Parameterized complexity of scheduling problems with precedence delays

Maher Mallem[1], Claire Hanen[2,3], Alix Munier-Kordon[2]

[1]Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France,
[2]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France,
[3]Université Paris Nanterre, F-92000 Nanterre, France,

12 June 2025

Outline

Outline

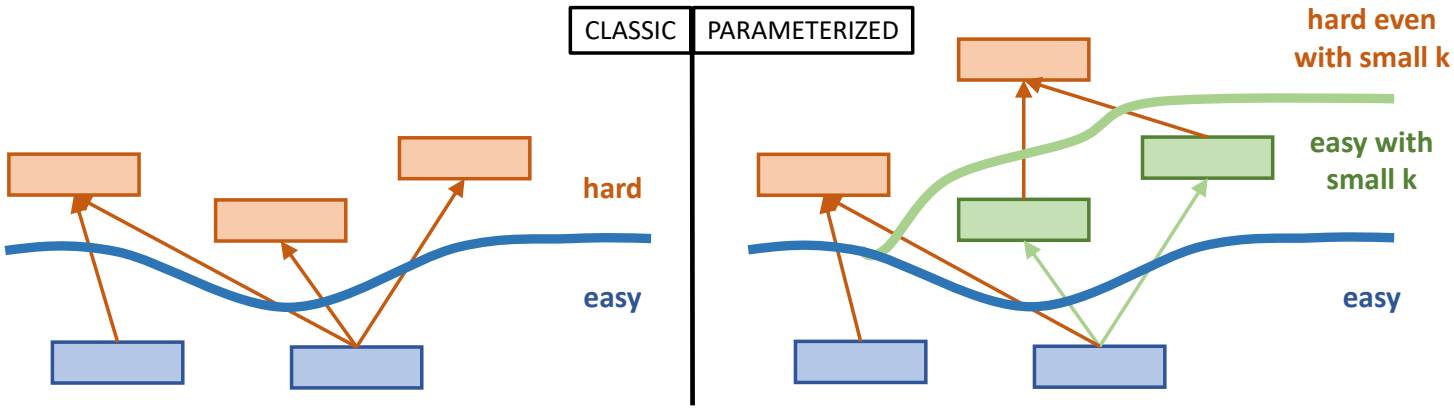
1. Introduction
   - The Problem
   - Parameterized Complexity
2. Parameterized Hardness
   - With Unbounded Precedence Delays
   - With Bounded Precedence Delays
3. Dynamic Programming Approach
   - Without Precedence Delays
   - With Precedence Delays
4. Conclusion

Motivation

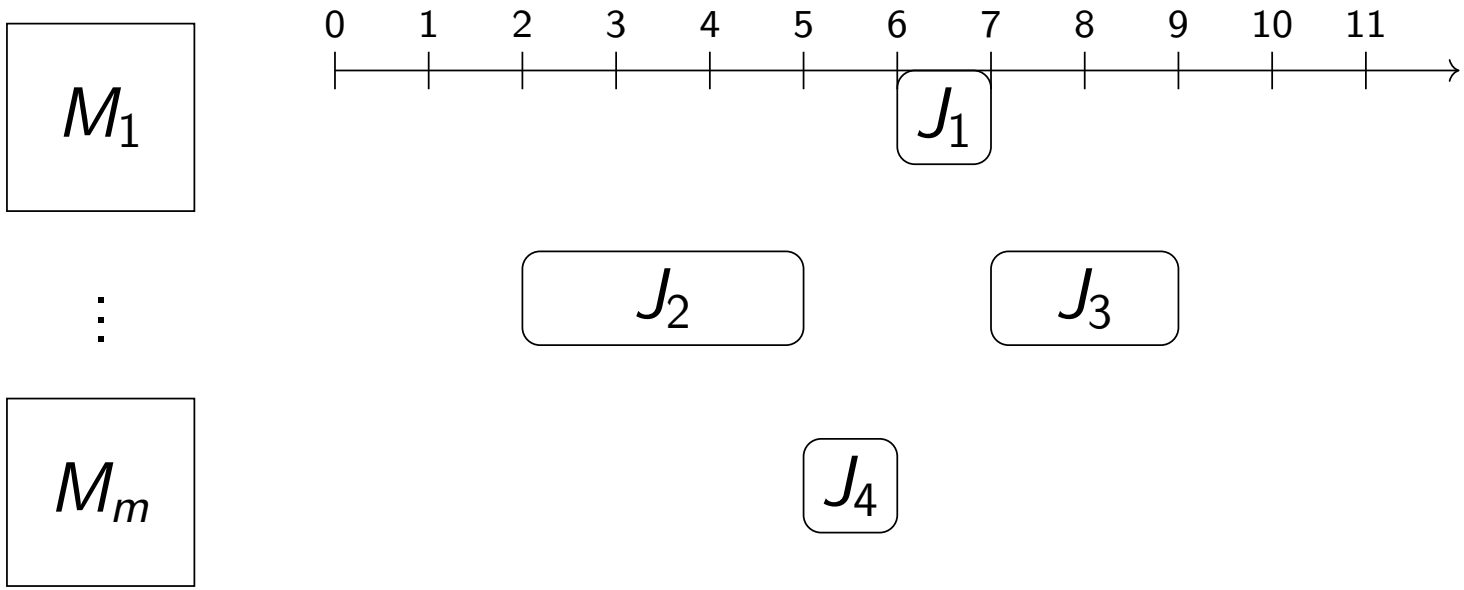
- From the "My Ph.D. thesis in 180s" session at EDITE day (2023):

