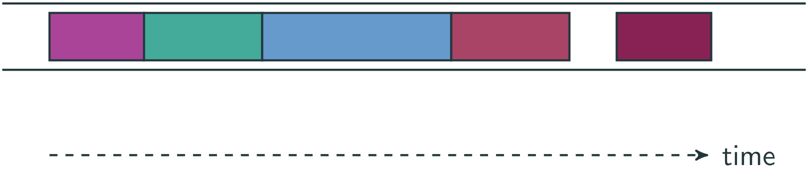
Recent Advances in Flow Time Scheduling

Lars Rohwedder

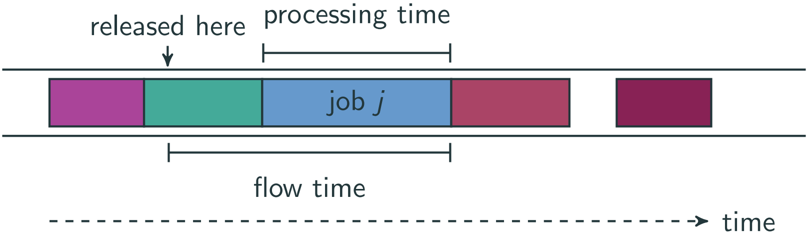
Maastricht University

Scheduling Seminar

flow time = time between arrival/release and completion



flow time = time between arrival/release and completion

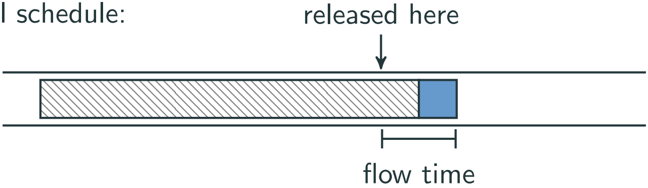


A few reasons why you should be interested:

- natural and classic measure of quality-of-service in scheduling
- related to completion time minimization
  (completion time = flow time if job released at zero)
- intuitively should be much harder than optimizing completion time, but there are surprises

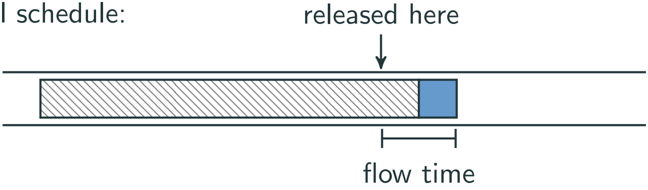
Why it seems harder

optimal schedule:

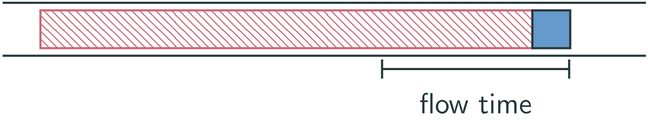


Why it seems harder

optimal schedule:

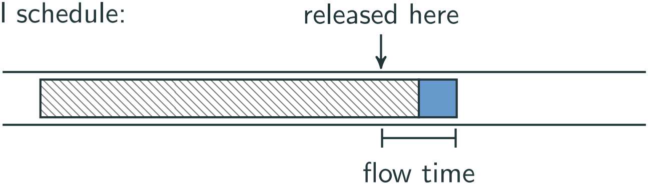


slightly suboptimal schedule:

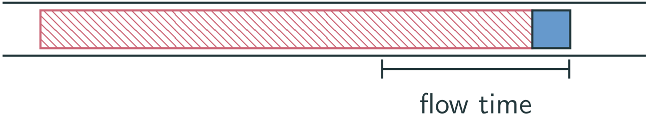


Why it seems harder

optimal schedule:



slightly suboptimal schedule:



$\rightsquigarrow$ breaks many known techniques

Three concrete examples

Fundamental problems in flow time scheduling

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$	NP-hard,	$\leq O(1)$	
$P \mid r_j \mid F_{\max}$	NP-hard,	≤ 3	(state: 3 years ago)
$R \mid r_j \mid F_{\max}$	≥ 1.5,	$\leq O(\log n)$	

Related problems considering completion time

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j C_j$	NP-hard,	PTAS
$P \mid\mid C_{\max}$	NP-hard,	PTAS
$R \mid\mid C_{\max}$	≥ 1.5,	≤ 2

Three concrete examples

Fundamental problems in flow time scheduling

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$	NP-hard,	PTAS
$P \mid r_j \mid F_{\max}$	NP-hard,	≤ 3
$R \mid r_j \mid F_{\max}$	≥ 1.5,	$\leq O(\log n)$

Related problems considering completion time

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j C_j$	NP-hard,	PTAS
$P \mid\mid C_{\max}$	NP-hard,	PTAS
$R \mid\mid C_{\max}$	≥ 1.5,	≤ 2

