$ and $M \in \mathbb{Z}_+$. Then we can rewrite constraint (1b) using new variables (h, g) as

$$q_{il} = \frac{1}{M}\sum_{t=1}^{M} t g_t, \;\; v_{ilj} = \frac{1}{M}\sum_{t=1}^{M} t h_t, \;\; \sum_{t=1}^{M} g_t \leq 1$$
$$h_t \leq y_{lj}, \;\; h_t \leq c_{lj} g_t, \;\; h_t \geq y_{lj} + c_{lj} g_t - c_{lj}, \;\; h_t \geq 0, \;\; g_t \in \{0,1\} \quad \forall t \in \{1, \ldots, M\}.$$

The above formulation is an example of the so-called *unary* method. Alfaki and Haugland (2011), Gupte et al. (2013), Pham et al. (2009) have evaluated a number of different MILP models and choices of variables to discretize for pooling and other bilinear programs.

2. Overview of Results

2.1. Results on MILP Based Relaxations

To the best of our knowledge, there has been no theoretical investigation of the quality of piecewise-linear relaxations for the pooling problem. We prove that the ratio of the upper bound obtained by solving any piecewise-linear relaxation (even the basic *pq*-relaxation) to the optimal objective function value of the pooling problem is at most n, where n is the number of output nodes. Furthermore, we show that this bound is tight: for any $\epsilon > 0$ and a given piecewise-linear relaxation scheme, there exists an instance of the pooling problem where the ratio of the optimal objective function value of the relaxation to that of the pooling problem is at least $n - \epsilon$. Section 3 presents these results. This result is interesting in the following sense: It is possible to construct more and more refined MILP relaxations of each bilinear term such that the absolute error in accounting for each bilinear term reduces. However in the worst case, these arbitrarily small errors in each bilinear term can propagate and lead to an overall error of approximately n for the entire piecewise linear MILP relaxation.

Since computational study of MILP based relaxations of the above type have been extensively conducted in literature (cf. Gounaris et al. 2009), we do not make an empirical study of these relaxations in this paper.

2.2. Results on MILP Based Restrictions

Our results on MILP based restrictions for the pooling problem are inspired by a simple approximation algorithm for the pooling problem: The analysis of bounds for MILP based relaxations discussed in the previous section, involves constructing a feasible solution for the pooling problem whose objective function value is at least $1/n$-times that of the optimal solution. Since this solution can be obtained by suitably updating an optimal solution of an LP of polynomial size, we obtain an n-approximation algorithm. At first sight, this appears to be a very poor approximation ratio. However, note that (1) this approximation ratio is independent of the number of the specifications ($|K|$), the number of input nodes ($|I|$) or the number of pool nodes ($|L|$); (2) Using an approximation factor preserving reduction from stable set problem, (this reduction was used in Alfaki and Haugland (2013) to show the NP-hardness of pooling problem), we ascertain that if there exists a polynomial-time approximation algorithm with guarantee better than $n^{1-\epsilon}$ (for any $\epsilon > 0$) for the pooling problem, then NP-complete problems have randomized polynomial algorithms. Section 4 presents these details.

Next we generalize the key ideas behind the n-approximation algorithm to construct a MILP whose feasible region is a restriction of the pooling problem. To the best of our knowledge, none of the MILP restrictions (Alfaki and Haugland, Gupte et al., Pham et al.) previously described for the pooling problem, has any guarantee on solution quality. In contrast, since the feasible region of MILP restriction proposed in this paper includes all solutions that can be obtained by the n-approximation algorithm, this MILP produces solutions that are at least as good as the n-approximation algorithm. Section 4 presents this MILP and related results.

Finally, Section 5 tests the MILP-based heuristic on large-scale pooling instances. Our empirical analysis shows that feasible solutions obtained by solving the proposed MILP are on average much superior compared to those from global solver and local heuristic methods.

3. Quality of Piecewise-Linear Relaxations

In this section, we will prove the following result.

THEOREM 1 **(Quality of Piecewise-Linear Relaxations)**. *Suppose we have a pre-specified piecewise-linear relaxation scheme. Let n represent the number of output nodes in a pooling problem. Given an instance of pooling problem, let z^S be the optimal objective function value of the piecewise-linear relaxation constructed using the piecewise-linear relaxation scheme and let z^* be the optimal objective function value. Then:*

1. $z^S \leq nz^*$.
2. *For any $\epsilon > 0$, there exists an instance of the pooling problem such that $z^S \geq (n-\epsilon)z^*$.*

In order to prove Theorem 1 we first describe a relaxation of the *pq*-relaxation, that we call as the *Inconsistent Pool-Outflow Problem (IPOP):* Construct this relaxation of the *pq*-relaxation by removing the q_{il} variables, for all $i \in I$ and $l \in L$, and by removing all constraints that involve these variables in the *pq*-relaxation. In particular, IPOP is the following LP:

$$\max_{y,v} \{(1a) : \text{s.t. } (1d) - (1h), (1j), y, v \geq \mathbf{0}\}. \tag{IPOP}$$

IPOP is a relaxation since the projection of any feasible solution of the *pq*-relaxation onto the space of y and v variables belongs to this relaxation. As discussed before, the constraints (1b) enforce consistency of the specification in the out-arcs of a pool node. Since (1b) is removed from the description of IPOP, there is no consistency of specification in different out-arcs of a given pool node, i.e., $\frac{\sum_{i \in I} \lambda_k^i v_{ilj}}{y_{lj}}$ can take different values in different out-arcs (l, j) of pool node $l \in L$.

In order to prove the first part of Theorem 1, the key insight is that if we take any feasible solution of IPOP and set the flows to all the output nodes except one to 0, then we trivially attain consistency of specification in the out-arcs of every pool, since there is at most one arc with positive flow among the out-arcs of every pool. We now formalize this observation in order to verify the first part of Theorem 1.

PROPOSITION 1. *Let n be the number of output nodes in a pooling problem. Let z^{pq} be the optimal objective function value of the pq-relaxation and let z^* be the optimal objective function value of the pooling problem. Then, $z^{pq} \leq n z^*$.*

Proof. Let z^{IPOP} be the optimal objective function value of IPOP problem. We will show that there exists a feasible solution of the pooling problem such that the objective function value of this point, say $\tilde{z}$, satisfies

$$z^{IPOP} \leq n\tilde{z}. \tag{3}$$

Since $z^{pq} \leq z^{IPOP}$ (IPOP is a relaxation of *pq*-relaxation) and $z^* \geq \tilde{z}$, we obtain that

$$z^{pq} \leq z^{IPOP} \leq n\tilde{z} \leq n z^*, \tag{4}$$

which is the desired result.

We now show the existence of a feasible solution of the pooling problem such that (3) holds. Let (v^*, y^*) be an optimal solution of IPOP. Then

$$z^{IPOP} = \sum_{j \in J} \left(\sum_{i \in I} \left(f_{ij} y^*_{ij} + \sum_{l \in L} (f_{il} + f_{lj}) v^*_{ilj} \right) \right). \tag{5}$$

Let

$$j^* \in \arg\max j \in J \left(\sum_{i \in I} \left(f_{ij} y^*_{ij} + \sum_{l \in L} (f_{il} + f_{lj}) v^*_{ilj} \right) \right). \tag{6}$$

Now construct the solution $(\bar{v}, \bar{y}, \bar{q})$ as

$$\bar{v}_{ilj} = \begin{cases} 0 & j \neq j^* \\ v^*_{ilj} & j = j^* \end{cases} \quad \bar{y}_{lj} = \begin{cases} 0 & j \neq j^* \\ y^*_{lj} & j = j^* \end{cases} \tag{7a}$$

$$\bar{y}_{ij} = \begin{cases} 0 & j \neq j^* \\ y^*_{ij} & j = j^* \end{cases} \quad \bar{q}_{il} = \begin{cases} \dfrac{\bar{v}_{ilj^*}}{\bar{y}_{lj^*}} & \bar{y}_{lj^*} > 0 \\ \dfrac{1}{|\{u \in I : (u,l) \in \mathcal{A}\}|} & \bar{y}_{lj^*} = 0. \end{cases} \tag{7b}$$

We need to verify that $(\bar{v}, \bar{y}, \bar{q})$ satisfy (1b) - (1i).

Constraint (1b): Consider any (i,l,j) such that $(i,l),(l,j) \in \mathcal{A}$. If $j \neq j^*$, then $\bar{v}_{ilj} = \bar{y}_{lj} = 0$ and (1b) is trivially satisfied. If $j = j^*$ and $\bar{y}_{lj^*} > 0$, then by construction of $\bar{q}$, we have that $\bar{v}_{ilj^*} = \bar{q}_{il}\bar{y}_{lj^*}$. If $j = j^*$ and $\bar{y}_{lj^*} = 0$, then $0 \leq \bar{v}_{ilj^*} = v^*_{ilj^*} \leq y^*_{lj^*} = \bar{y}_{lj^*} = 0$, where the second inequality follows from the fact that (v^*, y^*) satisfies (2). Therefore, $(\bar{q}, \bar{v}, \bar{y})$ satisfies (1b).

Constraint (1c): Consider any $l \in L$. If $\bar{y}_{lj^*} > 0$, then $\sum_{i \in I} \bar{q}_{il} = \sum_{i \in I} \frac{\bar{v}_{ilj^*}}{\bar{y}_{lj^*}} = 1$ since $\bar{v}_{ilj^*} = v^*_{ilj^*}$, $\bar{y}_{ilj^*} = y^*_{lj^*}$ and the fact that (v^*, y^*) satisfies (1j). If $\bar{y}_{lj^*} = 0$, then $\sum_{i \in I} \bar{q}_{il} = \sum_{i \in I, (i,l) \in \mathcal{A}} \frac{1}{|\{u \in I \,|\, (u,l) \in \mathcal{A}\}|} = 1$. Therefore, $(\bar{q}, \bar{v}, \bar{y})$ satisfies (1c).

Constraints (1d) and (1e): Consider any $j \neq j^*$. Since total flow into output node j is 0 in the flow vector $(\bar{v}, \bar{y})$, constraints (1d) and (1e) are trivially satisfied. If on the other hand $j = j^*$, then $\bar{v}_{ilj^*} = v^*_{ilj^*}$, $\bar{y}_{lj^*} = y^*_{lj^*}$ and $\bar{y}_{ij^*} = y^*_{ij^*}$ for all $i \in I, l \in L$, and since (v^*, y^*) satisfy (1d) and (1e), $(\bar{q}, \bar{v}, \bar{y})$ satisfy them.

Capacity Constraints: The vectors $\bar{v}$ and $\bar{y}$ trivially satisfy all the capacity constraints, since $\bar{v} \leq v^*$ and $\bar{y} \leq y^*$ and (v^*, y^*) satisfy the capacity constraints.

Therefore, $(\bar{q}, \bar{v}, \bar{y})$ is feasible solution of the pooling problem. Moreover (5) and (6) imply that the objective function value of $(\bar{q}, \bar{v}, \bar{y})$ satisfies (3), completing the proof. □

Even a very good piecewise-linear relaxation cannot avoid small errors in approximating the constraint (1b). The second part of Theorem 1 is essentially stating that it is possible to construct instances where these error 'accumulate' and therefore an overall error ratio $n - \epsilon$ for arbitrarily small ϵ can occur. Formally we verify the following result.

PROPOSITION 2. *Let S^1 be any pre-specified piecewise-linear approximation of $\mathcal{B}^1$. Given an instance of pooling problem with $c_{lj} = 1$ for all $l \in L$, $j \in J$, let z^S be the optimal objective function value of the piecewise-linear relaxation constructed by replacing (1b) by S^1 and let z^* be the optimal objective function value of the pooling problem. For any $n \in \mathbb{Z}_+$, $n \geq 2$ and any $\epsilon_0 > 0$, there exists an instance of the pooling problem with n output nodes such that $z^S \geq (n - \epsilon)z^*$ for some $0 < \epsilon \leq \epsilon_0$.*

Proof. Let us consider the piecewise-linear relaxation S^1 of $\mathcal{B}^1$ as presented in Definition 1. We will say that "a piecewise-linear function is composed of a finite list of affine functions

$\{\psi^1, \ldots, \psi^t\}$", implying that we can divide the domain of the function into a finite number of polytopes, such that the piecewise-linear function coincides with one of the affine functions in the list in each polytope and if two polytopes have non-empty intersection, then the corresponding affine functions of the two polytopes have the equal value for each point in the intersection.

CLAIM 1. *There exists* $\epsilon, \kappa, \alpha^*$ *satisfying* $0 < \epsilon \leq \epsilon_0$, $\kappa > 0$ *and* $0 < \alpha^* < 1$ *such that*

$$\left(\alpha^*, 1 - \frac{\epsilon}{n-1}, \alpha^*\left(1 - \frac{\epsilon}{n-1}\right) + \text{error}\right) \in S^1 \text{ and} \tag{8}$$

$$\left((1-\alpha^*), 1 - \frac{\epsilon}{n-1}, (1-\alpha^*)\left(1 - \frac{\epsilon}{n-1}\right) + \text{error}\right) \in S^1 \tag{9}$$

for all $|\text{error}| \leq \kappa$. Let the over-estimating and under-estimating piecewise-linear functions, respectively g^1 and h^1, be composed of a finite number of affine functions, respectively $\{\psi^s\}_{s=1}^{n_1}$ and $\{\sigma^s\}_{s=1}^{n_2}$. Then the list $\left\{\frac{\partial \psi^s}{\partial \alpha}\right\}_{s=1}^{n_1}$ and $\left\{\frac{\partial \sigma^s}{\partial \alpha}\right\}_{s=1}^{n_2}$ is also finite. In particular, there exists ϵ satisfying $0 < \epsilon \leq \epsilon_0$ such that $1 - \frac{\epsilon}{n-1} \notin \left(\cup_{s=1}^{n_1}\left\{\frac{\partial \psi^s}{\partial \alpha}\right\}\right) \bigcup \left(\cup_{s=1}^{n_2}\left\{\frac{\partial \sigma^s}{\partial \alpha}\right\}\right)$. Thus, the restriction of the functions g^1 and h^1 to $\left\{(\alpha, \beta) \in [0,1] \times [0,1] \,|\, \beta = 1 - \frac{\epsilon}{n-1}\right\}$, are piecewise-linear continuous functions composed of affine functions whose slope do not match $\left(1 - \frac{\epsilon}{n-1}\right)$, while on the other hand, these functions are over and under approximations of the linear function $\chi = \left(1 - \frac{\epsilon}{n-1}\right)\alpha$. Thus, $g^1(\alpha, 1 - \frac{\epsilon}{n-1}) = \alpha(1 - \frac{\epsilon}{n-1})$ and $h^1(\alpha, 1 - \frac{\epsilon}{n-1}) = \alpha(1 - \frac{\epsilon}{n-1})$ can hold for a finite list of α's. Therefore there must exists $0 < \alpha^* < 1$ such that

$$g^1\left(\alpha^*, 1 - \frac{\epsilon}{n-1}\right) > \alpha^*\left(1 - \frac{\epsilon}{n-1}\right) > h^1\left(\alpha^*, 1 - \frac{\epsilon}{n-1}\right)$$

and

$$g^1\left(1-\alpha^*, 1 - \frac{\epsilon}{n-1}\right) > (1-\alpha^*)\left(1 - \frac{\epsilon}{n-1}\right) > h^1\left(1-\alpha^*, 1 - \frac{\epsilon}{n-1}\right).$$

To complete the proof, set κ to be the minimum of the four terms $g^1(\alpha^*, 1 - \frac{\epsilon}{n-1}) - \alpha^*\left(1 - \frac{\epsilon}{n-1}\right)$, $\alpha^*\left(1 - \frac{\epsilon}{n-1}\right) - h^1(\alpha^*, 1 - \frac{\epsilon}{n-1})$, $g^1(1-\alpha^*, 1 - \frac{\epsilon}{n-1}) - (1-\alpha^*)\left(1 - \frac{\epsilon}{n-1}\right)$ and $(1-\alpha^*)\left(1 - \frac{\epsilon}{n-1}\right) - h^1(1-\alpha^*, 1 - \frac{\epsilon}{n-1})$. $\diamond$

By construction of S^1 the functions $g(\alpha) := \max\{\chi \,|\, (\alpha, 1 - \frac{\epsilon}{n-1}, \chi) \in S^1\}$ and $h(\alpha) := \min\{\chi \,|\, (\alpha, 1 - \frac{\epsilon}{n-1}, \chi) \in S^1\}$ are continuous. Therefore, there exists a neighborhood N_{α^*} of α^* such that $(\alpha, 1 - \frac{\epsilon}{n-1}, \alpha\left(1 - \frac{\epsilon}{n-1}\right) + \text{error}) \in S^1$, for all $|\text{error}| \leq \kappa/2$ and for all $\alpha \in N_{\alpha^*}$. Let $\alpha^u \in N_{\alpha^*}$ for $u \in \{1, \ldots, n-1\}$ be $n-1$ distinct points in this neighborhood that satisfy

$$|\alpha^u - \alpha^*| \;\leq\; \min\left\{\frac{\kappa/2}{1 - \epsilon/(n-1)}, \frac{\epsilon\alpha^*}{(n-1)}, \frac{\epsilon(1-\alpha^*)}{(n-1)}\right\}, \quad \forall u \in \{1, \ldots, n-1\}. \tag{10}$$

Define

$$a^0 \;=\; (\alpha^*, 1-\alpha^*) \qquad a^u \;=\; (\alpha^u, 1-\alpha^u),\ \forall u \in \{1, \ldots, n-1\}. \tag{11}$$

Now we present a family of instances for which the result holds. Consider the graph $G = (\mathcal{N}, \mathcal{A})$, with $|I| = 2$, $|L| = 2$ and $|J| = n$ and $\mathcal{A} = \{(I \times L) \cup (L \times J)\}$. We let $|K| = 2$ and $\lambda^i = \mathbb{1}_i$, where $\mathbb{1}_i \in \mathbb{R}^{|K|}$ is the unit vector in the direction of the i^{th} coordinate axis. Let $b^u = a^u$ for all $u \in \{0, \ldots, n\}$. Set the capacity of arcs (i, l) at n for all $i \in I$ and $l \in L$ and set the capacity of the arcs (l, j) at 1 for all $l \in L$ and $j \in J$. Set capacity of the input nodes to be $2n$, capacity of the pool nodes to be n and the capacity of the output nodes to 1. Let

$$\delta \ := \ \min\{ \|a^u - a^v\|_2 : \ u, v \in \{0, \ldots, n-1\}, \, u \neq v\}.$$

Set $f_{il} = 0$ for all $i \in I, l \in L$. Set $f_{1j} = 1$ (here 1 in the subscript is the first pool) for all $j \in J$ and $f_{2j} = -2/\delta$ (here 2 in the subscript represents the second pool) for all $j \in J$.

