Analysis of MILP techniques for the Pooling Problem

Santanu S. Dey
Georgia Institute of Technology

Akshay Gupte
Clemson University

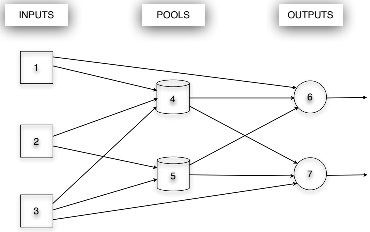
1 2

[1] Support from EXXONMOBIL Research and Engineering is gratefully acknowledged.
[2] Many thanks to Akshay Gupte for helping prepare the Figures.

1

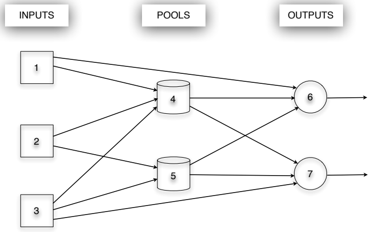
The Pooling Problem: Model, Relaxations and Restrictions

The Pooling Problem: Network Flow on Tripartite Graph



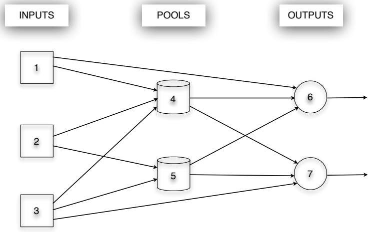
- Network flow problem on a tripartite directed graph, with three type of node: *Input* Nodes (I), *Pool* Nodes (L), *Output* Nodes (J).

The Pooling Problem: Network Flow on Tripartite Graph



- Network flow problem on a tripartite directed graph, with three type of node: *Input* Nodes (I), *Pool* Nodes (L), *Output* Nodes (J).
- Send flow from input nodes via pool nodes to output nodes.

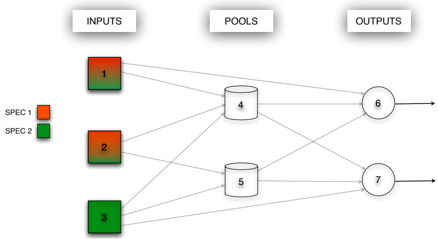
The Pooling Problem: Network Flow on Tripartite Graph



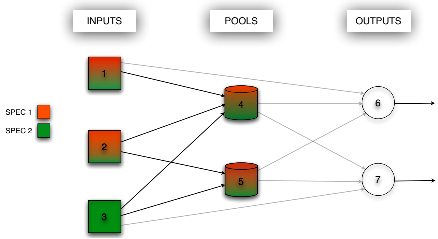
- Network flow problem on a tripartite directed graph, with three type of node: *Input* Nodes (I), *Pool* Nodes (L), *Output* Nodes (J).
- Send flow from input nodes via pool nodes to output nodes.
- Each of the arcs and nodes have capacities of flow.

The Pooling Problem: Other Constraints

- Raw material has specifications (like sulphur, carbon, etc.).

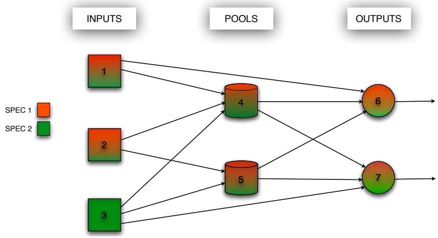


The Pooling Problem: Other Constraints



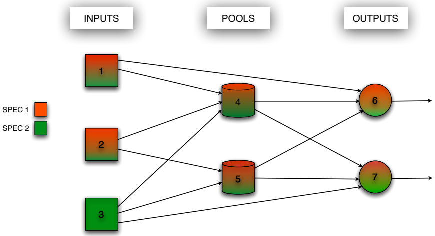
- Raw material has specifications (like sulphur, carbon, etc.).
- Raw material gets mixed at the pool producing new specification level at pools.

The Pooling Problem: Other Constraints



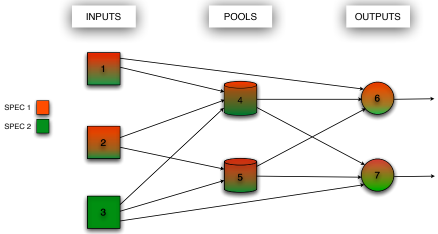
- Raw material has specifications (like sulphur, carbon, etc.).
- Raw material gets mixed at the pool producing new specification level at pools.
- The material gets further mixed at the output nodes.

The Pooling Problem: Other Constraints



- Raw material has specifications (like sulphur, carbon, etc.).
- Raw material gets mixed at the pool producing new specification level at pools.
- The material gets further mixed at the output nodes.
- The output node has required levels for each specification.

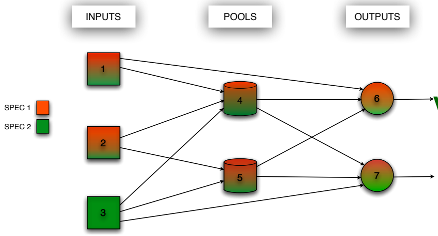
Tracking Specification



Data:

- λ_i^k: The value of specification k at input node i.

Tracking Specification



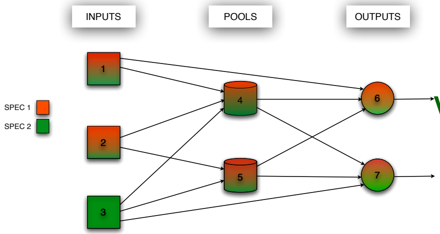
Data:

- λ_i^k: The value of specification k at input node i.

Variable:

- p_l^k: The value of specification k at node l
- y_{ab}: Flow along the arc (ab).

Tracking Specification



Data:

- λ_i^k: The value of specification k at input node i.

Variable:

- p_l^k: The value of specification k at node l
- y_{ab}: Flow along the arc (ab).