Three concrete examples

Fundamental problems in flow time scheduling

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$	NP-hard,	PTAS
$P \mid r_j \mid F_{\max}$	NP-hard,	≤ 3
$R \mid r_j \mid F_{\max}$	≥ 1.5,	$\leq O(\sqrt{\log n})$*

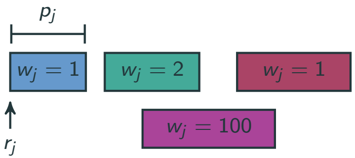
Related problems considering completion time

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j C_j$	NP-hard,	PTAS
$P \mid\mid C_{\max}$	NP-hard,	PTAS
$R \mid\mid C_{\max}$	≥ 1.5,	≤ 2

* non-constructive integrality gap

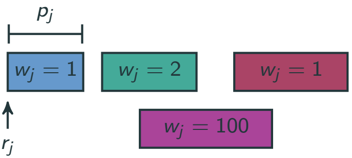
$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$

jobs:



$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$

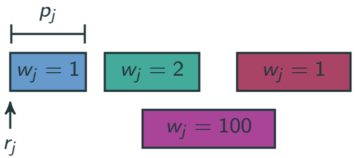
jobs:



schedule:

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$

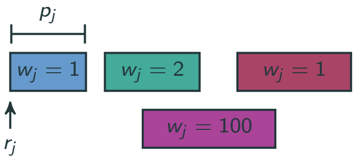
jobs:



schedule:

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$

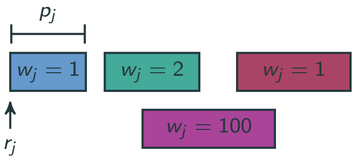
jobs:



schedule:

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$

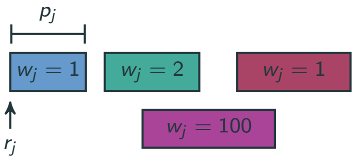
jobs:



schedule:

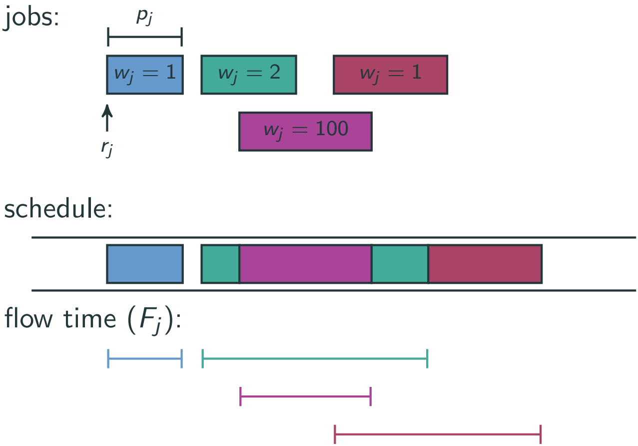
$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$

jobs:



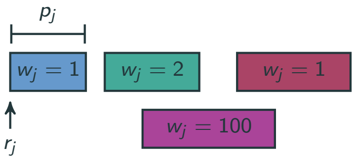
schedule:

$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$



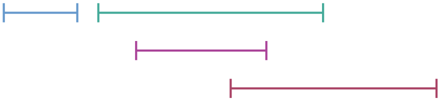
$1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$

jobs:



schedule:

flow time (F_j):



objective:

minimize $\sum_j w_j F_j$

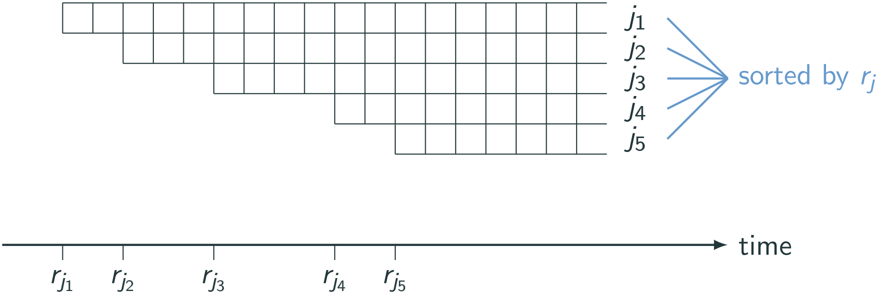
- Batra, Garg, Kumar (STOC'18): Framework and first $O(1)$-approximation (pseudo-poly. time)
- Feige, Kulkarni, Li (SODA'19): Assume w.l.o.g. that input numbers are polynomially bounded
- R., Wiese (STOC'21): $(2+\epsilon)$-approximation
- Armbruster, R., Wiese (unpublished): PTAS