We will prove that $z^* = 1$ while $z^S \geq n - \epsilon$.

Claim 2. $z^* = 1$. It is straightforward to construct a feasible solution whose objective function value is 1. Thus $z^* \geq 1$. We will prove that in any optimal solution to the above pooling problem, exactly one output node receives positive flow via the first pool only. Since the capacity is 1 for any output node, this implies that $z^* \leq 1$, completing the proof. We first show that in any optimal pooling solution to the above pooling problem, there is no flow via the second pool. Assume by contradiction that there exists $i \in I := \{1, 2\}$ such that $\sum_{j \in J} v_{i2j} > 0$. Then observe that there must be some $i \in I$ such that $\sum_{j \in J} v_{i1j} > 0$ as otherwise the objective function value of the pooling problem is negative, which cannot be optimal (since sending zero flow in the network is a feasible solution). Due to these positive flows to each of the pools, let $\mu^l \in \mathbb{R}^2$ be the resulting specification vector at pool $l \in \{1, 2\}$. Note that $\mu^1, \mu^2 \in \text{conv}\,\{\lambda^1, \lambda^2\}$. Let $\{j_1, \ldots, j_t\} \subseteq J$ be the subset of output nodes that receive a positive flow. Then we have that $a^{j_q} = b^{j_q} \in \text{conv}\,\{\mu^1, \mu^2\}$ for all $q \in \{1, \ldots, t\}$. We will verify that $t = 1$. Assume by contradiction, $t \geq 2$. Without loss of generality assume that a^{j_1} be the closest point to μ^1. If the distance of a^{j_q} and μ^1 is d_1^q and that between a^{j_q} and μ^2 is d_2^q, then the objective function value per unit flow to the node $j_q \in J$ is

$$(1)\frac{d_2^q}{d_1^q + d_2^q} \ + \ (-2/\delta)\frac{d_1^q}{d_1^q + d_2^q}. \tag{12}$$

For $q \geq 2$, $d_1^q \geq \delta$, since d_1^q is the sum of distance between μ^1 and a^{j_1} and the distance between a^{j_1} and a^{j_q}. Moreover $d_2^q \leq \sqrt{2} - \delta$ since $\mu^1, \mu^2 \in \text{conv}\,\{\mathbb{1}_1, \mathbb{1}_2\}$. Thus, for $q \geq 2$

$$\frac{d_2^q}{d_1^q + d_2^q} \ + \ (-2/\delta)\frac{d_1^q}{d_1^q + d_2^q} \leq \frac{\sqrt{2} - \delta}{d_1^q + d_2^q} \ + \ (-2/\delta)\frac{\delta}{d_1^q + d_2^q} \ < \ 0. \tag{13}$$

Thus, we can improve the solution by not sending any flow to output nodes $j_2, \ldots, j_q$, a contradiction. Therefore, $t \leq 1$. Now observe that the objective function per unit flow to $j_1 \in J$ is $\frac{d_2^1}{d_1^1 + d_2^1} + (-2/\delta)\frac{d_1^1}{d_1^1 + d_2^1}$ which is maximized if $d_1^1 = 0$. Since there is sufficient capacity in pool 1,

there is no flow via the second pool. Finally, since flow is sent to the output nodes via only one pool, exactly one output node can receive positive flow since the a^u are distinct for all $u \in \{0, \ldots, n-1\}$. $\diamond$

CLAIM 3. $z^S \geq n - \epsilon$. We first provide a feasible solution $(\bar{q}, \bar{v}, \bar{y})$ to the pq-relaxation such that the objective function value of this solution is $n - \epsilon$. Set $\bar{q}_{1l} = \alpha^* = a^0_1, \bar{q}_{2l} = 1 - \alpha^* = a^0_2$ for all $l \in L$. Set $\bar{y}_{2j} = 0$ for all $j \in J$, $\bar{y}_{10} = 1$ and $\bar{y}_{1j} = 1 - \frac{\epsilon}{n-1}$ for $j \in \{1, \ldots, n-1\}$. Set $\bar{v}_{i2j} = 0$ for all $i \in I$ and for all $j \in J$. Set $\bar{v}_{i1j} = a^j_i \bar{y}_{1j}$. (Here the subscript i for the term $\bar{v}_{i1j}$ represents the input node and the subscript i for the term a^j_i is used to represent the specification $i \in \{1, 2\}$.). Clearly the objective function value of this solution is $n - \epsilon$. Constraint (1c), the capacity constraints, and the non-negativity constraints are trivially satisfied. Constraints (1d) and (1e) are satisfied, since $\lambda^i_k = 1$ if and only if $i = k$ and $\bar{v}_{i1j} = a^j_i \bar{y}_{1j}$. Constraint (1j) is satisfied trivially if $l = 2$ and is satisfied for $l = 1$ since $\|a^j\|_1 = 1$ and $\bar{v}_{i1j} = a^j_i \bar{y}_{1j}$ for all $j \in J$. Constraint (1k) is satisfied trivially if $l = 2$. It holds for $l = 1$ since

$$\begin{aligned}\sum_{j \in J} \bar{v}_{i1j} = \sum_{j \in J} a^j_i \bar{y}_{1j} &= a^0_i + \left(\sum_{j \in \{1, \ldots, n-1\}} a^j_i \right) \left(1 - \frac{\epsilon}{n-1} \right) \\ &\leq \bar{q}_{i1} + \sum_{j \in \{1, \ldots, n-1\}} \left(\bar{q}_{i1} + \bar{q}_{i1} \frac{\epsilon}{n-1} \right) \left(1 - \frac{\epsilon}{n-1} \right) \leq n \bar{q}_{i1},\end{aligned}$$

where the first inequality follows from the definition of $\bar{q}_{il}$ and (10). Similarly it can be verified that, (2) is satisfied by $(\bar{q}, \bar{v}, \bar{y})$ due to (10). Finally observe that $\bar{v}_{i2j} = \bar{q}_{i2} \bar{y}_{2j}$ for all $i \in I, j \in J$ and also $\bar{v}_{i10} = \bar{q}_{i1} \bar{y}_{10}$. For $l = 1$ and $j \in \{1, \ldots, n-1\}$, we have

$$|\bar{v}_{i1j} - \bar{q}_{i1} \bar{y}_{1j}| = |(a^j_i - \bar{q}_{i1}) \bar{y}_{1j}| = |a^j_1 - \alpha^*| \left(1 - \frac{\epsilon}{n-1} \right) \leq \kappa/2, \tag{14}$$

where the second equality is by definition of $\bar{y}$ and $\bar{q}$, while the inequality is due to (10). Since $\bar{q}_{11} = \alpha^*$ and $\bar{q}_{21} = 1 - \alpha^*$, (8) and (9) along with (14) imply that $(\bar{q}, \bar{v}, \bar{y})$ satisfy the piecewise-linear relaxation of the constraint (1b). $\diamond$

The above three claims complete our proof of Proposition 2. $\square$

Proof of Theorem 1. Since $z^S \leq z^{pq}$ for any piecewise-linear relaxation scheme S, Proposition 1 and Proposition 2 complete the proof of Theorem 1. $\square$

REMARK 1. We make a few comments regarding the instance used in the proof of Proposition 2. For simplicity, we assumed that $a^j = b^j$ for all $j \in J$. However, it is possible to construct similar instances that prove the same bound where $b^j_k - a^j_k = \theta^j_k > 0$ for a suitably small θ^j_k; indeed θ^j_k's can be selected so that the pairwise intersection of "output specification cubes" are empty, where the output specification cube for the j^{th} output node is given by $\{(x_1, ..., x_{|K|}) \,|\, a^j_k \leq x_k \leq b^j_k \; \forall \; k\}$. Secondly, with a little care, all the data in the pooling instance used in the proof of Proposition 2 can be selected to be rational.

4. An Approximation Algorithm and MILP Based Restriction

An n-approximation algorithm for the pooling problem where n is the number of output nodes, i.e., an algorithm which returns feasible solutions with objective function value z^a satisfying $z^a \geq \frac{1}{n} z^*$ where z^* is the optimal objective function value to the pooling problem, follows from the proof of Proposition 1. Indeed, we first solve IPOP, next pick j^* according to (6), and then construct a feasible solution to the pooling problem according to (7). Since IPOP is an LP of polynomial-size with respect to input data, we obtain that this approximation algorithm runs in polynomial-time. We consider two questions in the section:

1. Is it possible to obtain a polynomial-time algorithm with a better approximation guarantee? We show this to be improbable in Theorem 2.

2. The algorithm described above produces 'unreasonable' solutions since in any solution only one output node receives positive flow. Is it possible to fix this, i.e., produce solutions at least as good as that produced by the approximation algorithm but that are more reasonable? We achieve this by embedding all the feasible solutions produced by the approximation algorithm within a MILP to construct a *MILP based restriction of the pooling problem* in Section 4.2.

4.1. Approximation Algorithm and its Analysis

We will prove the following result in this section.

THEOREM 2 **(Approximation Algorithm for Pooling Problem)**. *Let n be the number of output nodes in a pooling problem. There exists a polynomial time algorithm for the pooling problem which guarantees an n-approximation. On the other hand, if there exists a polynomial time approximation algorithm with guarantee better than $n^{1-\epsilon}$ for any $\epsilon > 0$ for the pooling problem, then NP-complete problems have randomized polynomial time algorithms.*

As discussed in the introduction of this section, the existence of a polynomial algorithm is a consequence of the proof of Theorem 1. We prove here the last part of Theorem 2. The proof requires two preliminary results. First we depend upon the following well-known result.

THEOREM 3 **(Hardness to Approximate Max Stable Set (Håstad 1999))**. *In a graph with n nodes, the max stable set problem cannot be approximated in polynomial time within a factor $n^{1-\epsilon}$, for any constant $\epsilon > 0$, unless NP-complete problems can be solved in probabilistic polynomial time.*

The polynomial-time reduction of the stable set problem to the pooling problem, due to Alfaki and Haugland (2013), that shows the NP-hardness of the pooling problem, is also an approximation factor preserving reduction as presented next.

PROPOSITION 3 **(Approximation Factor Preserving Reduction)**. *Given a simple graph with n vertices, there exists an instance of the pooling problem with n output nodes and of size polynomial in the size of the input graph such that*

1. *The size of the maximum stable set of the input graph is less than or equal to the optimal objective function value of the instance of the pooling problem.*

2. *Given any feasible solution for the instance of the pooling problem with objective function value t, it is possible to construct a stable set in the input graph of cardinality greater than or equal to t in polynomial time.*

Proof. Consider a simple graph $G=(V,E)$, where $|V|=n$, corresponding to an instance of the stable set problem. We construct an instance of the pooling problem with $|I|=|J|=|K|=n$ and $|L|=1$ as follows: (i) there is a bijection between V and the sets I,J,K, respectively; (ii) there is an arc from every input node to the pool node and an arc from the pool node to every output node; (iii) the specification vector at input node i is $\mathbb{1}_i \in \mathbb{R}^{|K|}$, the i^{th} unit vector; (iv) for the j^{th} output node, we set the lower bound and the upper bound vectors as

$$a^j_k = \begin{cases} \frac{1}{n} & j=k \\ 0 & j\neq k \end{cases} \qquad b^j_k = \begin{cases} 1 & (j,k)\notin E \\ 0 & (j,k)\in E. \end{cases} \tag{15}$$

(v) all arcs and input and output nodes have unit capacity whereas the pool capacity is n; (vi) $f_{i1}=0$ for every $i\in I$ and $f_{1j}=1$ for every $j\in J$ (1 in the subscript denotes the pool node).

Clearly, the instance of the pooling problem constructed above is of polynomial size with respect to the size of the input graph corresponding to the stable set problem. Now we verify the two requirements for the above reduction to be an approximation factor preserving reduction:

1. Let $\tilde{T}\subseteq J$ be a stable set of maximum cardinality. Then the optimal value of the pooling problem is at least $|\tilde{T}|$ because the following solution

$$\tilde{q}_i = \begin{cases} 0 & i\notin\tilde{T} \\ \frac{1}{|\tilde{T}|} & i\in\tilde{T} \end{cases} \qquad \tilde{v}_{i1j} = \begin{cases} \frac{1}{|\tilde{T}|} & i\in\tilde{T}, j\in\tilde{T} \\ 0 & \text{otherwise} \end{cases} \qquad \tilde{y}_j = \begin{cases} 1 & j\in\tilde{T} \\ 0 & j\notin\tilde{T}. \end{cases}$$

has objective value $|\tilde{T}|$ and it is straightforward to verify its feasibility to the pooling instance.

2. Now consider any feasible solution $(\tilde{q},\tilde{v},\tilde{y})$ of the pooling problem. Its objective value is $\|\tilde{y}\|_1$. We first show that $\tilde{T}:=\{j\in V\colon \tilde{y}_{1j}>0\}$ is a stable set in the original graph. Suppose that $\tilde{y}_{1j},\tilde{y}_{1j'}>0$ for some $j,j'\in J$. Recall that $\lambda^i=\mathbb{1}_i\ \forall i\in I$. First, $b^j_k=0\ \forall (j,k)\in E$ and equation (1d) imply $v_{i1j}=0\ \forall i\in I$ with $(i,j)\in E$. Similarly, $v_{i1j'}=0\ \forall i\in I$ with $(i,j')\in E$. Next, $a^j_j=a^{j'}_{j'}=1/n$ and equation (1e) imply $v_{j1j},v_{j'1j'}>0$. Now equation (1b) gives us $v_{j'1j}=v_{j'1j'}y_{1j}/y_{1j'}$ and hence $v_{j'1j}>0$. Then (1d) enforces that $b^j_{j'}$ must be positive. Thus $(j,j')\notin E$ and since this is true for arbitrary $j,j'\in\tilde{T}$, it follows that $\tilde{T}$ is a stable set in G. Finally, $|\tilde{T}|\geq\|\tilde{y}\|_1$ since $\tilde{y}_{1j}\leq 1$ for all $j\in J$. $\square$

Theorem 3 and Proposition 3 imply the second half of Theorem 2.

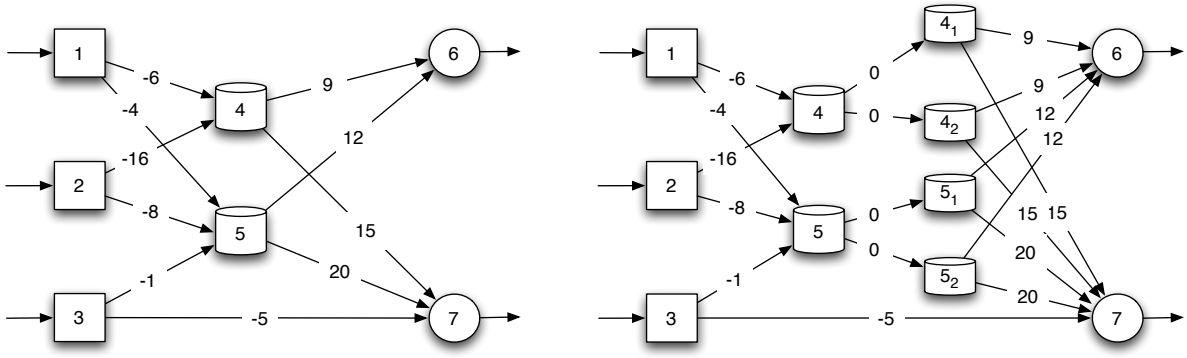
4.2. Using Approximation Algorithm to Construct a MILP Based Restriction

Our main objective in this section will be to construct a MILP, such that its feasible region is (1) a restriction of the feasible region of the pooling problem, and (2) contains all the solutions that are obtained by the n-approximation algorithm implied by the proof of Proposition 1. Moreover, we attempt to include as many feasible points of the pooling problem as possible in this MILP restriction in order to improve the chances of obtaining better solutions.

