Unrelated Machine Scheduling of Jobs with Uniform Smith Ratios

Jakub Tarnawski

joint work with Christos Kalaitzis and Ola Svensson

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

January 19, 2017

Scheduling on unrelated machines

- n jobs
- m machines
- allocate each job to one machine
- minimize certain objective

Scheduling on unrelated machines

- n jobs
- m machines
- allocate each job to one machine
- minimize certain objective
- p_{ij}: processing time of job j on machine i
  - jobs require different amounts of time, features of machine

Scheduling on unrelated machines

- n jobs
- m machines
- allocate each job to one machine
- minimize certain objective
- p_{ij}: processing time of job j on machine i
  - jobs require different amounts of time, features of machine

Scheduling on unrelated machines

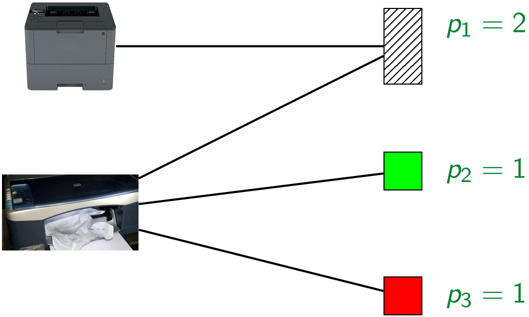
- n jobs
- m machines
- allocate each job to one machine
- minimize certain objective
- p_{ij}: processing time of job j on machine i
  - jobs require different amounts of time, features of machine

Scheduling on unrelated machines

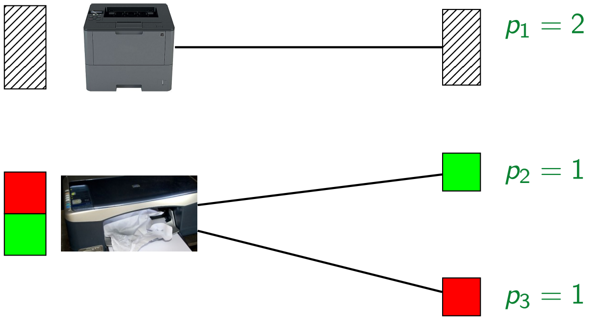
- n jobs
- m machines
- allocate each job to one machine
- minimize certain objective
- p_{ij}: processing time of job j on machine i
  - jobs require different amounts of time, features of machine

Objective: understand the approximability of these problems

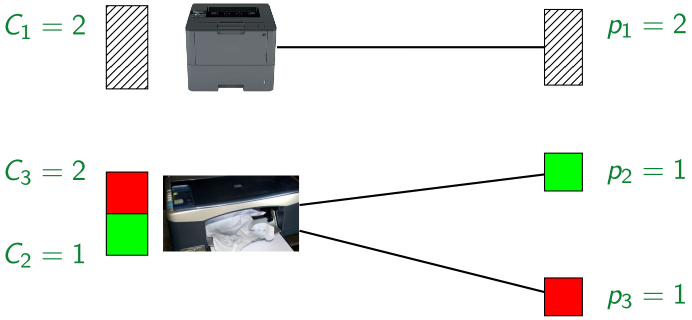
Scheduling on unrelated machines



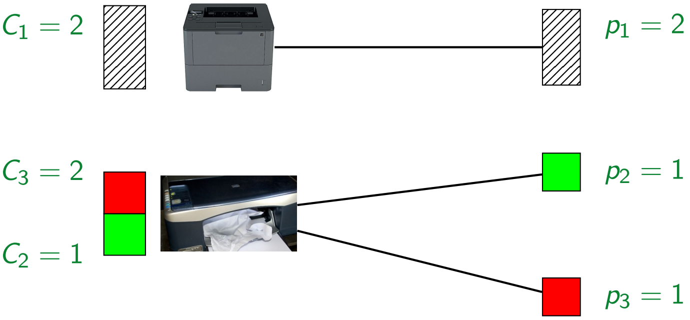
Scheduling on unrelated machines



Scheduling on unrelated machines



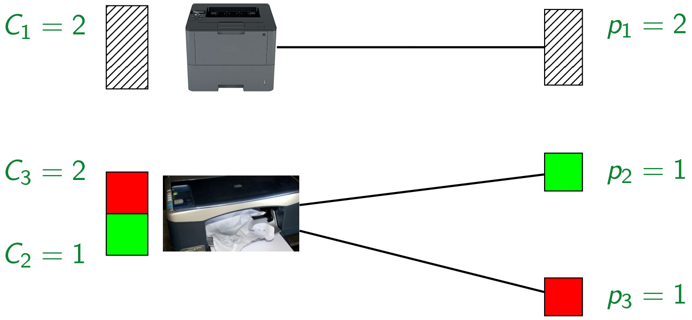
C_j: completion time of job j



C_j: completion time of job j

Minimize:

- **makespan**: $\max_j C_j$

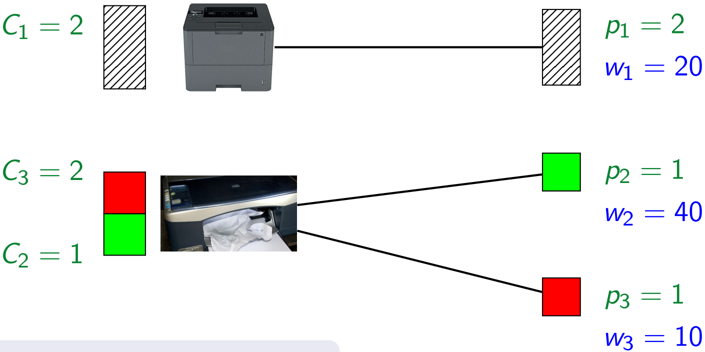


C_j: completion time of job j

Minimize:

- **makespan**: $\max_j C_j = 2$

Scheduling on unrelated machines

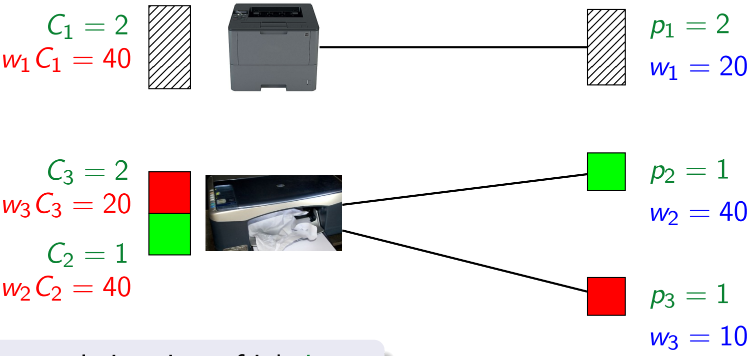


C_j: completion time of job j

Minimize:

- **makespan**: $\max_j C_j = 2$
- **weighted sum of completion times**: $\sum_j w_j C_j$
  - given weights w_j: importance of job j

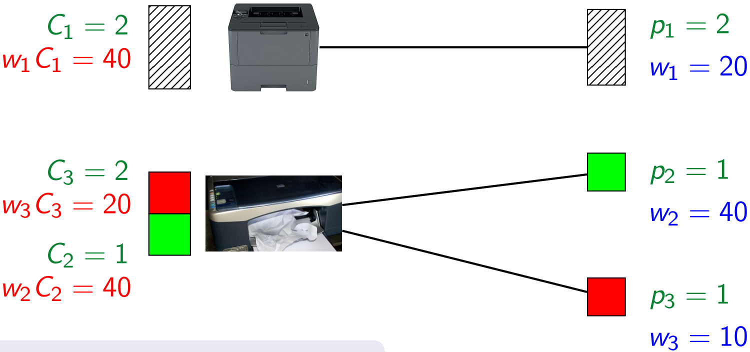
Scheduling on unrelated machines



C_j: completion time of job j

Minimize:

- **makespan**: $\max_j C_j = 2$
- **weighted sum of completion times**: $\sum_j w_j C_j = 100$
  - given weights w_j: importance of job j



C_j: completion time of job j

Minimize:

- ~~**makespan**: $\max_j C_j = 2$~~
- **weighted sum of completion times**: $\sum_j w_j C_j = 100$
  - given weights w_j: importance of job j

Single machine

To minimize weighted sum of completion times $\sum_j w_j C_j$:

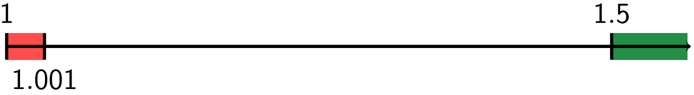
Smith's rule [1956]

Order jobs by w_j / p_j (Smith ratio)

So:

- allocate jobs to machines: **hard part**
- order jobs on every machine: **easy**

State of the art



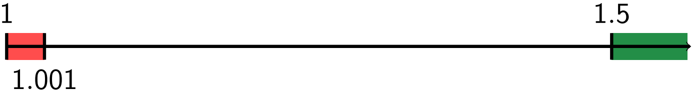
Hoogeveen et al. 2001

Minimizing $\sum_j w_j C_j$ is hard to approximate within 1.001.

Skutella 2001 / Sethuraman, Squillante 1999

There is a 1.5-approximation algorithm
(*independent* randomized rounding of convex relaxation).

State of the art



Hoogeveen et al. 2001

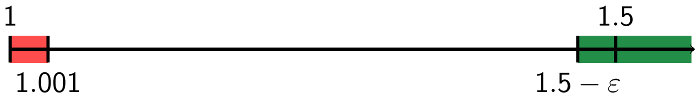
Minimizing $\sum_j w_j C_j$ is hard to approximate within 1.001.

Skutella 2001 / Sethuraman, Squillante 1999

There is a 1.5-approximation algorithm
(*independent* randomized rounding of convex relaxation).

Can't do better than 1.5 with independent rounding,
and these relaxations have integrality gap 1.5.

State of the art



Hoogeveen et al. 2001

Minimizing $\sum_j w_j C_j$ is hard to approximate within 1.001.

Skutella 2001 / Sethuraman, Squillante 1999

There is a 1.5-approximation algorithm
(*independent* randomized rounding of convex relaxation).

Can't do better than 1.5 with independent rounding,
and these relaxations have integrality gap 1.5.

Bansal et al. 2016

There is a $(1.5 - \varepsilon)$-approximation algorithm.

State of the art

Hoogeveen et al. 2001

Minimizing $\sum_j w_j C_j$ is hard to approximate within 1.001.

Skutella 2001 / Sethuraman, Squillante 1999

There is a 1.5-approximation algorithm
(*independent* randomized rounding of convex relaxation).

Can't do better than 1.5 with independent rounding,
and these relaxations have integrality gap 1.5.

Bansal et al. 2016

There is a $(1.5 - \varepsilon)$-approximation algorithm.

Our special case: uniform-Smith-ratios

For each machine i, Smith ratios are uniform: $p_{ij} \in \{\alpha_i w_{ij}, \infty\}$.

