UNIVERSITY OF TWENTE.
MISD

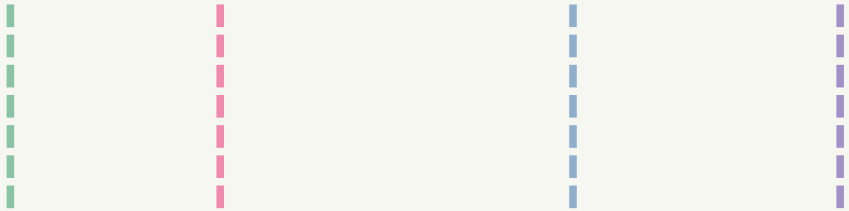
SPEED SCALING

DENISE GRAAFSMA

In collaboration with: Antonios Antoniadis, Ruben Hoeksma, Maria Vlasiou

Jobs

Release times

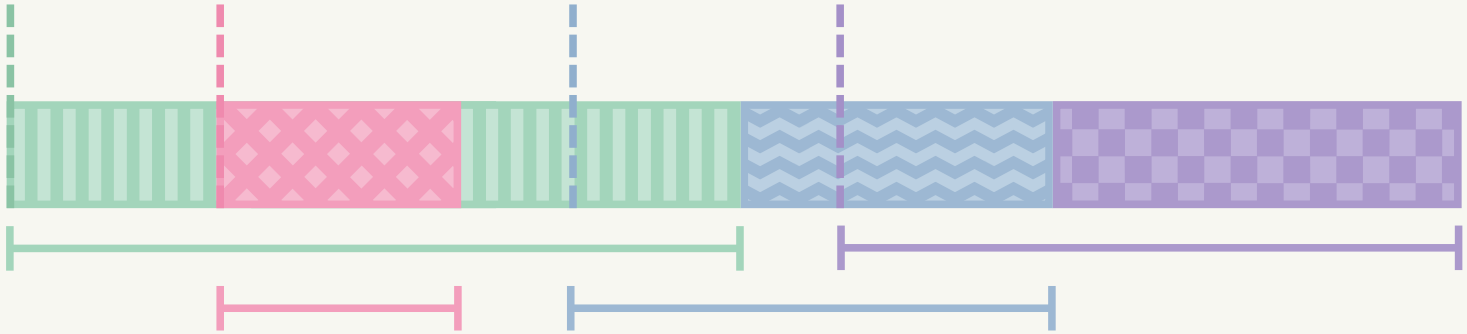


Jobs

Schedule

Jobs

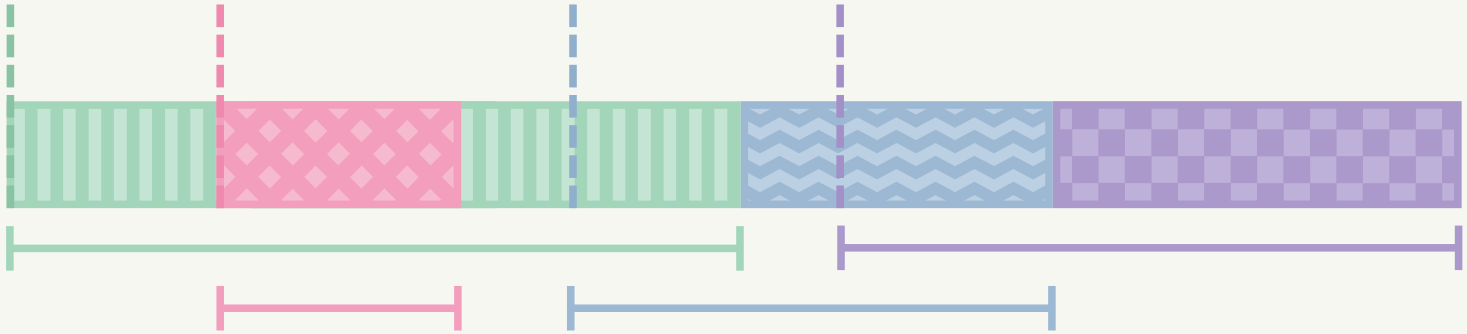
Schedule



Minimize: **total flow**

Jobs

Schedule



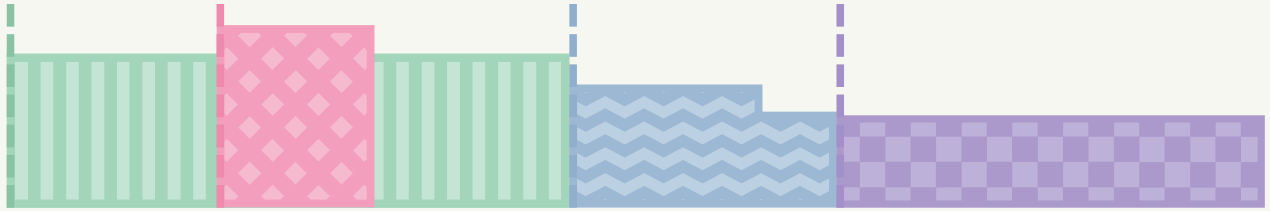
Minimize: **total flow**

Can be solved:

Shortest Remaining Processing Time (SRPT)

Jobs

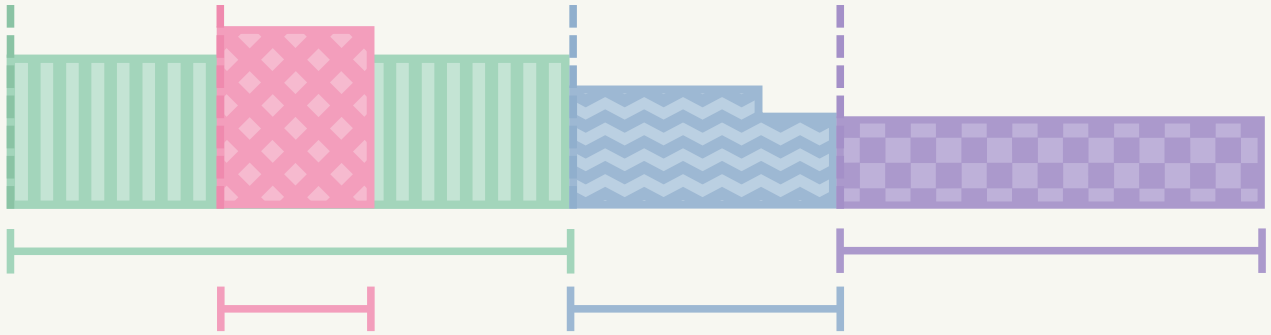
Speed scaling



Processor has **multiple speed settings**

Jobs

Speed scaling

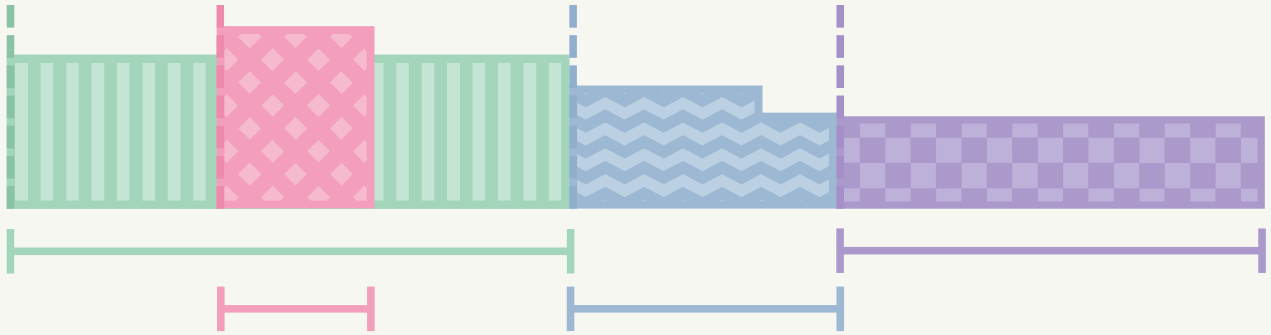


lower speeds: energy **efficient**

higher speeds: energy **inefficient**

Jobs

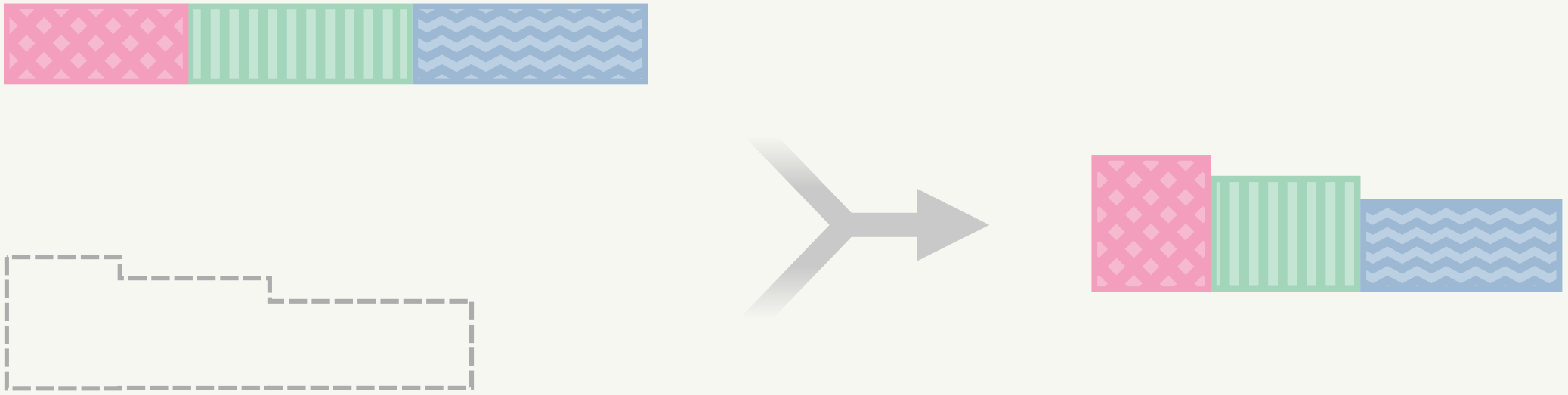
Speed scaling



Minimize: total flow **+ energy**

Speed scaling schedule

- In what **order** should the jobs complete?
- At what **speed** should they run?



