Approximation Algorithms for Process Systems Engineering

Dimitrios Letsios, Radu Baltean-Lugojan, Jeremy Bradley, Francesco Ceccon, Kristijonas Čyras, Georgia Kouyialis, Natasha Page, Johannes Wiebe, Francesca Toni & Ruth Misener

Funding EPSRC EP/P008739/1, EP/P016871/1, & EP/P029558/1

Tuesday 30th March, 2021

Paper Letsios et al., *Computers & Chemical Engineering*, 2020.

Visit London virtually this June . . . for the **MINLParty**!

MINLP workshop

- 28 - 29 June 2021 at Imperial,
- Organized by M Anjos, P Belotti, J Kronqvist & me,
- Mix of invited/contributed talks & videos,
- `https://optimisation.doc.ic.ac.uk/minlp-workshop-2020-june-11-12/`

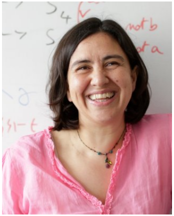
Team members

Georgia Kouyialis Dimtris Letsios

Radu Baltean-Lugojan

Natasha Page

Kristijonas Čyras



Francesca Toni

Heuristics & Approximation algorithms

Heuristic solutions

- Quickly address industrially-sized instances;
- Generate solutions with efficient running times;
- Enhance exact methods with good feasible solutions.

Heuristics & Approximation algorithms

Approximation algorithms – Heuristics with mathematical rigor

Want to find the minimum cost C_{OPT}. Prove a performance guarantee:

- Identify a good lower bound C_{LB};
- Design a heuristic computing good suboptimal solutions C_{ALG};
- Prove analytically that $C_{\mathsf{ALG}} \leq \rho \cdot C_{\mathsf{OPT}}$ for every instance.

Heuristic solutions

- Quickly address industrially-sized instances;
- Generate solutions with efficient running times;
- Enhance exact methods with good feasible solutions.

Heuristics & Approximation algorithms

Approximation algorithms – Heuristics with mathematical rigor

Want to find the minimum cost $\mathbf{C}_{\text{OPT}}$. Prove a performance guarantee:

- Identify a good lower bound C_{LB};
- Design a heuristic computing good suboptimal solutions C_{ALG};
- Prove analytically that $C_{\text{ALG}} \leq \rho \cdot C_{\text{OPT}}$ for every instance.

Heuristic solutions

- Quickly address industrially-sized instances;
- Generate solutions with efficient running times;
- Enhance exact methods with good feasible solutions.

Heuristics & Approximation algorithms

Approximation algorithms – Heuristics with mathematical rigor

Want to find the minimum cost C_{OPT}. Prove a performance guarantee:

- Identify a good lower bound $\mathbf{C}_{\mathsf{LB}}$;
- Design a heuristic computing good suboptimal solutions C_{ALG};
- Prove analytically that $C_{\mathsf{ALG}} \leq \rho \cdot C_{\mathsf{OPT}}$ for every instance.

Heuristic solutions

- Quickly address industrially-sized instances;
- Generate solutions with efficient running times;
- Enhance exact methods with good feasible solutions.

Heuristics & Approximation algorithms

Approximation algorithms – Heuristics with mathematical rigor

Want to find the minimum cost C_{OPT}. Prove a performance guarantee:

- Identify a good lower bound C_{LB};
- Design a heuristic computing good suboptimal solutions $\mathbf{C}_{\text{ALG}}$;
- Prove analytically that $C_{\text{ALG}} \leq \rho \cdot C_{\text{OPT}}$ for every instance.

Heuristic solutions

- Quickly address industrially-sized instances;
- Generate solutions with efficient running times;
- Enhance exact methods with good feasible solutions.

Heuristics & Approximation algorithms

Approximation algorithms – Heuristics with mathematical rigor

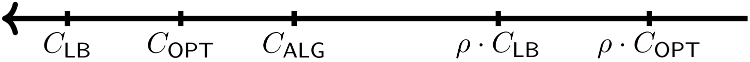
Want to find the minimum cost C_{OPT}. Prove a performance guarantee:

- Identify a good lower bound C_{LB};
- Design a heuristic computing good suboptimal solutions C_{ALG};
- Prove analytically that $\mathbf{C}_{\mathsf{ALG}} \leq \rho \cdot \mathbf{C}_{\mathsf{OPT}}$ for every instance.

Heuristic solutions

- Quickly address industrially-sized instances;
- Generate solutions with efficient running times;
- Enhance exact methods with good feasible solutions.

Practical applicability of approximation algorithms?

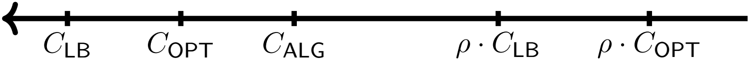


Useful for process systems engineering?

- Important optimization problems in PSE applications
  - Heat recovery networks
  - State-task network
  - Pooling problem
- Recovery & reoptimization
  - Royal Mail van allocation
- Explainable scheduling

Letsios, Kouyialis & Misener *Comput Chem Eng*, **113**:57-85, 2018.

Practical applicability of approximation algorithms?



Useful for process systems engineering?

- **Important optimization problems in PSE applications**
    - **Heat recovery networks**
    - State-task network
    - Pooling problem
- Recovery & reoptimization
    - Royal Mail van allocation
- Explainable scheduling

Letsios, Kouyialis & Misener *Comput Chem Eng*, **113**:57-85, 2018.

Heat recovery networks

Simultaneous method

Solve a mixed-integer nonlinear optimization problem, e.g. Ciric & Floudas [1989], Yee & Grossmann [1990], Papalexandri & Pistikopoulos [1994].

Heat recovery networks

Simultaneous method

Solve a mixed-integer nonlinear optimization problem, e.g. Ciric & Floudas [1989], Yee & Grossmann [1990], Papalexandri & Pistikopoulos [1994].

Sequential method

- Minimum utility cost — Linear program (LP)
- Minimum number of matches — Mixed-integer linear program (MILP)
  - Papoulias & Grossmann [1983], Cerda & Westerberg [1983], Anantharaman et al. [2010]
- Minimum investment cost — Nonlinear program (NLP)
  - Floudas et al. [1986]
  - **Goal** Generate many good candidate MILP solutions

Heat recovery networks

Simultaneous method

Solve a mixed-integer nonlinear optimization problem, e.g. Ciric & Floudas [1989], Yee & Grossmann [1990], Papalexandri & Pistikopoulos [1994].

Sequential method

- Minimum utility cost — Linear program (LP)
- Minimum number of matches — Mixed-integer linear program (MILP)
  - Papoulias & Grossmann [1983], Cerda & Westerberg [1983], Anantharaman et al. [2010]
- Minimum investment cost — Nonlinear program (NLP)
  - Floudas et al. [1986]
  - **Goal** Generate many good candidate MILP solutions

Heat recovery networks

Simultaneous method

Solve a mixed-integer nonlinear optimization problem, e.g. Ciric & Floudas [1989], Yee & Grossmann [1990], Papalexandri & Pistikopoulos [1994].

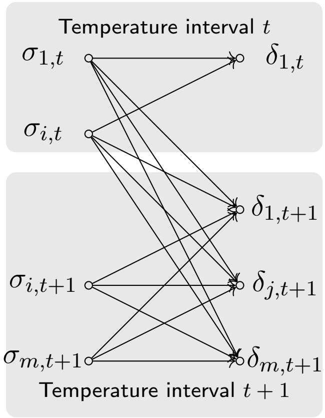
Sequential method

- Minimum utility cost — Linear program (LP)
- Minimum number of matches — Mixed-integer linear program (MILP)
  - Papoulias & Grossmann [1983], Cerda & Westerberg [1983], Anantharaman et al. [2010]
- Minimum investment cost — Nonlinear program (NLP)
  - Floudas et al. [1986]
  - **Goal** Generate many good candidate MILP solutions

Review Article

Furman & Sahinidis [*Ind Eng Chem Res*, 2002]

MILP Transportation Model [Cerda & Westerberg, 1983]



$$\min \sum_{i \in H} \sum_{j \in C} y_{i,j}$$

$$\sum_{j \in C} \sum_{t \in T} q_{i,s,j,t} = \sigma_{i,t} \quad i \in H, s \in T$$

$$\sum_{i \in H} \sum_{s \in T} q_{i,s,j,t} = \delta_{j,t} \quad j \in C, t \in T$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} y_{i,j} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \quad \forall\, i, s, j, t$$

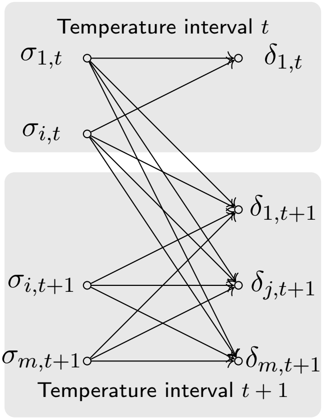
$$q_{i,s,j,t} = 0 \quad s, t \in T, s > t$$

$$y_{i,j} \in \{0,\, 1\} \quad i \in H, j \in C$$

Alternative MILP Transshipment Model [Papoulias & Grossmann, 1983]

Better experimental results, e.g. for CPLEX • Solves 1 additional problem

MILP Transportation Model [Cerda & Westerberg, 1983]



$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\sum_{j \in C} \sum_{t \in T} q_{i,s,j,t} = \sigma_{i,t} \quad i \in H, s \in T$$

$$\sum_{i \in H} \sum_{s \in T} q_{i,s,j,t} = \delta_{j,t} \quad j \in C, t \in T$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \quad \forall\, i, s, j, t$$

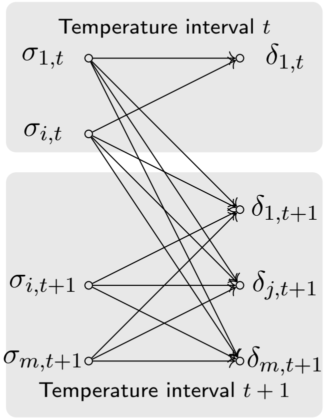
$$q_{i,s,j,t} = 0 \quad s, t \in T, s > t$$

$$\mathbf{y_{i,j}} \in \{\mathbf{0}, \mathbf{1}\} \quad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

Alternative MILP Transshipment Model [Papoulias & Grossmann, 1983]

Better experimental results, e.g. for CPLEX • Solves 1 additional problem

MILP Transportation Model [Cerda & Westerberg, 1983]



$$\min \sum_{i \in H} \sum_{j \in C} y_{i,j}$$

$$\sum_{j \in C} \sum_{t \in T} q_{i,s,j,t} = \sigma_{i,t} \qquad i \in H, s \in T$$

$$\sum_{i \in H} \sum_{s \in T} q_{i,s,j,t} = \delta_{j,t} \qquad j \in C, t \in T$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq \mathbf{U_{i,j}} y_{i,j} \qquad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \qquad \forall\, i, s, j, t$$

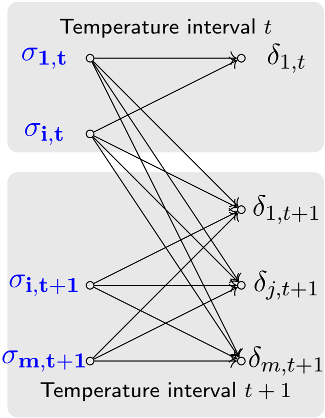
$$q_{i,s,j,t} = 0 \qquad s, t \in T, s > t$$

$$y_{i,j} \in \{0,\ 1\} \qquad i \in H, j \in C$$

Alternative MILP Transshipment Model [Papoulias & Grossmann, 1983]

Better experimental results, e.g. for CPLEX • Solves 1 additional problem

MILP Transportation Model [Cerda & Westerberg, 1983]



$$\min \sum_{i \in H} \sum_{j \in C} y_{i,j}$$

$$\sum_{j \in C} \sum_{t \in T} q_{i,s,j,t} = \sigma_{\mathbf{i,t}} \quad i \in H, s \in T$$

$$\sum_{i \in H} \sum_{s \in T} q_{i,s,j,t} = \delta_{j,t} \quad j \in C, t \in T$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} y_{i,j} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \quad \forall\, i, s, j, t$$