- order of jobs on machine doesn't matter
- natural: every unit of work has same weight
- jobs: time-consuming $\iff$ important

Our special case: uniform-Smith-ratios

For each machine i, Smith ratios are uniform: $p_{ij} \in \{\alpha_i w_{ij}, \infty\}$.

- order of jobs on machine doesn't matter
- natural: every unit of work has same weight
- jobs: time-consuming $\iff$ important

So:

- allocate jobs to machines: **hard part**
- order jobs on every machine: ~~**easy**~~ **irrelevant**

Hardness

Other special cases can be easy:

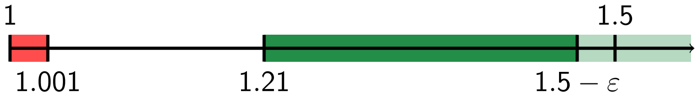
- all $w_j = 1$: in P
- identical parallel machines: has PTAS

but uniform-Smith-ratios inherits hardness of general version:

- still APX-hard
- still *independent* randomized rounding can only yield 1.5
- still the previous relaxations have integrality gap 1.5

Our result

Our result

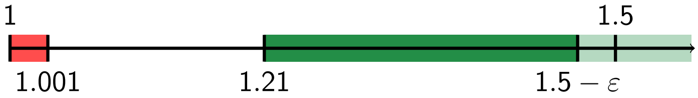


Our main result

There is a $\frac{1+\sqrt{2}}{2} \approx 1.21$-approximation algorithm for unrelated machine scheduling with uniform Smith ratios.

Analysis is **tight**.

Our result



Our main result

There is a $\frac{1+\sqrt{2}}{2} \approx 1.21$-approximation algorithm for unrelated machine scheduling with uniform Smith ratios.

Analysis is **tight**.

Bonus

Simultaneous 2-approximation for makespan and 1.21-approximation for $\sum_j w_j C_j$.

Plan of talk:

- Configuration-LP
  - assigns *configurations* (subsets of jobs) to machines
- Shmoys-Tardos rounding
  - randomized rounding of LP solution
- flavor of analysis
  - fix single machine
  - compare two probability distributions on configurations: from LP solution and from our rounding
  - bound ratio of their expected costs

Configuration-LP

Configuration-LP

Very strong LP relaxation which assigns whole *configurations* (subsets of jobs) to machines.

- variable $y_{iC} \geq 0$ for each machine i and configuration C
- intention: $y_{iC} = 1$ iff the set of jobs processed by machine i is C

Configuration-LP

Very strong LP relaxation which assigns whole *configurations* (subsets of jobs) to machines.

- variable $y_{iC} \geq 0$ for each machine i and configuration C
- intention: $y_{iC} = 1$ iff the set of jobs processed by machine i is C
- minimize cost: $\min \sum_{i,C} y_{iC} \cdot \mathrm{cost}_i(C)$
- constraints:
  - each machine processes exactly one configuration
  - each job is processed on exactly one machine

Configuration-LP

Very strong LP relaxation which assigns whole *configurations* (subsets of jobs) to machines.

- variable $y_{iC} \geq 0$ for each machine i and configuration C
- intention: $y_{iC} = 1$ iff the set of jobs processed by machine i is C
- minimize cost: $\min \sum_{i,C} y_{iC} \cdot \mathrm{cost}_i(C)$
- constraints:
  - each machine processes exactly one configuration
  - each job is processed on exactly one machine

exponential size but has PTAS to solve (Sviridenko, Wiese 2013)

Configuration-LP

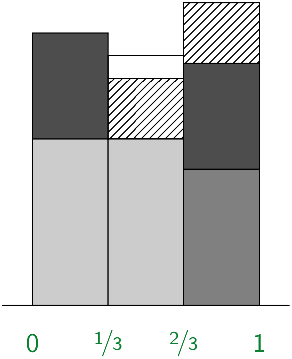
Very strong LP relaxation which assigns whole *configurations* (subsets of jobs) to machines.

- variable $y_{iC} \geq 0$ for each machine i and configuration C
- intention: $y_{iC} = 1$ iff the set of jobs processed by machine i is C
- minimize cost: $\min \sum_{i,C} y_{iC} \cdot \mathrm{cost}_i(C)$
- constraints:
  - each machine processes exactly one configuration
  - each job is processed on exactly one machine

exponential size but has PTAS to solve (Sviridenko, Wiese 2013)

$1.08 \leq$ integrality gap ≤ 1.21 (this work)

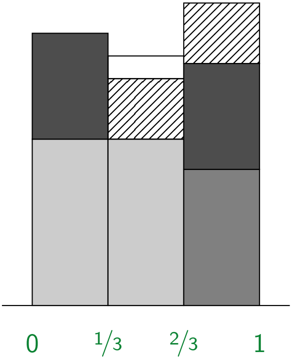
Configuration-LP outputs a distribution



Distribution on configurations for a fixed machine $i^\star$

- rectangle: job
- height of rectangle: processing time
- stack of rectangles: configuration
- width: probability

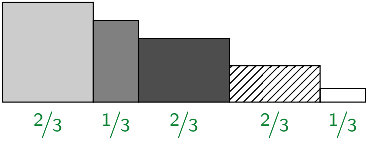
Configuration-LP outputs a distribution



Distribution on configurations for a fixed machine $i^\star$

- rectangle: job
- height of rectangle: processing time
- stack of rectangles: configuration
- width: probability

x_{ij} = marginal probability that machine i processes job j



$$x_{ij} = \sum_{C \ni j} y_{iC}$$

Shmoys-Tardos rounding

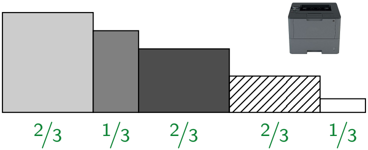
Shmoys-Tardos rounding

- map the marginals x to
  a fractional matching in a bipartite graph

- randomly round this fractional matching to
  an integral matching (which corresponds to a schedule)