Complexité paramétrée et nouveaux schémas énumératifs efficaces pour le RCPSP

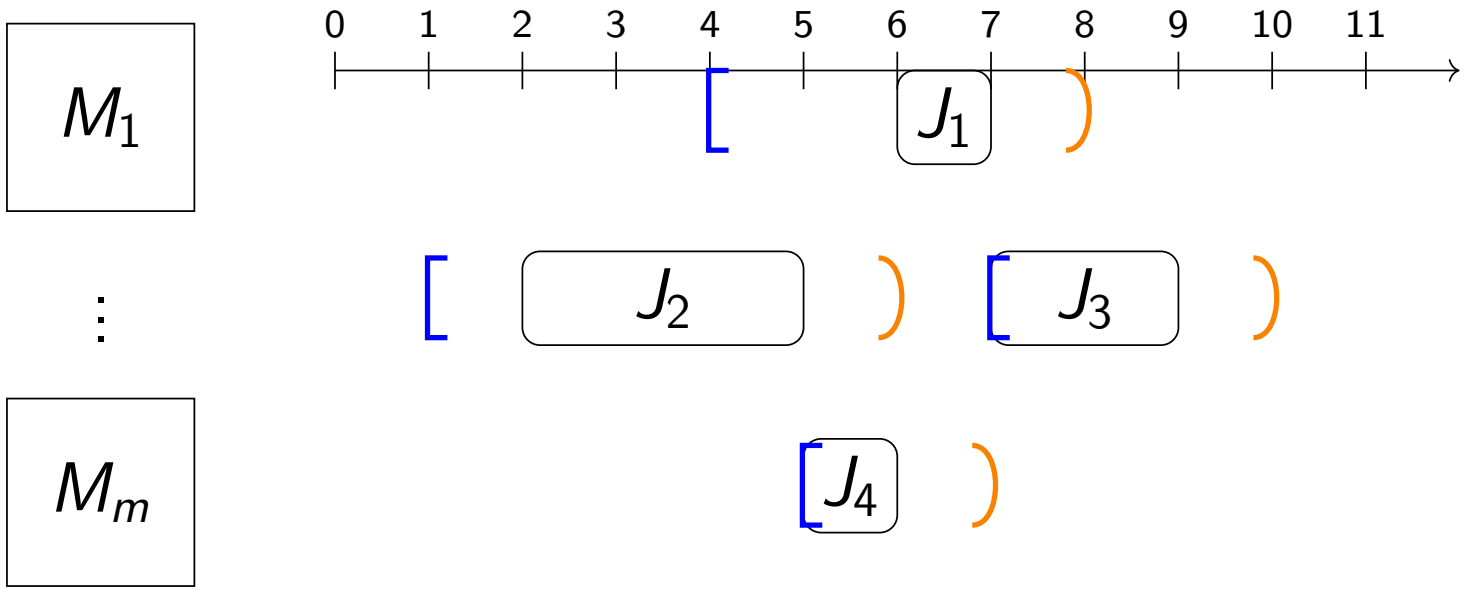
Maher MALLEM (LIP6, SU) sous la direction de Claire HANEN (UPL ; LIP6, SU)