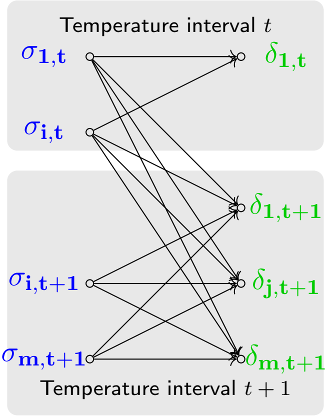
$$q_{i,s,j,t} = 0 \quad s, t \in T, s > t$$

$$y_{i,j} \in \{0,\ 1\} \quad i \in H, j \in C$$

Alternative MILP Transshipment Model [Papoulias & Grossmann, 1983]

Better experimental results, e.g. for CPLEX • Solves 1 additional problem

MILP Transportation Model [Cerda & Westerberg, 1983]



$$\min \sum_{i \in H} \sum_{j \in C} y_{i,j}$$

$$\sum_{j \in C} \sum_{t \in T} q_{i,s,j,t} = \sigma_{\mathbf{i,t}} \quad i \in H, s \in T$$

$$\sum_{i \in H} \sum_{s \in T} q_{i,s,j,t} = \delta_{\mathbf{j,t}} \quad j \in C, t \in T$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} y_{i,j} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \quad \forall\, i, s, j, t$$

$$q_{i,s,j,t} = 0 \quad s, t \in T, s > t$$

$$y_{i,j} \in \{0,\ 1\} \quad i \in H, j \in C$$

Alternative MILP Transshipment Model [Papoulias & Grossmann, 1983]

Better experimental results, e.g. for CPLEX • Solves 1 additional problem

Can't we just use state-of-the-art MILP solvers?

Test set of 48 minimum number of matches problems

- **Furman & Sahinidis [2004]** **Up to 38 streams, 357 binaries**
  - In 2004, 22 of 26 problems solve [7 hr timeout, CPLEX 7.0]
  - In 2017, 23 of 26 problems solve [30 min timeout, CPLEX 12.6.3]
- **Chen et al. [2015]** **Up to 43 streams, 462 binaries**
  - In 2017, 5 of 10 problems solve, 4 of 10 if using transportation model
- **Grossmann [2017]** **Up to 43 streams, 462 binaries**
  - In 2017, 0 of 12 problems solve

Can't we just use state-of-the-art MILP solvers?

Test set of 48 minimum number of matches problems

- **Furman & Sahinidis [2004] — Up to 38 streams, 357 binaries**
  - **In 2004, 22 of 26 problems solve [7 hr timeout, CPLEX 7.0]**
  - **In 2017, 23 of 26 problems solve [30 min timeout, CPLEX 12.6.3]**
- **Chen et al. [2015] — Up to 43 streams, 462 binaries**
  - In 2017, 5 of 10 problems solve, 4 of 10 if using transportation model
- **Grossmann [2017] — Up to 43 streams, 462 binaries**
  - In 2017, 0 of 12 problems solve

Can't we just use state-of-the-art MILP solvers?

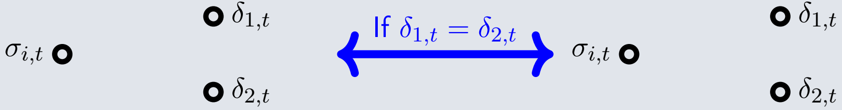
Test set of 48 minimum number of matches problems

- **Furman & Sahinidis [2004]** **Up to 38 streams, 357 binaries**
  - In 2004, 22 of 26 problems solve [7 hr timeout, CPLEX 7.0]
  - In 2017, 23 of 26 problems solve [30 min timeout, CPLEX 12.6.3]
- **Chen et al. [2015]** **Up to 43 streams, 462 binaries**
  - In 2017, 5 of 10 problems solve, 4 of 10 if using transportation model
- **Grossmann [2017]** **Up to 43 streams, 462 binaries**
  - In 2017, 0 of 12 problems solve

	FS04		LKM17		
Test Id	**Obj**	**CPU s**	**Obj**	**CPU s**	**Rel Gap**
20sp1	19	*	19	*	15%
22sp1	25	*	25	*	8%
23sp1	23	*	23	*	26%
37sp-yfyv	36	*	36	7.32	

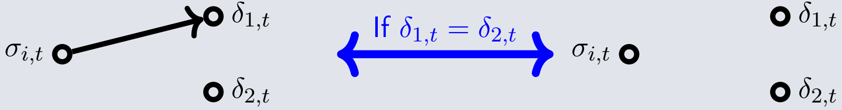
What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]



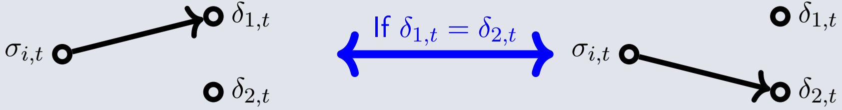
What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]



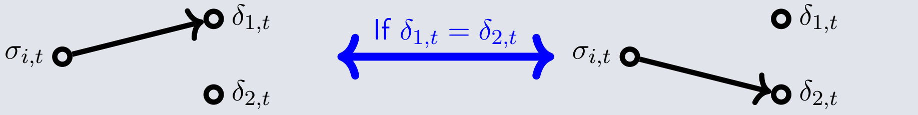
What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]

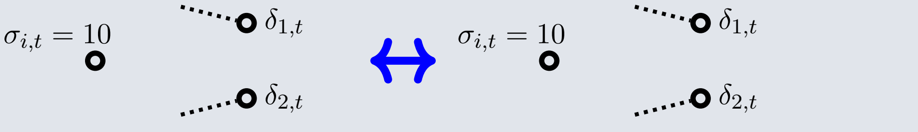


What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]

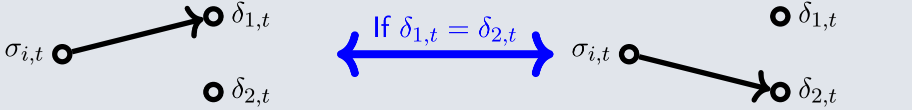


Degeneracy

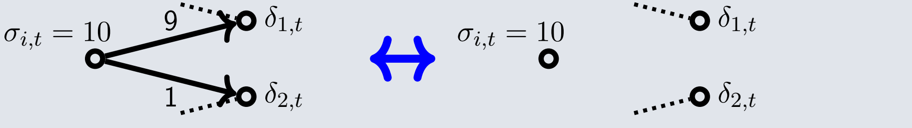


What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]

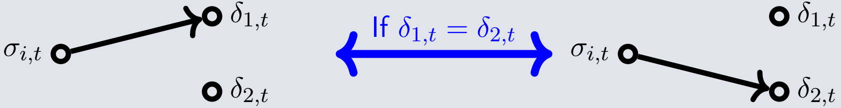


Degeneracy

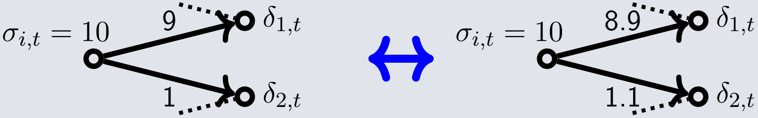


What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]

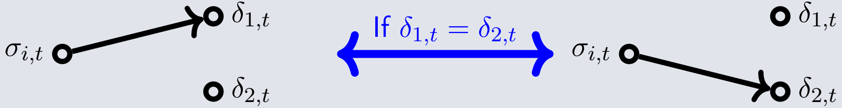


Degeneracy

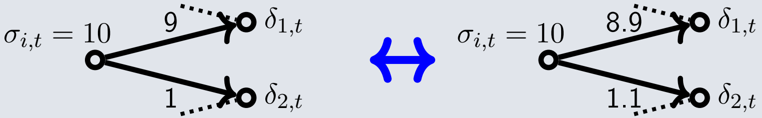


What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]



Degeneracy

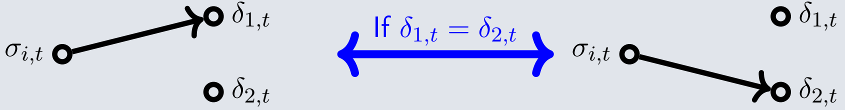


Strongly $\mathcal{NP}$-hard optimization problem [Furman & Sahinidis, 2001]

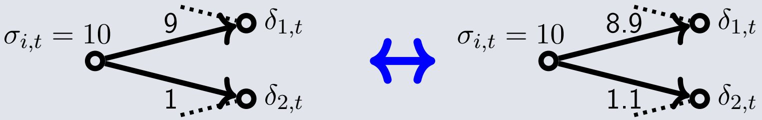
We developed an alternative $\mathcal{NP}$-hardness reduction to bin-packing.

What's the difficulty here?

Symmetry [Kouyialis & Misener, 2017]



Degeneracy



Strongly $\mathcal{NP}$-hard optimization problem [Furman & Sahinidis, 2001]

We developed an alternative $\mathcal{NP}$-hardness reduction to bin-packing.

Similar problems

Scheduling • Cloud computing • Bin packing

Three classes of heuristic methods

Relaxation rounding

Motivation Optimize a simpler, relaxed problem. Round the result.

- Fractional linear programming rounding†
- Lagrangian relaxation rounding†
- Covering relaxation rounding

† Extensions to Furman & Sahinidis [2004]

Water filling heuristics

Motivation Solve temperature intervals serially. Keep composition feasible.

Greedy packing heuristics

Motivation Bin packing $\iff$ minimum number of matches problem

Similar to Linnhoff & Hindmarsh [1983], Cerda, Westerberg, Mason & Linnhoff [1983]

Fractional linear programming rounding

Furman & Sahinidis [2004]

1. Original MILP

$$\min \sum_{i \in H} \sum_{j \in C} y_{i,j}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} y_{i,j} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \quad \forall\, i, s, j, t$$

$$y_{i,j} \in \{0,\, 1\} \quad i \in H, j \in C$$

Fractional linear programming rounding

Furman & Sahinidis [2004]

1. Original MILP

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \quad \forall\, i, s, j, t$$

$$\mathbf{y_{i,j}} \in \{\mathbf{0},\ \mathbf{1}\} \quad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

Fractional linear programming rounding

Furman & Sahinidis [2004]

1. Original MILP

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \qquad \forall\, i, s, j, t$$

$$\mathbf{y_{i,j}} \in \{\mathbf{0},\ \mathbf{1}\} \qquad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

2. Relax MILP integrality

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \geq 0 \qquad \forall\, i, s, j, t$$

$$\mathbf{y_{i,j}} \in [\mathbf{0},\ \mathbf{1}] \qquad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

Fractional linear programming rounding

Furman & Sahinidis [2004]

1. Original MILP

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \le U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \ge 0 \quad \forall\, i, s, j, t$$

$$\mathbf{y_{i,j}} \in \{\mathbf{0},\ \mathbf{1}\} \quad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

2. Relax MILP integrality

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \le U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \ge 0 \quad \forall\, i, s, j, t$$

$$\mathbf{y_{i,j}} \in [\mathbf{0},\ \mathbf{1}] \quad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

3. Solve the relaxed problem

Optimize the linear program.

Fractional linear programming rounding

Furman & Sahinidis [2004]

1. Original MILP

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \le U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \ge 0 \quad \forall\, i, s, j, t$$

$$\mathbf{y_{i,j} \in \{0,\ 1\}} \quad \mathbf{i \in H, j \in C}$$

2. Relax MILP integrality

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} q_{i,s,j,t} \le U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$q_{i,s,j,t} \ge 0 \quad \forall\, i, s, j, t$$

$$\mathbf{y_{i,j} \in [0,\ 1]} \quad \mathbf{i \in H, j \in C}$$

3. Solve the relaxed problem

Optimize the linear program.

4. Generate a feasible solution

If $\sum_{s,t \in T} q_{i,s,j,t} > 0$, $\quad \mathbf{1 \rightarrow y_{i,j}}$.