Covering integer program

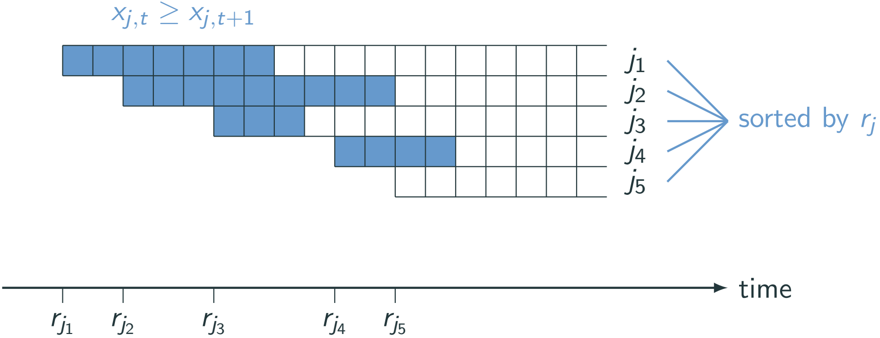
$$x_{j,t} \equiv \text{job } j \text{ has not finished at time } t$$

$$\begin{aligned}
&\min \sum_{j\in J}\sum_{t\geq r_j} w_j x_{j,t} \\
&\sum_{\substack{j\in J\\ s\leq r_j\leq t}} x_{j,t}\cdot p_j \geq \sum_{\substack{j\in J\\ s\leq r_j\leq t}} p_j - (t-s) && \forall s\leq t \\
&x_{j,t} \geq x_{j,t+1} && \forall j\in J, t>r_j \\
&x_{j,t} \in \{0,1\} && \forall j\in J, t\in\{r_j, r_j+1,\dots\}
\end{aligned}$$