Specification Tracking: $$\underbrace{\sum_{i \in I} \lambda_i^k y_{il}}_{\text{Inflow of Spec k}} = \underbrace{p_l^k \left(\sum_{j \in J} y_{lj} \right)}_{\text{Out flow of Spec k}}$$

The pooling problem: ‘P’ formulation

$$\max \quad \sum_{ij \in \mathcal{A}} w_{ij} y_{ij} \quad \text{(Maximize profit due to flow)}$$

The pooling problem: ‘P’ formulation

$$\max \quad \sum_{ij \in \mathcal{A}} w_{ij} y_{ij} \quad \text{(Maximize profit due to flow)}$$

Subject To:

1. Node and arc capacities.
2. Total flow balance at each node.
3. Specification balance at each pool.

$$\sum_{i \in I} \lambda_i^k y_{il} = p_l^k \left(\sum_{j \in J} y_{lj} \right)$$

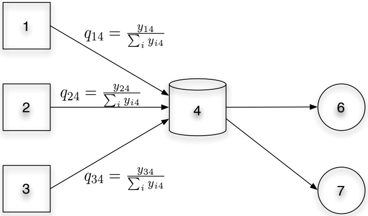
4. Bounds on p_j^k for all out put nodes j and specification k.

PQ Model

New Variable:

- q_{il} : fraction of flow to l from $i \in I$

$$\sum_{i \in I} q_{il} = 1, q_{il} \geq 0, i \in I.$$



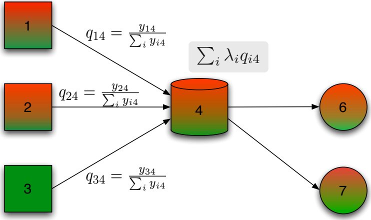
PQ Model

New Variable:

- q_{il} : fraction of flow to l from $i \in I$

$$\sum_{i \in I} q_{il} = 1, q_{il} \geq 0, i \in I.$$

- $p_l^k = \sum_{i \in I} \lambda_i^k q_{il}$



PQ Model

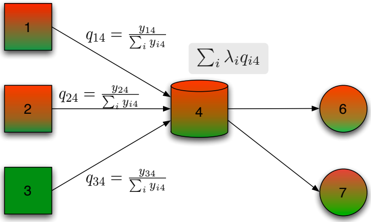
New Variable:

- q_{il} : fraction of flow to l from $i \in I$

$$\sum_{i \in I} q_{il} = 1, q_{il} \geq 0, i \in I.$$

  - $p_l^k = \sum_{i \in I} \lambda_i^k q_{il}$

- v_{ilj} : flow from input node i to output node j via pool node l.



PQ Model

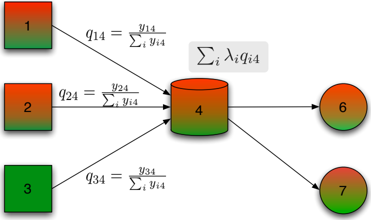
New Variable:

- q_{il} : fraction of flow to l from $i \in I$

  $$\sum_{i \in I} q_{il} = 1, q_{il} \geq 0, i \in I.$$

  - $p_l^k = \sum_{i \in I} \lambda_i^k q_{il}$

- v_{ilj} : flow from input node i to output node j via pool node l.
  - $v_{ilj} = q_{il} y_{lj}$



PQ Model Complete

$$\max \quad \sum_{i\in I, j\in J} w_{ij} y_{ij} + \sum_{i\in I, l\in L, j\in J} (w_{il} + w_{lj}) v_{ilj}$$

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

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

$$a_j^k \left(\sum_{i\in I} y_{ij} + \sum_{l\in L} y_{lj} \right) \le \sum_{i\in I} \lambda_i^k y_{ij} + \sum_{i\in I, l\in L} \lambda_i^k v_{ilj} \le b_j^k \left(\sum_{i\in I} y_{ij} + \sum_{l\in L} y_{lj} \right)$$

Capacity constraints

All variables are non-negative

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

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

What do we know?

1. The Pooling problem was formally introduced by Haverly (1978).

What do we know?

1. The Pooling problem was formally introduced by Haverly (1978).
2. The model is a Bilinear Model, this is a special case of Indefinite quadratic program.
3. Recently, Alfaki and Haugland (2012) formally proved that the pooling problem is NP-hard.

What do we know?

1. The Pooling problem was formally introduced by Haverly (1978).
2. The model is a Bilinear Model, this is a special case of Indefinite quadratic program.
3. Recently, Alfaki and Haugland (2012) formally proved that the pooling problem is NP-hard.
4. Numerous papers over the years have studied this problem.
5. This problem continues to remain *challenging* to solve to this day.

What do we know?

1. The Pooling problem was formally introduced by Haverly (1978).
2. The model is a Bilinear Model, this is a special case of Indefinite quadratic program.
3. Recently, Alfaki and Haugland (2012) formally proved that the pooling problem is NP-hard.
4. Numerous papers over the years have studied this problem.
5. This problem continues to remain *challenging* to solve to this day.

I am an integer programmer, so I am going to talk about IP methods....

Using Integer Linear Programming to Construct Relaxation

1. $S := \{(x, y, w) \in \mathbb{R}^3 \mid w = xy,\ x_l \le x \le x_u,\ y_l \le y \le y_u\}$.

Using Integer Linear Programming to Construct Relaxation

1. $S := \{(x, y, w) \in \mathbb{R}^3 \mid w = xy,\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$.
2. Let $g(x, y) : [x_l, x_u] \times [y_l, y_u] \to \mathbb{R}$ be a piece-wise linear continuous function (pwl) such that $g(x, y) \geq xy$.
3. Let $h(x, y) : [x_l, x_u] \times [y_l, y_u] \to \mathbb{R}$ piece-wise linear continuous function such that $h(x, y) \leq xy$.

Using Integer Linear Programming to Construct Relaxation

1. $S := \{(x, y, w) \in \mathbb{R}^3 \mid w = xy,\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$.
2. Let $g(x, y) : [x_l, x_u] \times [y_l, y_u] \to \mathbb{R}$ be a piece-wise linear continuous function (pwl) such that $g(x, y) \geq xy$.
3. Let $h(x, y) : [x_l, x_u] \times [y_l, y_u] \to \mathbb{R}$ piece-wise linear continuous function such that $h(x, y) \leq xy$.
4. Then $S \subseteq \{(x, y, w) \in \mathbb{R}^3 \mid h(x, y) \leq w \leq g(x, y),\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$.