Else $\quad \mathbf{0 \rightarrow y_{i,j}}$.

Fractional linear programming rounding

Furman & Sahinidis [2004]

1. Original MILP

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} \mathbf{q_{i,s,j,t}} \leq U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$\mathbf{q_{i,s,j,t}} \geq \mathbf{0} \qquad \forall i, s, j, t$$

$$\mathbf{y_{i,j}} \in \{\mathbf{0},\ \mathbf{1}\} \qquad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

2. Relax MILP integrality

$$\min \sum_{i \in H} \sum_{j \in C} \mathbf{y_{i,j}}$$

$$\vdots$$

$$\sum_{s,t \in T} \mathbf{q_{i,s,j,t}} \leq U_{i,j} \mathbf{y_{i,j}} \quad i \in H, j \in C$$

$$\mathbf{q_{i,s,j,t}} \geq 0 \qquad \forall i, s, j, t$$

$$\mathbf{y_{i,j}} \in [\mathbf{0},\ \mathbf{1}] \qquad \mathbf{i} \in \mathbf{H}, \mathbf{j} \in \mathbf{C}$$

3. Solve the relaxed problem

Optimize the linear program.

4. Generate a feasible solution

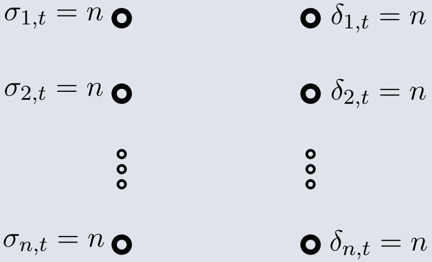
If $\sum_{s,t \in T} \mathbf{q_{i,s,j,t}} > 0$, $\quad \mathbf{1} \rightarrow \mathbf{y_{i,j}}$.

Else $\quad \mathbf{0} \rightarrow \mathbf{y_{i,j}}$.

Asymptotic behavior of Fractional LP Rounding?

FLPR is $\Omega(n)$-approximate

Consider 1 temperature interval ...

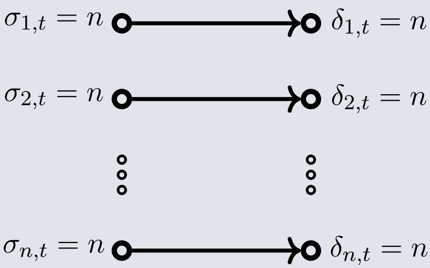


- **Min** n edges with capacity n;
- **Alg** n^2 edges with capacity 1;
- No approximation ratio asymptotically less than n.

Asymptotic behavior of Fractional LP Rounding?

FLPR is $\Omega(n)$-approximate

Consider 1 temperature interval ...

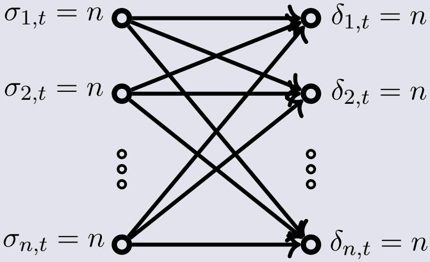


- **Min** n edges with capacity n;
- **Alg** n^2 edges with capacity 1;
- No approximation ratio asymptotically less than n.

Asymptotic behavior of Fractional LP Rounding?

FLPR is $\Omega(n)$-approximate

Consider 1 temperature interval ...

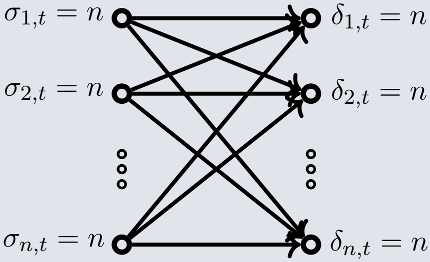


- **Min** n edges with capacity n;
- **Alg** n^2 edges with capacity 1;
- No approximation ratio asymptotically less than n.

Asymptotic behavior of Fractional LP Rounding?

FLPR is $\Omega(n)$-approximate

Consider 1 temperature interval ...



- **Min** n edges with capacity n;
- **Alg** n^2 edges with capacity 1;
- No approximation ratio asymptotically less than n.

FLPR is $O(\max_{(i,j)} U_{ij}/L_{ij})$ approx

Heuristic $y_{i,j}$ versus optimum $y^*_{i,j}$?

$$\begin{aligned}\sum_{i\in H, j\in C} y_{i,j} &= \sum_{i\in H, j\in C} \frac{U_{i,j}}{L_{i,j}} \sum_{s,t\in T} \frac{q_{i,s,j,t}}{U_{i,j}} \\ &\leq \left(\max_{(i,j)} \frac{U_{ij}}{L_{ij}}\right) \sum_{i\in H, j\in C} y^{LP}_{i,j} \\ &\leq \left(\max_{(i,j)} \frac{U_{ij}}{L_{ij}}\right) \sum_{i\in H, j\in C} y^{*}_{i,j}.\end{aligned}$$

$U_{ij} \equiv$ Max heat transfer $i \to j$

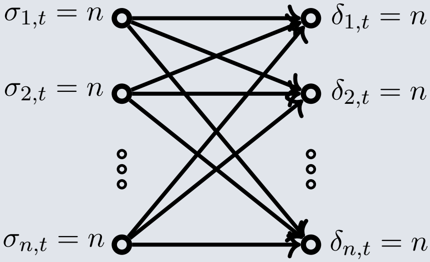
$L_{ij} \equiv$ Min heat transfer $i \to j$

Big-M parameter $U_{i,j}$ *critical*!

Asymptotic behavior of Fractional LP Rounding?

FLPR is $\Omega(n)$-approximate

Consider 1 temperature interval ...



- **Min** n edges with capacity n;
- **Alg** n^2 edges with capacity 1;
- No approximation ratio asymptotically less than n.

FLPR is $O(\max_{(i,j)} U_{ij}/L_{ij})$ approx

Heuristic $\mathbf{y_{i,j}}$ versus optimum $\mathbf{y^*_{i,j}}$?

$$\sum_{i\in H, j\in C} \mathbf{y_{i,j}} = \sum_{i\in H, j\in C} \frac{U_{i,j}}{L_{i,j}} \sum_{s,t\in T} \frac{q_{i,s,j,t}}{U_{i,j}}$$

$$\leq \left(\max_{(i,j)} \frac{U_{ij}}{L_{ij}}\right) \sum_{i\in H, j\in C} y^{LP}_{i,j}$$

$$\leq \left(\max_{(i,j)} \frac{U_{ij}}{L_{ij}}\right) \sum_{i\in H, j\in C} \mathbf{y^*_{i,j}}.$$

$U_{ij} \equiv$ Max heat transfer $i \to j$

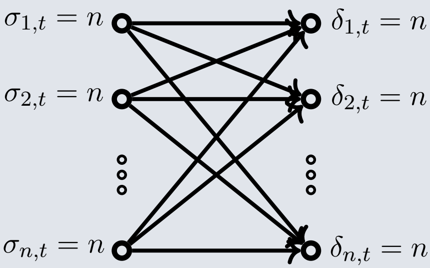
$L_{ij} \equiv$ Min heat transfer $i \to j$

Big-M parameter $U_{i,j}$ *critical*!

Asymptotic behavior of Fractional LP Rounding?

FLPR is $\Omega(n)$-approximate

Consider 1 temperature interval ...



- **Min** n edges with capacity n;
- **Alg** n^2 edges with capacity 1;
- No approximation ratio asymptotically less than n.

FLPR is $O(\max_{(i,j)} U_{ij}/L_{ij})$ approx

Heuristic $\mathbf{y_{i,j}}$ versus optimum $\mathbf{y^*_{i,j}}$?

$$\sum_{i\in H, j\in C} \mathbf{y_{i,j}} = \sum_{i\in H, j\in C} \frac{U_{i,j}}{L_{i,j}} \sum_{s,t\in T} \frac{q_{i,s,j,t}}{U_{i,j}}$$

$$\leq \left(\max_{(i,j)} \frac{U_{ij}}{L_{ij}}\right) \sum_{i\in H, j\in C} y^{LP}_{i,j}$$

$$\leq \left(\max_{(\mathbf{i,j})} \frac{\mathbf{U_{ij}}}{\mathbf{L_{ij}}}\right) \sum_{i\in H, j\in C} \mathbf{y^*_{i,j}}.$$

$\mathbf{U_{ij}} \equiv$ Max heat transfer $\mathbf{i} \to \mathbf{j}$

$\mathbf{L_{ij}} \equiv$ Min heat transfer $\mathbf{i} \to \mathbf{j}$

Big-M parameter $U_{i,j}$ *critical*!

Paper improves big-M values

Three classes of heuristic methods

Relaxation rounding

Motivation Optimize a simpler, relaxed problem. Round the result.

- Fractional linear programming rounding†
- Lagrangian relaxation rounding†
- Covering relaxation rounding

† Extensions to Furman & Sahinidis [2004]

Water filling heuristics

Motivation Solve temperature intervals serially. Keep composition feasible.

Greedy packing heuristics

Motivation Bin packing $\iff$ minimum number of matches problem

Similar to Linnhoff & Hindmarsh [1983], Cerda, Westerberg, Mason & Linnhoff [1983]

Three classes of heuristic methods

Relaxation rounding

Motivation Optimize a simpler, relaxed problem. Round the result.

- Fractional linear programming rounding†
- Lagrangian relaxation rounding†
- Covering relaxation rounding

† Extensions to Furman & Sahinidis [2004]

Water filling heuristics

Motivation Solve temperature intervals serially. Keep composition feasible.

Greedy packing heuristics

Motivation Bin packing $\iff$ minimum number of matches problem

Similar to Linnhoff & Hindmarsh [1983], Cerda, Westerberg, Mason & Linnhoff [1983]

Three classes of heuristic methods

Relaxation rounding

Motivation Optimize a simpler, relaxed problem. Round the result.

- Fractional linear programming rounding†
- Lagrangian relaxation rounding†
- Covering relaxation rounding

† Extensions to Furman & Sahinidis [2004]

Water filling heuristics

Motivation Solve temperature intervals serially. Keep composition feasible.

Greedy packing heuristics

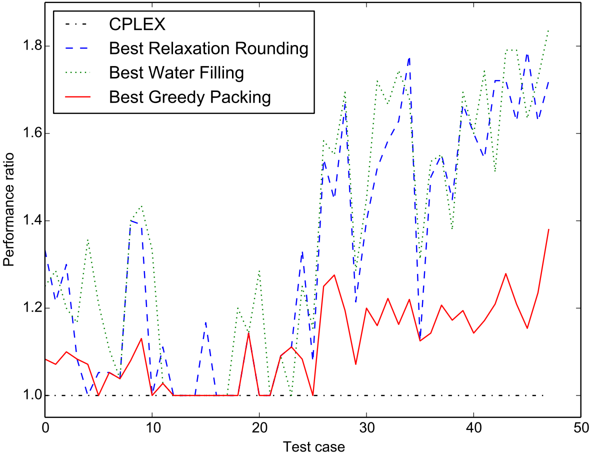
Motivation Bin packing $\iff$ minimum number of matches problem

Similar to Linnhoff & Hindmarsh [1983], Cerda, Westerberg, Mason & Linnhoff [1983]

Three classes of heuristics in competition

3 Relaxation rounding, 2 Water filling, 4 Greedy packing

Performance ratio heuristic value/best known sol'n



Performance guarantees

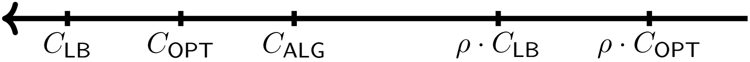
- **LP rounding** $\Omega(n)$
- **Greedy packing** $O(\log n + \log(h_{\max}/\epsilon))$
- Worst case **greedy packing** asymptotic ratio better than best case **LP rounding** in pathological example.

Heat exchanger networks – Larger instances [160 streams]

In all 3 cases, the high quality CPLEX solution took 2 hours to compute.
For 1-2 hours, the heuristic is better ($> 10\%$).