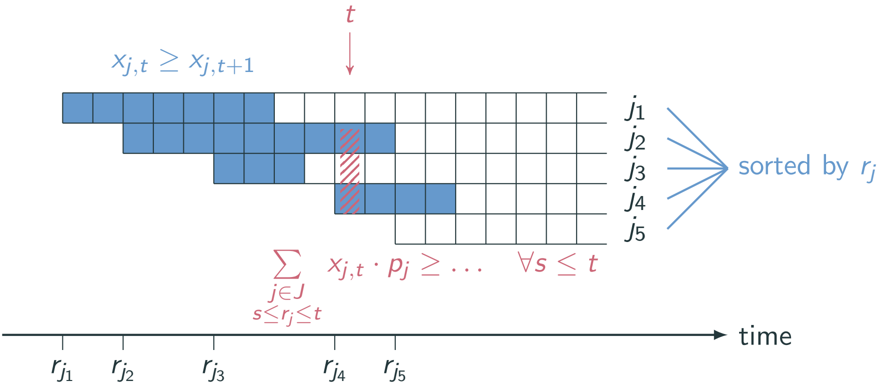
From a different perspective



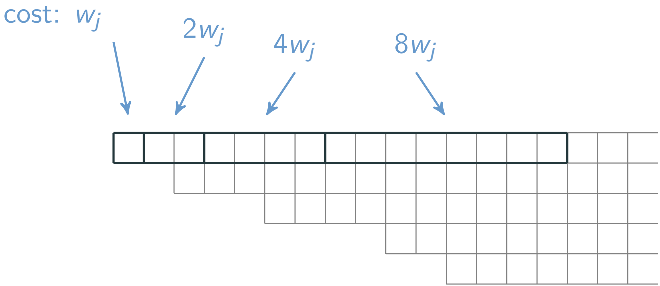
From a different perspective



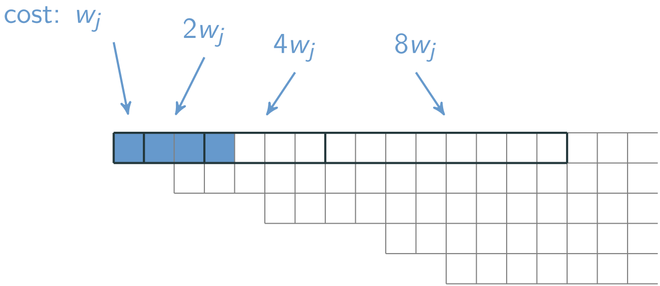
From a different perspective



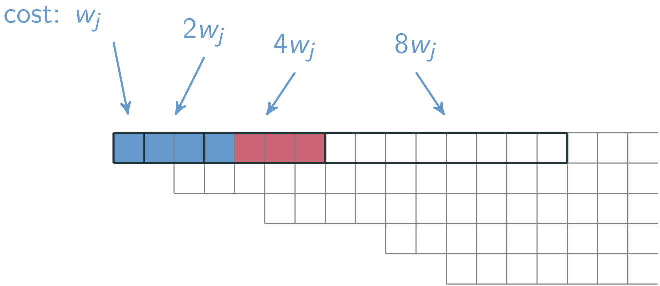
Avoiding the prefix constraint



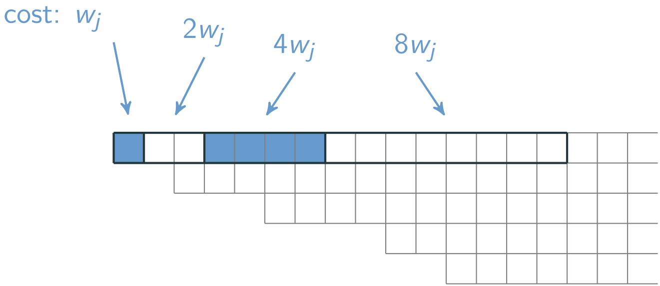
Avoiding the prefix constraint



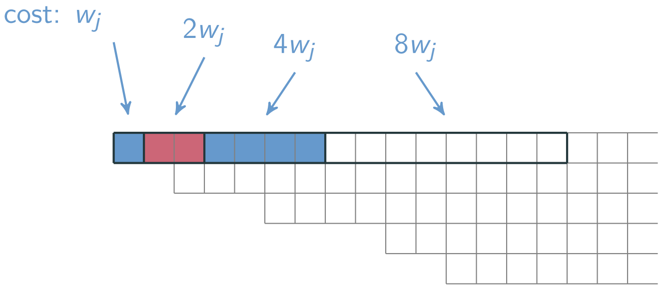
Avoiding the prefix constraint



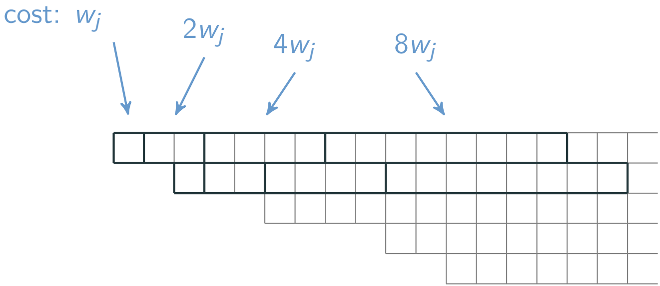
Avoiding the prefix constraint



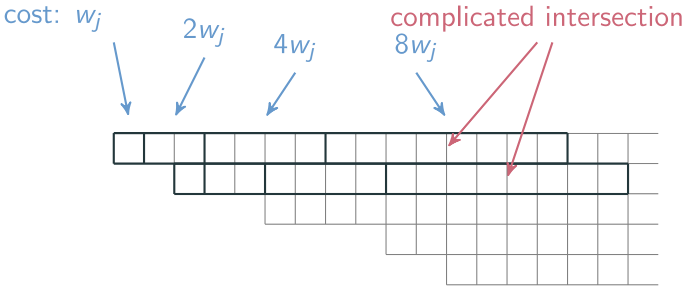
Avoiding the prefix constraint



Avoiding the prefix constraint

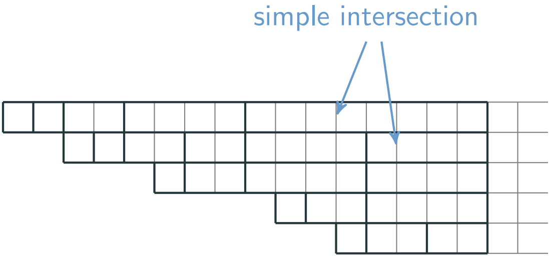


Avoiding the prefix constraint



Avoiding the prefix constraint (cont'd)

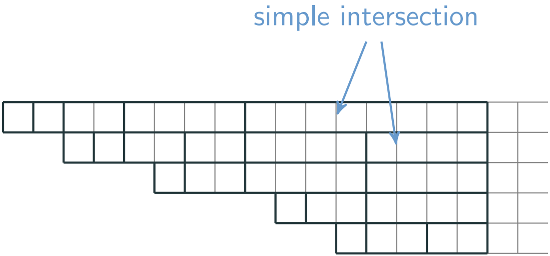
Using more involved (still exponentially growing) grouping.



$\rightsquigarrow$ loses factor 32

Avoiding the prefix constraint (cont'd)

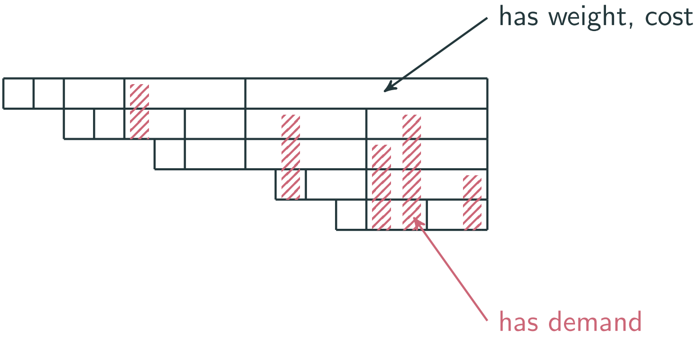
Using more involved (still exponentially growing) grouping.