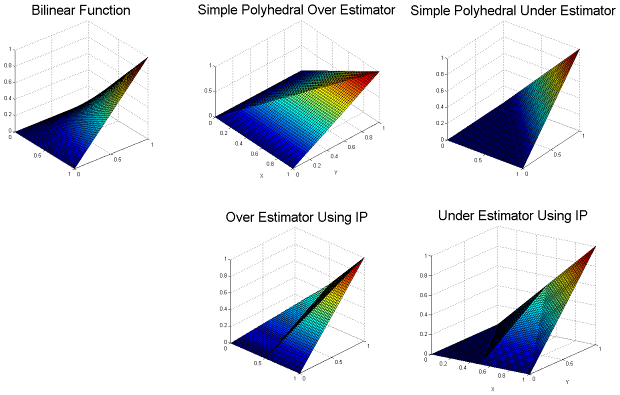
Using Integer Linear Programming to Construct Relaxation

1. $S := \{(x, y, w) \in \mathbb{R}^3 \,|\, w = xy,\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$.
2. Let $g(x, y) : [x_l, x_u] \times [y_l, y_u] \to \mathbb{R}$ be a piece-wise linear continuous function (pwl) such that $g(x, y) \geq xy$.
3. Let $h(x, y) : [x_l, x_u] \times [y_l, y_u] \to \mathbb{R}$ piece-wise linear continuous function such that $h(x, y) \leq xy$.
4. Then $S \subseteq \{(x, y, w) \in \mathbb{R}^3 \,|\, h(x, y) \leq w \leq g(x, y),\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$.
5. The set $\{(x, y, w) \in \mathbb{R}^3 \,|\, h(x, y) \leq w \leq g(x, y),\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$ is representable as a mixed integer liner set.

Using Integer Linear Programming to Construct Relaxation

1. $S := \{(x, y, w) \in \mathbb{R}^3 \mid w = xy,\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$.
2. Let $g(x, y) : [x_l, x_u] \times [y_l, y_u] \rightarrow \mathbb{R}$ be a piece-wise linear continuous function (pwl) such that $g(x, y) \geq xy$.
3. Let $h(x, y) : [x_l, x_u] \times [y_l, y_u] \rightarrow \mathbb{R}$ piece-wise linear continuous function such that $h(x, y) \leq xy$.
4. Then $S \subseteq \{(x, y, w) \in \mathbb{R}^3 \mid h(x, y) \leq w \leq g(x, y),\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$.
5. The set $\{(x, y, w) \in \mathbb{R}^3 \mid h(x, y) \leq w \leq g(x, y),\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}$ is representable as a mixed integer liner set.
6. **Gounaris et al. (2009), Misener and Floudas (2009)** used this for Pooling Problem very succesfully.

Using Integer Linear Programming to Construct Relaxation: Example



Using Mixed Integer Linear Programming to Construction Restriction

$$S := \{(x, y, w) \in \mathbb{R}^3 \mid w = xy,\ x_l \leq x \leq x_u,\ y_l \leq y \leq y_u\}.$$

1. Restrictions typically obtained by restricting a subset of variables, say y, to take values in a pre-determined finite set.

Using Mixed Integer Linear Programming to Construction Restriction

$$S := \{(x, y, w) \in \mathbb{R}^3 \,|\, w = xy,\; x_l \leq x \leq x_u,\; y_l \leq y \leq y_u\}.$$

1. Restrictions typically obtained by restricting a subset of variables, say y, to take values in a pre-determined finite set.
2. This restriction can be modeled using extra 0/1 variables and unary or binary expansion models.
3. **Pham et al. (2009), Alfaki et al. (2011), Gupte et al.(2012)** used for pooling problem with some success.

Using Mixed Integer Linear Programming to Construction Restriction: Example

- $w = xy, x \in [0, 1], y \in [0, 1]$

Using Mixed Integer Linear Programming to Construction Restriction: Example

- $w = xy, x \in [0, 1], y \in [0, 1]$
- Replace with $y \in \frac{1}{M}z,\ \ z \in \{0, \ldots, M\}$.

Using Mixed Integer Linear Programming to Construction Restriction: Example

- $w = xy, x \in [0,1], y \in [0,1]$
- Replace with $y \in \frac{1}{M}z,\ \ z \in \{0,\ldots,M\}$.
- Let $N = \lceil \log M \rceil$. Equivalently:

Using Mixed Integer Linear Programming to Construction Restriction: Example

- $w = xy, x \in [0,1], y \in [0,1]$
- Replace with $y \in \frac{1}{M}z,\ \ z \in \{0,\dots,M\}$.
- Let $N = \lceil \log M \rceil$. Equivalently:

$$\begin{aligned} x\left(\frac{1}{M}\sum_{i=1}^{N} 2^{i-1}u_i\right) &= w \\ \sum_{i=1}^{N} 2^{i-1}u_i &\le M \\ u_i \in \{0,1\}\ \forall i \in N \end{aligned}$$

Using Mixed Integer Linear Programming to Construction Restriction: Example

- $w = xy, x \in [0,1], y \in [0,1]$
- Replace with $y \in \frac{1}{M}z,\ \ z \in \{0,\ldots,M\}$.
- Let $N = \lceil \log M \rceil$. Equivalently:

$$\begin{array}{rcl} \frac{1}{M}\sum_{i=1}^{N} 2^{i-1} x u_i & = & w \\ \sum_{i=1}^{N} 2^{i-1} u_i & \leq & M \\ u_i \in \{0,1\}\ \forall i \in N & & \end{array}$$

Using Mixed Integer Linear Programming to Construction Restriction: Example

- $w = xy, x \in [0,1], y \in [0,1]$
- Replace with $y \in \frac{1}{M}z,\ \ z \in \{0, \ldots, M\}$.
- Let $N = \lceil \log M \rceil$. Equivalently:

$$\left.\begin{array}{rcl} \frac{1}{M}\sum_{i=1}^{N} 2^{i-1} v_i & = & w \\ \sum_{i=1}^{N} 2^{i-1} u_i & \leq & M \\ v_i \leq u_i, v_i \leq x, v_i & \geq & x + u_i - 1 \\ u_i \in \{0,1\}\ \forall i \in N & & \end{array}\right\}\text{MILP restriction}$$

2
Main Results

Main Results - Relaxations

Why do these MILP relaxations work so well?

Main Results - Relaxations

Why do these MILP relaxations work so well?

Theorem

Let n denote the number of output nodes.

Main Results - Relaxations

Why do these MILP relaxations work so well?

Theorem

Let n denote the number of output nodes. Let z^* denote the optimal solution of pooling problem.

Main Results - Relaxations

Why do these MILP relaxations work so well?