- originally used for 2-approximation for the makespan objective
  (applied to the so-called Assignment-LP)

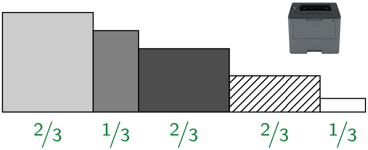
Shmoys-Tardos rounding



x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

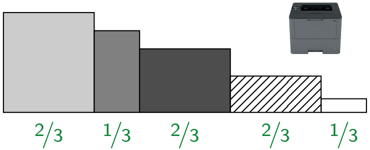
Shmoys-Tardos rounding



x_{ij} = marginal probability that machine i processes job j

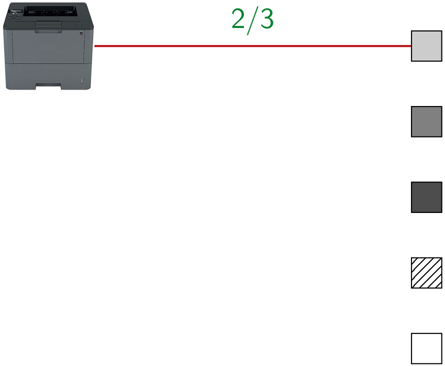
for each machine i:
for each job j in order of decreasing p_{ij}:

Shmoys-Tardos rounding

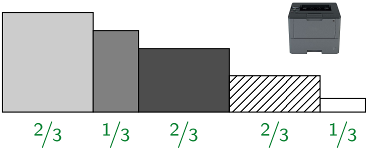


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

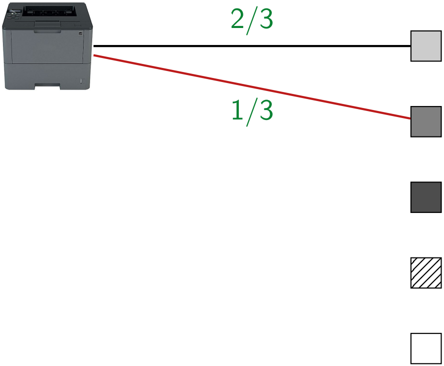


Shmoys-Tardos rounding

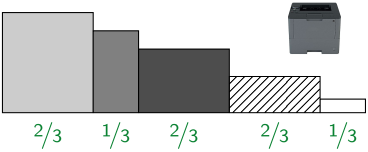


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

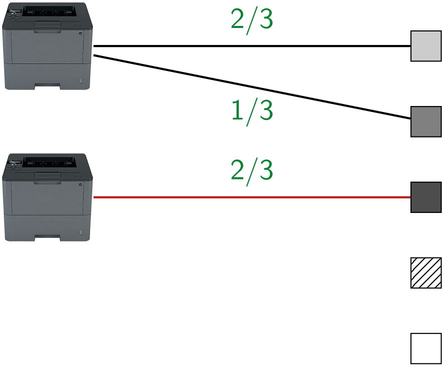


Shmoys-Tardos rounding

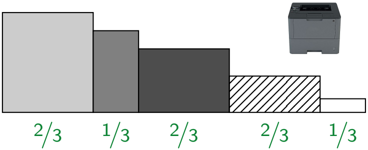


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

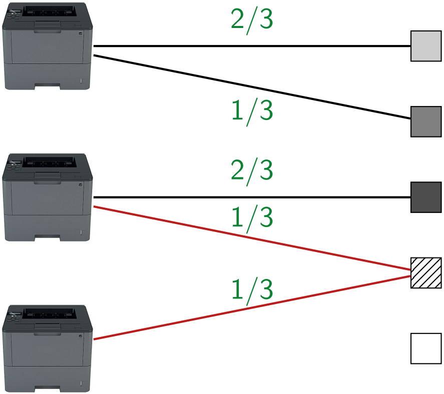


Shmoys-Tardos rounding

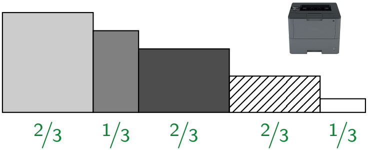


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

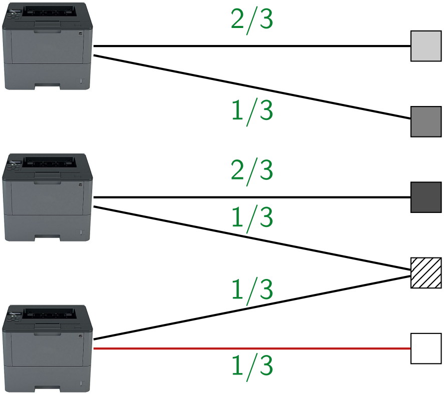


Shmoys-Tardos rounding

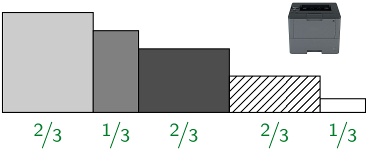


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

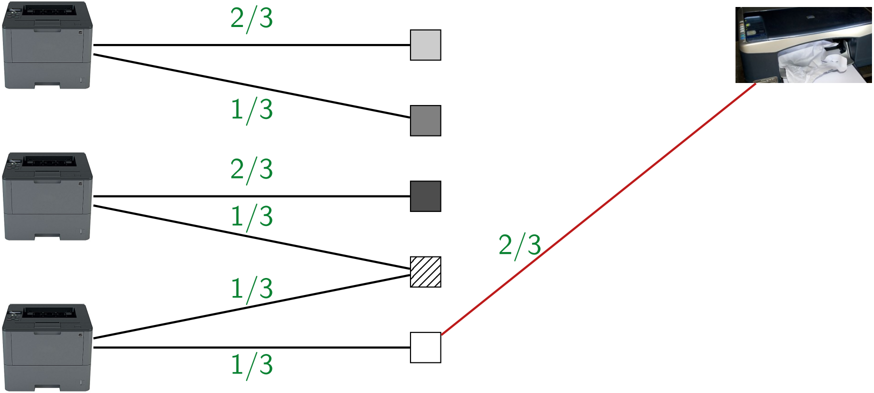


Shmoys-Tardos rounding

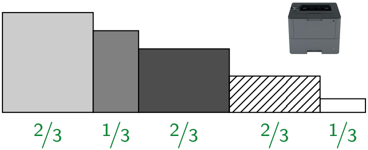


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

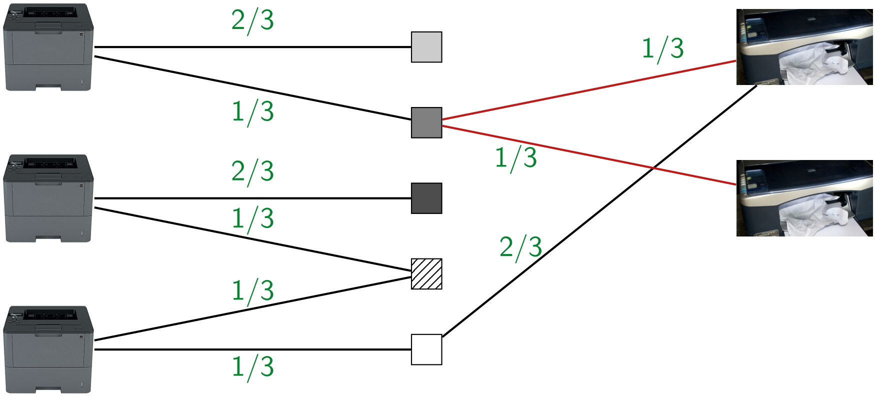


Shmoys-Tardos rounding

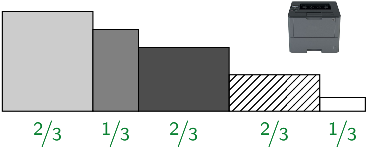


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j <u>in order of decreasing p_{ij}</u>:

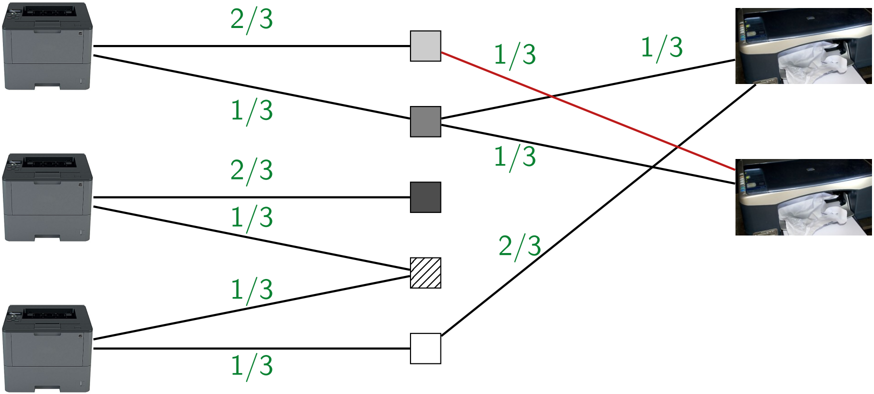


Shmoys-Tardos rounding

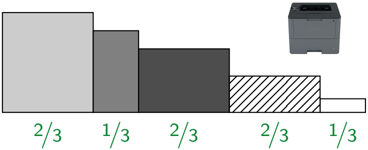


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

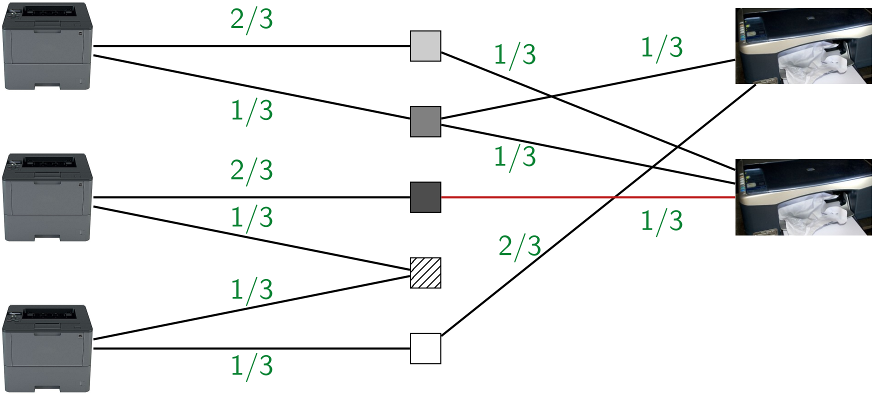


Shmoys-Tardos rounding

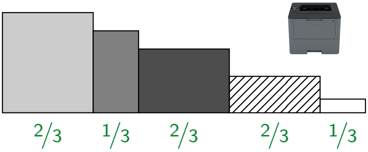


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

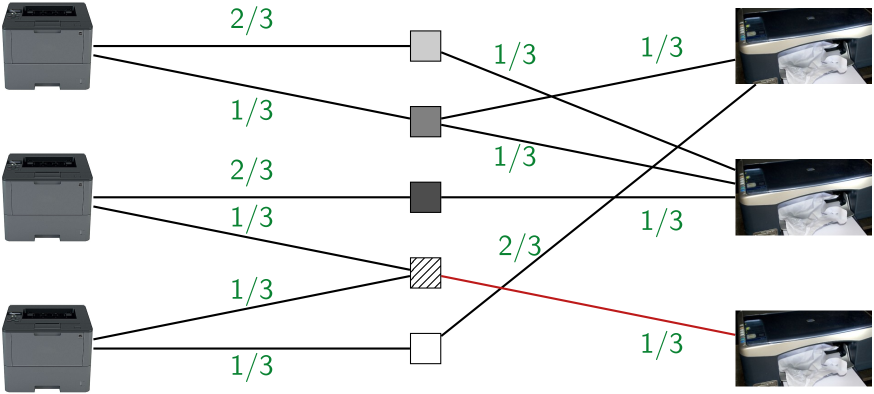


Shmoys-Tardos rounding

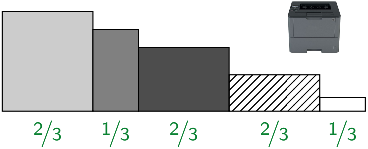


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j <u>in order of decreasing p_{ij}</u>:

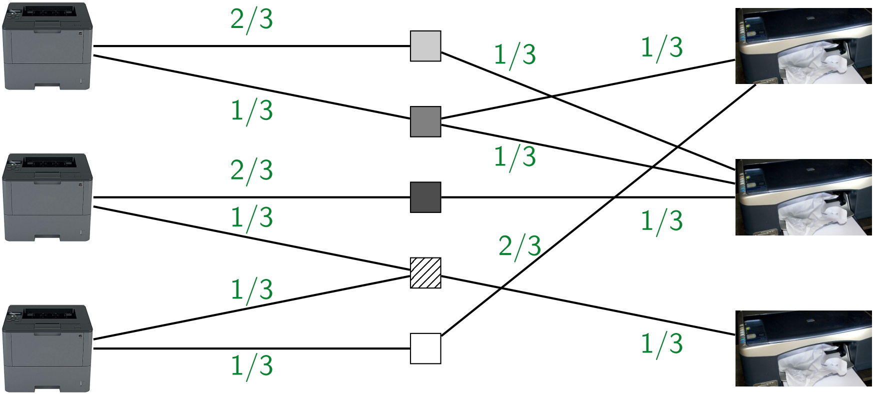


Shmoys-Tardos rounding

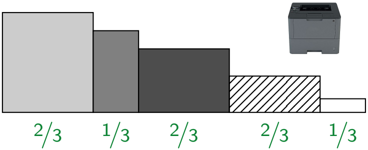


x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

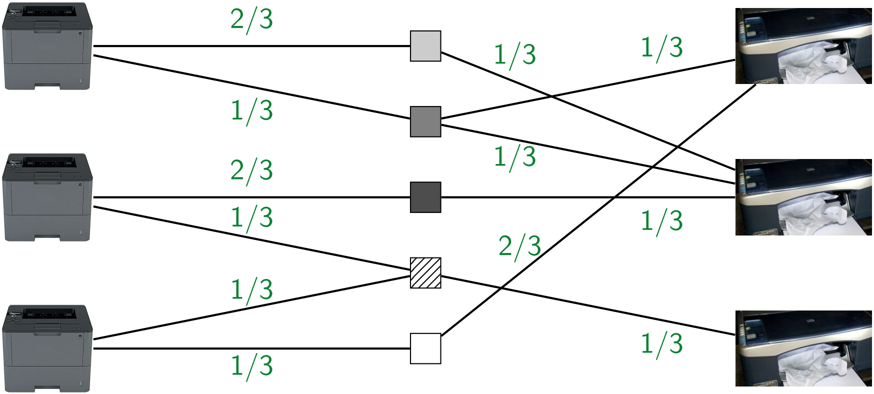


Shmoys-Tardos rounding



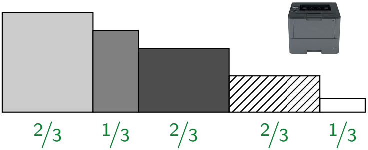
x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:



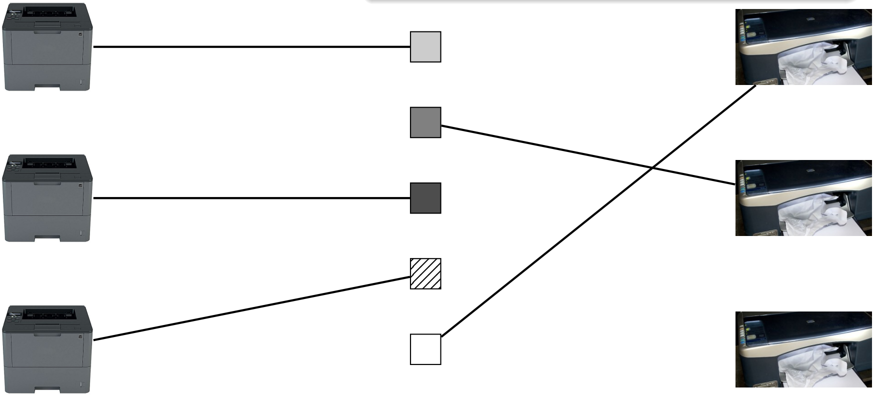
- Round fractional to integral matching, preserving marginals x_{ij}.