$\rightsquigarrow$ loses factor 32

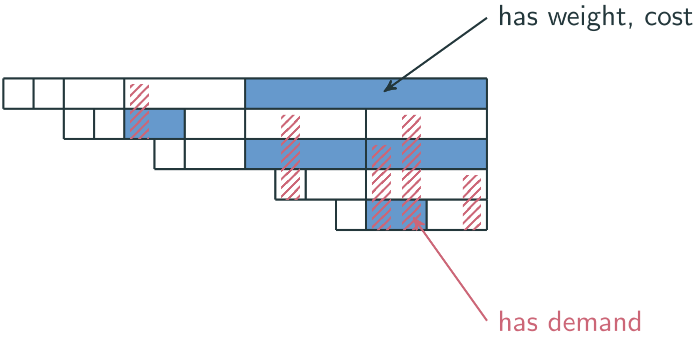
can be reduced to $(1 + \epsilon)$ with some extra technicalities.

Recap



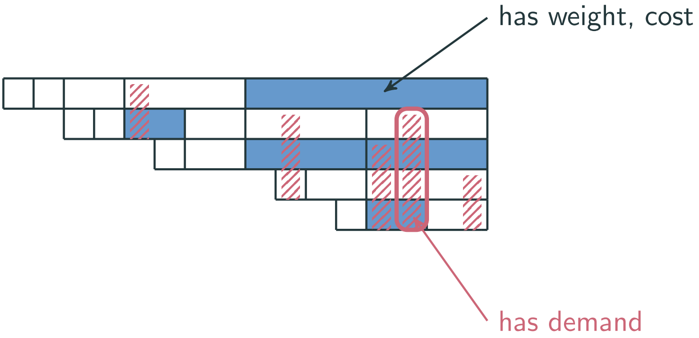
- Given hierarchically aligned rectangles
- Select a subset of rectangles of minimal cost
- Such that all demands are covered

Recap



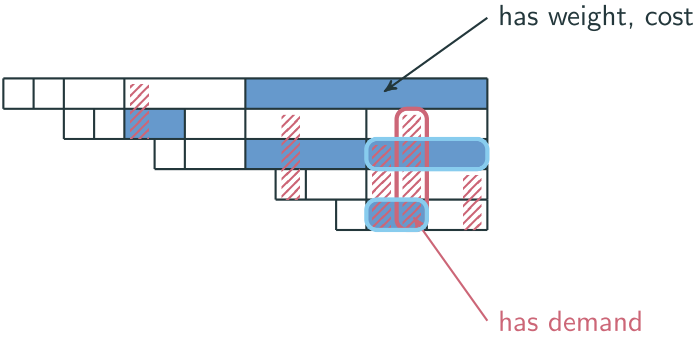
- Given hierarchically aligned rectangles
- Select a subset of rectangles of minimal cost
- Such that all demands are covered

Recap



- Given hierarchically aligned rectangles
- Select a subset of rectangles of minimal cost
- Such that all demands are covered

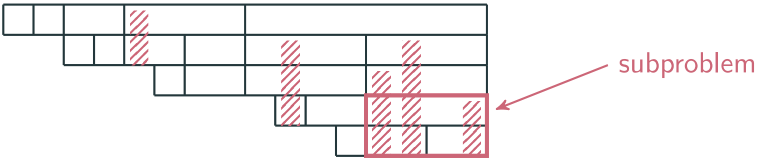
Recap



- Given hierarchically aligned rectangles
- Select a subset of rectangles of minimal cost
- Such that all demands are covered

Dynamic program

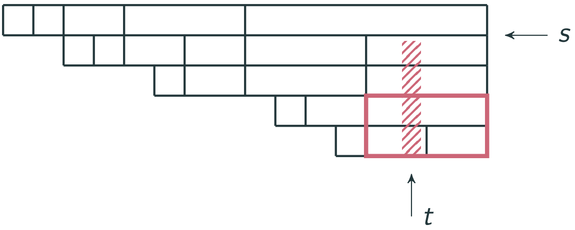
The hierarchical alignment allows for dynamic programming.



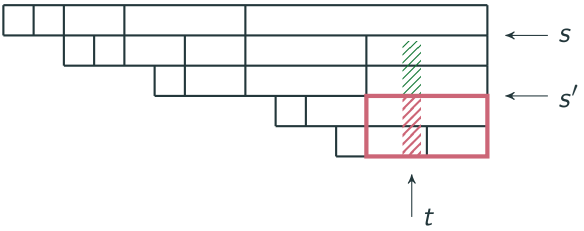
What do we need to know from the rest of the solution to determine the optimal solution of the subproblem?

Naive: how much is already covered for all demand rays that intersect the subproblem. $\rightsquigarrow$ too much information.