Test Case	Greedy Packing SS		CPLEX Transshipment	
	Value	Time	Value	Time
`large_scale0`	*233*	*642.94*	**175**	*
`large_scale1`	***218***	***652.00***	219	*
`large_scale2`	*242*	*670.32*	**239**	*

Practical applicability of approximation algorithms?

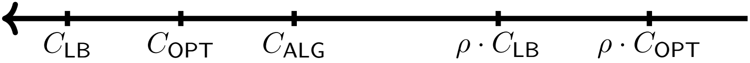


Useful for process systems engineering?

- **Important optimization problems in PSE applications**
  - **Heat recovery networks**
  - State-task network
  - Pooling problem
- Recovery & reoptimization
  - Royal Mail van allocation
- Explainable scheduling

Baltean-Lugojan & Misener, *J Global Optim*, **71**:655-690, 2018.

Practical applicability of approximation algorithms?

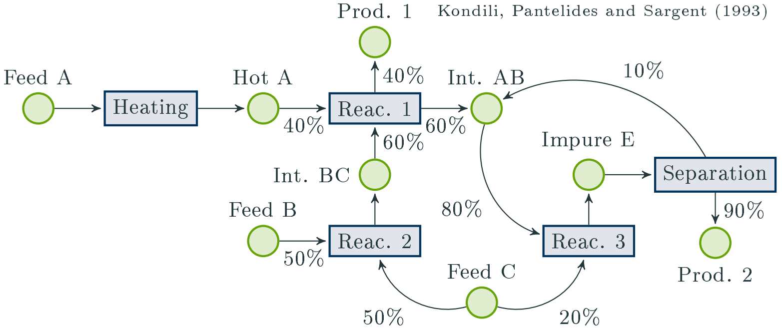


Useful for process systems engineering?

- **Important optimization problems in PSE applications**
  - Heat recovery networks
  - **State-task network**
  - **Pooling problem**
- Recovery & reoptimization
  - Royal Mail van allocation
- Explainable scheduling

Baltean-Lugojan & Misener, *J Global Optim*, **71**:655-690, 2018.

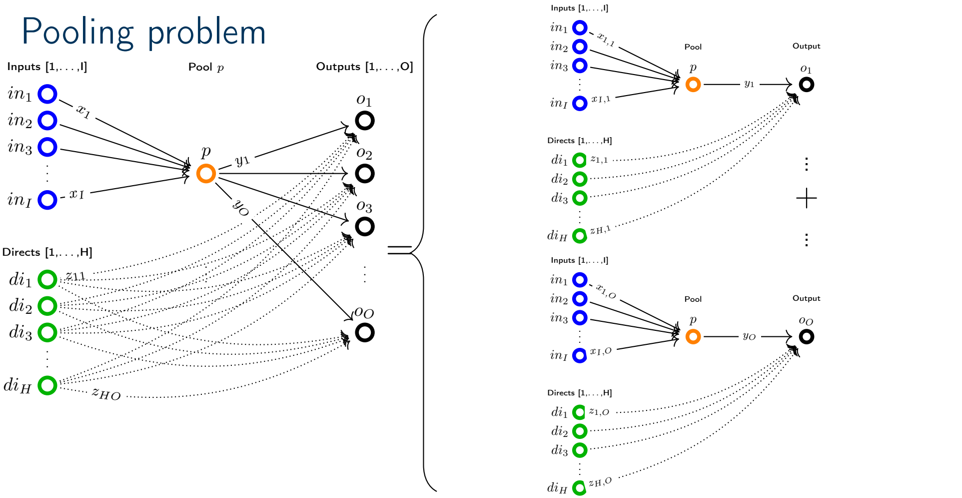
State-task network



STN complexity & efficient heuristics

- Modelling formulations [Maravelias, 2005]
- Generalises job-shop scheduling, so $\mathcal{NP}$-hard [Burkard et al., 1998]
- Special polynomial cases [Blömer & Günther, 2000]
- Efficient feas solutions [Burkard et. al, 1998, Blömer & Günther, 2000]

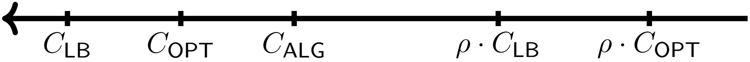
Pooling problem



Complexity & Heuristics https://github.com/cog-imperial/pooling-network

- Reduction from maximum independent set, so $\mathcal{NP}$-hard [Alfaki & Haugland, 2013]
- Polynomial cases [Haugland, 14; Boland et al., 17; Baltean-Lugojan & M, 18]
- MIP approximation heuristic [Dey & Gupte, 2015]

Practical applicability of approximation algorithms?

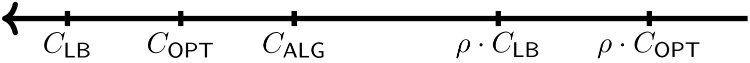


Useful for process systems engineering?

- **Important optimization problems in PSE applications**
  - **Heat recovery networks**
  - **State-task network**
  - **Pooling problem**
- Recovery & reoptimization
  - Royal Mail van allocation
- Explainable scheduling

Letsios & Misener, *European J Operational Research*, 2021.

Practical applicability of approximation algorithms?

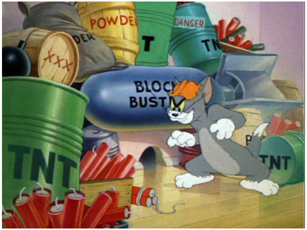


Useful for process systems engineering?

- Important optimization problems in PSE applications
  - Heat recovery networks
  - State-task network
  - Pooling problem
- **Recovery & reoptimization**
  - **Royal Mail van allocation**
- Explainable scheduling

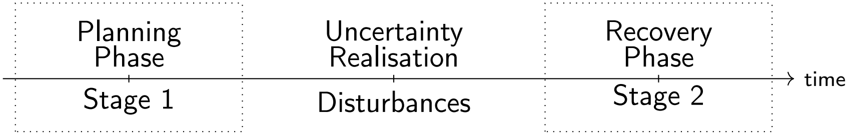
Letsios & Misener, *European J Operational Research*, 2021.

How to deal with highly uncertain environments?



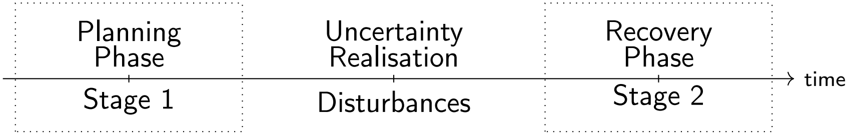
Planning & Recovery

Liebchen, Lübbecke, Möhring, Stiller [2009]



Planning & Recovery

Liebchen, Lübbecke, Möhring, Stiller [2009]



Sources of uncertainty

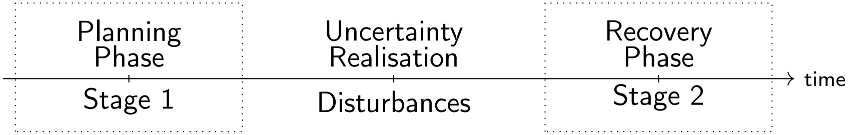
- Unexpected incidents,
- Erroneous input data,
- Future events.

Related Work

- 2-stage robust optimization,
- Recoverable robustness,
- Adjustable robustness.

Planning & Recovery

Liebchen, Lübbecke, Möhring, Stiller [2009]



Sources of uncertainty

- Unexpected incidents,
- Erroneous input data,
- Future events.

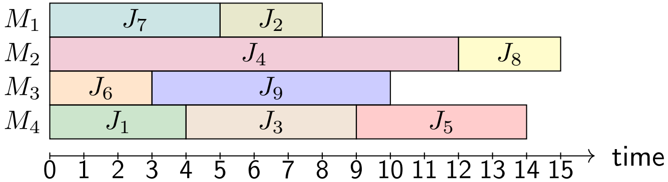
Related Work

- 2-stage robust optimization,
- Recoverable robustness,
- Adjustable robustness.

Benefit of reoptimization

Reactive response in case of unexpected disturbances.

Multiprocessor scheduling – Strongly $\mathcal{NP}$-hard



Input

- Set $J = \{1, 2, \ldots, n\}$ of jobs,
- Job j has a processing time p_j,
- Set $M = \{1, 2, \ldots, m\}$ of parallel identical machines.

Objective

- Construct a non-preemptive schedule with minimum makespan $C_{\max}$.

Multiprocessor scheduling – Mixed-integer optimization

$$
\begin{array}{lllll}
\min\limits_{\mathbf{x}, C_{\max}} & C_{\max} & & \\
 & \sum_{j=1}^{n} x_{i,j} \cdot p_j & \leq C_{\max} & i \in M \\
 & \sum_{i=1}^{m} x_{i,j} & = 1 & j \in J \\
 & x_{i,j} \in \{0,1\} & & j \in J, i \in M
\end{array}
$$

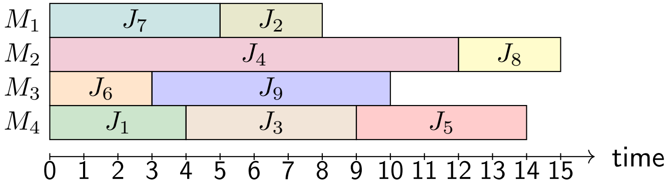
Input

- Set $J = \{1, 2, \ldots, n\}$ of jobs,
- Job j has a processing time p_j,
- Set $M = \{1, 2, \ldots, m\}$ of parallel identical machines.

Objective

- Construct a non-preemptive schedule with minimum makespan $C_{\max}$.

Multiprocessor scheduling – Perturbation types



Groups of perturbations

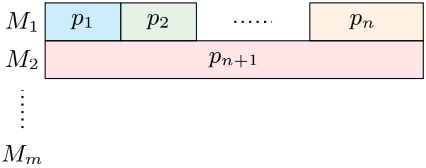
- Machine activation,
- Job arrival, processing time augmentation & machine failure,
- Job removal & processing time reduction,

Hardness of rescheduling

Job removal & processing time reduction

Job removal example

Instance I_{init} has m machines & $n+1$ jobs. One job has processing time $p_{n+1} = \sum_{j=1}^{n} p_j$. Perturb I_{init} by removing job J_{n+1}.

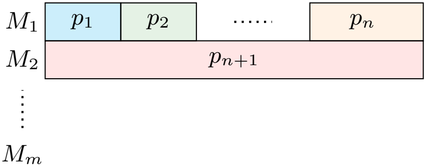


Hardness of rescheduling

Job removal & processing time reduction

Job removal example

Instance I_{init} has m machines & $n+1$ jobs. One job has processing time $p_{n+1} = \sum_{j=1}^{n} p_j$. Perturb I_{init} by removing job J_{n+1}.



Observations

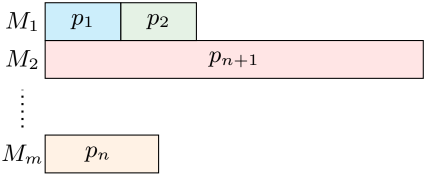
- The recovery problem is strongly $\mathcal{NP}$-hard.
- In a limited recovery setting, the initial schedule is a weak $\Omega(m)$ approximation for the new instance.

Hardness of rescheduling

Job removal & processing time reduction

Job removal example

Instance I_{init} has m machines & $n+1$ jobs. One job has processing time $p_{n+1} = \sum_{j=1}^{n} p_j$. Perturb I_{init} by removing job J_{n+1}.



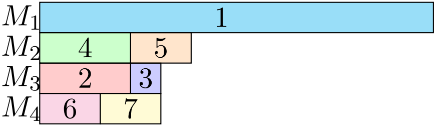
Observations

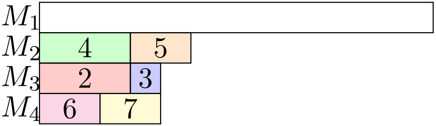
- The recovery problem is strongly $\mathcal{NP}$-hard.
- In a limited recovery setting, the initial schedule is a weak $\Omega(m)$ approximation for the new instance.