Theorem

Let n denote the number of output nodes. Let z^* denote the optimal solution of pooling problem.

1. *Bound: For any pwl MILP relaxation $\mathcal{P}$, let $z^{\mathcal{P}}$ be the optimal value of the MILP.*

Main Results - Relaxations

Why do these MILP relaxations work so well?

Theorem

Let n denote the number of output nodes. Let z^* denote the optimal solution of pooling problem.

1. *Bound: For any pwl MILP relaxation $\mathcal{P}$, let $z^{\mathcal{P}}$ be the optimal value of the MILP. Then*

$$z^* \leq z^{\mathcal{P}} \leq nz^*.$$

Main Results - Relaxations

Why do these MILP relaxations work so well?

Theorem

Let n denote the number of output nodes. Let z^* denote the optimal solution of pooling problem.

1. *Bound: For any pwl MILP relaxation $\mathcal{P}$, let $z^{\mathcal{P}}$ be the optimal value of the MILP. Then*

$$z^* \leq z^{\mathcal{P}} \leq nz^*.$$

2. *Quality of analysis: Suppose we choose a pwl MILP relaxation $\mathcal{P}$. Then for any $\epsilon > 0$, there exists an instance of the pooling problem with*

$$z^{\mathcal{P}} \geq (n - \epsilon)z^*.$$

Main Results - Computational Complexity

Corollary

There exists a polynomial-time algorithm that produces a feasible solution with objective function value $z^{\mathcal{A}}$ that satisfies $z^{\mathcal{A}} \geq \frac{z^*}{n}$.

Main Results - Computational Complexity

Corollary

There exists a polynomial-time algorithm that produces a feasible solution with objective function value $z^{\mathcal{A}}$ that satisfies $z^{\mathcal{A}} \geq \frac{z^*}{n}$.

1. Note that the dimension of the pooling problem is governed by $|I|$ (number of input nodes), $|L|$ (number of pools), $n := |J|$ (number of output nodes), $|K|$ (number of specs.).

Main Results - Computational Complexity

Corollary

There exists a polynomial-time algorithm that produces a feasible solution with objective function value $z^{\mathcal{A}}$ that satisfies $z^{\mathcal{A}} \geq \frac{z^*}{n}$.

1. Note that the dimension of the pooling problem is governed by $|I|$ (number of input nodes), $|L|$ (number of pools), $n := |J|$ (number of output nodes), $|K|$ (number of specs.).
2. But a factor of ‘n’ is still very bad, ...

Main Results - Computational Complexity

Corollary

There exists a polynomial-time algorithm that produces a feasible solution with objective function value $z^{\mathcal{A}}$ that satisfies $z^{\mathcal{A}} \geq \frac{z^*}{n}$.

1. Note that the dimension of the pooling problem is governed by $|I|$ (number of input nodes), $|L|$ (number of pools), $n := |J|$ (number of output nodes), $|K|$ (number of specs.).
2. But a factor of 'n' is still very bad, ...

Proposition

If there exists a polynomial-time approximation algorithm with guarantee $z^{\mathcal{A}} \geq \frac{z^*}{n-\epsilon}$ for any $\epsilon > 0$ for the pooling problem, then any problem in NP has randomized polynomial time algorithm.

Main results - Computational

1. Actually the ‘approximation algorithm’ is a **"rather silly algorithm".**

Main results - Computational

1. Actually the ‘approximation algorithm’ is a **"rather silly algorithm".**

2. We generalize the key ideas behind the n-approximation algorithm to construct a MILP s.t.

   2.1 MILP’s feasible region is a restriction of the pooling problem.

   2.2 All solution produced by the approximation algorithnm belong to the MILP $\Rightarrow$ **MILP produces solutions within a factor of n.**

Main results - Computational

1. Actually the ‘approximation algorithm’ is a **"rather silly algorithm".**
2. We generalize the key ideas behind the n-approximation algorithm to construct a MILP s.t.
   2.1 MILP's feasible region is a restriction of the pooling problem.
   2.2 All solution produced by the approximation algorithnm belong to the MILP $\Rightarrow$ **MILP produces solutions within a factor of n.**
3. The MILP produces good results in short time. In particular, for a number of problems in the literature, we have **found the best known solutions.**

3

Quality of MILP relaxation

Quality of dual bound

Proposition

Let n denote the number of output nodes. Let z^* denote the optimal solution of pooling problem. For any pwl MILP relaxation $\mathcal{P}$, let $z^{\mathcal{P}}$ be the optimal value of the MILP. Then

$$z^* \leq z^{\mathcal{P}} \leq nz^*.$$

High level proof technique

1. We construct a general relaxation $\mathcal{R}$ of the pooling problem, such that $\mathcal{R} \supseteq \mathcal{P}$, i.e. $\mathcal{R}$ is a relaxation of $\mathcal{P}$, for all $\mathcal{P}$.

High level proof technique

1. We construct a general relaxation $\mathcal{R}$ of the pooling problem, such that $\mathcal{R} \supseteq \mathcal{P}$, i.e. $\mathcal{R}$ is a relaxation of $\mathcal{P}$, for all $\mathcal{P}$.
2. Let $z^{\mathcal{R}}$ be the optimal solution of $\mathcal{R}$:

$$z^{\mathcal{P}} \leq z^{\mathcal{R}}$$

High level proof technique

1. We construct a general relaxation $\mathcal{R}$ of the pooling problem, such that $\mathcal{R} \supseteq \mathcal{P}$, i.e. $\mathcal{R}$ is a relaxation of $\mathcal{P}$, for all $\mathcal{P}$.
2. Let $z^{\mathcal{R}}$ be the optimal solution of $\mathcal{R}$:
$$z^{\mathcal{P}} \leq z^{\mathcal{R}}$$
3. We will show that
$$z^{\mathcal{R}} \leq n\tilde{z},$$
where $\tilde{z}$ is objective function value of a feasible solution of the pooling problem.