Shmoys-Tardos rounding



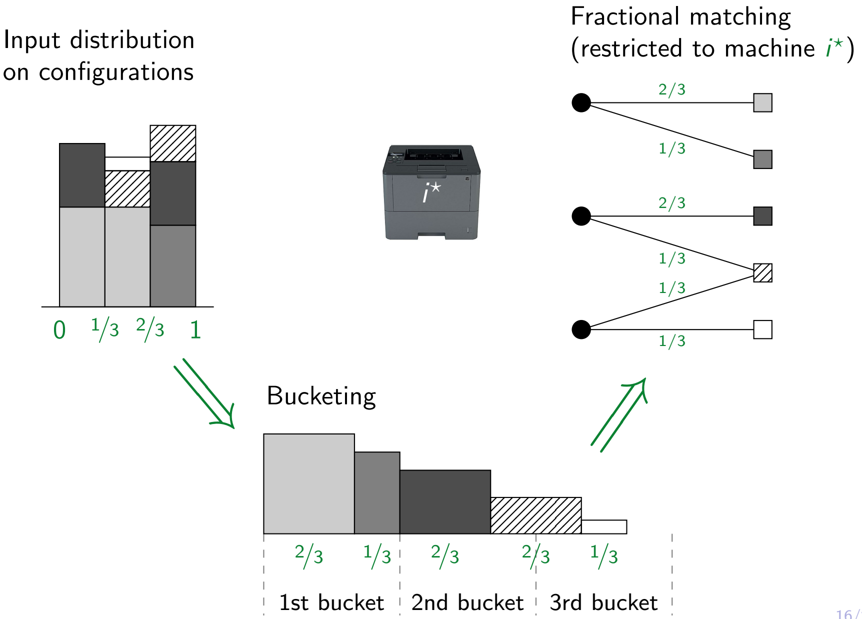
x_{ij} = marginal probability that machine i processes job j

for each machine i:
for each job j in order of decreasing p_{ij}:

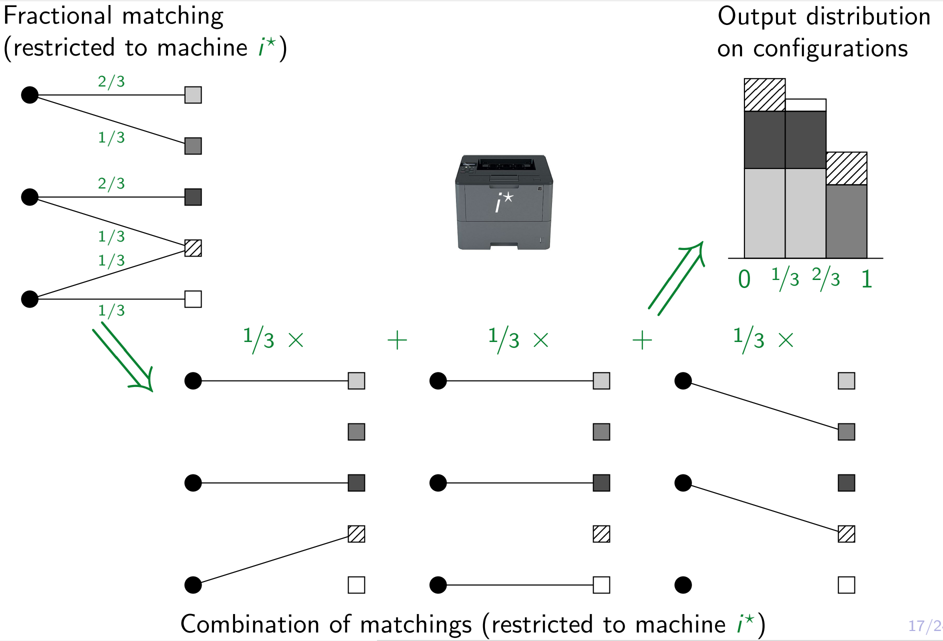


- Round fractional to integral matching, preserving marginals x_{ij}.

Shmoys-Tardos rounding



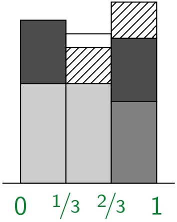
Shmoys-Tardos rounding



Our analysis

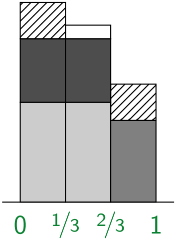
Two configurations

Input distribution
on configurations (y^{in})



$\text{cost}(y^{\text{in}})$: **LP lower bound**

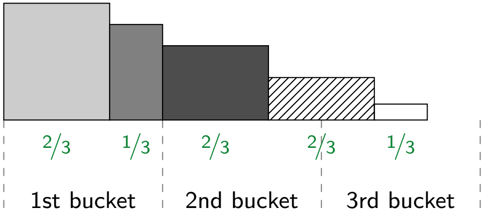
Output distribution
on configurations (y^{out})



$\text{cost}(y^{\text{out}})$: **cost of our solution**

- we want to bound $\frac{\text{cost}(y^{\text{out}})}{\text{cost}(y^{\text{in}})} \leq \frac{1+\sqrt{2}}{2} \approx 1.21$
- y^{in} and y^{out} have same marginals
- y^{out} has a nice bucket structure from our rounding