Reoptimization Travelling Salesman Problem (R-TSP)

R-TSP remains highly inapproximable even if all optimal solutions of the initial instance are known. *Böckenhauer, Hromkovič, Sprock [2011]*

Importance of lexicographic optimization

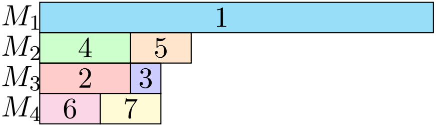


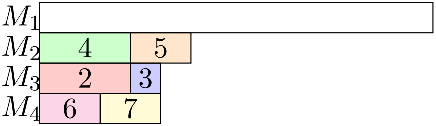


Re-scheduling with a lexicographic optimal schedule

If the initial schedule S_{init} is lexicographic optimal and 1 disturbance occurs, then S_{init} can become a 2-approximate schedule for the new instance.

Importance of lexicographic optimization





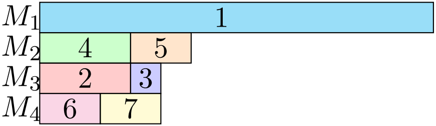
Re-scheduling with a lexicographic optimal schedule

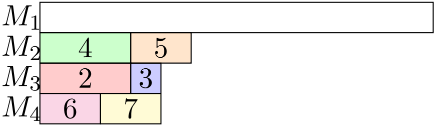
If the initial schedule S_{init} is lexicographic optimal and 1 disturbance occurs, then S_{init} can become a 2-approximate schedule for the new instance.

Lexicographic Optimization Definition

- m objective functions $F_1, F_2, \ldots, F_m$ ordered with respect to priority.
- $F_i : S \to \mathbb{R}_0^+$.
- lex $\min_{x \in S}\{F_1(x), F_2(x), \ldots, F_m(x)\}$computes a solution x^* where:
  - $F_1(x^*) = v_1^* = \min\{F_1(x) : x \in \mathcal{S}\}$, and

Importance of lexicographic optimization





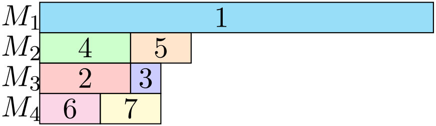
Re-scheduling with a lexicographic optimal schedule

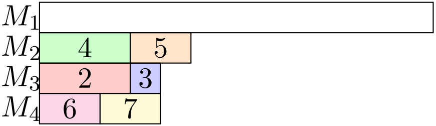
If the initial schedule S_{init} is lexicographic optimal and 1 disturbance occurs, then S_{init} can become a 2-approximate schedule for the new instance.

Lexicographic Optimization Definition

- m objective functions $F_1, F_2, \ldots, F_m$ ordered with respect to priority.
- $F_i : S \rightarrow \mathbb{R}_0^+$.
- lex $\min_{x \in S}\{F_1(x), F_2(x), \ldots, F_m(x)\}$computes a solution x^* where:
  - $F_1(x^*) = v_1^* = \min\{F_1(x) : x \in \mathcal{S}\}$, and
  - $F_2(x^*) = v_2^* = \min\{F_2(x) : x \in \mathcal{S}, F_1(x) = v_1^*\}$, and

Importance of lexicographic optimization





Re-scheduling with a lexicographic optimal schedule

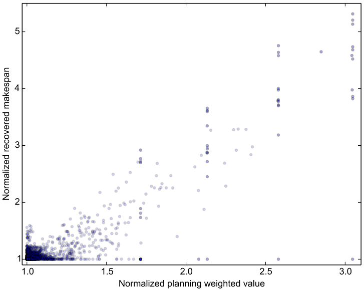
If the initial schedule S_{init} is lexicographic optimal and 1 disturbance occurs, then S_{init} can become a 2-approximate schedule for the new instance.

Lexicographic Optimization Definition

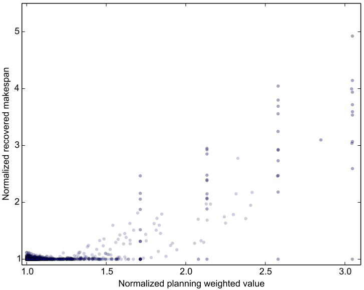
- m objective functions $F_1, F_2, \ldots, F_m$ ordered with respect to priority.
- $F_i : S \to \mathbb{R}_0^+$.
- $\text{lex}\min_{x \in S}\{F_1(x), F_2(x), \ldots, F_m(x)\}$computes a solution x^* where:
  - $F_1(x^*) = v_1^* = \min\{F_1(x) : x \in \mathcal{S}\}$, and
  - $F_2(x^*) = v_2^* = \min\{F_2(x) : x \in \mathcal{S}, F_1(x) = v_1^*\}$, and
  - $F_3(x^*) = v_3^* = \min\{F_3(x) : x \in \mathcal{S}, F_1(x) = v_1^*, F_2(x) = v_2^*\}$, ...

Practical applicability of lexicographic optimization?

Recover feasibility only



Allow limited recovery actions



Test conditions

- Medium instances only,
- Starting with the solution pool of possible heuristic solutions,
- Normalise the initial and recovered schedules.

Royal Mail's van allocation problem


Challenge

- At a delivery office in a morning $\implies$ deliver by afternoon
- 1250 delivery offices
- 37,000 vans; 90,000 drivers; 27 million locations

Letsios, Bradley, Suraj G, Misener & Page, *Journal of Scheduling*, 2021.

Bounded Job Start Scheduling Problem

$$\min_{x_{j,s},\, T} \quad T \tag{1a}$$

$$T \geq x_{j,s}(s + p_j) \qquad j \in \mathcal{J}, s \in D \tag{1b}$$

$$\sum_{j \in \mathcal{J}} \sum_{s \in A_{j,t}} x_{j,s} \leq m \qquad t \in D \tag{1c}$$

$$\sum_{s \in F_j} x_{j,s} = 1 \qquad j \in \mathcal{J} \tag{1d}$$

$$\sum_{j \in \mathcal{J}_s} x_{j,s} \leq g \qquad s \in D \tag{1e}$$

$$x_{j,s} \in \{0, 1\} \qquad j \in \mathcal{J}, s \in F_j \tag{1f}$$

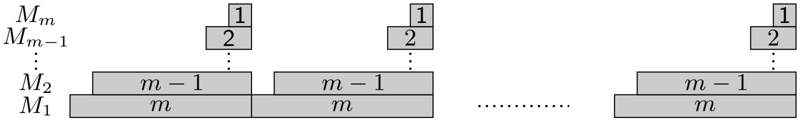
BJSP is strongly $\mathcal{NP}$-hard in the case $g = 1$, reduction to 3-Partition and ...

- Generalize fundamental makespan scheduling, i.e. $P||C_{\max}$,
- Relax forbidden sets scheduling, job subsets can't run in parallel [Schäffter, 1997],
- Relax scheduling with forbidden job start times [Billaut & Sourd, 2009; Rapine & Brauner, 2013; Gabay et al. 2016; Mnich & van Bevern, 2018] .

Comparing solutions

Worst case analysis

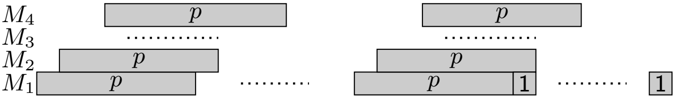
$P||C_{\max}$ optimum may be a factor $\Omega(m)$ from bounded job start optimum



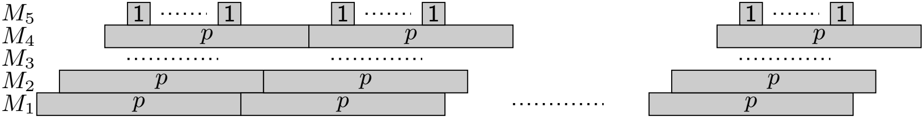
(a) Bounded job start optimal schedule

(b) $P||C_{\max}$ optimal schedule

Longest job processing time won't save us . . .



(a) LPT schedule S



(b) Optimal schedule S^*

Figure: LPT is 2-approximate for minimizing makespan and this ratio is tight.

How to get an approximation ratio better than 2?

Cases when longest job processing time is useful

Instance $\langle m, \mathcal{J} \rangle$ is *long* if $p_j \geq m, \forall\ j \in \mathcal{J}$ and *short* if $p_j < m, \forall\ j \in \mathcal{J}$.

- LPT is 5/3-approximate for long instances,
- LPT is optimal for short instances.

Shortest processing time first [good if $p_{\max}$ smaller than average load]

LSPT is 2-approximate for minimizing makespan. For long instances, LSPT is $(1 + \min\{1, 1/\alpha\})$-approximate, where $\alpha = (\frac{1}{m} \sum_{j \in \mathcal{J}} p_j)/p_{\max}$.

Mixing long & short jobs with machine augmentation

LSM computes a 1.985-approximate schedule with 1.2-machine augmentation by having some machines work with long jobs and some with short jobs. The bad case (needing machine augmentation) is with many very long jobs, i.e. more than $\lceil 5m/6 \rceil$ jobs with $p_j > T^*/2$.

LexOpt Scheduling with Machine Augmentation

$$\min_{x_{j,s},\, v,w} \quad v + \theta \left(\sum_{j,s} x_{j,s} w_{j,s+p_j} \right) \tag{2a}$$

$$v \geq \sum_{j,s} x_{j,s} \qquad j \in \mathcal{J}, s \in D \tag{2b}$$

$$x_{j,s}(s + p_j) \leq D \qquad j \in \mathcal{J}, s \in D \tag{2c}$$

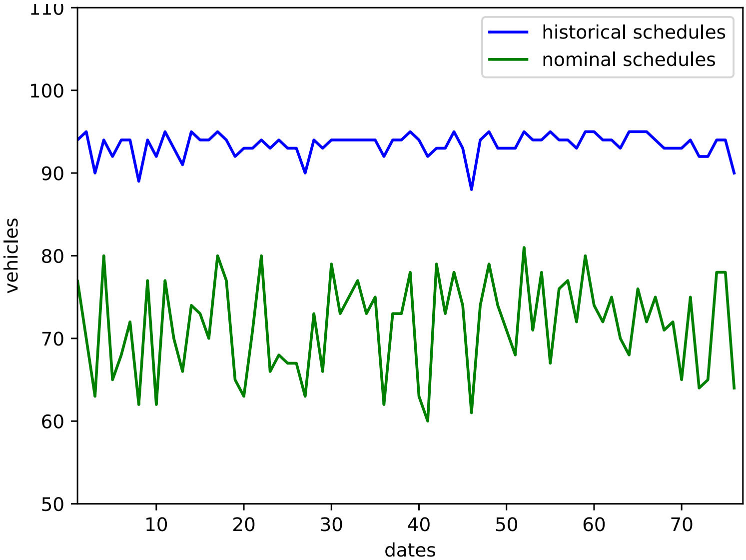
$$\sum_{j \in \mathcal{J}} \sum_{s \in A_{j,t}} x_{j,s} \leq m \qquad t \in \mathcal{D} \tag{2d}$$

$$\sum_{s \in F_j} x_{j,s} = 1 \qquad j \in \mathcal{J} \tag{2e}$$

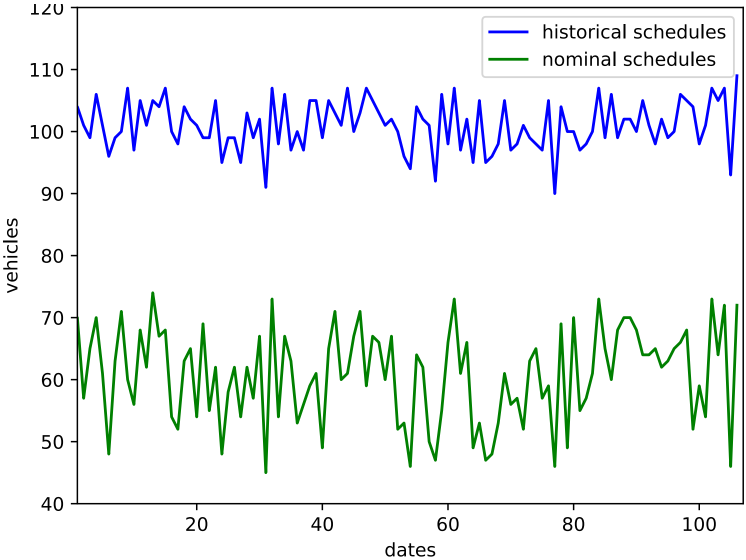
$$\sum_{j \in \mathcal{J}_s} x_{j,s} \leq g \qquad s \in D \tag{2f}$$