In Proposition 1, we constructed a feasible solution (7) of the pooling problem such that only one output node $j^* \in J$ receives positive flow. A MILP problem which includes such solutions only (that is set of all solutions of the IPOP that have positive flow to exactly one output node), will provide solutions that are guaranteed to be within a factor of n with respect to the optimal solution of the pooling problem[4]. We next make two observations that allows us to enlarge the set of feasible solutions considered while maintaining MILP representability.

1. The set of feasible solutions of the IPOP, with the restriction that each pool sends flow to only one (but not necessarily the same) output node, is also a restriction of the pooling problem since it yields solutions satisfying the bilinear equality constraint (1b).

2. Next 'split' each pool node into multiple copies. Each copy of the l^{th} pool node receives a predetermined ratio of the total flow received by the l^{th} pool. Each copy is restricted to send flow to at most one output node. This scheme of sending flow to only one output node ensures that (1b) is satisfied. On the other hand, each pool now effectively sends flow to multiple output nodes, since the flow at each pool node was split into copies.



(a) Sample pooling instance. Three inputs, two pools, and two outputs.

(b) Splitting each pool into $\tau = 2$ duplicate nodes. $(4_1, 4_2)$ and $(5_1, 5_2)$ are the duplicates at pools 4 and 5, respectively.

Figure 1 Network flow structure for MILP restriction of sample pooling problem. Weight per unit flow denoted on each arc.

[4] It is easy to verify that this set of solutions is MILP representable.

The basic idea behind constructing the MILP restriction is illustrated in Figure 1. We now formally present the proposed MILP. Let $\tau \in \mathbb{Z}_{++}$ be a chosen level of discretization, that is, the number of copies each pool node is split into. Consider a pool $l \in L$. We create τ duplicate nodes at this pool, denoted as $\{l_t\}_{t=1}^{\tau}$. Introduce an arc of zero cost between l and each node $\{l_t\}_{t=1}^{\tau}$. For every $j \in J$ such that $(l,j) \in \mathcal{A}$, delete the old arc (l,j) and instead introduce the arcs (l_t,j) for every $t \in \{1,\ldots,\tau\}$ with cost $f_{tj} = f_{lj}$. We define the variables w_{il_tj} to be the flow to the j^{th} output node from the i^{th} input node via the t^{th} copy of pool node l. As before, let y_{ij} be the direct flow from the i^{th} input node to the j^{th} output node, y_{lj} be the total effective flow from pool l to output node j and v_{ilj} be the total effective flow to the j^{th} output node from the i^{th} input node via the pool node l. Finally let ζ_{l_tj} be a binary variable that takes a value 1 if positive flow is sent to the j^{th} output node from the t^{th} copy of pool node l. We restrict l_t to receive a predefined fraction γ_{lt}^{th} of the incoming flow at pool l. For every $l \in L$, the vector of fractions $(\gamma_{l1},\ldots,\gamma_{l\tau})$ must be chosen such that it belongs to a set $\Gamma_\tau := \{\chi \in \mathbb{Q}_+^\tau : \sum_{t=1}^{\tau} \chi_t = 1\}$. The MILP restriction is

$$\max_{y,v,w,\zeta} \sum_{i\in I, j\in J} f_{ij} y_{ij} + \sum_{i\in I, l\in L, j\in J} (f_{il} + f_{lj})\, v_{ilj} \tag{16a}$$

$$\text{s.t.} \quad v_{ilj} = \sum_{t=1}^{\tau} w_{il_tj} \qquad \forall i \in I, l \in L, j \in J \tag{16b}$$

$$\sum_{j\in J} w_{il_tj} = \gamma_{lt} \sum_{j\in J} v_{ilj} \qquad \forall i \in I, l \in L, t \in \{1,\ldots,\tau\} \tag{16c}$$

$$y_{lj} = \sum_{i\in I} v_{ilj} \qquad \forall l \in L, j \in J \tag{16d}$$

$$0 \le w_{il_tj} \le c_{lj}\zeta_{l_tj} \qquad \forall i \in I, l \in L, t \in \{1,\ldots,\tau\}, j \in J \tag{16e}$$

$$\sum_{j\in J} \zeta_{l_tj} = 1 \qquad \forall l \in L, t \in \{1,\ldots,\tau\} \tag{16f}$$

$$\zeta_{l_tj} \in \{0,1\} \qquad \forall l \in L, t \in \{1,\ldots,\tau\}, j \in J \tag{16g}$$

$$\text{constraints (1d)} - \text{(1i)}.$$

We will use the following notation. We denote by $\mathbb{PQ}(\tau,\gamma)$ as the above MILP (16), $PQ(\tau,\gamma)$ as the projection of the feasible region of $\mathbb{PQ}(\tau,\gamma)$ on to the space of (y,v) variables, and PQ as the projection of the feasible region of the original pooling problem (1) on to the space of (y,v) variables. Note that when $\tau = 1$, there is only one duplicate node at each pool and hence each pool is constrained to send at most one positive outflow. We refer to the resulting MILP $\mathbb{PQ}(1,\mathbb{1})$, where $\mathbb{1}$ is a vector of ones, as the *SOS-1 restriction* of the pooling problem.

Next, we provide an intuition into the discretization imposed by $\mathbb{PQ}(\tau,\gamma)$. Consider a pool $l \in L$. Since ζ is a binary variable, (16f) implies that each duplicate node l_t must send outflow to

exactly one output node. Consider an arbitrary output $j' \in J$ and suppose that for some $1 \leq \kappa \leq \tau$, the duplicate nodes $l_1, \ldots, l_\kappa$ send their outflow to j'. Then, equation (16c) implies $w_{il_tj'} = \gamma_{lt} \sum_{j \in J} v_{ilj}$ $\forall i \in I, t \in \{1, \ldots, \kappa\}$, and consequently (16b) implies $v_{ilj'} = \sum_{t=1}^{\kappa} \gamma_{lt} \sum_{j \in J} v_{ilj}$ $\forall i \in I$. Note that $\sum_{j \in J} v_{ilj}$ is the total flow on arc (i, l). Thus, the fraction of total incoming flow from input i that is redirected to output j' is equal to the quantity $\sum_{t=1}^{\kappa} \gamma_{lt}$. Different choices of duplicate nodes sending their flow to j', leads to different values of this fraction of outflow from l to j. Hence, we can think of $\mathbb{PQ}(\tau, \gamma)$ as a MILP that discretizes at each pool the fraction of outgoing flow to each of the output nodes. This is in contrast to other discretization techniques in literature mentioned in Section 1.3, which involve restricting some of the problem variables in (1) to take discrete values. One other major difference of our approach is that $\mathbb{PQ}(\tau, \gamma)$ retains a network flow structure (cf. Figure 1(b)) for the discretized pooling problem, where equations (16b) - (16d) represent flow balance constraints.

Remark 2. In $\mathbb{PQ}(\tau, \gamma)$, we have replaced the bilinear constraints (1b) with a discretization formulated using linear constraints and integer variables. Hence, this MILP restriction can be generalized to accommodate linear combinatorial constraints, such as demands or fixed charge costs (D'Ambrosio et al. 2011, Gupte et al. 2013, Meyer and Floudas 2006), and arcs between two pools that are not present in the classical pooling problem defined in Section 1.

We now formally verify the correctness of $\mathbb{PQ}(\tau, \gamma)$ and also present worst case bounds on the solutions provided by $\mathbb{PQ}(\tau, \gamma)$ for the pooling problem. Since the objective function remains unchanged from (1a), we only need to compare the feasible sets for showing (16) is a valid restriction of (1).

Theorem 4. *The following properties are true for $PQ(\tau, \gamma)$.*

1. *$PQ(1, \mathbb{1})$ contains the solution constructed in (7).*
2. *$PQ(1, \mathbb{1}) \subseteq PQ(\tau, \gamma) \subseteq PQ$ for any $\tau \in \mathbb{Z}_{++}$ and $\gamma \in \Gamma_\tau$.*
3. *For any $\tau \in \mathbb{Z}_{++}$ and $\gamma \in \Gamma_\tau$, we have that $z(\tau, \gamma) \geq \dfrac{z^*}{n}$.*
4. *For every positive integer n, and given (τ, γ) where γ is rational, there exists a instance of the pooling problem with n output nodes such that $z(\tau, \gamma) = z^*/n$.*

Proof. We prove these four properties in the following four claims, respectively.

Claim 4. The solution $(\bar{y}, \bar{v})$ constructed in (7) was shown to belong to $PQ(1, \mathbb{1})$ in Proposition 1. In particular, all pools send outflow to the same output j^* constructed in (6). $\diamond$

Claim 5. Take a $\tau \in \mathbb{Z}_{++}$ and $\gamma \in \Gamma_\tau$. Let $(y, v) \in PQ(1, \mathbb{1})$. For every $l \in L$, let $j(l) \in J$ be such that $y_{lj} = 0$ for all $j \in J \setminus j(l)$. Such a $j(l)$ indeed exists for every $l \in L$ by construction

of $\mathbb{PQ}(1,\mathbb{1})$. Let (v,y,w,ζ) be the corresponding point in the feasible region of $\mathbb{PQ}(1,\mathbb{1})$. We construct a solution of $(\hat{y},\hat{v},\hat{w},\hat{\zeta})$ for $\mathbb{PQ}(\tau,\gamma)$ by setting (for all $i\in I, l\in L, j\in J, 1\leq t\leq\tau$)

$$\hat{v}_{il_tj}=\begin{cases}0 & j\neq j(l)\\ \gamma_{lt}v_{ilj} & j=j(l)\end{cases}\quad \hat{\zeta}_{l_tj}=\begin{cases}0 & j\neq j(l)\\ 1 & j=j(l)\end{cases}\quad \hat{y}_{lj}=y_{lj},\quad \hat{v}_{ilj}=v_{ilj}\quad \hat{y}_{ij}=y_{ij}. \tag{17}$$

It is straightforward to verify that this construction satisfies (16b) – (16f) and (1d) - (1i), thereby giving us $PQ(1,\mathbb{1})\subseteq PQ(\tau,\gamma)$.

Now we must show that if $(y,v)\in PQ(\tau,\gamma)$, then $(y,v)\in PQ$. By definition, (y,v) satisfies the constraints (1d) - (1i). We need to construct the matrix $q\in\mathbb{R}^{|I|\times|L|}$ such that (v,y,q) satisfy (1b) and (1c). First observe that there exists w and ζ such that (y,v,w,ζ) belongs to the feasible region of $\mathbb{PQ}(\tau,\gamma)$.

If for some $l\in L$, we have $w_{il_tj}=0$ for all $i\in I$, $1\leq t\leq\tau, j\in J$, then set $q_{il}=1/|\{u\in I\,|\,(u,l)\in\mathcal{A}\}|$ for all $i\in I, l\in L$. It is straightforward to verify that (v_{ilj},y_{lj},q_{il}) satisfy (1b) and (1c).

We now assume that for a given $l\in L$, $w_{il_tj}>0$ for some $i\in I$, $1\leq t\leq\tau$, $j\in J$. To verify (1b), we first show that for all $j\in J$ such that $y_{lj}>0$, the ratio v_{ilj}/y_{lj} depends only on i and l. Define $x_{il}:=\sum_{j\in J}v_{ilj}, \gamma_{lj}:=\sum_{t:\,\zeta_{l_tj}=1}\gamma_{lt}$. Then observe that

$$v_{ilj}=\sum_{t=1}^{\tau}w_{il_tj}=\sum_{\substack{t:\\ \zeta_{l_tj}=1}}w_{il_tj}=\sum_{\substack{t:\\ \zeta_{l_tj}=1}}\left(\sum_{j'\in J}w_{il_tj'}\right)=\sum_{\substack{t:\\ \zeta_{l_tj}=1}}\left(\gamma_{lt}\sum_{j'\in J}v_{ilj'}\right)=\sum_{\substack{t:\\ \zeta_{l_tj}=1}}\gamma_{lt}x_{il}=\gamma_{lj}x_{il} \tag{18}$$

where the first equality follows from (16b), the second equality follows from (16e), the third equality follows from the fact that each of the nodes l_t sends positive flow to at most one output node (cf. 16f), the fourth equality follows from (16c) and the fifth and sixth equalities follow from the definitions of x_{il} and γ_{lj}. Therefore,

$$\frac{v_{ilj}}{y_{lj}}=\frac{v_{lj}}{\sum_{i\in I}v_{ilj}}=\frac{\gamma_{lj}x_{il}}{\sum_{i\in I}\gamma_{lj}x_{il}}=\frac{x_{il}}{\sum_{i\in I}x_{il}} \tag{19}$$

where the first equation follows from (16d) and the second equation follows from (18). Thus, we have that the ratio v_{ilj}/y_{lj} is same for all $j\in J$ such that $y_{lj}>0$. Set $q_{il}:=\frac{x_{il}}{\sum_{i\in I}x_{il}}$. Then by (19) we have that $v_{ilj}=q_{il}y_{lj}$ where $y_{lj}>0$. Moreover, if $y_{lj}=0$, then $v_{ilj}=0$ due to (16d), and therefore again we have $v_{ilj}=q_{il}y_{lj}$. Finally, (1j) and (19) give us $\sum_{i\in I}q_{il}=1$. $\diamond$

Claim 6. Claim 4 and Proposition 1 imply $z(1,1)\geq z^*/n$. By Claim 5, we have $z(\tau,\gamma)\geq z(1,1)$. Thus, we obtain $z(\tau,\gamma)\geq z^*/n$. $\diamond$

Claim 7. Since γ is rational, there exists a positive integer κ such that all possible values of $\frac{y_{lj'}}{\sum_{j\in J}y_{lj}}$ for all $l\in L$ and all $j'\in J$ are integer multiples of $\frac{1}{\kappa}$. Let $0<\epsilon<\min\{\frac{1}{2\kappa},\frac{1}{2(n-1)}\}$. Consider a graph $G=(\mathcal{N},\mathcal{A})$ with $|I|=|K|=|J|=|L|=n$. The arc set $\mathcal{A}=\{(u,u)\,|\,u\in I, u\in L\}\cup(L\times J)$.

Let $\lambda_k^u = 1$ if $u = k$ and $\lambda_k^u = 0$ if $u \neq k$ for all $u \in I$. Let $b_k^j = a_k^j = 1-(n-1)\epsilon$ for $k = j$ and $b_k^j = a_k^j = \epsilon$ for $j \neq k$ for all $j \in J$. Set all arc capacities to 1, pool capacity to n and all input and output node capacities also to 1. The incoming arcs (i,l) have zero weight for all $i \in I$ and the outgoing arcs (l,j) have a unit weight for all $j \in J$.

One feasible solution to the pooling problem is: $q_{il} = 1$ if $i = l$ and $q_{il} = 0$ if $i \neq l$; $y_{lj} = 1-(n-1)\epsilon$ if $l = j$ and $y_{lj} = \epsilon$ for $j \neq l$ and $v_{ilj} = q_{il}y_{lj}$. Since the flows at the output node are at their upper bounds, this solution is optimal with value $z^* = n$.