Bucket structure



Each configuration gets:

- one job from 1st bucket
- one job from 2nd bucket
- one job from 3rd bucket (or none)

k-th largest job of any configuration
$$\geq$$
$(k+1)$-th largest job of any configuration

Nonexistent bad example

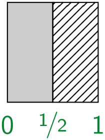
Input distribution on configurations

Output distribution on configurations

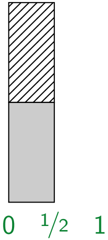
- left: best possible distribution with marginals $1/2$ on both jobs
- right: worst possible such distribution (would give ratio 1.5)
- good if small variance

Nonexistent bad example

Input distribution
on configurations



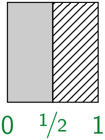
Output distribution
on configurations (impossible)



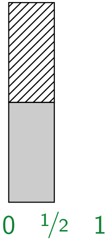
- left: best possible distribution with marginals $1/2$ on both jobs
- right: worst possible such distribution (would give ratio 1.5)
- good if small variance
- this cannot happen in our algorithm:
  no bucket structure

Nonexistent bad example

Input distribution
on configurations



Output distribution
on configurations (impossible)



- left: best possible distribution with marginals $1/2$ on both jobs
- right: worst possible such distribution (would give ratio 1.5)
- good if small variance
- this cannot happen in our algorithm:
  no bucket structure

Our analysis

- we transform both y^{in} and y^{out} while making the ratio worse

$$(y^{\text{in}}, y^{\text{out}}) \to (y_1^{\text{in}}, y_1^{\text{out}}) \to (y_2^{\text{in}}, y_2^{\text{out}}) \to \dots$$

$$\frac{\text{cost}(y^{\text{out}})}{\text{cost}(y^{\text{in}})} \le \frac{\text{cost}(y_1^{\text{out}})}{\text{cost}(y_1^{\text{in}})} \le \frac{\text{cost}(y_2^{\text{out}})}{\text{cost}(y_2^{\text{in}})} \le \dots$$

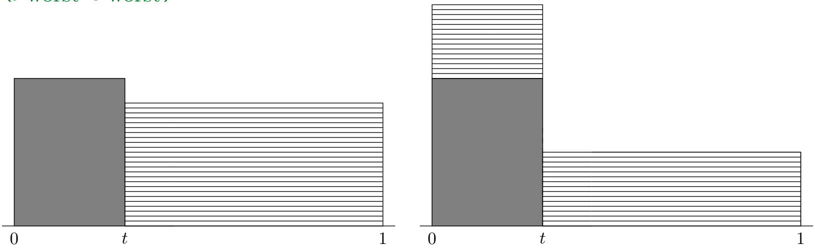
  (main technical part, uses uniform Smith ratios)
- we arrive at a "worst-case" pair for which we can bound the ratio by $\frac{1+\sqrt{2}}{2}$

$$\dots \to (y_{\text{worst}}^{\text{in}}, y_{\text{worst}}^{\text{out}})$$

$$\dots \le \frac{\text{cost}(y_{\text{worst}}^{\text{out}})}{\text{cost}(y_{\text{worst}}^{\text{in}})} \le \frac{1+\sqrt{2}}{2}$$

Our analysis

$(y^{\text{in}}_{\text{worst}}, y^{\text{out}}_{\text{worst}})$ looks like this:



- gray part: single large job
- striped parts: many jobs with *infinitesimal size* $\varepsilon \to 0$

$$\frac{\text{cost}(y^{\text{out}}_{\text{worst}})}{\text{cost}(y^{\text{in}}_{\text{worst}})} \leq \sup_{t\in[0,1),\gamma\geq 0,\lambda\geq 0} \frac{t\gamma^2 + t\gamma\lambda + \frac{\lambda^2}{2}}{t\gamma^2 + \frac{\lambda^2}{2(1-t)}} \leq \frac{1+\sqrt{2}}{2}$$

- analysis **tight**: this corresponds to a scheduling instance

Summary

Our main result

There is a $\frac{1+\sqrt{2}}{2} \approx 1.21$-approximation algorithm for unrelated machine scheduling with uniform Smith ratios.

Summary

Our main result

There is a $\frac{1+\sqrt{2}}{2} \approx 1.21$-approximation algorithm for unrelated machine scheduling with uniform Smith ratios.

Compared to Bansal et al. (2016):

- $-$ only for case of uniform Smith ratios
- $+$ 1.21 apx ratio vs $1.5 - \varepsilon$
- $+$ much simpler algorithm and analysis

Summary

Our main result

There is a $\frac{1+\sqrt{2}}{2} \approx 1.21$-approximation algorithm for unrelated machine scheduling with uniform Smith ratios.

Compared to Bansal et al. (2016):

- – only for case of uniform Smith ratios
- \+ 1.21 apx ratio vs $1.5 - \varepsilon$
- \+ much simpler algorithm and analysis

Open directions:

- best approximation factor for uniform-Smith-ratios?
- approximation factor of this/similar algorithm for general case?
- more applications of such an analysis

Summary

Our main result

There is a $\frac{1+\sqrt{2}}{2} \approx 1.21$-approximation algorithm for unrelated machine scheduling with uniform Smith ratios.

Compared to Bansal et al. (2016):

– only for case of uniform Smith ratios

+ 1.21 apx ratio vs $1.5 - \varepsilon$

+ much simpler algorithm and analysis

Open directions:

- best approximation factor for uniform-Smith-ratios?
- approximation factor of this/similar algorithm for general case?
- more applications of such an analysis

Thank you!