High level proof technique

1. We construct a general relaxation $\mathcal{R}$ of the pooling problem, such that $\mathcal{R} \supseteq \mathcal{P}$, i.e. $\mathcal{R}$ is a relaxation of $\mathcal{P}$, for all $\mathcal{P}$.
2. Let $z^{\mathcal{R}}$ be the optimal solution of $\mathcal{R}$:
$$z^{\mathcal{P}} \leq z^{\mathcal{R}}$$
3. We will show that
$$z^{\mathcal{R}} \leq n\tilde{z},$$
where $\tilde{z}$ is objective function value of a feasible solution of the pooling problem.
4. Therefore, $\tilde{z} \leq z^*$

High level proof technique

1. We construct a general relaxation $\mathcal{R}$ of the pooling problem, such that $\mathcal{R} \supseteq \mathcal{P}$, i.e. $\mathcal{R}$ is a relaxation of $\mathcal{P}$, for all $\mathcal{P}$.
2. Let $z^{\mathcal{R}}$ be the optimal solution of $\mathcal{R}$:
$$z^{\mathcal{P}} \leq z^{\mathcal{R}}$$
3. We will show that
$$z^{\mathcal{R}} \leq n\tilde{z},$$
where $\tilde{z}$ is objective function value of a feasible solution of the pooling problem.
4. Therefore, $\tilde{z} \leq z^*$
5. Combining we obtain: $z^{\mathcal{P}} \leq z^{\mathcal{R}} \leq n\tilde{z} \leq nz^*$, i.e., $\boxed{z^{\mathcal{P}} \leq nz^*}$

High level proof technique

1. We construct a general relaxation $\mathcal{R}$ of the pooling problem, such that $\mathcal{R} \supseteq \mathcal{P}$, i.e. $\mathcal{R}$ is a relaxation of $\mathcal{P}$ as well.
2. Let $z^{\mathcal{R}}$ be the optimal objective function value of $\mathcal{R}$:
$$z^{\mathcal{P}} \leq z^{\mathcal{R}}$$
3. We will show that
$$z^{\mathcal{R}} \leq n\tilde{z},$$
where $\tilde{z}$ is objective function value of a feasible solution of the pooling problem.
4. Therefore, $\tilde{z} \leq z^*$
5. Combining we obtain: $z^{\mathcal{P}} \leq z^{\mathcal{R}} \leq n\tilde{z} \leq nz^*$, i.e., $z^{\mathcal{P}} \leq nz^*$

Inconsistent Pool Outflow Problem (IPOP)

$$
\begin{aligned}
\max \quad & \sum_{i \in I, j \in J} w_{ij} y_{ij} + \sum_{i \in I, l \in L, j \in J} (w_{il} + w_{lj}) v_{ilj} \\
\text{s.t.} \quad & v_{ilj} = q_{il} y_{lj} \ \forall i \in I, l \in L, j \in J \\
& \sum_{i \in I} q_{il} = 1 \ \forall l \in L \\
& a_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \leq \sum_{i \in I} \lambda_i^k y_{ij} + \sum_{i \in I, l \in L} \lambda_i^k v_{ilj} \leq b_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \\
& \text{Capacity constriants} \\
& \text{All variables are non-negative} \\
& \sum_{i \in I} v_{ilj} = y_{lj} \ \forall l \in L, j \in J \\
& \sum_{j \in J} v_{ilj} \leq c_l q_{il} \ \forall i \in I, l \in L.
\end{aligned}
$$

Inconsistent Pool Outflow Problem (IPOP)

IPOP:

$$\max \sum_{i \in I, j \in J} w_{ij} y_{ij} + \sum_{i \in I, l \in L, j \in J} (w_{il} + w_{lj}) v_{ilj}$$

s.t.

$$a_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \leq \sum_{i \in I} \lambda_i^k y_{ij} + \sum_{i \in I, l \in L} \lambda_i^k v_{ilj} \leq b_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right)$$

Capacity constraints

All variables are non-negative

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

Inconsistent Pool Outflow Problem (IPOP)

IPOP:

$$\max \sum_{i \in I, j \in J} w_{ij} y_{ij} + \sum_{i \in I, l \in L, j \in J} (w_{il} + w_{lj}) v_{ilj}$$

s.t.

$$a_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \leq \sum_{i \in I} \lambda_i^k y_{ij} + \sum_{i \in I, l \in L} \lambda_i^k v_{ilj} \leq b_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right)$$

Capacity constraints

All variables are non-negative

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

Bilinear constraint relaxed: $v_{ilj} = q_{il} y_{lj}$

Inconsistent Pool Outflow Problem (IPOP)

IPOP:

$$\max \sum_{i \in I, j \in J} w_{ij} y_{ij} + \sum_{i \in I, l \in L, j \in J} (w_{il} + w_{lj}) v_{ilj}$$

$$\text{s.t.} \quad a_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \leq \sum_{i \in I} \lambda_i^k y_{ij} + \sum_{i \in I, l \in L} \lambda_i^k v_{ilj} \leq b_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right)$$

Capacity constraints

All variables are non-negative

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

Bilinear constraint relaxed: $v_{ilj} = q_{il} y_{lj}$

1. This is a relaxation of $\mathcal{P}$ (any piecewise linear MILP relaxation).

Inconsistent Pool Outflow Problem (IPOP)

IPOP:

$$\max \sum_{i\in I, j\in J} w_{ij} y_{ij} + \sum_{i\in I, l\in L, j\in J} (w_{il} + w_{lj}) v_{ilj}$$

$$\text{s.t.} \quad a_j^k \left(\sum_{i\in I} y_{ij} + \sum_{l\in L} y_{lj} \right) \leq \sum_{i\in I} \lambda_i^k y_{ij} + \sum_{i\in I, l\in L} \lambda_i^k v_{ilj} \leq b_j^k \left(\sum_{i\in I} y_{ij} + \sum_{l\in L} y_{lj} \right)$$

Capacity constraints

All variables are non-negative

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

Bilinear constraint relaxed: $v_{ilj} = q_{il} y_{lj}$

1. This is a relaxation of $\mathcal{P}$ (any piecewise linear MILP relaxation).
2. $\frac{v_{ilj}}{y_{lj}} = q_{il}$

Inconsistent Pool Outflow Problem (IPOP)

IPOP:

$$\max \sum_{i \in I, j \in J} w_{ij} y_{ij} + \sum_{i \in I, l \in L, j \in J} (w_{il} + w_{lj}) v_{ilj}$$

s.t. $$a_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right) \leq \sum_{i \in I} \lambda_i^k y_{ij} + \sum_{i \in I, l \in L} \lambda_i^k v_{ilj} \leq b_j^k \left(\sum_{i \in I} y_{ij} + \sum_{l \in L} y_{lj} \right)$$

Capacity constraints

All variables are non-negative

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

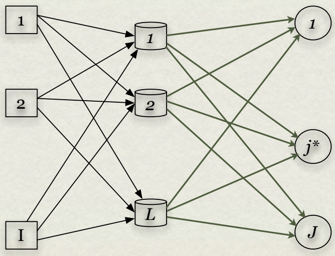
Bilinear constraint relaxed: $v_{ilj} = q_{il} y_{lj}$

1. This is a relaxation of $\mathcal{P}$ (any piecewise linear MILP relaxation).
2. $\frac{v_{ilj}}{y_{lj}} = q_{il}$: Implies the spec value in the different outer arc from a pool to be consistent.