We now claim that in any solution of the MILP restriction, only one of the output nodes will receive positive flow. Assume by contradiction that output node 1 and output node 2 receive positive flow of $\delta_1 > 0$ and $\delta_2 > 0$. Then, by construction of arc set $\mathcal{A}$ and specification values (λ, a, b), it must be that pools 1 and 2 send positive flows to outputs 1 and 2 in any feasible solution to $\mathbb{PQ}(\tau,\gamma)$. For $\tau = 1$, since (16f) implies that pools 1 and 2 send outflow to exactly one output node, we immediately have a contradiction. Otherwise $\tau \geq 2$ and let the total flow into pool l be $\phi_l > 0$, for $l \in \{1,2\}$, and let the ratio of flow from pool l to output node j be $\frac{r_{lj}}{\kappa}$ (i.e. $\frac{r_{lj}}{\kappa} = \frac{y_{lj}}{\phi_l}$). Note that $r_{lj} \in \{1,\ldots,\kappa-1\}$ for $l,j \in \{1,2\}$. Then observe that we require

$$\phi_1 \frac{r_{11}}{\kappa} = \delta_1(1-(n-1)\epsilon),\ \phi_2 \frac{r_{21}}{\kappa} = \delta_1\epsilon,\ \phi_1 \frac{r_{12}}{\kappa} = \delta_2\epsilon,\ \phi_2 \frac{r_{22}}{\kappa} = \delta_2(1-(n-1)\epsilon). \tag{20}$$

Using the first and third equation in (20) we obtain

$$\frac{\delta_1}{\delta_2} = \frac{\epsilon}{1-(n-1)\epsilon}\frac{r_{11}}{r_{12}} < 2\epsilon\kappa, \tag{21}$$

where the last strict inequality follows from the definition of ϵ (since $1-(n-1)\epsilon > 1/2$) and the construction of r_{12} and r_{11}. Similarly, second and fourth equation give us

$$\frac{\delta_1}{\delta_2} = \frac{1-(n-1)\epsilon}{\epsilon}\frac{r_{21}}{r_{22}} > \frac{1}{2\epsilon\kappa}. \tag{22}$$

However, note that (21) and (22) contradict each other due to $\epsilon < \frac{1}{2\kappa}$. Thus the MILP solution sends flow to only one output node, implying $z(\tau,\gamma) = 1$ and hence $z(\tau,\gamma) = z^*/n$. $\diamond$ $\square$

REMARK 3. For ease of exposition we assume that the level of discretization τ is same across all pools, i.e. $\tau_l = \tau$ for all $l \in L$. However, one can easily formulate $\mathbb{PQ}(\tau,\gamma)$ with τ as a $|L|$-dimensional vector and different values of τ_l's across the pools. The resulting MILP is still a restriction since Claim 5 proves its validity individually for every pool $l \in L$. The tightness of the approximation factor in Claim 7 also holds true.

Theorem 4 does not compare $PQ(\tau,\gamma)$ and $PQ(\tau',\gamma')$ for any two arbitrary discretization schemes (τ,γ) and (τ',γ'). Indeed, if we consider the MILP restriction as discretizing the outflow fractions from each pool, such a comparison may not be possible.

EXAMPLE 1. Consider an arbitrary instance of the pooling problem and let $\tau = 2, \tau' = 3, \gamma_{lt} = 1/2 \ \forall l \in L, t = 1, 2$, and $\gamma'_{lt} = 1/3 \ \forall l \in L, t = 1, 2, 3$. Then for any $i \in I, l \in L, j' \in J$, $(y, v) \in PQ(2, \frac{1}{2})$ will imply $v_{ilj'} / \sum_{j \in J} v_{ilj} \in \{0, 1/2, 1\}$ whereas $(y', v') \in PQ(3, \frac{1}{3})$ will imply $v'_{ilj'} / \sum_{j \in J} v'_{ilj} \in \{0, 1/3, 2/3, 1\}$, and it is obvious that the two solution sets are incomparable.

For any positive integer τ, there are infinitely many possible values for $\gamma \in \Gamma_\tau$, each of which yields a MILP restriction as per Theorem 4. We present two common choices for γ.

DEFINITION 2. The uniform and asymmetric MILP restrictions are defined as follows.
Uniform model $\mathbb{U}(\tau)$*:* Set $\gamma_{lt} = 1/\tau \ \forall t \in \{1, \ldots, \tau\}, l \in L$.
Asymmetric model $\mathbb{A}(\tau)$*:* Set $\gamma_{lt} = 1/2^t \ \forall t \in \{1, \ldots, \tau - 1\}, l \in L$, and $\gamma_{l\tau} = 1/2^{\tau - 1} \ \forall l \in L$.

Let $z^{\mathbb{U}}(\tau)$ and $z^{\mathbb{A}}(\tau)$ be the respective optimal values of $\mathbb{U}(\tau)$ and $\mathbb{A}(\tau)$. Let the projection to (y, v)-space of their respective feasible sets be $U(\tau)$ and $A(\tau)$. Note that $\mathbb{U}(\tau)$ and $\mathbb{A}(\tau)$ are equivalent for $\tau = 1, 2$. The next result is straightforward to verify.

PROPOSITION 4. *$\mathbb{U}(\tau)$ and $\mathbb{A}(\tau)$ can be compared as follows.*

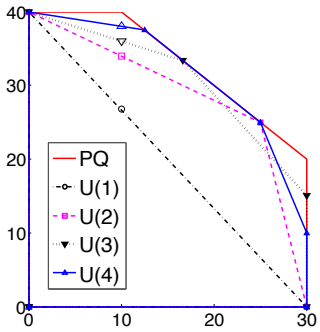
1. *For every integer $\tau \geq 3$, $A(\tau) \subseteq U(2^{\tau - 1})$.*
2. *For every integer $\tau \geq 1$, $U(\tau) \subseteq U(2^\tau)$.*

In general, the restrictions proposed in Definition 2 can be strict subsets of the actual feasible set PQ. We illustrate this point on an example constructed with $|L| = 1$ and $\mathcal{A} \cap (I \times J) = \emptyset$. Note that the set of feasible flows is usually non-polyhedral, however under these conditions it is a polyhedral set. This is because, under the assumptions $|L| = 1$ and $\mathcal{A} \cap (I \times J) = \emptyset$, it is possible to write a MILP representation of problem (1). Details of this reformulation are provided in Appendix A.

EXAMPLE 2. Consider a simple pooling instance with $|I| = 2, |L| = 1, |J| = 2$ and $\mathcal{A} = (I \times L) \cup (L \times J)$. The pool capacity is 50 and capacities at the two outputs are 30 and 40, respectively. The specification vectors at the two inputs are $\lambda^1 = (3, 27, 75), \lambda^2 = (2.25, 5, 40)$ and upper bounds at the outputs are $b^1 = (2.75, 20, 50), b^2 = (2.4, 15, 60)$. In order to provide a clear illustration in $\Re^2$, we project the respective feasible sets to (y_1, y_2), which are the flows to the two outputs from the solitary pool. From Figure 2 it is clear that (a) $\text{Proj}_y U(1) \subset \text{Proj}_y U(\tau)$ for $\tau = 2, 3, 4$, (b) $\text{Proj}_y U(2) \not\subset \text{Proj}_y U(3) \not\subset \text{Proj}_y U(4)$, (c) $\text{Proj}_y U(2) \subset \text{Proj}_y U(4)$ and (d) $\text{Proj}_y U(\tau) \subset \text{Proj}_y PQ$ for $\tau = 1, \ldots, 4$. In fact, it turns out that for this small example, $PQ = U(5)$ and hence $\text{Proj}_y PQ = \text{Proj}_y U(5)$.

In the next section, we will empirically compare the quality of solutions obtained from our MILP restrictions. One striking conclusion from our experiments is that $\mathbb{U}(1), \mathbb{U}(2)$ and $\mathbb{U}(3)$ give very good primal bounds (typically $\leq 10\%$ of the global optimum) within a reasonable amount

Figure 2 Comparing $\text{Proj}_y PQ = \text{Proj}_y U(5)$ and $\text{Proj}_y U(\tau)$ for $\tau = 1, \ldots, 4$ in Example 2.



of time; this practical performance of $\mathbb{U}(\cdot)$ is in stark contrast to the approximation factor from Theorem 4. That being said, we now show that on a specific family of instances, which is probably not representative of the real-world problems, $\mathbb{U}(\cdot)$ can give a very weak bound. These instances are motivated by the stable set instances used in the factor preserving reduction of Proposition 3.

Let $n' \geq 3$ be a positive integer. Consider a graph $G(n')$ with $3n'$ vertices and $4n'$ edges constructed as follows: the vertices are labeled as $1a, 1b, 1c, 2a, 2b, 2c, \ldots, n'a, n'b, n'c$ (for convenience we denote $(n'+1)c := 1c$ and $0c := n'c$); the edges are $(ia, ib), (ib, ic), (ia, ic)$ and $(ic, (i+1)c)$, for every $i = 1, \ldots, n'$. The stability number of $G(n')$ is equal to n'. Using $G(n')$, we construct an instance of the pooling problem with arbitrarily many pools and $|I| = |J| = |K| = 3n'$ as follows: there is an arc from every input to every pool and every pool to every output; each arc has unit capacity whereas each node capacity is redundant; $f_{il} = 0, f_{lj} = 1 \ \forall i \in I, l \in L, j \in J$; the specification vector at input node i is the i^{th} unit vector $\mathbb{1}_i \in \mathbb{R}^{3n'}$; for the j^{th} output node, we set the lower bound and upper bound vectors as per equation (15). From our construction of $G(n')$ it follows that b^j_k is equal to 1 except that for every $i = 1, \ldots, n'$,

$$b^{ic}_k = 0 \ \ k \in \{ia, ib, (i-1)c, (i+1)c\}, \quad b^{ia}_k = 0 \ \ k \in \{ib, ic\}, \quad b^{ib}_k = 0 \ \ k \in \{ia, ic\}.$$

For such instances of the pooling problem, we prove the following.

PROPOSITION 5. *For any integer $n' \geq 3$ and $\tau \in \{1, 2, \ldots, n'\}$ and an instance of the pooling problem constructed from the graph $G(n')$, we have $z^*/z^{\mathbb{U}}(\tau) = n'/\tau$.*

Proof. Let there be $|L|$ pools in the problem instance. We make two claims that complete the proof: $z^* = |L|n'$ and $z^{\mathbb{U}}(\tau) = |L|\tau$. Consider some $i \leq n'$ and pool $l \in L$. Since $G(n')$ contains the edges $(ia, ib), (ib, ic), (ia, ic)$, applying similar arguments as in the second claim of Proposition 3 to the pool l implies that in any feasible solution to this instance, there can be positive flow on at most one of the arcs in the triplet $\{(l, ia), (l, ib), (l, ic)\}$. This, along with unit arc capacities and $f_{lj} = 1 \ \forall l \in L, j \in J$, leads to $z^* \leq |L|n'$. The following feasible solution yields $z^* = |L|n'$: for

every $l \in L$ and $i \leq n'$, send unit flow from input ia to l and unit outflow from l to output ia. For the MILP $\mathbb{U}(\cdot)$, we can similarly argue that any two distinct copies l_t and $l_{t'}$ of the pool l cannot send positive flow to two different output nodes in the triplet $\{ia, ib, ic\}$. Since every copy of pool l must send outflow to exactly one outflow (cf. (16f)) and $\tau \leq n'$, we get that the total outflow from l is at most τ and hence $z^{\mathbb{U}}(\tau) \leq |L|\tau$. The MILP feasible solution - $v_{ia,l,ia} = w_{ia,l_i,ia} = 1$ for every $l \in L$ and $i \leq \tau$, gives us $z^{\mathbb{U}}(\tau) = |L|\tau$.

Thus on such instances we have $\lim_{n' \to \infty} z^*/z^{\mathbb{U}}(\tau) = \infty$ for fixed τ.

5. Computational Experiments

In this section we report computational results for several test instances of the nonconvex pooling problem (1). Our purpose is to assess the practical usefulness of the MILP restriction developed in §4.2. Towards this end, we compare the best feasible solutions obtained within stipulated times from the following approaches: i) a generic global solver `BARON` 12.7.3; (ii) local search solvers/techniques - `SNOPT` 7.2, Successive LP (SLP) solver of `Xpress` 7.5 (FICO 2013), Alternating LP (AltLP) technique; (iii) flow-augmentation (FlowAug) heuristic of Alfaki and Haugland (2013a); (iv) uniform and asymmetric MILP restrictions (cf. Definition 2) $\mathbb{U}(\tau)$ and $\mathbb{A}(\tau)$ for various choices of τ. Another objective of this empirical analysis for different a-priori selected values of τ is to test whether the proposed MILP performs much better in practice than suggested by the theoretical approximation factor of $1/n$ in Theorem 4.

A few comments are in order about our comparison algorithms. We experimented with the specialized global solver `APOGEE` (Misener et al. 2011) for pooling problems but did not observe any performance benefit compared to `BARON`. The SLP algorithm has been widely used in the industry (see Baker and Lasdon 1985) to find feasible solutions for large-scale real-life instances of the pooling problem and has been well-implemented in the `Xpress` solver. The Alternating LP method was used by Audet et al. (2004) as a heuristic for pooling problems and alternates between fixing q and y variables in (1b) and solving corresponding LPs. For given values $(\overline{y}, \overline{v})$ with $\sum_j \overline{y}_{lj} > 0$, the value for q can be calculated simply as $\overline{q}_{il} = \sum_j \overline{v}_{ilj} / \sum_j \overline{y}_{lj}$ since q_{il} denotes the ratio of flow on arc (i,l) to the total incoming flow to pool l. If $\sum_j \overline{y}_{lj} = 0$ then we choose an index $i^* \in I$ uniformly at random such that $(i^*, l) \in \mathcal{A}$ and set $\overline{q}_{i^*l} = 1$ and $\overline{q}_{il} = 0$, $i \in I \setminus i^*$. We also experimented with a few variations without much success in improving on the above implementation: (i) first fixing y instead of q; (ii) fixing the least number of variables at any iteration: for any pool l such that $|\{i \in I\colon (i,l) \in \mathcal{A}\}| \leq |\{j \in J\colon (l,j) \in \mathcal{A}\}|$, we fix q_{il} $\forall i$; otherwise we fix y_{lj} $\forall j$.

5.1. Experimental setup

For SLP and AltLP (implemented in AMPL), the flows (y, v) were warm-started with the optimal solution of pq-relaxation and 4 random initial solutions satisfying capacity constraints (1f)-(1j). Note that these warm-starting values may possibly not correspond to a feasible solution of (1). We did not observe much benefit to warm-starting `SNOPT` and ran it without any initial values. For each method, we report the best lower bounds amongst all warm-starting approaches.

Alfaki and Haugland (2013a)'s heuristic was implemented in GAMS and run on the NEOS server invoking `BARON` 12.7.3 to solve bilinear subproblems. For each of the `std*` instances, we report the best solution between our implementation and that of Alfaki and Haugland; we were able to find better feasible solutions on 7 out of these 20 instances.

We tested $\tau = 1, \ldots, 5$ for $\mathbb{U}(\tau)$ and $\tau = 3, 4, 5$ for $\mathbb{A}(\tau)$. For every fixed discretization level τ, each of $\mathbb{U}(\tau)$ and $\mathbb{A}(\tau)$ is solved with `Cplex` 12.2 for 1 hour. Proposition 4 tells us $\mathbb{A}(3) \subseteq \mathbb{U}(4)$ but since we do not run the MILPs to termination, their time limited performance may be (and indeed is) different.

Since `BARON` is a branch-and-bound based global solver whose algorithm finds feasible solutions among many other things, such as tight bounds via node relaxations, variable bounding tightening, branching decisions etc., it is impossible to know exactly how much time it spent to only find feasible solutions. For this purpose, we gave longer time limits for global optimization. `BARON`, with `Cplex` as the LP solver and `SNOPT` as the NLP solver, was run for 6 hours on the NEOS server. AltLP and `Xpress` SLP were run until termination whereas FlowAug, `SNOPT` and our MILPs were given a time limit of 1 hour.

Our experiments with `SNOPT`, AltLP, SLP and MILP were run on a Linux machine with 64-bit x86 processor and 32GB RAM. To ensure numerical consistency among the different solvers, we used the following algorithmic parameters: `feasibility tolerance` $= 10^{-6}$, `relative optimality gap` $= 0.01\%$, and `absolute optimality gap` $= 10^{-3}$. For our MILPs, `Cplex` was run in single-threaded mode with `integrality tolerance` $= 10^{-5}$ and `MIPEmphasis = Feasibility` to find good feasible solutions. We do not know of a similar feasibility emphasis parameter for `BARON`. All other parameters were set to their default values. In our preliminary experiments with solving $\mathbb{U}(\cdot)$, we did not find any obvious benefit of tuning the `symmetry` parameter in `Cplex`.

5.2. Test instances

The pooling instances commonly used in literature mostly comprise of small-scale problems proposed many years ago; see for example Tawarmalani and Sahinidis (2002). Since these problems are solved in a matter of seconds by `BARON`, we created a test suite of 70 medium- to large-scale

Table 1 Characteristics of the pooling instances.

Source	Label	Inputs	Pools	Outputs	Specs	Avg. num. vars.	
		$\|I\|$	$\|L\|$	$\|J\|$	$\|K\|$	q	y
Alfaki and Haugland (2013)	`stdA0-9`	20	10	15	12	110	157
Alfaki and Haugland (2013)	`stdB0-5`	35	17	21	17	333	388
Alfaki and Haugland (2013)	`stdC0-3`	60	30	40	20	498	655
Random	`randstd11-20`	25	18	25	8	203	205
Random	`randstd21-30`	25	22	30	10	240	302
Random	`randstd31-40`	30	22	35	10	289	360
Random	`randstd41-50`	40	30	45	10	500	635
Random	`randstd51-60`	40	30	50	14	501	708

instances, consisting of 20 instances from Alfaki and Haugland (2013) and 50 randomly generated instances. The problem sizes for these instances are presented in Table 1. Details about our random instance generator are provided in Appendix B. An LP preprocessing technique to reduce sizes of some of the instances is described in Appendix C.