A closer look at the demands

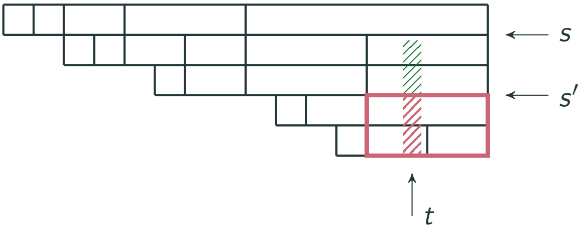


$$\sum_{\substack{j\in J: s\le r_j\le t\\ \text{rect. selected}}} p_j \ge \sum_{j\in J: s\le r_j \le t} p_j - (t-s)$$



$$\underset{\text{rect. selected}}{\sum_{j\in J:s\le r_j<s'} p_j} + \underset{\text{rect. selected}}{\sum_{j\in J:s'\le r_j\le t} p_j} \ge \sum_{j\in J:s\le r_j<s'} p_j - (s'-s) + \sum_{j\in J:s'\le r_j\le t} p_j - (t-s')$$

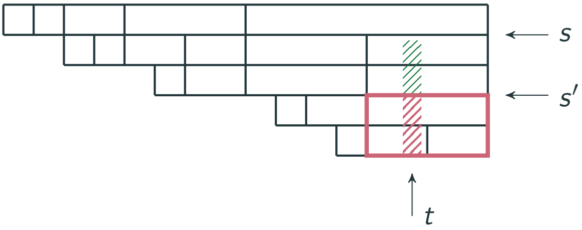
A closer look at the demands



$$\underbrace{\sum_{j \in J : s \le r_j < s'} p_j}_{\text{rect. selected}} + \underbrace{\sum_{j \in J : s' \le r_j \le t} p_j}_{\text{rect. selected}} \ge \sum_{j \in J : s \le r_j < s'} p_j - (s' - s) + \sum_{j \in J : s' \le r_j \le t} p_j - (t - s')$$

Fixing the solution outside the subproblem, we only need to satisfy constraint for s', t with a certain extra slack (same for all t)

A closer look at the demands



$$\underbrace{\sum_{j \in J : s \le r_j < s'} p_j}_{\text{rect. selected}} + \underbrace{\sum_{j \in J : s' \le r_j \le t} p_j}_{\text{rect. selected}} \ge \sum_{j \in J : s \le r_j < s'} p_j - (s' - s) + \sum_{j \in J : s' \le r_j \le t} p_j - (t - s')$$

Fixing the solution outside the subproblem, we only need to satisfy constraint for s', t with a certain extra slack (same for all t)

$\rightsquigarrow$ in DP "guess" the extra slack.

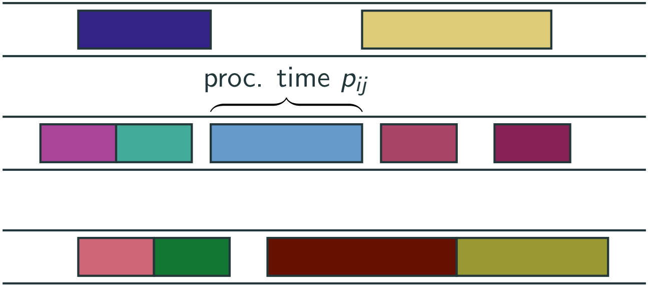
Summary

- Problem can be solved in pseudopolynomial time using DP over a tree + structural insights.
- Reduction not lossless, but error can be made $(1+\epsilon)$ using additional technical work

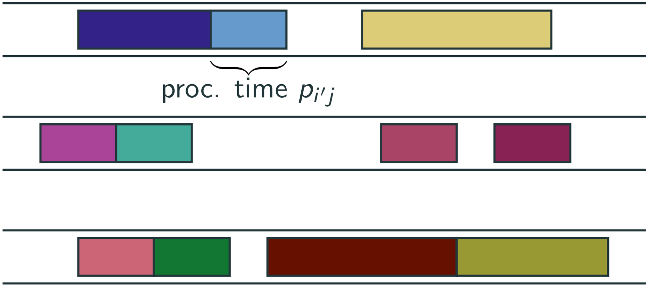
$\rightsquigarrow$ PTAS for sum of weighted flow time on a single machine.

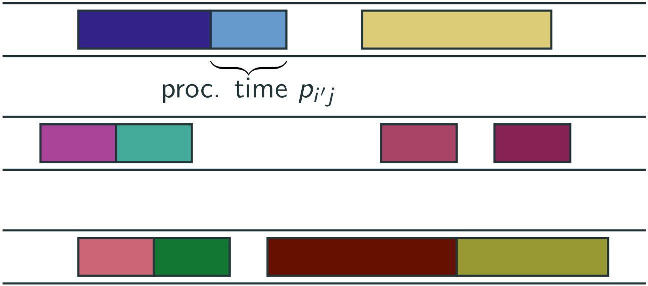
parallel machines

$R \mid r_j \mid F_{\max}$



$R \mid r_j \mid F_{\max}$



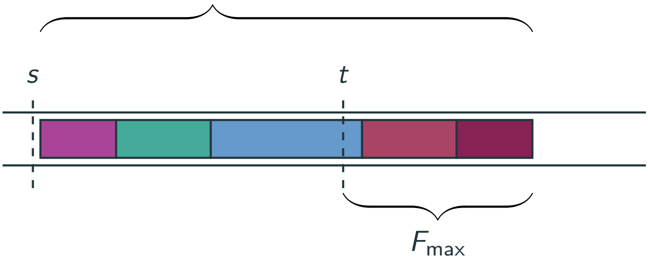


Goal: minimize max flow time $\max_j F_j$.