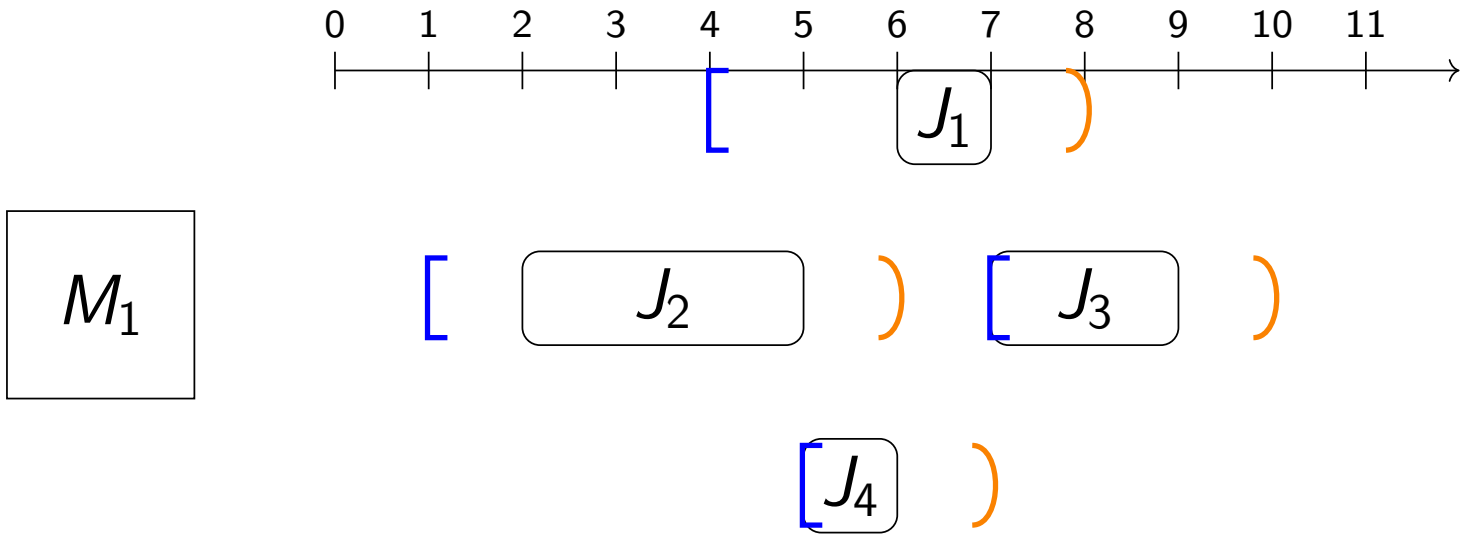
SORBONNE UNIVERSITÉ cnrs LIP6 Université Paris Nanterre



n jobs to schedule on m machines $M_1, \ldots, M_m$.