5.3. Numerical results

For each approach (global, MILP, and other heuristics), we calculate the % gap as

$$\% \text{ gap} = 100 \times \left(1 - \frac{BestFeas}{BestRelax}\right), \tag{23}$$

where $BestRelax$ is the best known relaxation bound for (1) returned by `BARON` (local search does not provide relaxation values) and $BestFeas$ is the best feasible solution provided by the corresponding approach. Recall that the $BestFeas$ values for `BARON` are obtained after 6hr. whereas for all other methods, these values are obtained after a maximum of 1hr.

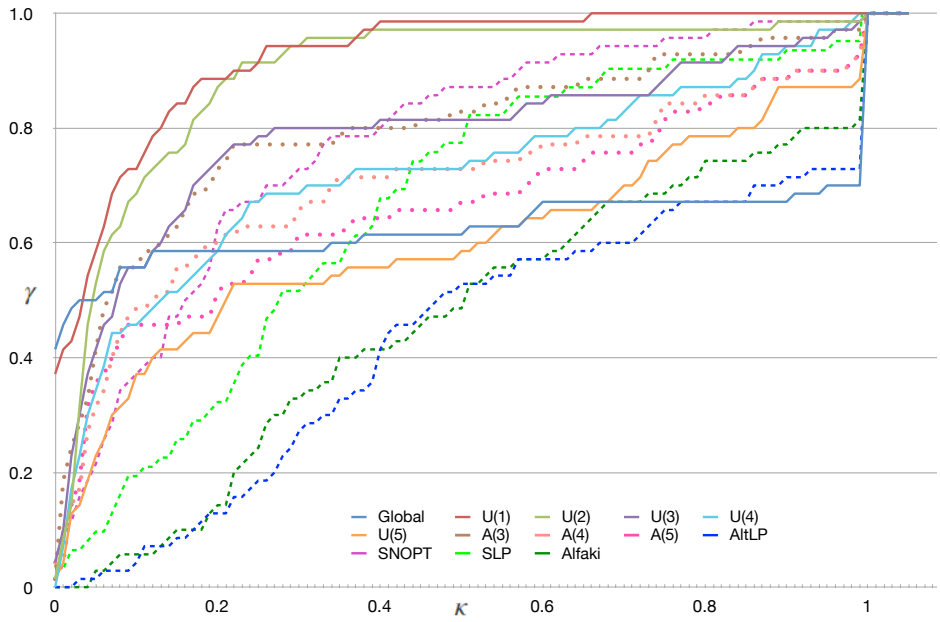
The exact output values and % gaps for each instance are presented in Appendix D. Only 4 instances were solved to within 0.01% gap by global solvers. Table 2 presents the number of nodes in branch-and-bound tree, computational time and optimal value for global optimization of these 4 instances. Columns 5 to 10 report the % gaps of feasible solutions from best uniform MILP, best asymmetric MILP and the local heuristics. The best MILP gaps are quite close to the tolerance 0.01%. Amongst the other heuristics, `SNOPT` seems to perform the best and although its gaps are not too far (except `stdA2`) from optimality, the MILP gaps are consistently better.

Next, we present an overall comparison of the different methods on all the instances (including the ones solved in Table 2). To do so, we plot the performance profiles (cf. Dolan and Moré 2002) of the feasible solutions obtained with the different methods upon termination. Let $\nu_{\mathscr{M}}(\mathcal{I})$ be the best solution value for instance $\mathcal{I}$ upon termination of method $\mathscr{M}$. We calculate the relative metric $\eta_{\mathscr{M}}(\mathcal{I}) := \frac{\nu^{\max}(\mathcal{I}) - \nu_{\mathscr{M}}(\mathcal{I})}{\nu^{\max}(\mathcal{I}) - \nu^{\min}(\mathcal{I})}$, where $\nu^{\max}(\mathcal{I}) = \max_{\mathscr{M}} \nu_{\mathscr{M}}(\mathcal{I})$ and $\nu^{\min}(\mathcal{I}) = \min_{\mathscr{M}} \nu_{\mathscr{M}}(\mathcal{I})$. Any

Table 2 Instances solved to global optimality by BARON.

Instance	Nodes	Time	Opt val	% gaps for					
		(min:sec)	z^*	Best $z^{\mathbb{U}}(\tau)$	Best $z^{\mathbb{A}}(\tau)$	SNOPT	AltLP	SLP	FlowAug
stdA2	109	07:23	23044.16	1.32	1.15	15	36	10	16
stdA9	89	10:44	21933.33	0.25	0.15	1.20	1.80	4.72	3.13
randstd22	11	28:40	67328.70	1.51	1.45	3.44	4.27	4.27	4.27
randstd26	58	55:02	87949.15	0.04	0.01	4.14	16	7	22

Figure 3 Performance profile of best feasible solutions for all instances.



Note. Profiles of $\mathbb{U}(\cdot)$ are solid, of $\mathbb{A}(\cdot)$ are dotted and of other heuristics are dashed lines.

point (κ, γ) on the performance profile for method $\mathscr{M}$ indicates that $\eta_{\mathscr{M}}(\mathcal{I})$ was at most κ for a fraction γ of the 70 instances. Note that $\eta_{\mathscr{M}}(\mathcal{I}) = 0$ means that the best solution for $\mathcal{I}$ was obtained by $\mathscr{M}$. The profiles are drawn in Figure 3.

The performance profiles indicate that in general, the two uniform MILPs $\mathbb{U}(1)$ and $\mathbb{U}(2)$ produce very good quality solutions. $\mathbb{A}(3)$ does better than $\mathbb{U}(4)$ in 1hr., contrary to the expectation from the containment $A(3) \subseteq U(4)$. We also conclude that for any discretization $\tau = 1, \ldots, 4$, $\mathbb{U}(\tau)$ performs better than $\mathbb{U}(\tau+1)$ within the given time limit of 1 hour. Similarly for $\mathbb{A}(\tau)$. This is perhaps not too surprising since a smaller discretization implies smaller size of the resulting MILP. Cplex seems to be able to find good quality solutions much quicker on smaller MILPs. AltLP and FlowAug are the poorest of all the methods whereas SNOPT does reasonably well. Finally, we note that although BARON does not possess the best profile, it does find the best solution on the maximum fraction of instances ($\gamma \approx 0.42$ for $\kappa = 0$). The next best method in

terms of this metric is $\mathbb{U}(1)$ with $\gamma \approx 0.38$ for $\kappa = 0$. As we will see next, BARON's profile has $\gamma \approx 0.42$ for $\kappa = 0$ because it is able to find good solutions for medium sized instances, but its performance significantly deteriorates for larger instances.

We further investigate the profiles in Figure 3 by recording the geometric average % gap for the different methods as a function of the number of outputs n in the instance. For each method, the average is taken over unsolved instances only. We provide % gaps for only four MILPs - $\mathbb{U}(1), \mathbb{U}(2), \mathbb{U}(3)$ and $\mathbb{A}(3)$, since they seem to be the best ones in Figure 3. Recall that according to Theorem 4, our proposed MILP restrictions have a tight approximation factor of $1/n$, which translates to a % gap of approximately $100(n-1)$ %. Looking at the % gaps in Table 3 we see that the MILPs behave much better in practice. In Table 4, we compare the running time of different methods for finding feasible solutions. Note that the running time of BARON is 6 hours for all instances except the ones solved to optimality in Table 2. Overall, we can see that the other heuristic methods run quicker than our MILPs but provide poorer solutions.

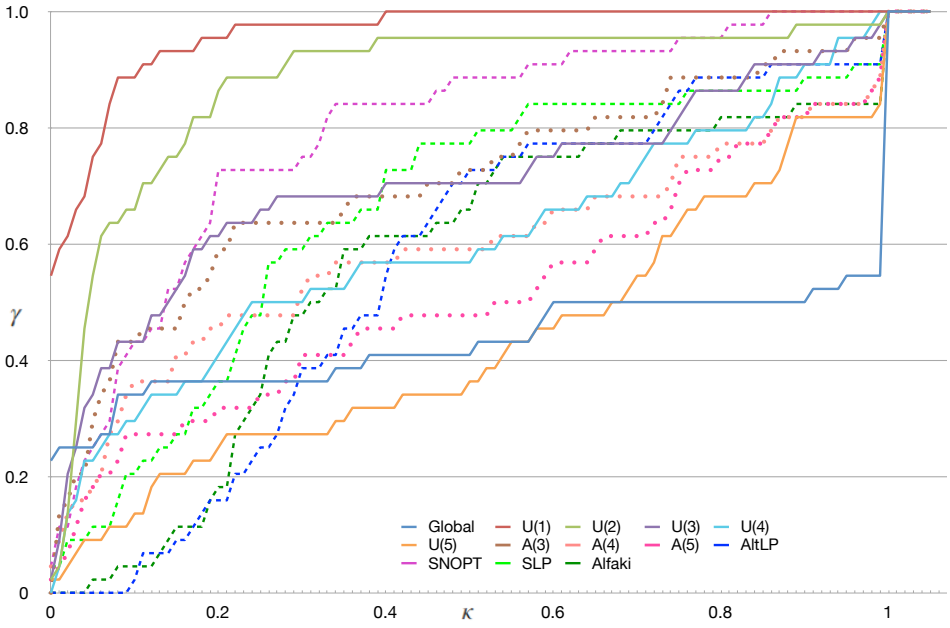
Table 3 Geometric average % gap versus number of outputs $|J| = n$.

n	Global	$\mathbb{U}(1)$	$\mathbb{U}(2)$	$\mathbb{U}(3)$	$\mathbb{A}(3)$	SNOPT	AltLP	SLP	FlowAug
15	1.50	2.75	1.90	1.49	1.42	2.54	11	6.88	9.52
21	2.79	1.01	1.59	1.76	1.93	4.88	5.86	4.38	7.84
25	0.64	3.43	1.99	1.97	1.66	8.98	30	22	25
30	2.77	2.59	1.60	1.72	1.69	3.83	16	11	17
35	7.26	2.70	3.45	5.28	4.48	5.35	19	16	18
40	95	11	14	14	16	47	24	17	29
45	75	5.21	20	61	51	18	37	23	33
50	47	1.92	8.82	19	28	10	25	17	20
Avg	7.47	2.91	3.91	5.31	5.28	7.18	19	13	18

Table 4 Geometric average time (sec.) versus number of outputs $|J| = n$. For AltLP and SLP, average is of total time for all warm-starts.

n	$\mathbb{U}(1)$	$\mathbb{U}(2)$	$\mathbb{U}(3)$	$\mathbb{A}(3)$	SNOPT	AltLP	SLP	FlowAug
15	10	200	520	602	5	42	47	6
21	603	3083	3341	2977	588	107	128	31
25	34	1433	3597	3597	45	26	39	11
30	187	2459	3597	3547	118	38	61	19
35	526	3162	3588	3590	274	51	77	20
40	3458	3572	3589	3598	3179	280	345	87
45	1229	3596	3598	3596	978	174	202	61
50	1176	3598	3597	3599	2412	274	304	169
Avg	273	1914	2710	2735	211	77	108	30

Figure 4 Performance profile of best feasible solutions for 44 large instances ($n \geq 30$).



Note. Profiles of $\mathbb{U}(\cdot)$ are solid, of $\mathbb{A}(\cdot)$ are dotted and of other heuristics are dashed lines.

Table 3 also shows us that for small values of n, the feasible solutions from `BARON` are very good and sometimes better than those from other methods. This also explains the observation that `BARON` found best quality solutions on the largest fraction of instances in Figure 3. As the value of n increases ($n \geq 30$), the performance of `BARON` deteriorates. We plot the performance profiles on these large instances in Figure 4. Notice that now $\mathbb{U}(1)$ not only has the best profile but also provides the best solution on largest fraction of instances ($\gamma \approx 0.55$). The profile of `BARON` becomes considerably worse than in Figure 3. $\mathbb{U}(1)$ seems to be doing the best, followed by $\mathbb{U}(2)$ and `SNOPT`.

The small % gaps for MILPs in Table 3 indicate that not only is the feasible solution from MILP very good, but also the *BestRelax* value from `BARON` is very close to the global optimal value. Thus `BARON` seems to be doing a remarkably good job of obtaining tight relaxation bounds for the pooling problem. The high average % gaps for large instances implies that the feasible solutions found by `BARON` are not as good. In our experiments, we observed that strong relaxation bounds were obtained initially, perhaps at or very close to the root node after preprocessing and bound tightening. Similarly, the solver was unable to significantly improve the feasible solutions found during initial stages of branch-and-bound. It seems that a generic solver like `BARON` faces considerable difficulty in finding good quality feasible solutions to large-scale pooling problems.

Finally, we remark that for all the 20 `std*` instances, our MILPs provide better solutions than Alfaki and Haugland (2013a) and on 10 of these instances, we are able to obtain best

known solutions; see Alfaki and Haugland (2011), Gupte et al. (2013) for previous best solutions obtained using variable discretization strategies.

6. Comments

6.1. On the Analytical results

The key results we proved in this paper are the following: We first showed that MILP based relaxations have a performance guarantee of n, i.e. the ratio of the objective function value of the relaxation to the optimal objective function value of the pooling problem is n, where n is the number of output nodes. Moreover this bound is tight. We also used this result to construct a n-approximation algorithm, showed that it is unlikely that a polynomial algorithm can have a better approximation guarantee, and more importantly used this approximation algorithm to construct a new class of MILP restriction for the pooling problem.

We make two comments regarding these results. First, note that these result imply the following: The complexity of the problem depends very differently on the number of pool and output nodes. On the one hand the pooling problem is NP-hard (Alfaki and Haugland 2013) even if there is one pool. On the other hand, if there is only one output node, then Theorem 1 implies that the pooling problem is polynomially solvable.

Our final comment is regarding a slight strengthening of Theorem 1 in some specific cases. Consider the bipartite sub-graph $G'(L \cup J, \mathcal{A}')$ corresponding to the pool nodes and the output nodes and suppose that there are q disconnected components. Theorem 1 and consequently Theorem 2 can be strengthened when $q \geq 2$: Let $\{t_p\}_{p=1}^{q}$ be the number of output nodes in different disconnected components of G' and let $t^* = \max_{1 \leq p \leq q}\{t_p\}$. Then by a straightforward change in the proof of Theorem 1 (instead of updating the optimal solution of the IPOP by sending flow only to the 'most productive' output node among all output nodes and zero flow to others, send flow to 'most productive' output node within each disconnected component and zero to other output nodes), it can be shown that $z^{pq}/z^* \leq t^*$. Consequently, we have the following result.

COROLLARY 1. *Let t^* be the maximum number of output nodes in any disconnected component of the subgraph defined by pool nodes and output nodes of the pooling problem. Then there exists a t^*-approximation algorithm to solve the pooling problem. In particular, if every pool node has an out-degree of* 1*, then the pooling problem can be solved as an LP in polynomial-time.*

We end this section with a few open questions.

QUESTION 1. Complexity: There is a need to better understand the complexity of the pooling problem. For example, if we fix the number of specifications, what is the complexity of the

problem? (If in addition the number of pool nodes is 1 and there are no arcs between the input and output nodes, then the problem can be solved in polynomial time; see Alfaki and Haugland (2013)). For any fixed $\Delta_{\max}$, consider instances of the pooling problem where the out-degree of the pool nodes is at most $\Delta_{\max}$. Then (for a fixed value of $\Delta_{\max}$), what is the complexity status of this problem? In particular, is there a polynomial-time algorithm?

Question 2. Better MILP restrictions: In Theorem 4 it was shown that the performance guarantee of n for the proposed MILP based restriction is tight in the worst case. Is it possible to obtain a better guarantee than n (n is the number of output nodes), by using some other MILP restrictions of the pooling problem?

6.2. On the Empirical Results

If we view the performance of the MILP based restriction of the pooling problem proposed in this paper on its own, we observe that the quality of solutions that are produced by MILP are significantly better than the theoretical performance guarantee of n.

The MILP based restriction also performed surprisingly well in comparison to all the other methods. Indeed, our MILPs are able to find the best known solutions for half the instances from Alfaki and Haugland (2011). We also note that the dual bounds obtained by BARON (even after only 1 hr of running time) are very good. This is a very impressive performance for a general purpose solver. On the other hand, the overall poor performance of BARON is perhaps to be expected since it is a generic global solver and hence may be unable to exploit the specific structure of the pooling problem. Similarly, the performance of alternating LP and successive LP methods are less impressive, perhaps again due to their lack of ability to take advantage of the specific structure of the pooling problem. While `SNOPT` produces much better results that successive and alternative LP methods, the solution produced by the MILP model $\mathbb{U}(1)$ are almost consistently better than those by `SNOPT` for larger instances. Other than the better quality of solutions produced by the MILP based restriction, we argue that there are two other advantages of using MILP techniques as against the local search techniques: (1) The quality of solutions produced by local search techniques is very dependent on the initial solution. On the other hand, using powerful MILP solvers, one can be more certain of obtaining a global solution (for the MILP). (2) The pooling problem is essentially a stylized model and actual applications have various side constraints. In particular, this includes the presence of integer variables, such as, for modeling fixed charge costs or modeling operational features such as turning an arc "on or off" (cf. D'Ambrosio et al. 2011, Meyer and Floudas 2006). In such cases, using MILP approach is more useful than a NLP solver.