Rounding IPOP Solution

Let (y, v) be optimal to IPOP and $j^* \in J$ be most 'profitable output':

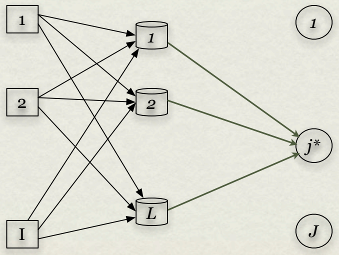
$$j^* \in \text{argmax}_{j \in J} \left\{ \sum_{i \in I} w_{ij} y_{ij} + \sum_{i \in I, l \in L} (w_{il} + w_{lj}) v_{ilj} \right\}$$



Rounding IPOP Solution

Let (y, v) be optimal to IPOP and $j^* \in J$ be most 'profitable output':

$$j^* \in \text{argmax}_{j \in J} \left\{ \sum_{i \in I} w_{ij} y_{ij} + \sum_{i \in I, l \in L} (w_{il} + w_{lj}) v_{ilj} \right\}$$



$$\bar{v}_{ilj} = \begin{cases} 0 & j \neq j^* \\ v^*_{ilj} & j = j^* \end{cases}$$

$$\bar{y}_{lj} = \begin{cases} 0 & j \neq j^* \\ y^*_{lj} & j = j^* \end{cases}$$

$$\bar{q}_{il} = \begin{cases} \dfrac{\bar{v}_{ilj^*}}{\bar{y}_{lj^*}} & \bar{y}_{lj^*} > 0 \\ \dfrac{1}{\text{indeg}(l)} & \bar{y}_{lj^*} = 0 \end{cases}$$

Final details

Lemma

$(\bar{y}, \bar{v}, \bar{q})$ *is a valid solution to the pooling problem.*

Final details

Lemma

$(\bar{y}, \bar{v}, \bar{q})$ *is a valid solution to the pooling problem.*

Proposition

$z^{\mathcal{R}} \leq n\tilde{z}$

Proof:

Final details

Lemma

$(\bar{y}, \bar{v}, \bar{q})$ is a valid solution to the pooling problem.

Proposition

$z^{\mathcal{R}} \leq n\tilde{z}$

Proof:

1. (y, v) is optimal solution of *IPOP*
2. $(\bar{y}, \bar{v}, \bar{q})$ is a valid solution for pooling problem.

Final details

Lemma

$(\bar{y}, \bar{v}, \bar{q})$ is a valid solution to the pooling problem.

Proposition

$z^{\mathcal{R}} \leq n\tilde{z}$

Proof:

1. (y, v) is optimal solution of *IPOP*
2. $(\bar{y}, \bar{v}, \bar{q})$ is a valid solution for pooling problem.
3. Because j^* was picked greedily, we have that $n(\text{Obj. value}(\bar{y}, \bar{v}, \bar{q})) \geq (\text{Obj. value}(y, v))$.

Analysis is tight

Proposition (Quality of analysis)

Suppose we choose a pwl MILP relaxation $\mathcal{P}$. Then for any $\epsilon > 0$, there exists an instance of the pooling problem with

$$z^{\mathcal{P}} \geq (n - \epsilon) z^*.$$

Analysis is tight : High level idea

Construct a pooling problem with:

1. 2 input nodes, 1 pool node, *n* out put nodes, 2 specs, no direct arcs between input and output nodes.

Analysis is tight : High level idea

Construct a pooling problem with:

1. 2 input nodes, 1 pool node, n out put nodes, 2 specs, no direct arcs between input and output nodes.
2. Each output node has a capacity of 1.

Analysis is tight : High level idea

Construct a pooling problem with:

1. 2 input nodes, 1 pool node, n out put nodes, 2 specs, no direct arcs between input and output nodes.
2. Each output node has a capacity of 1.
3. The spec requirement of the n output nodes to not match; If $j \neq k$:

$$\begin{aligned} \{(u, v) \in \mathbb{R}^2 \mid a_j^1 \leq u \leq b_j^1, a_j^2 \leq u \leq b_j^2\} & \cap \\ \{(u, v) \in \mathbb{R}^2 \mid a_k^1 \leq u \leq b_k^1, a_k^2 \leq u \leq b_k^2\} & = \quad \emptyset \end{aligned}$$

Analysis is tight : High level idea

Construct a pooling problem with:

1. 2 input nodes, 1 pool node, n out put nodes, 2 specs, no direct arcs between input and output nodes.
2. Each output node has a capacity of 1.
3. The spec requirement of the n output nodes to not match; If $j \neq k$:

$$\begin{aligned} \{(u, v) \in \mathbb{R}^2 \,|\, a_j^1 \leq u \leq b_j^1, a_j^2 \leq u \leq b_j^2\} \quad &\cap \\ \{(u, v) \in \mathbb{R}^2 \,|\, a_k^1 \leq u \leq b_k^1, a_k^2 \leq u \leq b_k^2\} \quad &= \quad \emptyset \end{aligned}$$

On the other hand the are very "similar": $a_j \approx a_k, \; b_j \approx b_k$.

Analysis is tight : High level idea

Construct a pooling problem with:

1. 2 input nodes, 1 pool node, n out put nodes, 2 specs, no direct arcs between input and output nodes.
2. Each output node has a capacity of 1.
3. The spec requirement of the n output nodes to not match; If $j \neq k$:

$$\begin{aligned} \{(u,v) \in \mathbb{R}^2 \mid a_j^1 \leq u \leq b_j^1, a_j^2 \leq u \leq b_j^2\} & \quad \cap \\ \{(u,v) \in \mathbb{R}^2 \mid a_k^1 \leq u \leq b_k^1, a_k^2 \leq u \leq b_k^2\} & \quad = \quad \emptyset \end{aligned}$$

   On the other hand the are very "similar": $a_j \approx a_k,\ b_j \approx b_k$.
4. Maximize the total flow through the pool.