Variants

- **Flow + energy** or **budget** objective

Flow + energy variant:

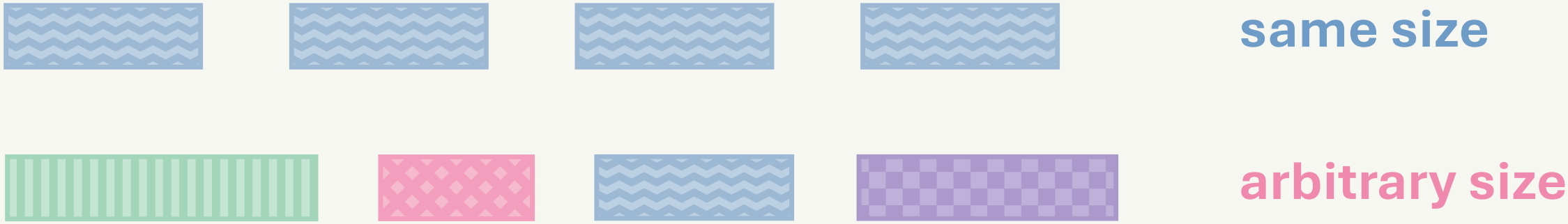
- Minimize **flow + energy**
- **No constraint** on **energy** usage

Budget variant:

- Minimize **flow**
- **Energy** has to stay **within budget**

Variants

- **Flow + energy** or **budget** objective
- **Same size** or **arbitrary size** jobs



Variants

- **Flow + energy** or **budget** objective
- **Same size** or **arbitrary size** jobs
- **Unweighted** or **weighted** jobs

Unweighted jobs:

- All jobs are **equally** important

Weighted jobs:

- Some job flows are weighed **more heavily** in the **objective**

Complexity landscape

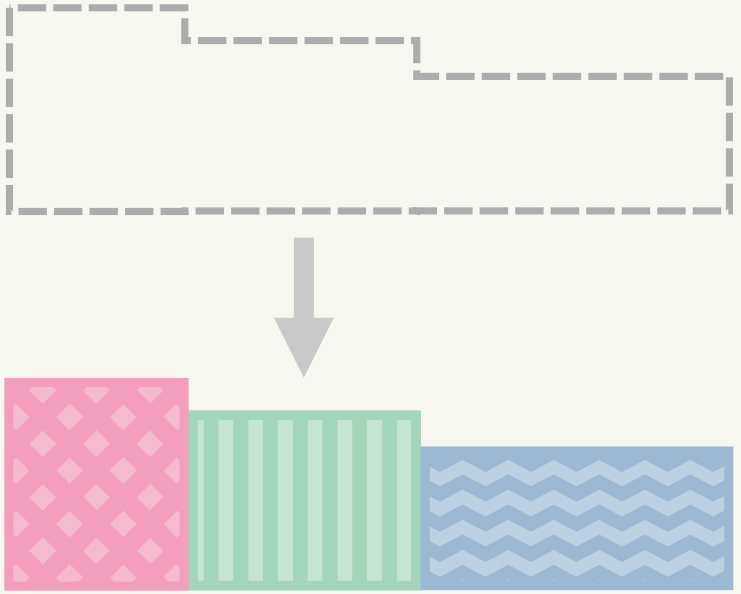
	Flow + energy		Budget	
	Same size	**Arbitrary size**	**Same size**	**Arbitrary size**
Unweighted	P [1]	?	P [1]	NP-hard [1]
Weighted	?	NP-hard [2]	NP-hard [1]	NP-hard [2]

[1] Barcelo et al., 2015
[2] Labetoulle et al., 1984

Known:

For **fixed speeds**

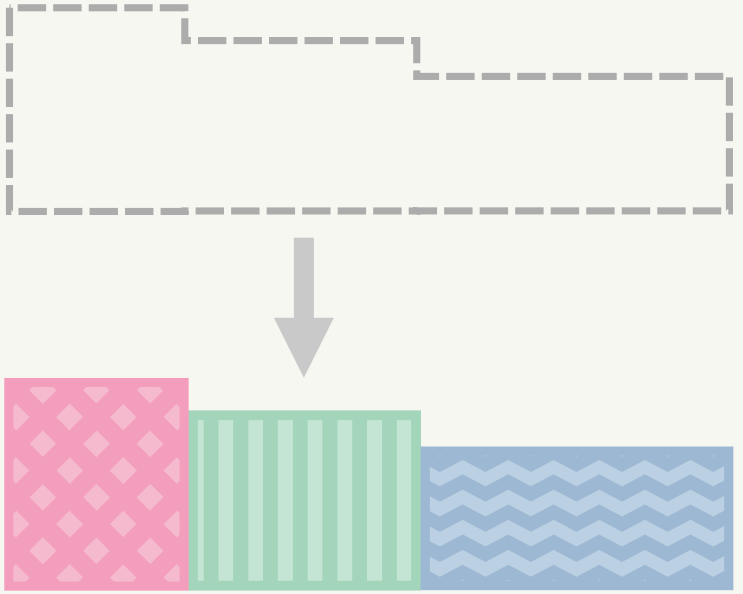
- Can find **optimal order** with **SRPT**

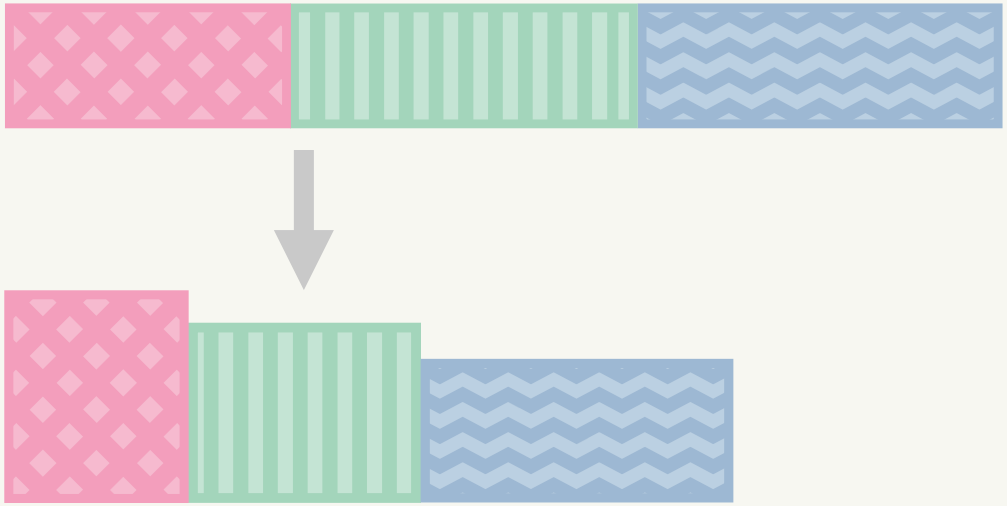


Known:

For **fixed speeds**

- Can find **optimal order** with **SRPT**



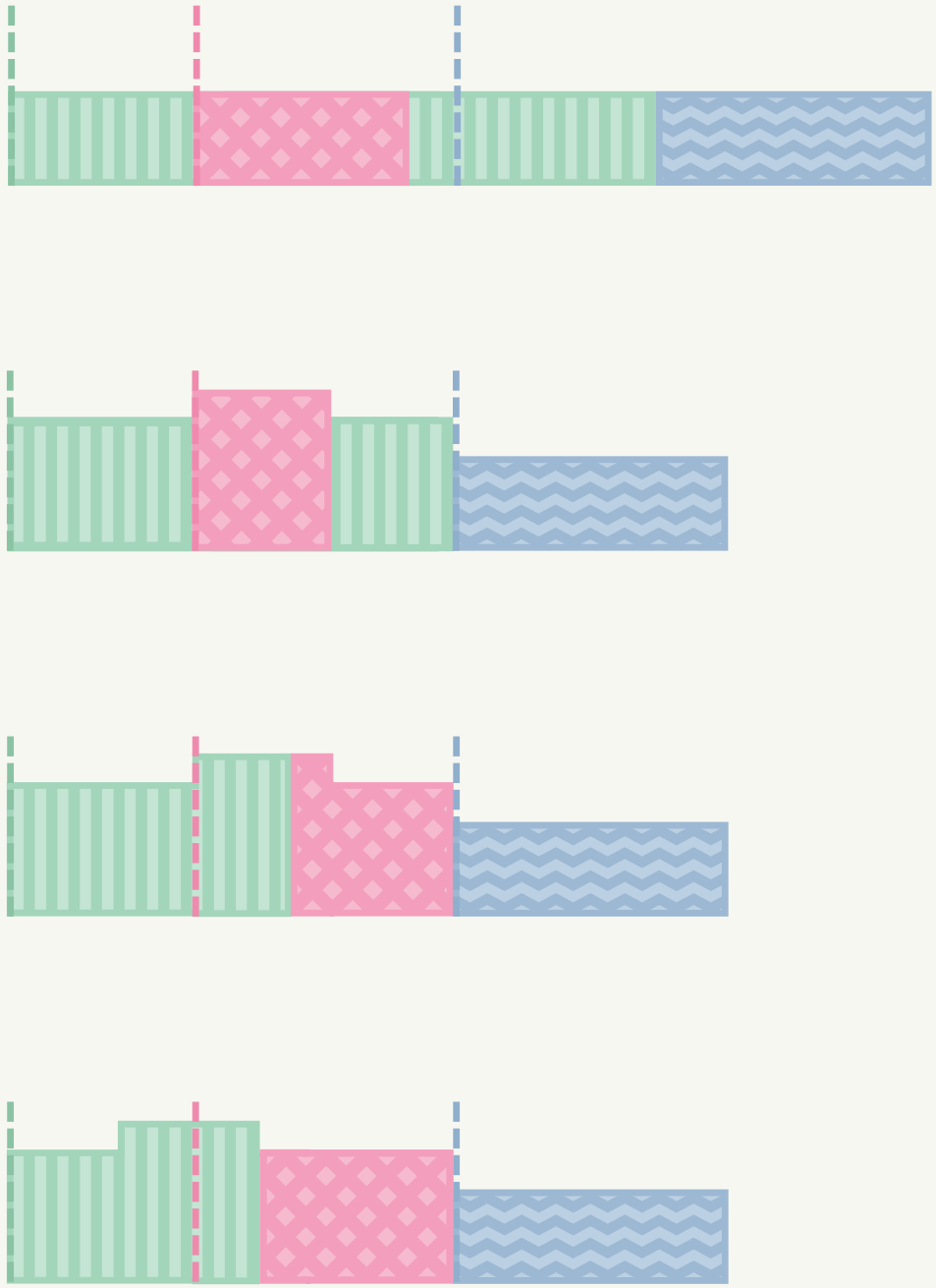


We found:

For **fixed order**

- Can find **optimal speeds** with **LP**

  (holds for any variant)



Optimize **speeds**

Optimize **order**

Optimize **speeds**

Does **not always** converge to optimal schedule.

Problem is **NP-hard**.

NP-Hardness

- **Budget**
- **Unweighted**
- **Arbitrary size**
- **Two** speeds

- **Flow + energy**
- **Unweighted**
- **Arbitrary size**
- **Two** speeds

NP-hard

(Barcelo et al., 2015)

NP-Hardness

Main idea:

*"Simulate **budget** in **flow + energy** setting"*

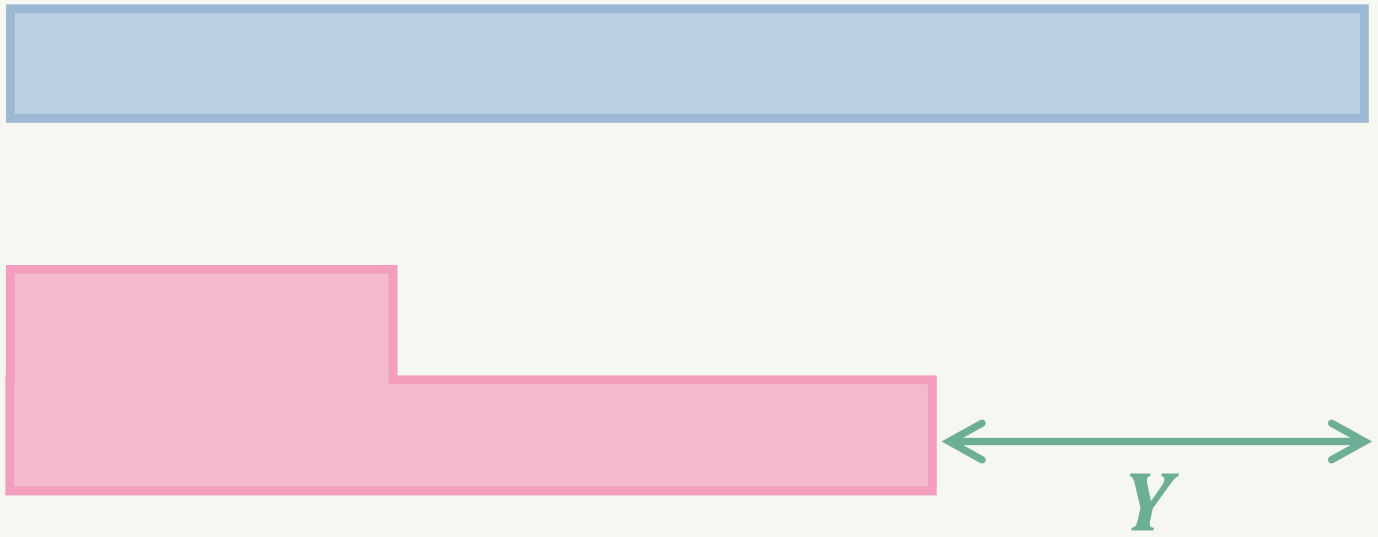
- Start with everything at **lowest speed**
- **Before** spending budget:

  energy cost < improvement in flow
- **After** spending budget:

  energy cost > improvement in flow

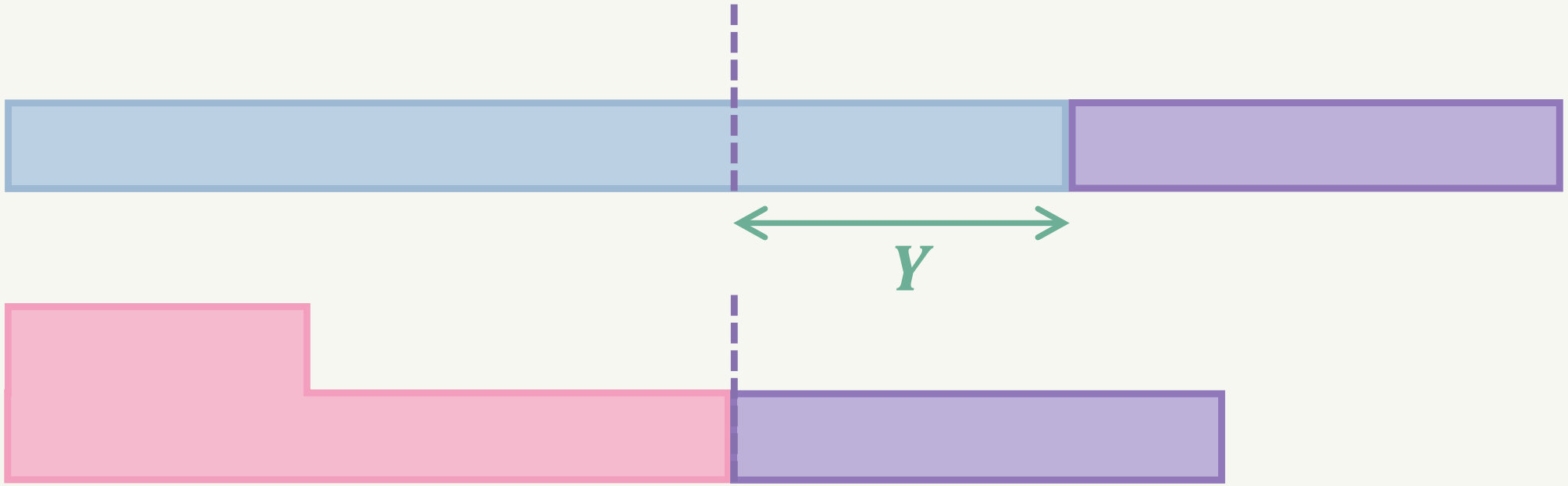
NP-Hardness

Y = total processing time at **lowest speed** − total processing time after **spending budget**