Condition for maximum flow time

Need to bound load on each interval of release times:

$$\sum_{\substack{s \le r_j \le t \\ j \to i}} p_{ij} \le t - s + F_{\max}.$$



Bansal-Kulkarni (STOC'15): Iterative rounding.

$\rightsquigarrow$ rounding **half** of the variables incurs error $O(\mathrm{OPT})$

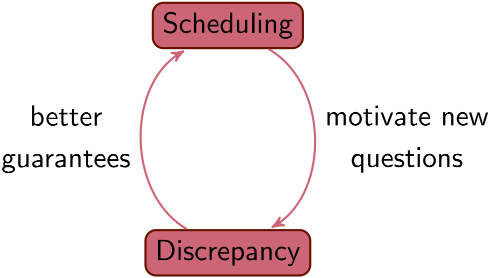
$\rightsquigarrow$ final solution worse by factor $O(\log n)$

Bansal-Kulkarni (STOC'15): Iterative rounding.

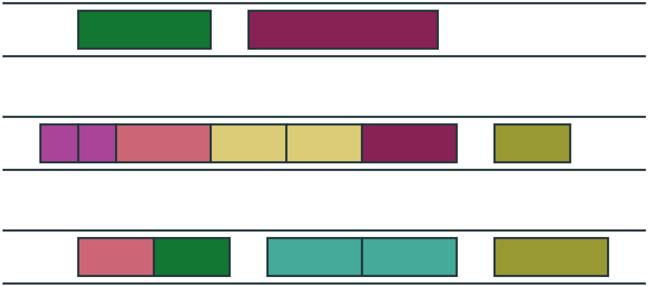
$\rightsquigarrow$ rounding **half** of the variables incurs error $O(\mathrm{OPT})$

$\rightsquigarrow$ final solution worse by factor $O(\log n)$

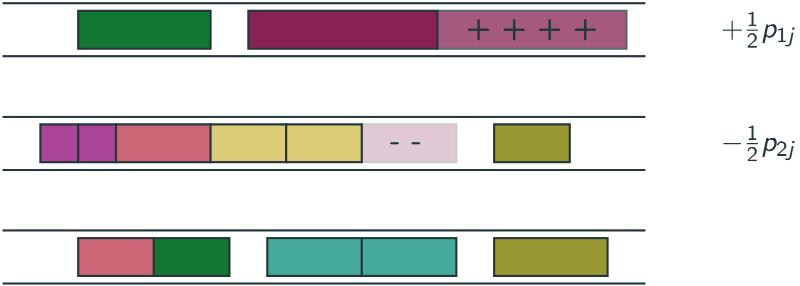
Bansal, R., Svensson (STOC'22): $O(\sqrt{\log n})$ using results from discrepancy (that rely on different methods from convex geometry).



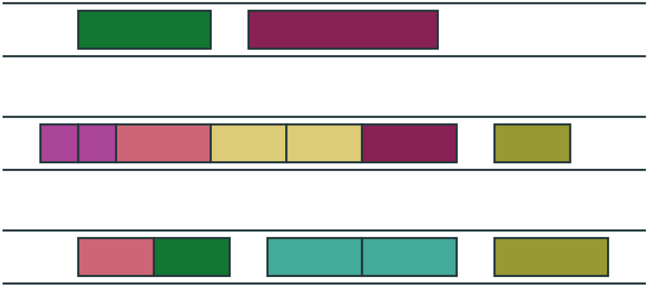
Assume for simplicity we already have a half integral solution



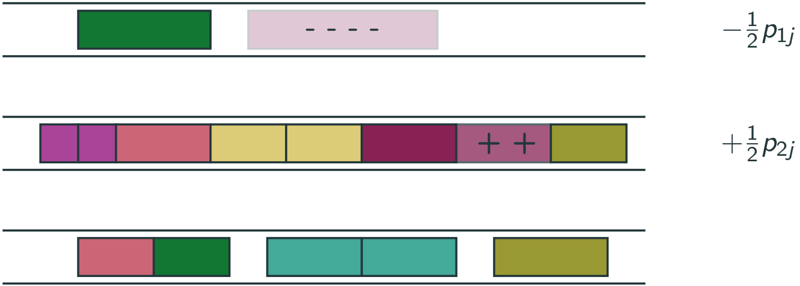
Assume for simplicity we already have a half integral solution