$$x_{j,s} \in \{0, 1\} \qquad j \in \mathcal{J}, s \in F_j \tag{2g}$$

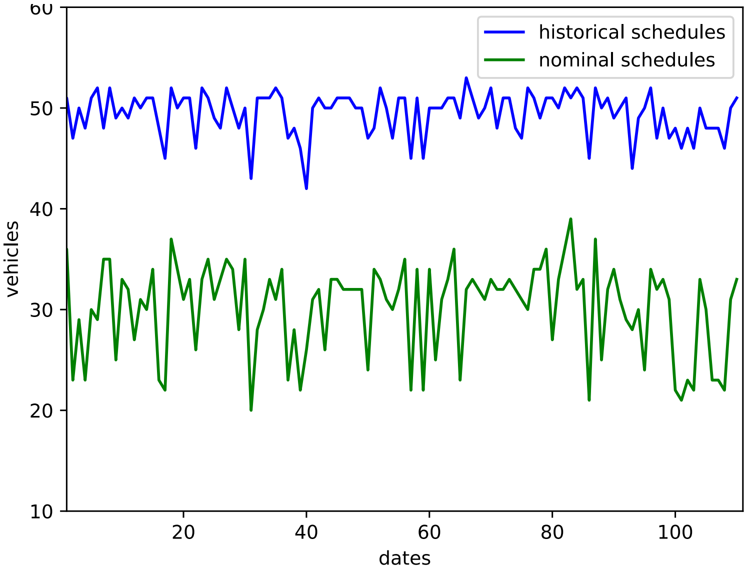
Evaluating historical schedules . . .



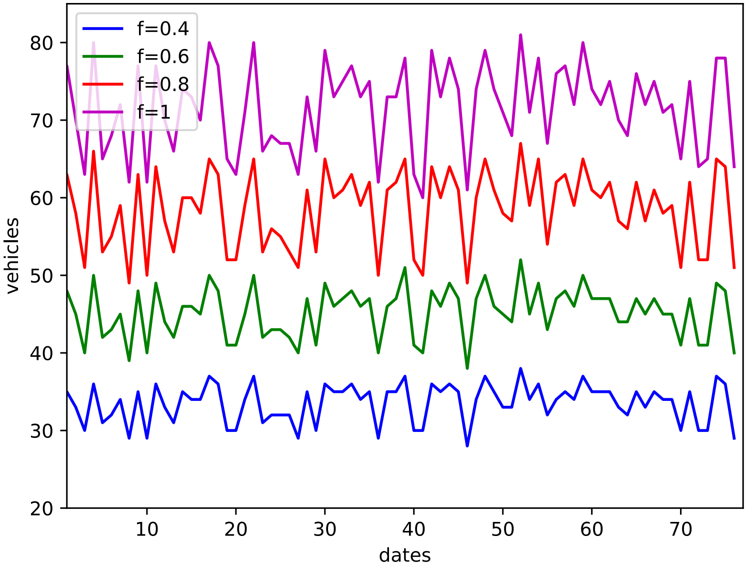
Evaluating historical schedules . . .



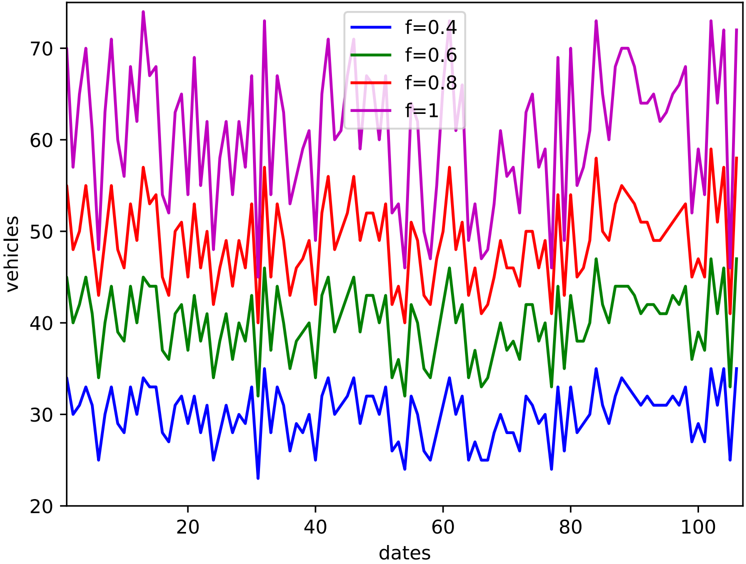
Evaluating historical schedules . . .



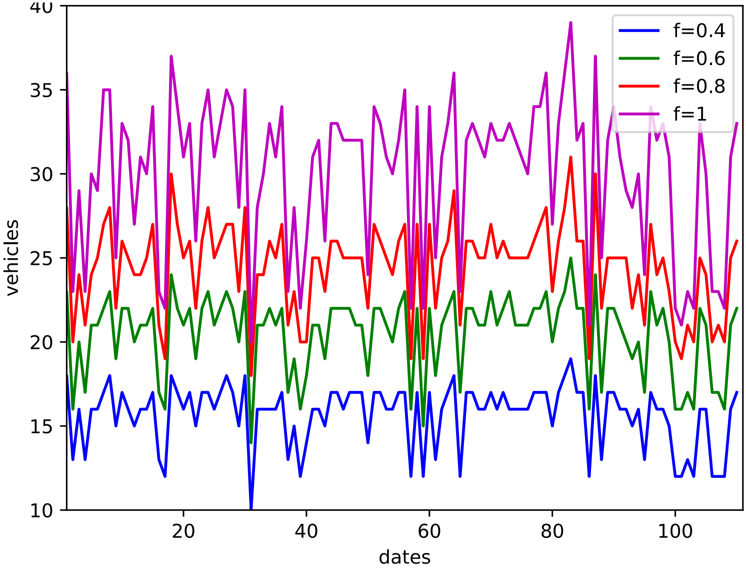
Sensitivity analysis Delivery Office 1



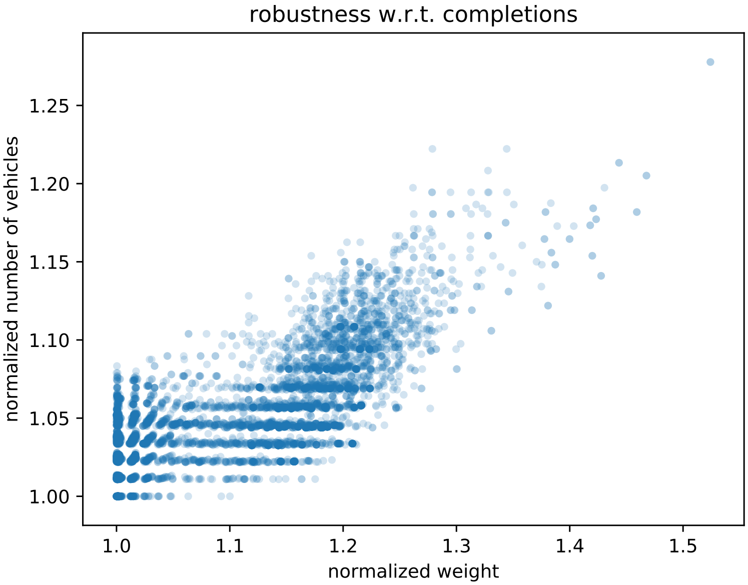
Sensitivity analysis Delivery Office 2



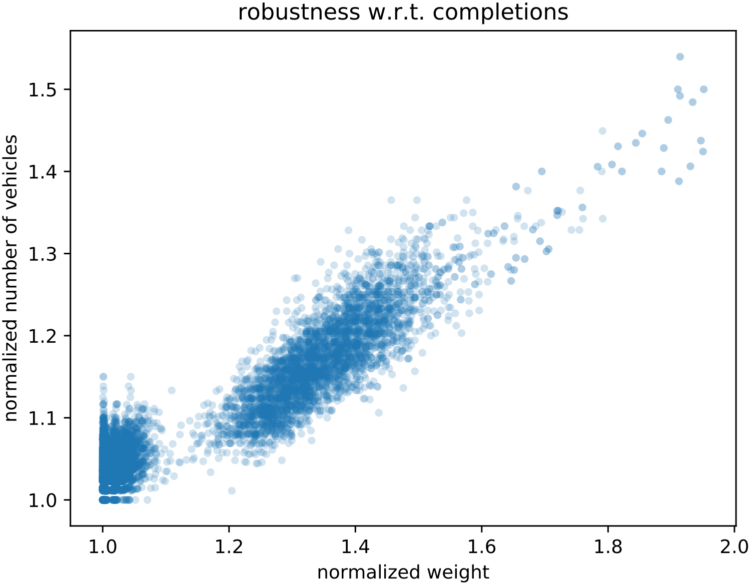
Sensitivity analysis Delivery Office 3



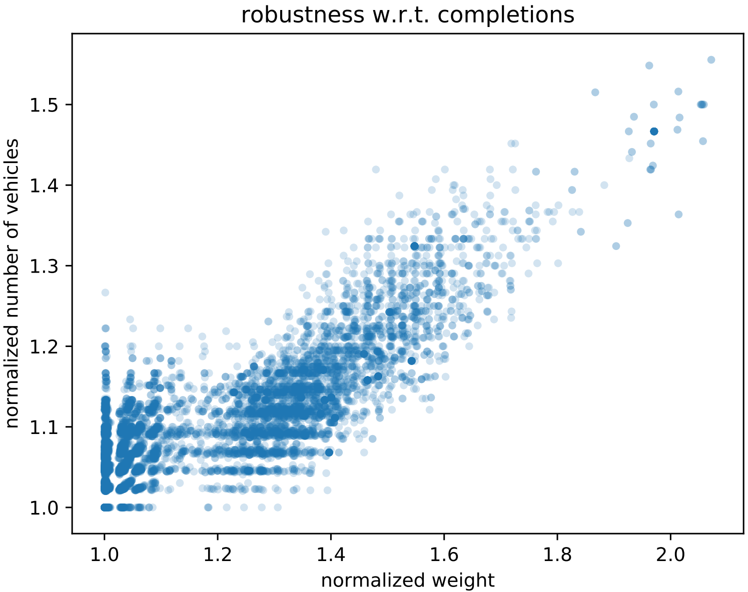
Advantage of lexicographic optimization Delivery Office 1



Advantage of lexicographic optimization Delivery Office 2



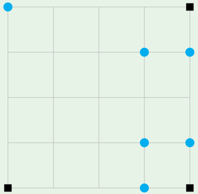
Advantage of lexicographic optimization Delivery Office 3



Extensions to other applications

Facility Location

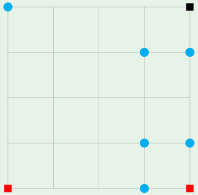
- n customers
- m facility locations
- Open $k < m$ facilities.
- Minimise maximum distance of a customer to its closest facility.



Extensions to other applications

Facility Location

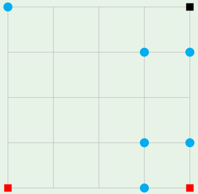
- n customers
- m facility locations
- Open $k < m$ facilities.
- Minimise maximum distance of a customer to its closest facility.



Extensions to other applications

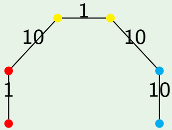
Facility Location

- n customers
- m facility locations
- Open $k < m$ facilities.
- Minimise maximum distance of a customer to its closest facility.