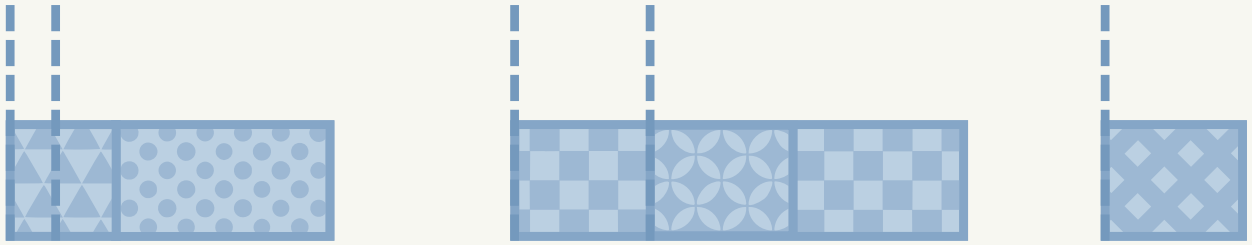


NP-Hardness

Until **budget** is spent:
flow **new job** also improves

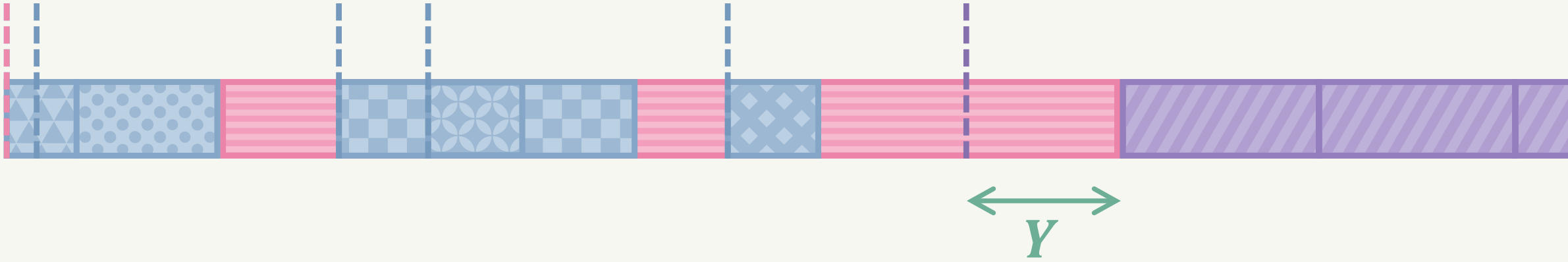


NP-Hardness



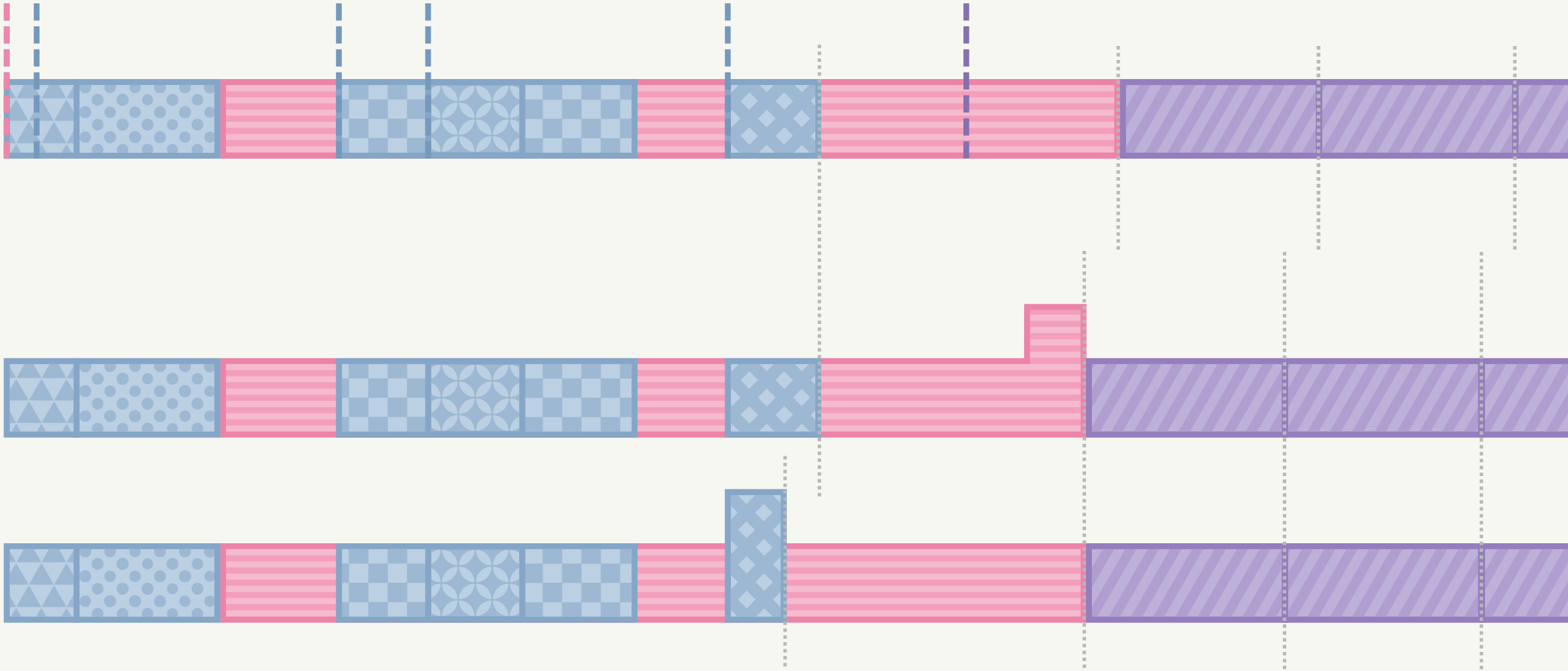
Budget instance with **lowest speed**

NP-Hardness



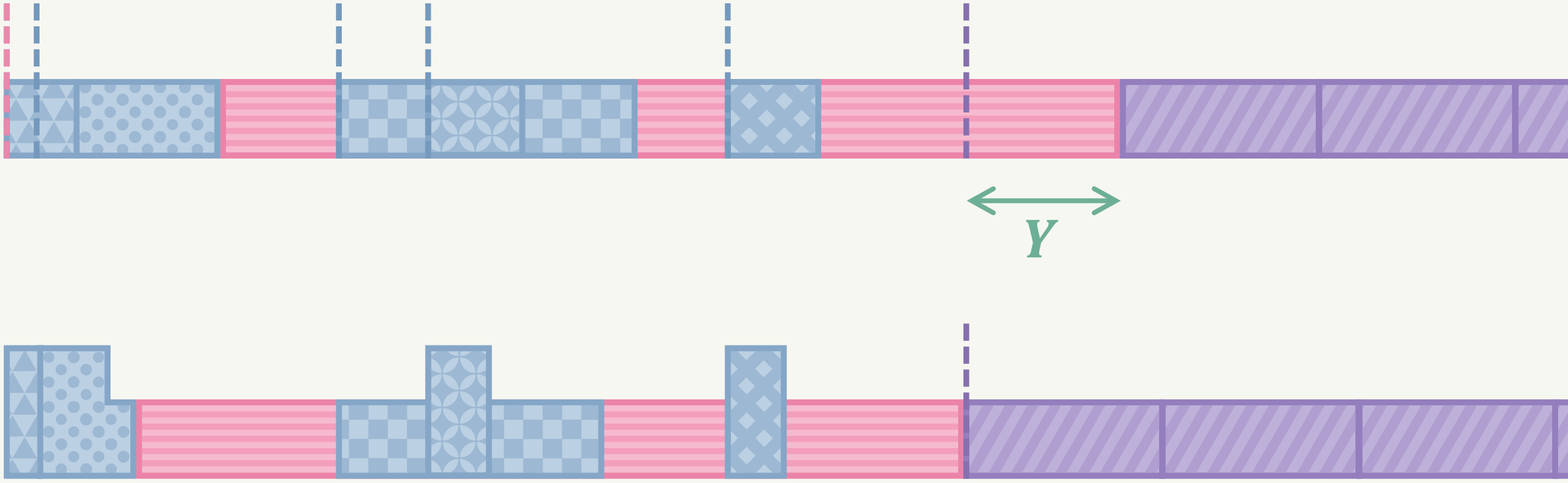
Add **long job** and **jobs with late release time**

NP-Hardness



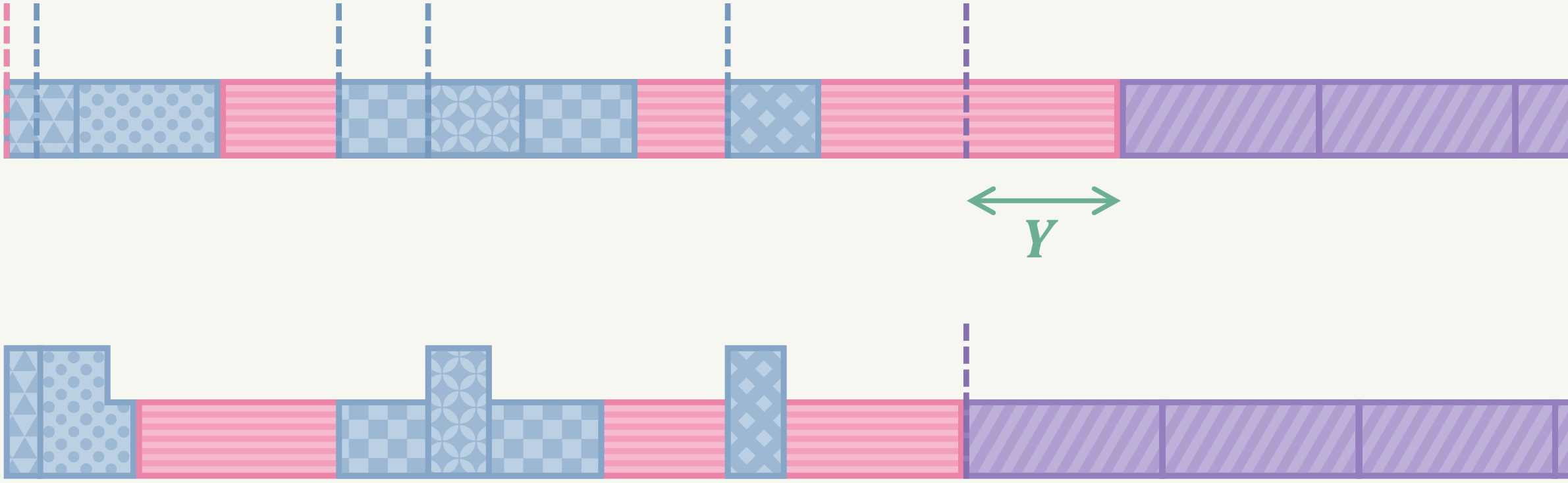
Speeding up **original** jobs is better

NP-Hardness



After **spending budget**:
long job finished before **release time**

NP-Hardness



Change **cost** of higher speed:

Speed up **only worth it** if flow of **all added jobs** improves.

Complexity landscape

	Flow + energy		Budget	
	Same size	**Arbitrary size**	**Same size**	**Arbitrary size**
Unweighted	P [1]	NP-hard	P [1]	NP-hard [1]
Weighted	NP-hard	NP-hard [2]	NP-hard [1]	NP-hard [2]

[1] Barcelo et al., 2015
[2] Labetoulle et al., 1984

In conclusion

- All variants are **NP-hard**,
  except if all jobs are the **same size** and **unweighted**

- With a **fixed completion order**,
  any variant can be **solved with an LP**