In general, these results (and the arguments above) make us optimistic about using MILP based restriction techniques that combine the power of MILP solvers together with the use of the structure of the problem to solve bilinear programs. Finally, we note that it may be worthwhile to study the polyhedral and symmetry structure (especially when the discretization is symmetric) of the MILP based restriction proposed in this paper. In particular, this may help in solving these instances even quicker than default `Cplex`, in turn becoming more useful for finding good solutions for the pooling problem.

Acknowledgments

We thank Shabbir Ahmed and Myun-Seok Cheon for many useful discussions on this topic. We also gratefully acknowledge ExxonMobil Research and Engineering for support.

References

Alfaki, M., D. Haugland. 2011. Comparison of discrete and continuous models for the pooling problem. A. Caprara, S. Kontogiannis, eds., *11th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*. OpenAccess Series in Informatics, 112–121.

Alfaki, M., D. Haugland. 2013a. A cost minimization heuristic for the pooling problem. *Annals of Operations Research* 1–15doi:10.1007/s10479-013-1433-1.

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

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

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

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

D'Ambrosio, C., J. Linderoth, J. Luedtke. 2011. Valid inequalities for the pooling problem with binary variables. Oktay Günlük, Gerhard Woeginger, eds., *Integer Programming and Combinatorial Optimization*, *Lecture Notes in Computer Science*, vol. 6655. Springer, 117–129.

Dolan, E. D., J. J. Moré. 2002. Benchmarking optimization software with performance profiles. *Mathematical Programming* **91** 201–213.

FICO. 2013. FICO Xpress Optimization Suite 7.5. URL `www.fico.com/xpress`.

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

Gupte, A., S. Ahmed, M. S. Cheon, S. Dey. 2013. Pooling problem: relaxations and discretizations. *Optimization and Analytics in the Oil and Gas Industry*, chap. *accepted*. International Series in Operations Research and Management Science, Springer, 1–38. URL `http://www.optimization-online.org/DB\_HTML/2012/10/3658.html`.

Håstad, J. 1999. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica* **182** 105–142.

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

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

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

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

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

Nemhauser, G.L., L.A. Wolsey. 1988. *Integer and combinatorial optimization*. Wiley-Interscience.

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

Quesada, I., I.E. Grossmann. 1995. Global optimization of bilinear process networks with multicomponent flows. *Computers and Chemical Engineering* **19**(12) 1219–1242.

Sherali, H.D., W.P. Adams. 1998. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, *Nonconvex Optimization and its Applications*, vol. 31. Kluwer Academic Publishers.

Tawarmalani, M., N.V. Sahinidis. 2002. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, chap. 9: The Pooling problem. Kluwer Academic Publishers, 281–310.

Wicaksono, D. S., I. A. Karimi. 2008. Piecewise MILP Under- and Overestimators for Global Optimization of Bilinear Programs. *AIChE Journal* **54** 991–1008.

BIOGRAPHIES

Santanu S. Dey is a Fouts Family Associate Professor in the Stewart School of Industrial and Systems Engineering at Georgia Institute of Technology. His research interest is in the area of optimization, specifically mixed integer linear programming, mixed integer nonlinear programming and global optimization. His honors include being winner of the INFORMS George Nicholson Student Paper Competition, one IBM Faculty award and the NSF CAREER award.

Akshay Gupte is an Assistant Professor in the Department of Mathematical Sciences at Clemson University. His research interests are in the areas of discrete optimization and mixed integer nonlinear programming.

Supplementary information

Appendix A: MILP representability of the single pool case

Consider the pooling problem defined on a graph G with $|L| = 1$ and $\mathcal{A} \cap (I \times J) = \emptyset$, i.e. flows from all inputs nodes are mixed at a single pool and then sent to multiple output nodes. Proposition 3 proves NP-hardness of the pooling problem using a reduction from the stable set problem to this specific instance of pooling where values for (λ, a, b) are assigned based on the connectivity of G. We argue that assuming this special structure on G, the pooling problem (1) can be written as a 0/1-MILP for arbitrary values of (λ, a, b). We accomplish this by defining a new variable[5] $x_i := \sum_{j \in J} v_{ij}$ for the total flow from input i to the solitary pool (since $|L| = 1$, we drop subscript l for convenience). Then, (1b) transforms (1d) and (1e) to

$$a_k^j y_j \;\le\; \left[\sum_{i \in I} \lambda_k^i q_i\right] y_j \;\le\; b_k^j y_j \qquad \forall k \in K, j \in J.$$

Hence, we must ensure that for any $j \in J$, $y_j > 0$ implies $\sum_{i \in I} \lambda_k^i q_i \in [a_k^j, b_k^j]$ for all $k \in K$. Thus, the pooling problem becomes combinatorial in nature: find the output nodes to send flows of maximum weight on capacitated graph G such that for every $k \in K$, the value of specification k produced at the solitary pool satisfies the requirements for all the output nodes receiving positive flow. This combinatorial problem can be formulated as a MILP by introducing new binary variables $\xi \in \{0,1\}^{|J|}$ and imposing $0 \le y_j \le c_j \xi_j$, $\forall j$. Output j receives a positive flow if and only if $\xi_j = 1$. Furthermore, since $q_i = x_i / \sum_{t \in J} y_t$ by definition, we want

$$\left[a_k^j \sum_{t \in J} y_t\right] \xi_j \;\le\; \left[\sum_{i \in I} \lambda_k^i x_i\right] \xi_j \;\le\; \left[b_k^j \sum_{t \in J} y_t\right] \xi_j \qquad \forall k \in K, j \in J.$$

In the above equation, product terms between a $\{0,1\}$ variable and continuous variables can be linearized by adding new variables and constraints corresponding to McCormick envelopes. Finally, the MILP formulation of the pooling problem is

$$\max_{x,y,\xi,\omega,\rho} \sum_{i \in I} f_i x_i + \sum_{j \in J} f_j y_j \tag{EC.1a}$$

$$\text{s.t.} \quad \sum_{i \in I} x_i = \sum_{j \in J} y_j, \quad \sum_{j \in J} y_j \le c, \quad x_i \le c_i \;\forall i \in I, \quad y_j \le c_j \xi_j,\; \xi_j \in \{0,1\} \;\forall j \in J \tag{EC.1b}$$

$$\sum_{i \in I} \lambda_k^i \rho_{ij} \;\ge\; a_k^j \omega_j, \qquad \sum_{i \in I} \lambda_k^i \rho_{ij} \;\le\; b_k^j \omega_j \qquad \forall k \in K, j \in J \tag{EC.1c}$$

$$\sum_{t \in J} y_t + c\xi_j - c \le \omega_j, \quad \omega_j \le \sum_{t \in J} y_t, \quad \omega_j \le c\xi_j \qquad \forall j \in J \tag{EC.1d}$$

$$x_i + c_i \xi_j - c_i \le \rho_{ij}, \quad \rho_{ij} \le x_i, \quad \rho_{ij} \le c_i \xi_j \qquad \forall i \in I, j \in J \tag{EC.1e}$$

$$\text{All variables non-negative.} \tag{EC.1f}$$

[5] The idea of using x variables, instead of (q, v), is similar to the so-called p-formulation for pooling (Tawarmalani and Sahinidis 2002, cf.).

Appendix B: Generating Random Instances.

We generated fifty random instances as follows. Arcs were randomly generated with the following probabilities:

$$\mathbb{P}\{(u,v)\in\mathcal{A}\} = \begin{cases} 0.35 & u\in I, v\in L \\ 0.42 & u\in L, v\in J \\ 0.04 & u\in I, v\in J. \end{cases}$$

The choices for edge probabilities are arbitrary up to the following motivating factors: 1) We wish to construct large and relatively sparse graphs. Although dense graphs may lead to more bilinear terms, the real-world instances are typically sparse and the practical challenges in solving them arise from their large size. 2) For any pool $l\in L$, the number of bilinear terms in (1b) depends on the number of output arcs from l. Hence, we wish to have more output arcs, generated w.p. 0.42, than input arcs, generated w.p. 0.35. 3) Since bypass arcs $(i,j)\in\mathcal{A}$, for some $i\in I$ and $j\in J$, can always be modeled by introducing an extra pool with single input and output arc, we generate very few (w.p. 0.04) connections of this type. We also ensure that $|\{i\in I\colon (i,l)\in\mathcal{A}\}|\geq 0.4|I|$ for $l\in L$, i.e. each pool is connected to at least 40% of the total number of inputs. The arc weights are given by

$$f_{uv} = \begin{cases} f_v - f_u, & u\in I, v\in J \\ -f_u, & u\in I, v\in L \\ f_v, & u\in L, v\in J \end{cases}$$

where $f_u\geq 0$ is the cost associated with sending a unit outflow from input $u\in I$ and $f_v\sim\lfloor U(15,100)\rfloor$ is the revenue of a unit inflow into output $v\in J$. The costs at input nodes are built as follows. For each $i\in I$, we evaluate the set $J_i := \{j\in J\colon \exists \text{ path from } i \text{ to } j\}$ and sort it in increasing order of f_j. Then, we partition J_i into sets J_i^- and J_i^+ as follows : Each $j\in\{1,\ldots,0.3|J_i|\}$ can belong to J_i^- w.p. 0.25. All other $j\in J_i$ belong to J_i^+. Define

$$f_i^{\min} := \max_{j\in J_i^-} f_j \qquad\qquad f_i^{\max} := \min_{j\in J_i^+\colon f_j > f_i^{\min}} f_j.$$

The input cost is generated as $f_i\sim\lfloor U(f_i^{\min}, f_i^{\max})\rfloor$. The motivation for this approach is to create at most 30% of the paths from each input with negative cost so that an optimal solution may try to send very little flow to some of the outputs (objective is maximization type). We believe that this might make it considerably harder to find an optimal solution than say simply trying to send as much flow as possible from inputs to outputs.

The node capacities are $c_l\in\lfloor U(50,140)\rfloor$, for $l\in L$, and $c_j\in\lfloor U(70,220)\rfloor$, for $j\in J$. For $i\in I$, c_i is chosen uniformly at random from the discrete set $\{10,12,17,21,28,34,37,42\}$ w.p. 0.15, otherwise it is chosen as $\lfloor U(60,180)\rfloor$. We do this to create a some nodes in the graph with a low capacity. The arc capacities are $c_{uv} = \min\{c_u, c_v\}$ for every $(u,v)\in\mathcal{A}$.

Specification values. The specification values at the inputs are $\lambda^i_k \sim U(8,75)$. To generate the values at the outputs, we first observe from our preprocessing technique that

$$a^j_k \geq \lambda^{\min}_{jk} := \min\{\lambda^i_k \colon i \in I_j\} \qquad\qquad b^j_k \leq \lambda^{\max}_{jk} := \max\{\lambda^i_k \colon i \in I_j\}$$

where I_j is the subset of inputs from which there exists a path to output j. Now pick a random direction $\xi \in \mathbb{R}^{|\mathcal{A}|}$ such that $\xi_{uv} \in U(-100,100)$ for all $(u,v) \in \mathcal{A}$. Then, we solve an LP

$$\begin{aligned} \max_{y,v} \quad & \sum_{i\in I, j\in J} \xi_{ij} y_{ij} \; + \sum_{i\in I, l\in L, j\in J} (\xi_{il} + \xi_{lj})\, v_{ilj} \\ \text{s.t.} \quad & \text{constraints (1f)} - \text{(1j)} \end{aligned} \tag{RandomFlowLP}$$

whose optimal solution $(\hat{y}, \hat{v})$ may or may not satisfy the flow consistencies (1b) and (1c) and specification requirements (1d) and (1e). By definition, q_{il} is the fraction of flow on arc (i,l) to the total incoming flow to pool l. Here, $\sum_j v_{ilj}$ is the flow on (i,l), whereas $\sum_j y_{lj} = \sum_{i,j} v_{ilj}$ is the total incoming flow to (and outgoing flow from) pool l. Hence, the flow ratio variables $\hat{q}$ corresponding to a feasible flow $(\hat{y}, \hat{v})$ are given as follows: for any pool l with a positive total incoming flow, we set

$$\hat{q}_{il} := \frac{\sum_j \hat{v}_{ilj}}{\sum_{i,j} \hat{v}_{ilj}} \qquad \text{if } \sum_{i,j} \hat{v}_{ilj} > 0,$$

otherwise, we set the ratio equal to 1 on the first incoming arc to l, say $q_{1l} = 1$, and equal to 0 on all other incoming arcs to l. By construction, we have $\sum_i \hat{q}_{il} = 1$ for each l. Define

$$\overline{v}_{ilj} \;:=\; \hat{q}_{il}\hat{y}_{lj}, \qquad \forall i \in I, l \in L, j \in J.$$

We verify that $(\hat{y}, \overline{v})$ satisfies the flow constraints (1f) - (1j). It suffices to show that at each pool l, we have $\sum_i \overline{v}_{ilj} = \sum_i \hat{v}_{ilj}$ and $\sum_j \overline{v}_{ilj} = \sum_j \hat{v}_{ilj}$. We only argue for pools with positive incoming flow; the other case is trivial.

$$\sum_i \overline{v}_{ilj} \;=\; \hat{y}_{lj} \sum_i \hat{q}_{il} \;=\; \hat{y}_{lj} \;=\; \sum_i \hat{v}_{ilj}$$

where the last equality is due to $(\hat{y}, \hat{v})$ being feasible to RandomFlowLP.

$$\sum_j \overline{v}_{ilj} \;=\; \hat{q}_{il} \sum_j \hat{y}_{lj} \;=\; \hat{q}_{il} \sum_j \sum_i \hat{v}_{ilj} \;=\; \frac{\sum_j \hat{v}_{ilj}}{\sum_{i,j} \hat{v}_{ilj}} \sum_{i,j} \hat{v}_{ilj} \;=\; \sum_j \hat{v}_{ilj}$$

where second equality is by substitution from (1j) and third equality is by construction of $\hat{q}_{il}$.

For output $j \in J$, calculate the specification $k \in K$ produced by the flow $(\hat{y}, \overline{v})$ as the quantity

$$\hat{\mu}^j_k := \begin{cases} \dfrac{\sum_{i\in I} \lambda^i_k \hat{y}_{ij} + \sum_{i\in I, l\in L} \lambda^i_k \overline{v}_{ilj}}{\sum_{i\in I\cup L} \hat{y}_{ij}} & \text{if } \sum_{i\in I\cup L} \hat{y}_{ij} > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Next we solve the following LP relaxation of the pq-formulation,

$$\max_{y,v} \text{(1a)} \quad \text{s.t. (1f)} - \text{(1j)} \tag{FlowLP}$$

which has the same constraints as RandomFlowLP but the objective is the original function (1a). As before, compute the three quantities

1. $\tilde{q}$ corresponding to the LP solutions $(\tilde{y}, \tilde{v})$,
2. $\widetilde{v}_{ilj} = \tilde{q}_{il}\tilde{y}_{lj}, \forall i, l, j$, and
3. $\tilde{\mu}^j_k, \forall j, k$.

We would like to construct hard instances of the pooling problem with the following two properties.

Nontrivial feasible region: We have chosen $(\hat{y}, \overline{v}, \hat{q})$ to be a non-trivial feasible solution to the instance. However, note that this solution may not be an optimal solution, since it is constructed using a random direction ξ, which is different from the objective function f.

Nontrivial optimal solution: In order to make the instance considerably harder, we want to impose the condition that a global solver cannot find a optimal solution at the root node by simply solving FlowLP. Hence, we want to cut off the point $(\tilde{y}, \widetilde{v}, \tilde{q})$.

The above two properties can be satisfied by constructing a nonempty interval around $\hat{\mu}^j_k$ that does not include $\tilde{\mu}^j_k$ and such that the lower (resp. upper) limit of this interval corresponds to the minimum (resp. maximum) specification requirement for k at output j. Thus, we have

$$a^j_k = \begin{cases} 1.1\lambda^{\min}_{jk} & \text{if } \hat{\mu}^j_k = 0 \\ \hat{\mu}^j_k & \text{if } \hat{\mu}^j_k \geq \tilde{\mu}^j_k \\ \hat{\mu}^j_k - \epsilon^0_{jk}(\hat{\mu}^j_k - \lambda^{\min}_{jk}) & \text{if } 0 < \hat{\mu}^j_k < \tilde{\mu}^j_k \end{cases}$$

$$b^j_k = \begin{cases} 0.9\lambda^{\max}_{jk} & \text{if } \hat{\mu}^j_k = 0 \\ \hat{\mu}^j_k + \epsilon^1_{jk}(\lambda^{\max}_{jk} - \hat{\mu}^j_k) & \text{if } \hat{\mu}^j_k \geq \tilde{\mu}^j_k \\ \hat{\mu}^j_k & \text{if } 0 < \hat{\mu}^j_k < \tilde{\mu}^j_k \end{cases}$$

where $\epsilon^0_{jk} \sim U(0.05, 0.13)$ and $\epsilon^1_{jk} \sim U(0.08, 0.15)$. The smaller these numbers, the smaller is the range on the specification requirements in (1d) and (1e) and hence harder are the instances. The scaling factors 0.9 and 1.1 are valid as long as $\lambda^{\max}_{jk}/\lambda^{\min}_{jk} \geq 11/9$ and can be modified suitably.