Min-Max Graph Partitioning

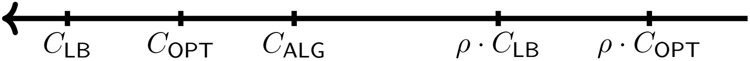
- G: graph with edge weights,
- Partition the vertices into equal-sized subsets,
- Minimise the maximum total weight of the edges leaving a single part.



Commonalities:

- Partitioning problems with a cost tied to each partition component,
- Min-max problems.

Practical applicability of approximation algorithms?

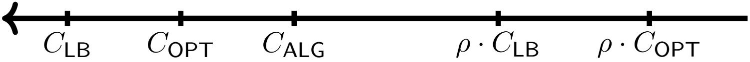


Useful for process systems engineering?

- Important optimization problems in PSE applications
  - Heat recovery networks
  - State-task network
  - Pooling problem
- **Recovery & reoptimization**
  - **Royal Mail van allocation**
- Explainable scheduling

Čyras, Letsios, Misener & Toni, *AAAI* [oral], 2019.

Practical applicability of approximation algorithms?



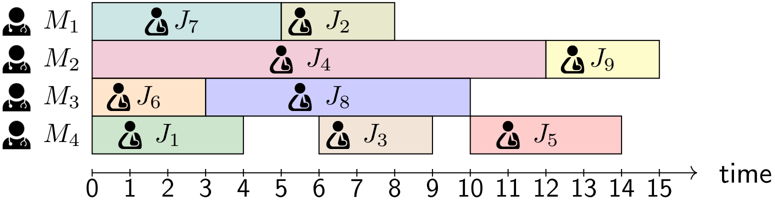
Useful for process systems engineering?

- Important optimization problems in PSE applications
  - Heat recovery networks
  - State-task network
  - Pooling problem
- Recovery & reoptimization
  - Royal Mail van allocation
- **Explainable scheduling**

Čyras, Letsios, Misener & Toni, *AAAI* [oral], 2019.

Makespan scheduling

Example: nurse rostering



Input

- Set $J = \{J_1, J_2, \ldots, J_n\}$ of jobs
- Job J_j has a processing time p_j
- Set $M = \{M_1, M_2, \ldots, M_m\}$ of machines

Objective

- Construct a schedule S with minimum makespan ($\mathcal{NP}$-hard)

Challenge: Explain this to a nurse!

Why am I
to do job j?

$$\min_{\mathbf{x}, C_{\max}} \quad C_{\max}$$

$$\star \quad \begin{array}{lll} \sum_{j=1}^{n} x_{i,j} \cdot p_j & \leq C_{\max} & i \in M \\ \sum_{i=1}^{m} x_{i,j} & = 1 & j \in J \\ x_{i,j} \in \{0,1\} & & j \in J, i \in M \end{array}$$

Schedule S is efficient iff

- Feasible $\star$
- No job can be moved from a busiest machine: $C_i - C_{i'} \leqslant p_j$
- No jobs can be exchanged with any busiest machine: for $j' \neq j$ with $x_{i',j'} = 1$, if $p_j > p_{j'}$, then $C_i + p_{j'} \leqslant C_{i'} + p_j$

for any $j \in J$ such that $x_{i,j} = 1$ and $C_i = C_{\max}$.

Explanation desiderata

Cognitive tractability

Explanations pertaining to schedule S are concise (polynomial in size)

Computational tractability

Explaining whether and why schedule S is (not) good can be performed efficiently (in polynomial time)

Soundness & completeness

Given schedule S, there exists an explanation why S is (not) good **iff** S is (not) good

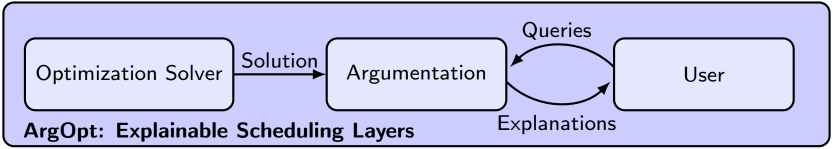
Build an interpretive model for classification

Dash, Günlük & Wei. Boolean decision rules via column generation. Advances in Neural Information Processing Systems (NeurIPS). 2018.

ArgOpt: Argumentation-Optimization

Argumentation

Explainable abstraction paradigm for reasoning with incomplete and conflicting information



Explanations with respect to

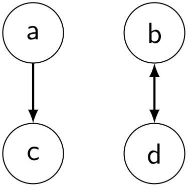
- Schedules from the optimization solver
- Schedules from the user

Argumentation

Argumentation framework

Directed graph with:

- nodes – *arguments*
- edges – *attacks*

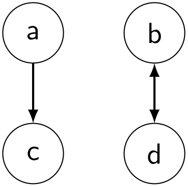


Argumentation

Argumentation framework

Directed graph with:

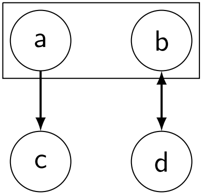
- nodes – *arguments*
- edges – *attacks*



Stable extension of AF

A set S of arguments such that:

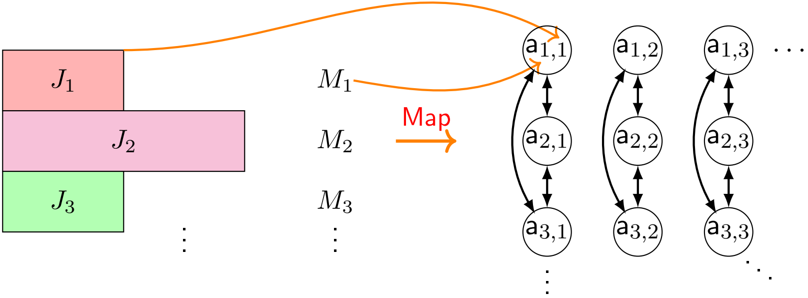
- no attacks between arguments in S
    - internally consistent (*conflict-free*)
- attacks all arguments not in S
    - externally aggressive, global



{a, b} is stable
(so is {a, d})

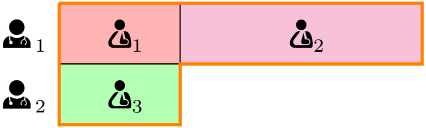
Mapping makespan scheduling argumentation frameworks

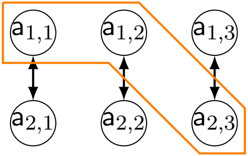
An argumentation framework models decisions with arguments, and incompatibilities with attacks:



- Assignments $x_{i,j}$ become arguments $a_{i,j}$
- $a_{i,j}$ attacks $a_{k,l}$ iff $i \neq k$ and $j = l$
  - Different machines compete for the same job
- Stable extensions are 'good' schedules

Nurse: Can I do this?

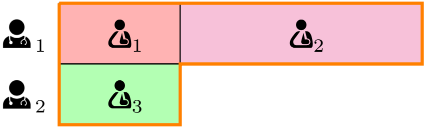


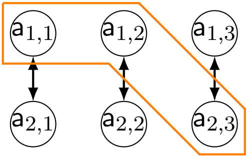


Feasible...

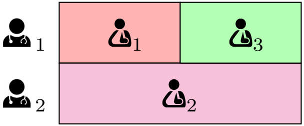
Stable

Nurse: Can I do this?

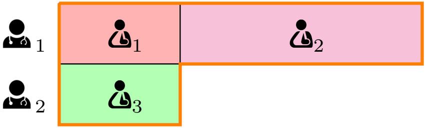


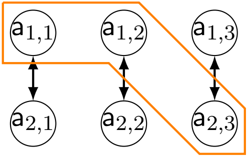


But not efficient! Swap jobs.

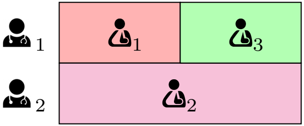


Nurse: Can I do this?

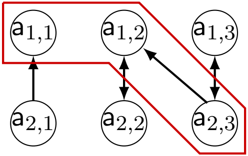




But not efficient! Swap jobs.



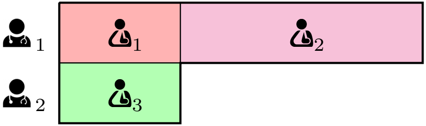
Not stable

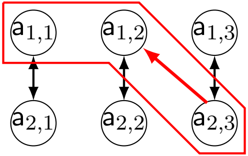


An attacked argument that does not counter-attack represents an inefficient allocation!

Natural language explanations

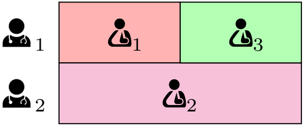
Natural language explanations extracted from AFs



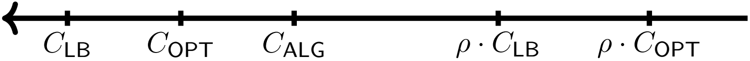


The attack from $a_{2,3}$ to $a_{1,2}$ explains why $S \approx \{a_{1,1}, a_{1,2}, a_{2,3}\}$ is not efficient:

> *Because S can be improved by swapping jobs 3 and 2 between nurses 2 and 1.*



Practical applicability of approximation algorithms?



Useful for process systems engineering?

- Important optimization problems in PSE applications
  - Heat recovery networks
  - State-task network
  - Pooling problem
- Recovery & reoptimization
  - Royal Mail van allocation
- Explainable scheduling

Lexicographic optimal scheduling

MILP reformulation

- Machines ordered in non-increasing order of completion times.
- Completion time bound strengthening constraints.

$$
\begin{aligned}
\text{lex min}_{\mathbf{x},\mathbf{C}} \quad & \{C_1, C_2, \ldots, C_m\} & \\
C_i & \geq C_{i+1} & i \in M \setminus m \\
C_i & \geq \frac{1}{m-i+1}\left(\sum_{j=1}^{n} p_j - \sum_{i'=1}^{i-1} C_{i'}\right) & i \in M \\
C_i & = \sum_{j=1}^{n} x_{i,j} \cdot p_j & i \in M \\
\sum_{i=1}^{m} x_{i,j} & = 1 & j \in J \\
x_{i,j} & \in \{0,1\} & j \in J, i \in M
\end{aligned}
$$

Input

- Set $J = \{1, 2, \ldots, n\}$ of jobs,
- Job j has a processing time p_j,
- Set $M = \{1, 2, \ldots, m\}$ of parallel machines with completion time C_i.

State-of-the-art lexicographic optimization methods

Sequential method

- $v_1^* = \min\{F_1(\vec{x}, \vec{C}) : (\vec{x}, \vec{C}) \in \mathcal{S}\}$.
- For $i = 2, \ldots, m$,
  - $v_i^* = \min\{F_i(\vec{x}, \vec{C}) : x \in S, F_1(\vec{x}, \vec{C}) = v_1^*, \ldots F_{i\text{-}1}(\vec{x}, \vec{C}) = v_{i\text{-}1}^*\}$
- Return the last computed solution.

State-of-the-art lexicographic optimization methods

Sequential method

- $v_1^* = \min\{F_1(\vec{x}, \vec{C}) : (\vec{x}, \vec{C}) \in \mathcal{S}\}$.
- For $i = 2, \ldots, m$,
  - $v_i^* = \min\{F_i(\vec{x}, \vec{C}) : x \in S, F_1(\vec{x}, \vec{C}) = v_1^*, \ldots F_{i\text{-}1}(\vec{x}, \vec{C}) = v_{i\text{-}1}^*\}$
- Return the last computed solution.

Simultaneous (highest rank objective) method

- Solve $v_1^* = \min\{C_1 : (\vec{x}, \vec{C}) \in \mathcal{S}\}$.
- Compute the solution pool $\mathcal{P} = \{(\vec{x}, \vec{C}) \in S : C_1 = v_1^*\}$.
- Return the lexicographically smallest solution in $\mathcal{P}$.

State-of-the-art lexicographic optimization methods

Sequential method

- $v_1^* = \min\{F_1(\vec{x},\vec{C}) : (\vec{x},\vec{C}) \in \mathcal{S}\}$.
- For $i = 2,\ldots,m$,
  - $v_i^* = \min\{F_i(\vec{x},\vec{C}) : x \in S, F_1(\vec{x},\vec{C}) = v_1^*, \ldots F_{i\text{-}1}(\vec{x},\vec{C}) = v_{i\text{-}1}^*\}$
- Return the last computed solution.