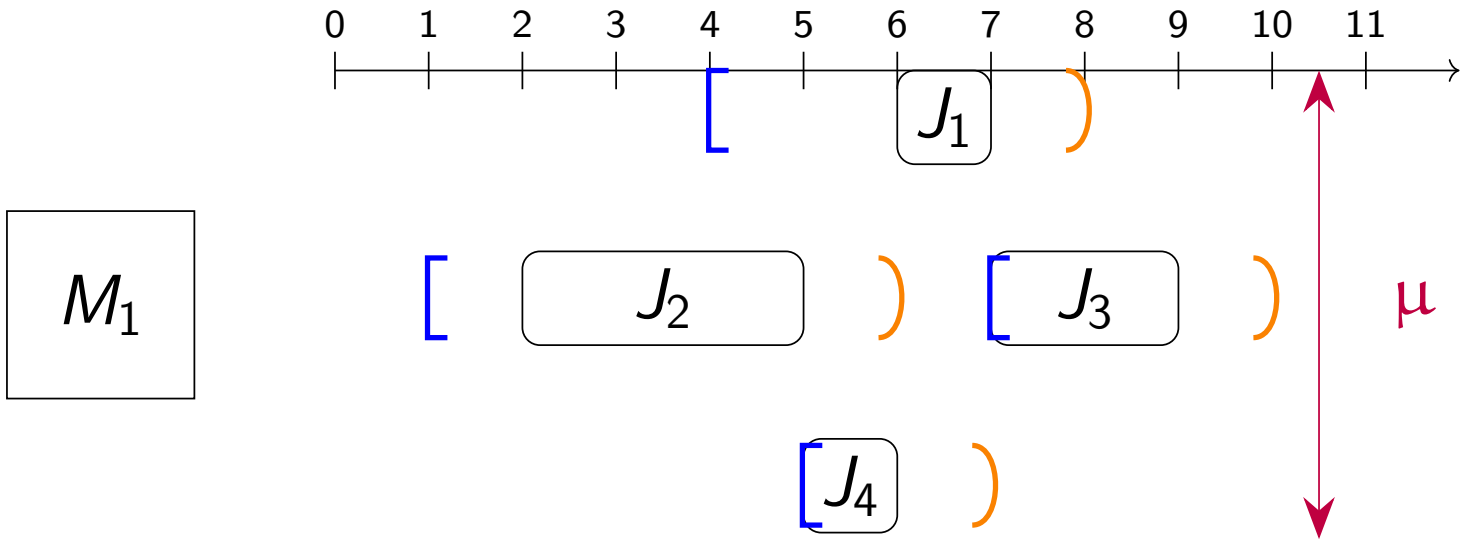


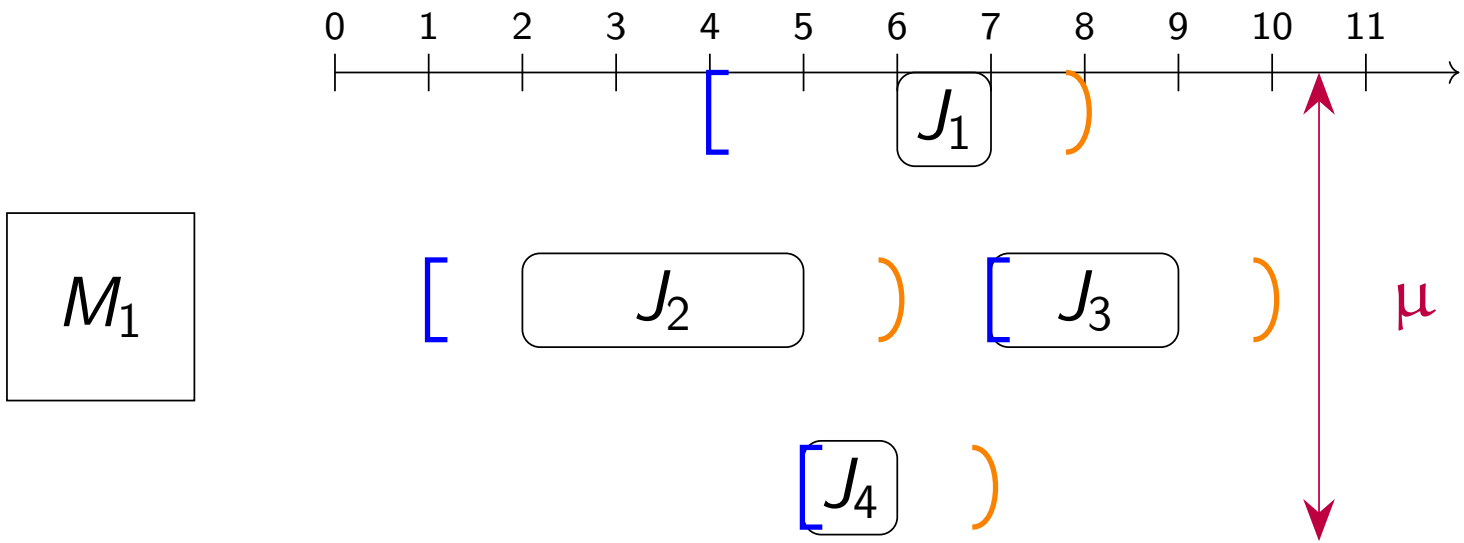
Time windows $[r_j, d_j)$.