Appendix C: Preprocessing

Recall the specification requirement constraints (1d) and (1e) for any $j \in J, k \in K$. After substituting $v_{ilj} = q_{il}y_{lj}$, we get

$$a^j_k \sum_{i \in I \cup L} y_{ij} \leq \sum_{i \in I} \lambda^i_k y_{ij} + \sum_{i \in I, l \in L} \lambda^i_k q_{il} y_{lj} \leq b^j_k \sum_{i \in I \cup L} y_{ij}$$

If $\sum_{i\in I\cup L} y_{ij} > 0$, then scaling both sides of above by $\sum_{i\in I\cup L} y_{ij}$ gives the following interpretation:

$$\sum_{i\in I\cup L} y_{ij} > 0 \quad \Rightarrow \quad p_{j\cdot} \in \text{conv}\left(\cup_{i\in I_j} \lambda_{i\cdot}\right) \text{ and } p_{j\cdot} \in [a^j_{\cdot}, b^j_{\cdot}],$$

where $p_{j\cdot}$ is the vector of concentration values produced at output j and $I_j := \{i \in I\colon \exists \text{ path from } i \text{ to } j\}$. Thus, we want to check if $\sum_{i\in I\cup L} y_{ij} > 0$ implies $\text{conv}\left(\cup_{i\in I_j} \lambda_{i\cdot}\right) \cap [a^j_{\cdot}, b^j_{\cdot}] \neq \emptyset$. This can be easily verified by solving an LP.

Observation 1 **(LP Preprocessing)**. *For any $j \in J$ such that the system of linear inequalities*

$$a^j_{\cdot} \leq \sum_{i\in I_j} \lambda_{i\cdot}\xi \leq b^j_{\cdot}, \quad \sum_{i\in I_j} \xi_i = 1,\ \xi \geq \mathbf{0} \tag{EC.2}$$

is infeasible, then $\sum_{i\in I\cup L} y_{ij} = 0$ is valid to the pooling problem and hence output node j can be deleted from the graph.

If (EC.2) *is feasible, then for any $k \in K$ such that $\max_{i\in I_j} \lambda_{ik} \leq b^j_k$, or $\min_{i\in I_j} \lambda_{ik} \geq a^j_k$, we can relax* (1d) *or* (1e) *for the specific $k \in K$ and $j \in J$, respectively.*

Table EC.1 Effects of Observation 1 on instances from Alfaki and Haugland (2013).

#	Deleted		#	Deleted	
	$j \in J$	Constr.		$j \in J$	Constr.
stdA0	3	20	stdB0	5	15
stdA1	5	21	stdB1	3	9
stdA2	5	13	stdB2	0	15
stdA3	3	15	stdB3	4	15
stdA4	2	9	stdB4	5	10
stdA5	4	16	stdB5	2	8
stdA6	0	15	stdC0	5	15
stdA7	3	6	stdC1	4	18
stdA8	1	3	stdC2	3	10
stdA9	6	11	stdC3	7	13

For every instance, we report the number of deleted outputs ($j \in J$) and the number of deleted constraints of the type (1d) or (1e).

Using the above preprocessing technique, we are able to reduce the sizes of some of the test instances from Alfaki and Haugland (2013); see Table EC.1. Our random instance generator is designed in such a way that the sizes of the random instances cannot be further reduced by the above preprocessing technique.

Appendix D: Detailed computational results

The detailed outputs from our experiments of Section 5 are provided in Tables EC.2 - EC.4. For each instance, we report the best feasible solution value (BestFeas) at termination of the different methods and the % gap with respect to upper bound (BestRelax) from `BARON` (cf. (23)). For the MILP approximations, we also report in parenthesis either the % termination gap of feasible solution with respect to dual bound of `Cplex` after 1hr or the solution time in seconds if `Cplex` termination gap was less than our threshold of 0.01%.

References

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

Tawarmalani M, Sahinidis N (2002) *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*, chap. 9: The Pooling Problem (Kluwer Academic Publishers).

Table EC.2 Best feasible solutions at termination for BARON, SNOPT, AltLP, SLP and FlowAug.

#		BARON		SNOPT		AltLP		SLP		FlowAug	
	BestRelax	BestFeas	% gap	BestFeas	% gap	BestFeas	% gap	BestFeas	% gap	BestFeas	% gap
stdA0	36967.37	35812.33	3.12	35137.80	2.47	30781.18	15	33291.59	7.60	32947.22	8.55
stdA1	30470.29	29276.56	3.92	29207.42	2.95	16364.99	46	23974.72	20	22696.24	25
stdA2	23046.46	23044.16	0.01	19627.77	15	14608.02	37	20731.62	10	19317.88	16
stdA3	40507.89	39301.98	2.98	37532.87	5.42	36612.12	7.74	36302.87	8.52	32423.06	18
stdA4	42727.63	41257.14	3.44	39219.64	8.21	36689.01	14	39443.92	7.69	36130.45	15
stdA5	28257.75	27901.14	1.26	27288.91	3.43	23399.90	17	25091.27	11	23466.98	17
stdA6	42463.05	42176.94	0.67	42079.94	0.90	41234.16	2.89	41925.01	1.27	41566.46	2.11
stdA7	44682.25	44352.79	0.74	44357.37	0.73	40056.74	10	36719.41	18	41187.37	7.82
stdA8	30666.87	30569.03	0.32	30396.84	0.88	27164.19	11	30061.61	1.97	28431.88	7.29
stdA9	21933.99	21933.33	0.00	21671.47	1.20	21538.91	1.80	20898.01	4.72	21247.18	3.13
stdB0	45457.05	42693.4	6.08	42427.66	6.66	34879.19	23	37527.71	17	37054.06	18
stdB1	65467.99	63377.41	3.19	61212.44	6.50	55645.62	15	58129.17	11	56381.25	14
stdB2	56452.21	54228.13	3.94	48428.91	14	50573.65	10	51978.37	7.93	49594.48	12
stdB3	74050.47	0	100	58591.57	21	72102.53	2.63	68880.37	6.98	70211.56	5.18
stdB4	59469.66	59410.05	0.10	58893.49	0.97	58552.82	1.54	59260.75	0.35	56520.41	4.96
stdB5	60696.36	60319.69	0.62	60037.14	1.09	59027.95	2.75	59561.18	1.87	58942.86	2.89
stdC0	98290.76	3971.64	96	70008.32	29	66569.63	32	83853.83	15	69431.52	29
stdC1	119569.06	16401.81	86	53423.71	55	88589.97	26	94313.96	21	71995.17	40
stdC2	136102.76	0	100	17549.43	87	107763.04	21	114423.52	16	93738.48	31
stdC3	130315.02	0	100	82743.17	37	103997.43	20	108574.91	17	106645.56	18
randstd11	70406.04	61579.76	13	52589.47	25	44763.55	36	54069.07	23	43323.84	38
randstd12	57850.31	57513.39	0.58	53711.45	7.15	40713.28	30	40713.28	30	44538.2	23
randstd13	73140.43	71998.69	1.56	57630.01	21	37021.00	49	37021.00	49	37021	49
randstd14	77577.49	77402.09	0.23	70319.62	9.36	60846.00	22	71468.09	7.88	70239.94	9.46
randstd15	95223.38	94679.27	0.57	75497.99	21	61087.97	36	79338.33	17	74216.35	22
randstd16	65078.08	64998.27	0.12	64410.85	1.03	52933.37	19	52933.37	19	55848.56	14
randstd17	65689.93	64905.8	1.19	64167.92	2.32	50966.61	22	50966.61	22	52131.62	21
randstd18	59274.44	58811.76	0.78	46855.55	21	46976.21	21	53139.54	10	40896.62	31
randstd19	83659.75	83195.13	0.56	74579.17	11	38689.48	54	60553.34	28	53994.18	35
randstd20	68836.6	68751.62	0.12	62967.56	8.53	44254.23	36	40168.05	42	44263.02	36
randstd21	89682.96	86150.67	3.94	85265.04	4.93	73972.92	18	84547.95	5.73	63267.19	29
randstd22	67335.44	67328.7	0.01	65022.22	3.44	64458.59	4.27	64458.59	4.27	64458.59	4.27
randstd23	94186.37	92644.35	1.64	82150.30	13	80174.10	15	80174.10	15	70731.39	25
randstd24	74020.43	73932.76	0.12	69688.74	5.85	57207.26	23	52165.04	30	62255.83	16
randstd25	75419.74	74756.56	0.88	66275.10	12	54553.20	28	47749.01	37	49790.27	34
randstd26	87949.16	87949.15	0.00	84309.53	4.14	74143.11	16	81780.42	7.01	68815.3	22
randstd27	56406.56	52063.59	7.70	53758.78	4.69	50777.00	9.98	51758.55	8.24	42563.43	25
randstd28	96159.65	76367.98	21	96082.49	0.08	87287.73	9.23	85148.23	11	74249.69	23
randstd29	82991.4	81911.52	1.30	81850.26	1.38	51231.57	38	79307.10	4.44	66907.72	19
randstd30	81110.45	60546.45	25	64401.45	21	59446.97	27	59446.97	27	76462.61	5.73
randstd31	104751.06	102383.79	2.26	89613.90	14	81070.10	23	98580.36	5.89	85714.79	18
randstd32	98374.7	0	100	85863.63	13	79549.40	19	86191.74	12	84199.04	14
randstd33	79676.37	77298.16	2.98	77403.37	2.85	72338.78	9.21	73019.77	8.35	66267.29	17
randstd34	90621.44	71264.55	21	88506.19	2.33	73839.18	19	75573.10	17	77769.09	14
randstd35	73655.67	72530.66	1.53	72154.58	2.04	59712.12	19	57071.80	23	55314.15	25
randstd36	98416.54	98156.08	0.26	98071.84	0.35	92801.00	5.71	87356.84	11	90835	7.70
randstd37	94087.5	92110.42	2.10	87606.48	6.89	69608.49	26	74905.37	20	75887.93	19
randstd38	111367.18	101667.06	8.71	92928.58	17	66694.90	40	66694.90	40	88320.4	21
randstd39	81651	50282.23	38	69532.14	15	55475.29	32	57214.81	30	52567.37	36
randstd40	124416	0	100	108147.67	13	90576.47	27	99057.68	20	96638.31	22
randstd41	89315.91	66736.84	25	76238.47	15	69799.85	22	65198.80	27	70604.44	21
randstd42	99160	0	100	77340.35	22	71581.63	28	70349.24	29	64839.99	35
randstd43	108014	0	100	84750.76	22	74153.79	31	102177.30	5.40	50824.09	53
randstd44	117167.43	70651	40	109848.05	6.25	82055.72	30	103493.05	12	97175.27	17
randstd45	101281	0	100	91763.78	9.40	64957.98	36	78630.27	22	64933.12	36
randstd46	113212.55	51154.54	55	88232.55	22	66066.89	42	83516.23	26	78281.43	31
randstd47	108612	0	100	70701.45	35	55152.41	49	78530.45	28	71517.04	34
randstd48	115470	0	100	97588.47	15	72045.48	38	74892.20	35	56381.6	51
randstd49	102147	0	100	61679.53	40	24716.46	76	51746.03	49	60372.46	41
randstd50	143113	0	100	117015.56	18	82042.64	43	100575.95	30	105761.08	26
randstd51	137423	0	100	103650.99	25	97237.07	29	129437.26	5.81	119777.21	13
randstd52	108653	0	100	102732.12	5.45	93723.72	14	103962.93	4.32	100891.17	7.14
randstd53	110433.47	79793.55	28	102656.30	7.04	87300.33	21	89285.00	19	88689.60	20
randstd54	88151.77	4920	94	86422.97	1.96	62468.42	29	71946.70	18	69103.22	22
randstd55	61516.27	55681.99	9.48	57525.40	6.49	42132.00	32	35913.30	42	37778.91	39
randstd56	131087.35	123789.56	5.57	104964.07	20	101870.34	22	106045.17	19	104792.72	20
randstd57	105147	0	100	56201.85	47	62377.11	41	87650.23	17	80068.29	24
randstd58	106591.6	0	100	101375.68	4.89	77802.27	27	78485.74	26	78280.70	27
randstd59	159035.34	104581	34	140022.56	12	121474.82	24	115808.59	27	117508.46	26
randstd60	112463	0	100	99999.53	11	90227.25	20	85105.00	24	86641.68	23

Table EC.3 Best feasible solutions at termination for $\mathbb{U}(\cdot)$. Time (sec.) or % termination gap in paranthesis.