Simultaneous (highest rank objective) method

- Solve $v_1^* = \min\{C_1 : (\vec{x},\vec{C}) \in \mathcal{S}\}$.
- Compute the solution pool $\mathcal{P} = \{(\vec{x},\vec{C}) \in S : C_1 = v_1^*\}$.
- Return the lexicographically smallest solution in $\mathcal{P}$.

Weighting method

- Set big-M parameter $M = 2$.
- For $i = 2,\ldots,m$, set machine weight $w_i = M^{m-i}$.
- Solve $\min\{\sum_{i=1}^{m} w_i \cdot C_i : (\vec{x},\vec{C}) \in \mathcal{S}\}$.

Novel bounding technique

Can we develop methodology for bounding the best solution?

Let's develop strong lexicographic optimization lower bounding technique to solve the lex optimization problem exactly.

Novel bounding technique

Can we develop methodology for bounding the best solution?

Let's develop strong lexicographic optimization lower bounding technique to solve the lex optimization problem exactly.

Vectorial lower bound of schedule S

- A vector $\vec{L} = (L_1, \ldots, L_m)$, s.t. $L_i \leq C_i(S)$, for all $i = 1, 2, \ldots, m$ (both vectors $\vec{L}$ and $\vec{C}(S)$ are sorted in non-increasing order).

Novel bounding technique

Can we develop methodology for bounding the best solution?

Let's develop strong lexicographic optimization lower bounding technique to solve the lex optimization problem exactly.

Vectorial lower bound of schedule S

- A vector $\vec{L} = (L_1, \ldots, L_m)$, s.t. $L_i \leq C_i(S)$, for all $i = 1, 2, \ldots, m$ (both vectors $\vec{L}$ and $\vec{C}(S)$ are sorted in non-increasing order).

Vectorial bounds may enforce exact, branch-and-cut methods

- Better convergence to efficient solutions,
- Improved global optimality proving.

Lexicographic branch-and-bound method

Branch-and-bound ingredients

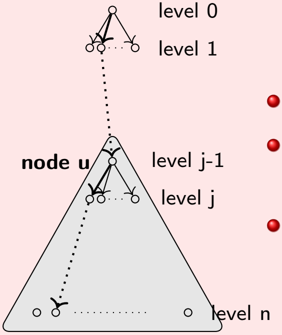
- Sort the jobs $p_1 \geq \ldots \geq p_n$,
- Search a tree with $n+1$ levels. Level ℓ has assigned jobs $J_1, \ldots, J_\ell$,
- Depth first search.

Lexicographic branch-and-bound method

Branch-and-bound ingredients

- Sort the jobs $p_1 \geq \ldots \geq p_n$,
- Search a tree with $n+1$ levels. Level ℓ has assigned jobs $J_1, \ldots, J_\ell$,
- Depth first search.

Pruning using vectorial lower bounds



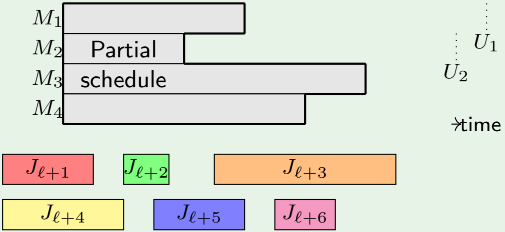
S(u): set of all schedules below node **u**

- S_{inc}: Best found (*incumbent*) solution,
- At node u, compute a vectorial lower bound $\vec{L}(u)$ of the lex best schedule in $\mathcal{S}(u)$,
- If $\vec{C}(S_{inc}) \leq_{\text{lex}} \vec{L}(u)$, then prune the subtree.

Vectorial lower bound computation

- In our concrete scheduling problem:
  - Approximate scheduling problem with job rejections,
  - Use knapsack-like bounding approaches,
  - Equivalent to constructing a pseudo-schedule which is feasible except that some jobs are scheduled fractionally.

L_3 Computation



Vectorial lower bound computation [cont.]

Computation of the k-th component of vectorial lower bound

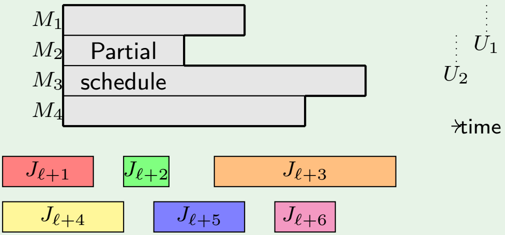
1: Select job index $q = \min\{j : \sum_{j'=\ell+1}^{j} p_{j'} \geq \sum_{i=1}^{k-1}(U_i - t_i)\}$.

2: Compute remaining load $\lambda = \sum_{j=q+1}^{n} p_j$.

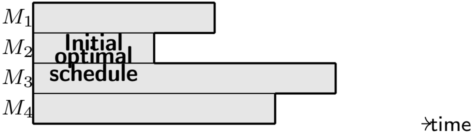
3: Return the maximum among:

- $\min_{k \leq i \leq m}\{t_i\} + p_{q+1}$, and
- $\max_{k \leq i \leq m}\{t_i\} + \max\left\{\frac{1}{m-k+1}\left(\lambda - \sum_{i=k+1}^{m}(\max_{k \leq i \leq m}\{t_i\} - t_i)\right), 0\right\}$.

L_3 Computation

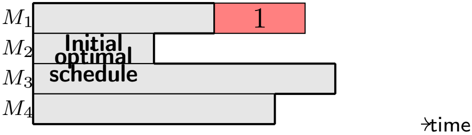


Longest-processing time first heuristic – Add k new jobs



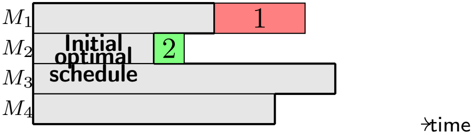
1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

Longest-processing time first heuristic – Add k new jobs



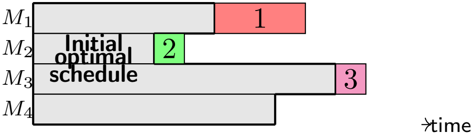
1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

Longest-processing time first heuristic – Add k new jobs



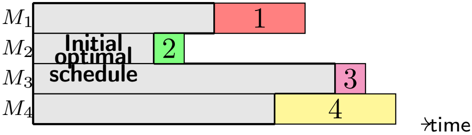
1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

Longest-processing time first heuristic – Add k new jobs



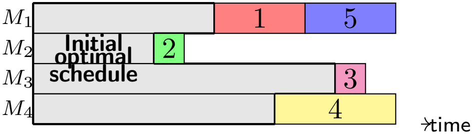
1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

Longest-processing time first heuristic – Add k new jobs



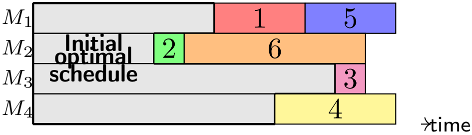
1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

Longest-processing time first heuristic – Add k new jobs



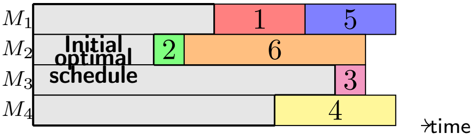
1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

Longest-processing time first heuristic – Add k new jobs



1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

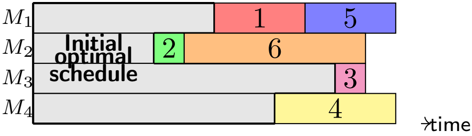
Longest-processing time first heuristic – Add k new jobs



1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

- If $k \leq m$, then it is 2-approximate.
- If k is large, then the approximation ratio can be arbitrarily bad.

Longest-processing time first heuristic – Add k new jobs



1) **Round-robin** algorithm is $O(k)$-time $\left(1 + \left\lceil \frac{k}{m} \right\rceil\right)$-approximate.

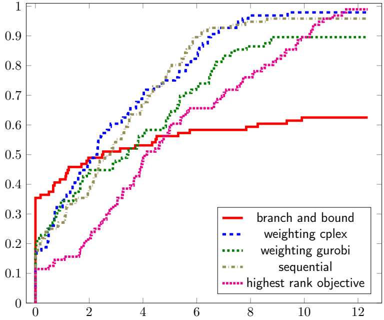
- If $k \leq m$, then it is 2-approximate.
- If k is large, then the approximation ratio can be arbitrarily bad.

2) **List scheduling** algorithm is $O(k \log m)$-time **2-approximate (tight)**.

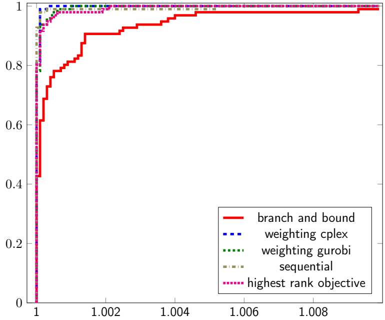
- Sort the jobs $p_1 \geq \ldots \geq p_n$,
- Schedule next job to machine with lowest current completion time.

Numerical results: Moderate test set

Elapsed times on $\log_2$ scale



Upper bounds on $[1, 1.01]$

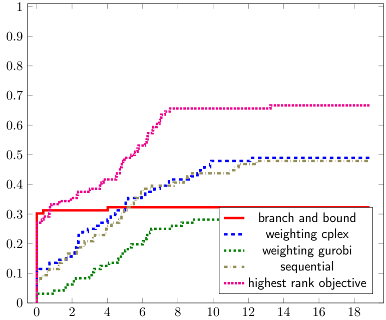


Random instance set-up & solution termination criteria

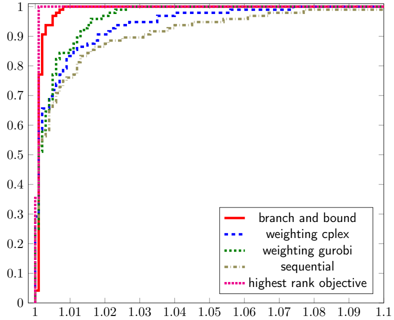
Number of machines	$3, 4, 5, 6$
Number of jobs	$20, 30, 40, 50$
Processing time parameter	$100, 1000$
Processing time distributions	Uniform, normal, symmetric normal
Relative error	0.0001
Time limit	10^4 seconds

Numerical results: Hard test set

Elapsed times on $\log_2$ scale



Upper bounds on $[1, 1.08]$

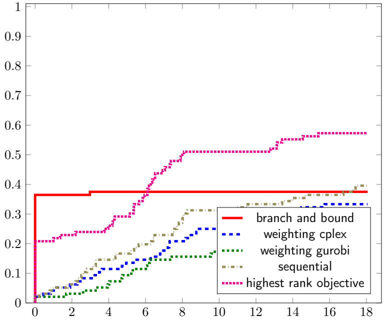


Random instance set-up & solution termination criteria

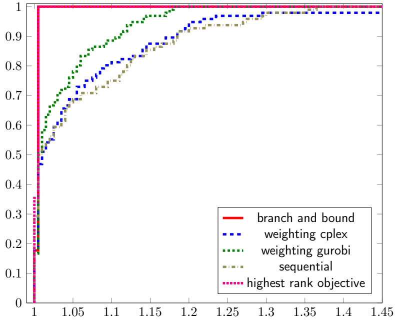
Number of machines	$10, 12, 14, 16$
Number of jobs	$100, 200, 300, 400$
Processing time parameter	$1000, 10000$
Processing time distributions	Uniform, normal, symmetric normal
Relative error	0.0001
Time limit	10^4 seconds

Numerical results: Challenging test set

Elapsed times on $\log_2$ scale



Upper bounds on $[1, 1.4]$



Random instance set-up & solution termination criteria

Number of machines	$10, 15, 20, 25$
Number of jobs	$200, 300, 400, 500$
Processing time parameter	$1000, 10000$
Processing time distributions	Uniform, normal, symmetric normal
Relative error	0.0001
Time limit	10^4 seconds