Analysis is tight : High level idea

Construct a pooling problem with:

1. 2 input nodes, 1 pool node, n out put nodes, 2 specs, no direct arcs between input and output nodes.
2. Each output node has a capacity of 1.
3. The spec requirement of the n output nodes to not match; If $j \neq k$:

$$\begin{aligned} \{(u,v) \in \mathbb{R}^2 \mid a_j^1 \leq u \leq b_j^1, a_j^2 \leq u \leq b_j^2\} &\cap \\ \{(u,v) \in \mathbb{R}^2 \mid a_k^1 \leq u \leq b_k^1, a_k^2 \leq u \leq b_k^2\} &= \emptyset \end{aligned}$$

   On the other hand the are very "similar": $a_j \approx a_k,\ b_j \approx b_k$.

4. Maximize the total flow through the pool.

1. Pooling Problem: In the actual problem flow can be sent to at most one output node. Therefore max flow $= 1$.
2. IPOP: On the other hand in the pwl relaxation since $v_{ilj} \approx q_{il} y_{lj}$ and since $a_j \approx a_k,\ b_j \approx b_k$, we can set $y_{lj} \approx 1$ for all $j \in J$.

4 Approximation Algorithm

Algorithm

'Algorithm':

1. Solve IPOP
2. Round it to make it feasible for pooling problem.

Algorithm

‘Algorithm’:

1. Solve IPOP
2. Round it to make it feasible for pooling problem.

Is it possible to obtain a better approximation ratio in polynomial time?

Approximation preserving reduction from Stable Set

Theorem (Hardness to Approximate Max Stable Set (Håstad))

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 any problem in NP can be solved in probabilistic polynomial time.

Approximation preserving reduction from Stable Set

Theorem (Hardness to Approximate Max Stable Set (Håstad))

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 any problem in NP can be solved in probabilistic polynomial time.

Proposition (Approximation Factor Preserving Reduction from Stable Set Problem)

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

Approximation preserving reduction from Stable Set

Theorem (Hardness to Approximate Max Stable Set (Håstad))

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 any problem in NP can be solved in probabilistic polynomial time.

Proposition (Approximation Factor Preserving Reduction from Stable Set Problem)

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.*

Approximation preserving reduction from Stable Set

Theorem (Hardness to Approximate Max Stable Set (Håstad))

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 any problem in NP can be solved in probabilistic polynomial time.

Proposition (Approximation Factor Preserving Reduction from Stable Set Problem)

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.*

5
A new MILP restriction and computational result

‘Improving’ the solutions from the approximation algorithm - I

1. Construct an IP whose feasible region contains all the feasible solutions generated by the approximation algorithm.

‘Improving’ the solutions from the approximation algorithm - I

1. Construct an IP whose feasible region contains all the feasible solutions generated by the approximation algorithm.
2. Improve further by having "as many as possible" solutions of the pooling problem not generated by the approximation algorithm.

‘Improving’ the solutions from the approximation algorithm - I

1. Construct an IP whose feasible region contains all the feasible solutions generated by the approximation algorithm.
2. Improve further by having "as many as possible" solutions of the pooling problem not generated by the approximation algorithm.

We constructed a feasible solution of the pooling problem such that only one output node $j^* \in J$ receives positive flow.

$$\Updownarrow$$

Solve a multicommodity flow problem with the additional condition that all flow must go to one output node

'Improving' the solutions from the approximation algorithm - II

Solve a multicommodity flow problem with the additional condition that all flow must go to one output node

'Improving' the solutions from the approximation algorithm - II

Solve a multicommodity flow problem with the additional condition that all flow must go to one output node

$\subseteq$ Solve a multicommodity flow problem with the additional condition that flow from every pool goes to atmost one output node.

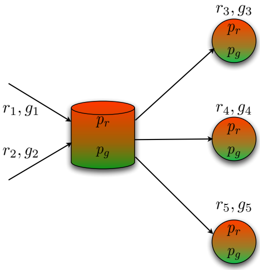
‘Improving’ the solutions from the approximation algorithm - II

Solve a multicommodity flow problem with the additional condition that all flow must go to one output node

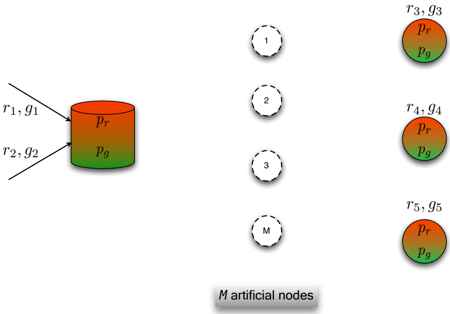
$\subseteq$ Solve a multicommodity flow problem with the additional condition that flow from every pool goes to atmost one output node.

$\boxed{\text{Can we try and do better?}}$

Using IP technology: A new method to discretize

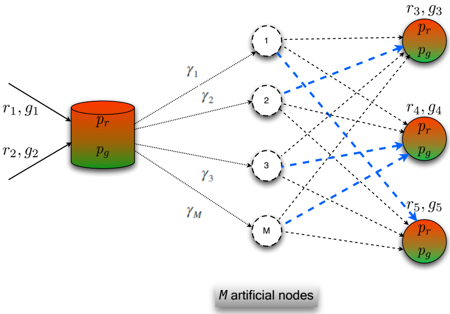


Using IP technology: A new method to discretize



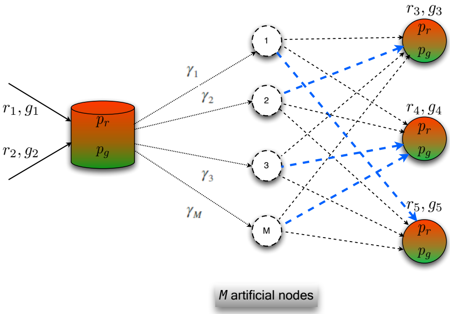
1. Introduce *M* artificial nodes.

Using IP technology: A new method to discretize



1. Introduce *M* artificial nodes.
2. Each artificial node *t* receives γ_t fraction of total flow.

Using IP technology: A new method to discretize



1. Introduce *M* artificial nodes.
2. Each artificial node *t* receives γ_t fraction of total flow.
3. Each artificial node *t* sends flow to one output only.