#	$\mathbb{U}(1)$		$\mathbb{U}(2)$		$\mathbb{U}(3)$		$\mathbb{U}(4)$		$\mathbb{U}(5)$	
	BestFeas	% gap	BestFeas	% gap	BestFeas	% gap	BestFeas	% gap	BestFeas	% gap
stdA0	33906.94 (1sec)	5.89	34988.17 (40sec)	2.89	35436.59 (203sec)	1.64	35516.68 (0.15)	1.42	35551.68 (1.23)	1.32
stdA1	28515.22 (1sec)	5.25	28598.72 (2sec)	4.97	28661.31 (18sec)	4.76	28677.01 (86sec)	4.71	28721.45 (144sec)	4.56
stdA2	22721.6 (1sec)	1.40	22721.6 (2sec)	1.40	22721.6 (5sec)	1.40	22729.68 (59sec)	1.36	22740.95 (70sec)	1.32
stdA3	37523 (1sec)	5.44	38525.55 (16sec)	2.91	39095.25 (38sec)	1.48	38919.79 (69sec)	1.92	38993.97 (278sec)	1.73
stdA4	39376.44 (6sec)	7.84	40087.95 (312sec)	6.18	40549.31 (1.36)	5.10	40584.58 (3.05)	5.02	40790.56 (3.67)	4.53
stdA5	26766.96 (13sec)	5.28	27059.73 (2.8)	4.24	27113.01 (3.68)	4.05	26847.33 (5.15)	4.99	26944.73 (4.81)	4.65
stdA6	41555.84 (60sec)	2.14	41888.22 (0.52)	1.35	41973.12 (0.87)	1.15	41979.64 (0.97)	1.14	41979.86 (1)	1.14
stdA7	43446.34 (32sec)	2.77	44074.18 (0.09)	1.36	44261.69 (0.78)	0.94	44287.72 (0.88)	0.88	44335.47 (0.78)	0.78
stdA8	30460.45 (46sec)	0.67	30482.21 (0.45)	0.60	30489.47 (0.55)	0.58	30504.35 (0.53)	0.53	30502.69 (0.54)	0.54
stdA9	21794.46 (13sec)	0.64	21854.86 (0.36)	0.36	21879.31 (0.25)	0.25	21864.87 (0.32)	0.32	21861.63 (0.33)	0.33
stdB0	41776.96 (32sec)	8.10	42166.06 (3.33)	7.24	42659.63 (4.18)	6.15	42812.55 (4.62)	5.82	42801.71 (5.22)	5.84
stdB1	62482.74 (16sec)	4.56	62747.62 (0.7)	4.16	62861.57 (1.61)	3.98	62906.32 (2.45)	3.91	63009.41 (2.96)	3.76
stdB2	53263.93 (1.25)	5.65	52913.47 (4.5)	6.27	53104.75 (4.72)	5.93	53290.67 (4.7)	5.60	53337.46 (4.52)	5.52
stdB3	73778.68 (0.33)	0.37	73723.39 (0.44)	0.44	73833.89 (0.29)	0.29	73638.63 (0.56)	0.56	73008.83 (1.43)	1.41
stdB4	59445.31 (0.04)	0.04	59259.6 (0.35)	0.35	58948.33 (0.88)	0.88	58920.93 (0.93)	0.92	58866.69 (1.02)	1.01
stdB5	60488.2 (0.34)	0.34	60358.09 (0.56)	0.56	60217.61 (0.8)	0.79	59810.29 (1.48)	1.46	34884.58 (74)	43
stdC0	80206.61 (8.9)	18	82091.03 (12)	16	82269.87 (13)	16	80805.63 (17)	18	80933.69 (17)	18
stdC1	100935.17 (10)	16	98880 (16)	17	97625.83 (19)	18	95075.43 (22)	20	87324.41 (34)	27
stdC2	119528.78 (10)	12	115661.45 (16)	15	116408.77 (15)	14	109724.3 (23)	19	106005.81 (27)	22
stdC3	124531.25 (4.57)	4.44	118687.78 (9.79)	8.92	120119.49 (8.48)	7.82	105740.96 (23)	19	73412.32 (78)	44
randstd11	61630.25 (100sec)	12	63439.2 (1.31)	9.90	62274.87 (14)	12	58544.95 (22)	17	59477.79 (20)	16
randstd12	53406.97 (15sec)	7.68	54642.59 (0.55)	5.54	55472.92 (4.12)	4.11	55343.83 (4.51)	4.33	55891.28 (3.37)	3.39
randstd13	70039.57 (7sec)	4.24	70346.91 (1145)	3.82	70370.52 (2.13)	3.79	70443.51 (3.18)	3.69	70232.72 (3.47)	3.98
randstd14	76759.2 (23sec)	1.05	77028.33 (0.4)	0.71	77205.97 (0.45)	0.48	77181.22 (0.6)	0.51	77327.16 (0.26)	0.32
randstd15	91967.15 (72sec)	3.42	93565.25 (0.99)	1.74	94040.97 (1.16)	1.24	93671.98 (1.66)	1.63	93374.7 (1.86)	1.94
randstd16	63278.96 (4sec)	2.76	63971.81 (273sec)	1.70	63959.16 (0.55)	1.72	64203.53 (0.19)	1.34	64138.23 (0.34)	1.44
randstd17	64135.33 (1033sec)	2.37	64980.18 (0.13)	1.08	65049.61 (0.15)	0.97	64493.99 (1.91)	1.82	64148.92 (2.7)	2.35
randstd18	58351.94 (120sec)	1.56	58691.39 (0.88)	0.98	58699.9 (1.02)	0.97	58553.92 (1.27)	1.22	58533.33 (1.31)	1.25
randstd19	75738.61 (249sec)	9.47	82085.74 (1.4)	1.88	79401.48 (5.15)	5.09	81349.85 (3.11)	2.76	79372.19 (5.09)	5.12
randstd20	67735.53 (1sec)	1.60	68063.29 (15sec)	1.12	68152.36 (0.11)	0.99	68112.36 (0.22)	1.05	68195.07 (0.11)	0.93
randstd21	85273.46 (26sec)	4.92	85265.97 (5.24)	4.93	84895.27 (6.37)	5.34	83580.82 (8.36)	6.80	83298.87 (8.64)	7.12
randstd22	66195.24 (10sec)	1.69	66216.78 (0.08)	1.66	66208.62 (1.69)	1.67	66298.88 (1.7)	1.54	66317.6 (1.53)	1.51
randstd23	90399.84 (10sec)	4.02	92238.84 (937sec)	2.07	86750.95 (10)	7.89	73677.06 (30)	22	73263 (31)	22
randstd24	72985.55 (1434sec)	1.40	73626.53 (0.47)	0.53	73516.68 (0.69)	0.68	73622.72 (0.53)	0.54	73591.34 (0.59)	0.58
randstd25	71251.87 (0.34)	5.53	73070.26 (1.22)	3.12	72495.3 (3.36)	3.88	71739.38 (4.32)	4.88	49860.9 (50)	34
randstd26	87675.53 (0.04)	0.31	87862.33 (751sec)	0.10	87913.66 (0.04)	0.04	87864.08 (0.1)	0.10	87528.41 (0.48)	0.48
randstd27	52719.64 (35sec)	6.54	53824.69 (1471sec)	4.58	54677.63 (3.16)	3.07	55004.17 (2.57)	2.49	9724.67 (480)	83
randstd28	94495.07 (0.25)	1.73	95304.89 (0.28)	0.89	95050.66 (0.56)	1.15	95242.52 (0.72)	0.95	93559.57 (2.17)	2.70
randstd29	80042.87 (284sec)	3.55	80421.44 (0.02)	3.10	80375.17 (2.51)	3.15	80219.6 (3.45)	3.34	79801.73 (3.85)	3.84
randstd30	77757.4 (30sec)	4.13	78505.72 (3.29)	3.21	78900.35 (2.65)	2.72	65717.43 (23)	19	79006.56 (2.56)	2.59
randstd31	102960 (69sec)	1.71	103189.3 (1699sec)	1.49	103124.09 (1.19)	1.55	103015.92 (1.46)	1.66	101398 (3.23)	3.20
randstd32	96761.13 (0.44)	1.64	94473.86 (3.81)	3.97	80995.75 (21)	18	95748.41 (2.71)	2.67	48713.08 (102)	50
randstd33	77103.68 (2367sec)	3.23	76003.18 (4.2)	4.61	71076.8 (12)	11	75364.06 (5.71)	5.41	73628.64 (8.2)	7.59
randstd34	87692.94 (102sec)	3.23	88454.26 (1.58)	2.39	88598.46 (1.82)	2.23	88660.26 (1.39)	2.16	85383.88 (5.97)	5.78
randstd35	71358.65 (69sec)	3.12	72092.32 (0.07)	2.12	72301.95 (1.4)	1.84	72306.02 (1.83)	1.83	63277.47 (16)	14
randstd36	95860.36 (25sec)	2.60	95926.81 (2093sec)	2.53	95982.13 (2.51)	2.47	95713.27 (2.81)	2.75	96076.18 (2.44)	2.38
randstd37	88889.33 (830sec)	5.52	90475.24 (2.7)	3.84	89903.27 (3.6)	4.45	78196.47 (20)	17	23294.23 (301)	75
randstd38	109775.86 (1472sec)	1.43	103852.83 (7.18)	6.75	92538.73 (20)	17	86314.04 (29)	22	37020.52 (201)	67
randstd39	79713.51 (1.09)	2.37	78971.18 (3.25)	3.28	70285.95 (16)	14	49404.68 (65)	39	28291.16 (189)	65
randstd40	118575.91 (1.12)	4.69	114287.01 (8.69)	8.14	117757.99 (5.28)	5.35	33626.2 (269)	73	33626.2 (269)	73
randstd41	83015.31 (895sec)	7.05	40168 (122)	55	36679.63 (143)	59	41650.73 (114)	53	34555.15 (158)	61
randstd42	91139.16 (346sec)	8.09	76958.33 (28)	22	78670.2 (26)	21	42504.94 (133)	57	29078.51 (241)	71
randstd43	103570.98 (1.27)	4.11	74430.89 (44)	31	16761.79 (541)	84	16726.68 (543)	85	13426.94 (701)	88
randstd44	115910.95 (3251)	1.07	105591.1 (11)	9.88	39457.35 (197)	66	88297.69 (33)	25	39461.27 (197)	66
randstd45	99740.31 (0.61)	1.52	95250.9 (6.3)	5.95	24454.4 (314)	76	30328.62 (234)	70	25441.05 (298)	75
randstd46	99678.51 (0.25)	12	83483.22 (33)	26	24121.64 (364)	79	22186.75 (408)	80	17287.95 (554)	85
randstd47	102566.1 (0.18)	5.57	74050.11 (46)	32	18300.29 (492)	83	78415.97 (38)	28	23503.39 (362)	78
randstd48	102780.09 (108sec)	11	85819.16 (34)	26	77424.63 (48)	33	37099.31 (210)	68	40127.36 (187)	65
randstd49	91569.63 (120sec)	10	76602.09 (32)	25	21150.11 (380)	79	21849.31 (367)	79	12062.32 (746)	88
randstd50	135784.18 (0.65)	5.12	131023.45 (8.76)	8.45	35230.8 (305)	75	39591.23 (261)	72	36776.97 (289)	74
randstd51	106198.79 (249sec)	23	126741.65 (8.36)	7.77	55208.65 (149)	60	55208.65 (149)	60	55208.65 (149)	60
randstd52	105107.4 (1.7)	3.26	98076.53 (11)	9.73	89518.29 (21)	18	51646.23 (110)	52	47941.07 (127)	56
randstd53	108766.04 (0.04)	1.51	93536.52 (18)	15	103036.22 (7.12)	6.70	40218.59 (175)	64	44872.05 (146)	59
randstd54	87097.3 (45sec)	1.20	87097.75 (0.54)	1.20	26376.57 (233)	70	83830.6 (5.12)	4.90	24731.65 (256)	72
randstd55	60906.33 (0.08)	0.99	58711.85 (4.62)	4.56	16112.69 (281)	74	18372.33 (234)	70	20611.21 (198)	66
randstd56	129640.22 (59sec)	1.10	52760.98 (148)	60	129737.99 (0.88)	1.03	54172.45 (142)	59	53635.74 (144)	59
randstd57	102633.71 (0.13)	2.39	91356.4 (15)	13	85457.18 (22)	19	71183.64 (46)	32	28235.91 (271)	73
randstd58	105688.16 (0.12)	0.85	94769.53 (13)	11	42259.06 (154)	60	84663.45 (27)	21	32521.12 (230)	69
randstd59	156920.01 (3596sec)	1.33	147654.31 (7.69)	7.16	144777.7 (9.84)	8.97	55347.42 (187)	65	53038.91 (200)	67
randstd60	110492.28 (1.03)	1.75	104301.35 (7.8)	7.26	98124.46 (15)	13	14532 (674)	87	53769.2 (109)	52

Table EC.4 Best feasible solutions at termination for $\mathbb{A}(\cdot)$. Time (sec.) or % termination gap in paranthesis.

#	$\mathbb{A}(3)$		$\mathbb{A}(4)$		$\mathbb{A}(5)$	
	BestFeas	% gap	BestFeas	% gap	BestFeas	% gap
stdA0	35516.68 (278sec)	1.42	35626.3 (2524sec)	1.12	35657.63 (0.45)	1.03
stdA1	28677.01 (14sec)	4.71	28868.12 (36sec)	4.08	28868.12 (71sec)	4.08
stdA2	22729.68 (7sec)	1.36	22760.75 (19sec)	1.23	22778.54 (61sec)	1.15
stdA3	38919.79 (105sec)	1.92	39077.87 (839sec)	1.52	39128.11 (1368sec)	1.40
stdA4	40584.58 (1.89)	5.02	40878.77 (3.08)	4.33	40918.93 (3.26)	4.23
stdA5	27140.82 (3.86)	3.95	27368.53 (3.06)	3.15	27003.48 (4.54)	4.44
stdA6	41979.64 (0.82)	1.14	41968.65 (0.86)	1.16	41993.76 (0.83)	1.11
stdA7	44287.72 (0.77)	0.88	43991.39 (1.53)	1.55	44101.75 (1.3)	1.30
stdA8	30502.29 (0.52)	0.54	30506.67 (0.53)	0.52	30518.72 (0.49)	0.48
stdA9	21893.63 (0.18)	0.18	21900.82 (0.15)	0.15	21883.86 (0.23)	0.23
stdB0	42849.18 (3.55)	5.74	42886.81 (3.89)	5.65	42751.43 (4.73)	5.95
stdB1	62929.04 (1.86)	3.88	63000.95 (2.31)	3.77	63029.61 (2.44)	3.72
stdB2	53359.9 (4.21)	5.48	52296.56 (6.85)	7.36	52779.07 (5.95)	6.51
stdB3	73634.53 (0.56)	0.56	73200.42 (1.16)	1.15	73334.81 (0.98)	0.97
stdB4	59065.84 (0.68)	0.68	59014.9 (0.77)	0.76	58840.37 (1.07)	1.06
stdB5	60016.32 (1.13)	1.12	58717.22 (3.37)	3.26	60139.9 (0.93)	0.92
stdC0	82789.83 (14)	16	82384.58 (15)	16	79833.6 (19)	19
stdC1	94325.5 (23)	21	85513.77 (36)	28	87890.52 (33)	26
stdC2	111011.86 (21)	18	108475.33 (24)	20	28728.32 (368)	79
stdC3	116701.51 (12)	10	116866.16 (12)	10	73156.21 (78)	44
randstd11	63877.75 (11)	9.27	48836.94 (46)	31	53770.95 (32)	24
randstd12	55671.64 (3.51)	3.77	51731.89 (12)	11	56254.02 (2.78)	2.76
randstd13	70443.51 (1.29)	3.69	65843.07 (10)	9.98	70779.36 (3.11)	3.23
randstd14	77265.2 (0.47)	0.40	76886.63 (0.98)	0.89	77232.35 (0.54)	0.44
randstd15	94150.6 (0.82)	1.13	90316.76 (5.43)	5.15	86764.35 (9.75)	8.88
randstd16	64200.15 (0.07)	1.35	64068.31 (1.69)	1.55	64238.78 (0.12)	1.29
randstd17	65040.66 (0.29)	0.99	64290.25 (2.17)	2.13	64551.23 (1.75)	1.73
randstd18	58795.12 (0.78)	0.81	58377.41 (1.57)	1.51	58546.83 (1.28)	1.23
randstd19	81679.64 (2)	2.37	81844.34 (2.05)	2.17	81661.73 (2.27)	2.39
randstd20	68112.36 (0.02)	1.05	68231.53 (0.04)	0.88	68230.52 (0.06)	0.88
randstd21	85133.17 (5.88)	5.07	84195.67 (7.25)	6.12	85703.75 (5.48)	4.44
randstd22	66338.29 (1.49)	1.48	66357.02 (1.62)	1.45	66321.93 (1.52)	1.51
randstd23	92574.27 (2.58)	1.71	86130.9 (11)	8.55	86232.91 (11)	8.44
randstd24	73402.29 (0.81)	0.84	73470.68 (0.72)	0.74	73266.54 (1.03)	1.02
randstd25	73129.08 (1.81)	3.04	72978.8 (2.43)	3.24	46406.66 (61)	38
randstd26	87880.32 (0.07)	0.08	87943.94 (1543)	0.01	86367.4 (1.83)	1.80
randstd27	55213.2 (2.16)	2.12	55439.54 (1.74)	1.71	51347.46 (10)	8.97
randstd28	95070.26 (0.64)	1.13	94946.19 (0.84)	1.26	81851.77 (17)	15
randstd29	80676.99 (2.62)	2.79	80393.59 (3.22)	3.13	72948.85 (14)	12
randstd30	72160.29 (12)	11	77823.35 (4.13)	4.05	67656.26 (20)	17
randstd31	103383.99 (0.45)	1.31	103505.6 (1.08)	1.19	103440.91 (1.18)	1.25
randstd32	96244.37 (1.92)	2.17	96552.98 (1.86)	1.85	94405.22 (4.17)	4.04
randstd33	71094.14 (12)	11	76542.98 (4.09)	3.93	73321.38 (8.66)	7.98
randstd34	86272.09 (4.35)	4.80	86094.71 (4.56)	5.00	89011.8 (1.42)	1.78
randstd35	72388.26 (0.61)	1.72	72210.73 (1.95)	1.96	72052.6 (2.16)	2.18
randstd36	96048.8 (2.46)	2.41	93643.09 (5.1)	4.85	95203.14 (3.38)	3.27
randstd37	89304.56 (4.37)	5.08	85996.89 (8.39)	8.60	89173.91 (4.54)	5.22
randstd38	98192.23 (13)	12	708 (15630)	99	82651.79 (35)	26
randstd39	71890.93 (13)	12	69331.68 (18)	15	28931.2 (182)	65
randstd40	115185.55 (7.48)	7.42	101139.08 (23)	19	33866.09 (266)	73
randstd41	36679.63 (143)	59	35571.7 (151)	60	35571.7 (151)	60
randstd42	43386.41 (129)	56	37034.2 (168)	63	37034.2 (168)	63
randstd43	15978.25 (572)	85	15085.46 (612)	86	15085.46 (613)	86
randstd44	100654 (16)	14	93418.66 (25)	20	101338.96 (16)	14
randstd45	26481.85 (282)	74	26205.25 (286)	74	26205.25 (286)	74
randstd46	17035.95 (558)	85	17035.95 (561)	85	17035.95 (561)	85
randstd47	57122.2 (90)	47	9410.63 (1052)	91	9410.63 (-)	91
randstd48	36262.31 (216)	69	36363.86 (217)	69	36363.86 (217)	69
randstd49	47113.05 (115)	54	18987.43 (437)	81	18987.43 (438)	81
randstd50	105943.74 (35)	26	35229.53 (305)	75	35229.53 (306)	75
randstd51	57653.97 (138)	58	54784.17 (151)	60	54784.17 (151)	60
randstd52	85234.61 (27)	22	49557.32 (-)	54	49557.32 (-)	54
randstd53	103180.3 (7.03)	6.57	32089.15 (244)	71	32089.15 (244)	71
randstd54	82629.68 (6.43)	6.26	61926.85 (42)	30	19921.23 (342)	77
randstd55	15512.69 (296)	75	15512.69 (296)	75	15512.69 (296)	75
randstd56	113432.84 (15)	13	52760.98 (148)	60	52760.98 (-)	60
randstd57	26924.93 (289)	74	25920.3 (305)	75	25920.3 (305)	75
randstd58	29400.06 (265)	72	90068.28 (19)	16	98098.35 (9.34)	7.97
randstd59	139903.6 (14)	12	54681.13 (-)	66	153061.89 (3.9)	3.76
randstd60	14532 (674)	87	14532 (674)	87	14532 (698sec)	87