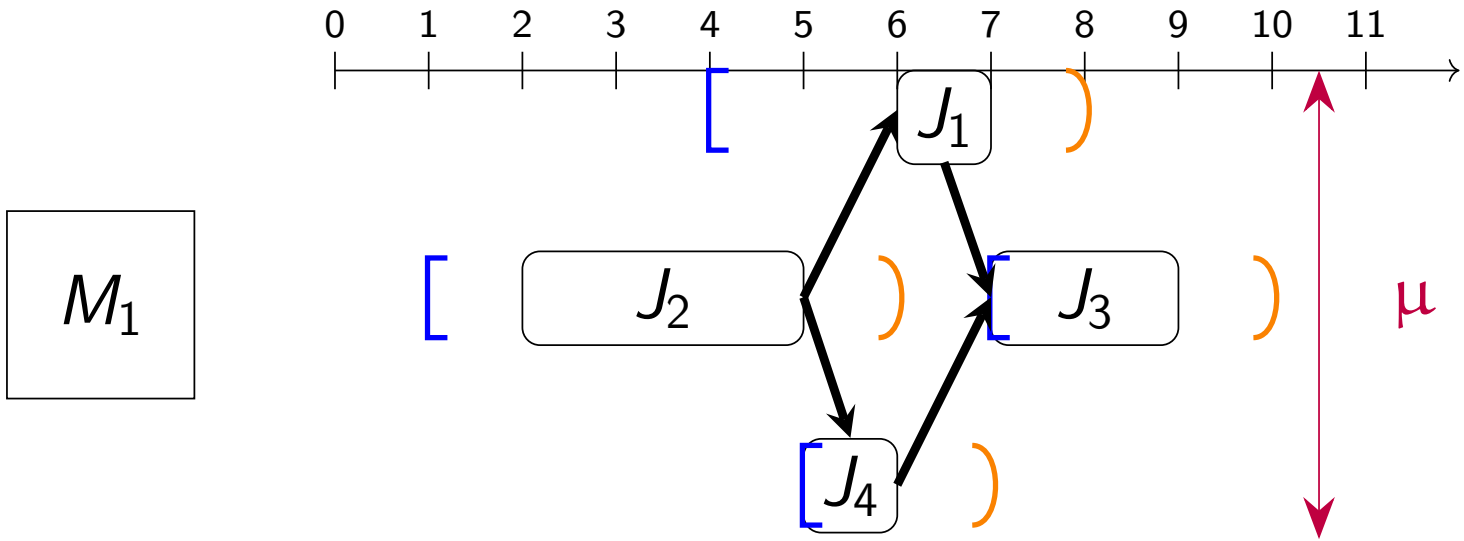
[Lenstra et al. '77] 1-machine with $[r_j, d_j)$: strongly NP-complete.