MILP formulation

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

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

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

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

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

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

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

Spec bounds for $j\in J, k\in K$, y is feasible flow.

More details

Some reasonable choices for γ:

1. $\gamma_t = M^{-1}$
2. $\gamma_t = \frac{1}{1-2^{-M}} 2^{-t}$

More details

Some reasonable choices for γ:

1. $\gamma_t = M^{-1}$
2. $\gamma_t = \frac{1}{1-2^{-M}} 2^{-t}$

Theorem

1. *Let $\tilde{z}$ be the optimal solution of the MILP restriction and let z^* be the optimal solution of the pooling problem. Then*

$$\tilde{z} \geq \frac{z^*}{n}.$$

More details

Some reasonable choices for γ:

1. $\gamma_t = M^{-1}$
2. $\gamma_t = \frac{1}{1-2^{-M}} 2^{-t}$

Theorem

1. *Let $\tilde{z}$ be the optimal solution of the MILP restriction and let z^* be the optimal solution of the pooling problem. Then*
$$\tilde{z} \geq \frac{z^*}{n}.$$
2. *Let γ be a rational vector. Then there exists a pooling instance such that*
$$\tilde{z} = \frac{z^*}{n}.$$

Computational Experiment

Instances:

Source	Label	Inputs	Pools	Outputs	Specs
		$\|I\|$	$\|L\|$	$\|J\|$	$\|K\|$
Alfaki et al. (2012)	stdA0-9	20	10	15	12
Alfaki et al. (2012)	stdB0-5	35	17	21	17
Alfaki et al. (2012)	stdC0-3	60	30	40	20
Random	randstd11-20	25	18	25	8
Random	randstd21-30	25	22	30	10
Random	randstd31-40	30	22	35	10
Random	randstd41-50	40	30	45	10
Random	randstd51-60	40	30	50	14

Experiment Details

Solvers

1. BARON 9.0.7: (24 hours)
2. SNOPT (1 hr)
3. Alternating LP technique
4. MILP (1hr) (CPLEX 12.2)

Metric

Experiment Details

Solvers

1. BARON 9.0.7: (24 hours)
2. SNOPT (1 hr)
3. Alternating LP technique
4. MILP (1hr) (CPLEX 12.2)

Metric

1. $\%\text{gap} = 100 \times \left(\frac{\mathit{BestRelax}}{\mathit{BestFeas}} - 1\right)$

Experiment Details

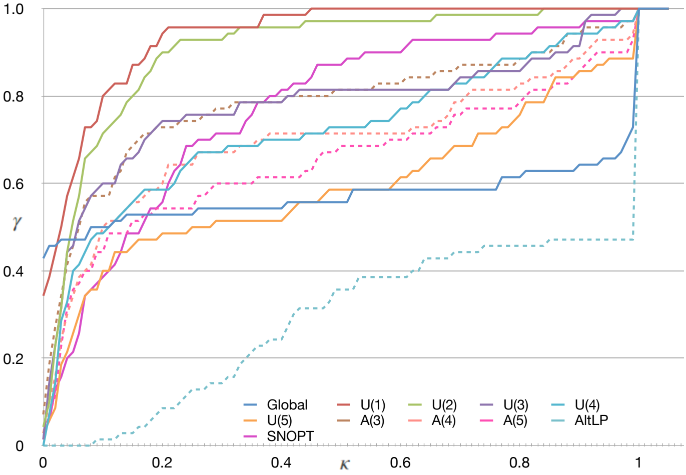
Solvers

1. BARON 9.0.7: (24 hours)
2. SNOPT (1 hr)
3. Alternating LP technique
4. MILP (1hr) (CPLEX 12.2)

Metric

1. $\%\text{gap} = 100 \times \left(\frac{BestRelax}{BestFeas} - 1\right)$
2. $\tau_M(I) :=$
   the best solution value for instance I until termination of method M.
3. $\eta_M(I) := \frac{\tau^{max}(I) - \tau_M(I)}{\tau^{max}(I) - \tau^{min}(I)}$

Performance profile



Geometric Average of % gap versus number of outputs $|J| = n$.

n	Global	$U(1)$	$U(2)$	$U(3)$	$A(3)$	SNOPT	AltLP
15	**1.13**%	2.95%	2.09%	1.70%	1.63%	2.81%	25%
21	8%	**1.00**%	1.57%	1.72%	1.89%	5.18%	12%
25	**0.60**%	3.75%	2.15%	2.18%	1.82%	10%	63%
30	3.50%	2.78%	**1.75**%	1.82%	1.87%	4.09%	39%
35	6%	**2.73**%	3.54%	5.63%	4.71%	5.79%	29%
40	370%	**12**%	15%	15%	18%	115%	38%
45	485%	**5**%	26%	203%	135%	22%	71%
50	550%	**1.87**%	10%	30%	59%	11%	35%
G. Ave.	15%	**3.00**%	4.36%	7.16%	7.18%	8.46%	36%

Discussion

1. For 19 out of the 20 Alfaki et al. instances, MILP heuristic produces the best known results.

Discussion

1. For 19 out of the 20 Alfaki et al. instances, MILP heuristic produces the best known results.

Open Problems:

1. The performance of the MILP heuristic is surprising. Can we better explain this?

Discussion

1. For 19 out of the 20 Alfaki et al. instances, MILP heuristic produces the best known results.

Open Problems:

1. The performance of the MILP heuristic is surprising. Can we better explain this?
2. For a fixed value of out degree of the pool nodes, what is the complexity status of this problem?

Discussion

1. For 19 out of the 20 Alfaki et al. instances, MILP heuristic produces the best known results.

Open Problems:

1. The performance of the MILP heuristic is surprising. Can we better explain this?
2. For a fixed value of out degree of the pool nodes, what is the complexity status of this problem?
3. If we fix the number of specifications, what is the complexity of the problem?

Discussion

1. For 19 out of the 20 Alfaki et al. instances, MILP heuristic produces the best known results.

Open Problems:

1. The performance of the MILP heuristic is surprising. Can we better explain this?
2. For a fixed value of out degree of the pool nodes, what is the complexity status of this problem?
3. If we fix the number of specifications, what is the complexity of the problem?
4. 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?

Thank you

Thank You!