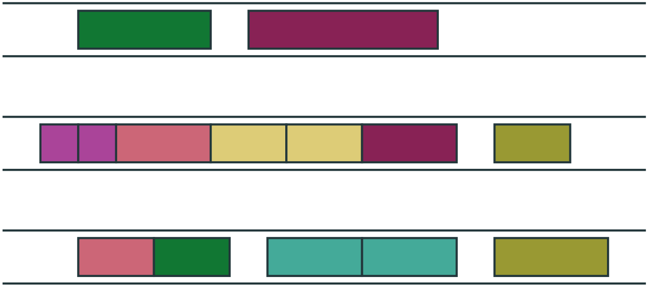
Assume for simplicity we already have a half integral solution



Assume for simplicity we already have a half integral solution



Assume for simplicity we already have a half integral solution



$$\pm \begin{pmatrix} \frac{1}{2} p_{1j} \\ -\frac{1}{2} p_{2j} \\ 0 \end{pmatrix}$$

Prefix Beck-Fiala

Given a series of n vectors with ℓ_1-norm $\leq T$

$$v_1 = \begin{pmatrix} 0 \\ 0.1 \\ 0 \\ -0.5 \\ 0 \end{pmatrix}, v_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.9 \\ -0.9 \end{pmatrix}, v_3 = \begin{pmatrix} 0.5 \\ 0 \\ -0.7 \\ 0 \\ 0 \end{pmatrix}, v_4 = \begin{pmatrix} 0 \\ 0 \\ 0.9 \\ -0.7 \\ 0 \end{pmatrix}, \ldots$$

show there exist signs $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \ldots \in \{-1, 1\}$ s.t. for all $\ell \leq h$

$$\|\varepsilon_\ell v_\ell + \varepsilon_{\ell+1} v_{\ell+1} + \cdots + \varepsilon_h v_h\|_\infty \leq \text{“some upper bound”}$$

Prefix Beck-Fiala

Given a series of n vectors with ℓ_1-norm $\leq T$

$$v_1 = \begin{pmatrix} 0 \\ 0.1 \\ 0 \\ -0.5 \\ 0 \end{pmatrix}, v_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.9 \\ -0.9 \end{pmatrix}, v_3 = \begin{pmatrix} 0.5 \\ 0 \\ -0.7 \\ 0 \\ 0 \end{pmatrix}, v_4 = \begin{pmatrix} 0 \\ 0 \\ 0.9 \\ -0.7 \\ 0 \end{pmatrix}, \dots$$

show there exist signs $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \dots \in \{-1, 1\}$ s.t. for all $\ell \leq h$

$$\|\varepsilon_\ell v_\ell + \varepsilon_{\ell+1} v_{\ell+1} + \cdots + \varepsilon_h v_h\|_\infty \leq \text{“some upper bound”}$$

Banaszczyk'98, Banaszczyk'12: $O(\sqrt{\log n} \cdot T)$ suffices

$\rightsquigarrow$ $O(\sqrt{\log n})$ integrality gap 🙂

(works also beyond the half-integral case)

Summary

Theorem
Integrality gap of LP for max flow time $\leq O(\sqrt{\log n})$*

Theorem
Integrality gap of LP for total flow time $\leq O(\log^{3/2} n)$*

(Known lower bound is $\Omega(\log n)$)

*** bounds are non-constructive, because Banaszczyk's proof does not yield an efficient algorithm**

Conclusion

Does the lack of good algorithms for optimizing flow time (compared to completion time) come from inherent hardness or have we simply not found the right techniques, yet?

Conclusion

Does the lack of good algorithms for optimizing flow time (compared to completion time) come from inherent hardness or have we simply not found the right techniques, yet?

Progress on $1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$ and $R \mid r_j \mid F_{\max}$ speaks in favor of the latter.

Conclusion

Does the lack of good algorithms for optimizing flow time (compared to completion time) come from inherent hardness or have we simply not found the right techniques, yet?

Progress on $1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$ and $R \mid r_j \mid F_{\max}$ speaks in favor of the latter.

Favorite open questions:

- constant approximation for $R \mid r_j \mid F_{\max}$ (linked to Discrepancy)
- 2.99-approximation for $P \mid r_j \mid F_{\max}$

Conclusion

Does the lack of good algorithms for optimizing flow time (compared to completion time) come from inherent hardness or have we simply not found the right techniques, yet?

Progress on $1 \mid \mathrm{pmpt}, r_j \mid \sum_j w_j F_j$ and $R \mid r_j \mid F_{\max}$ speaks in favor of the latter.

Favorite open questions:

- constant approximation for $R \mid r_j \mid F_{\max}$ (linked to Discrepancy)
- 2.99-approximation for $P \mid r_j \mid F_{\max}$

Thanks!