Ph.D. thesis: scheduling and parameterized complexity

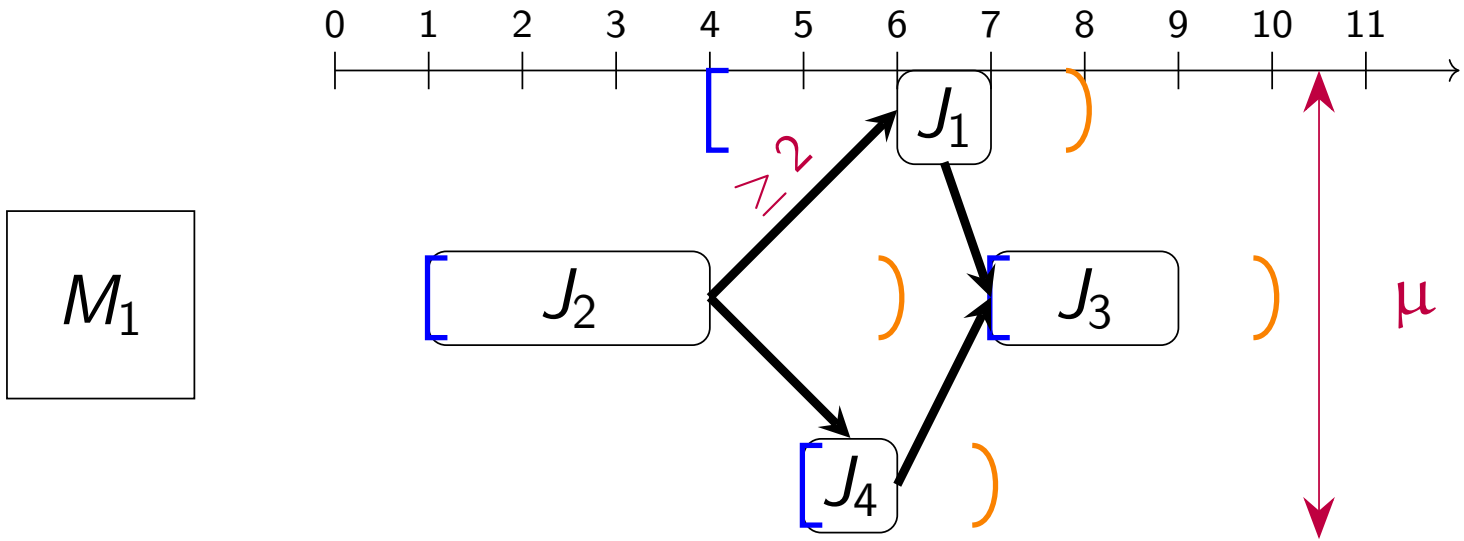




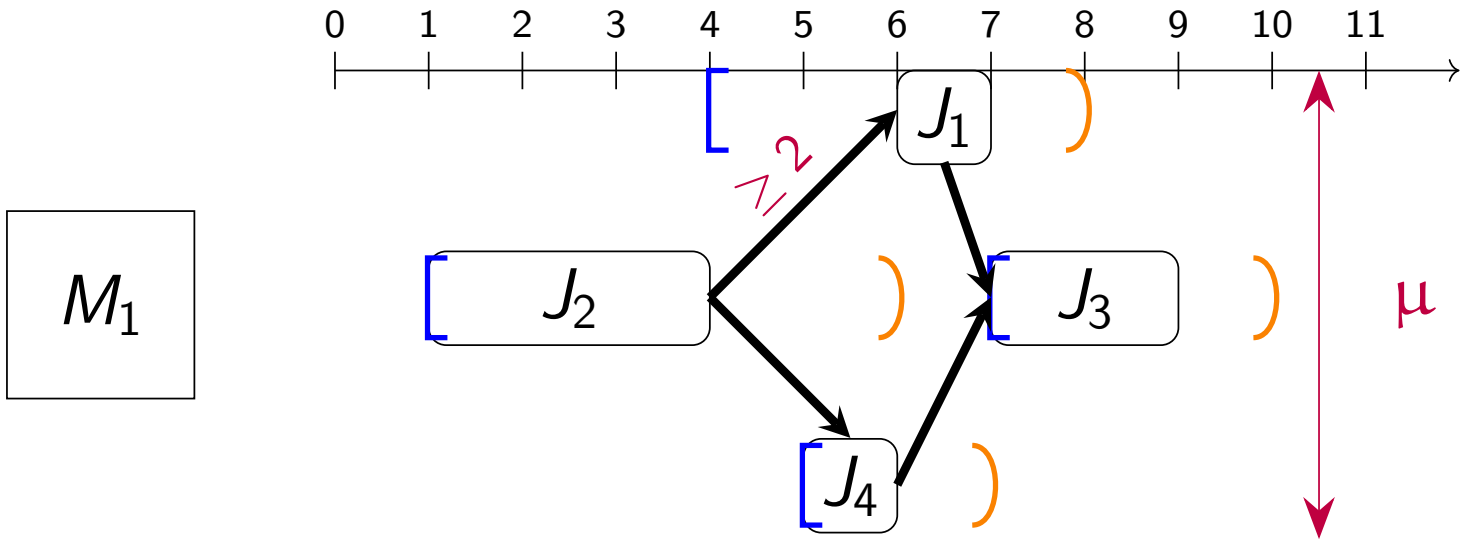
Exact resolution in FPT time:
$\mathcal{O}(\mu\ log(\mu)\ 4^{\mu} \cdot n + n^2)$.



Exact resolution in FPT time:
$\mathcal{O}(\mu^2\ 4^\mu \cdot n + n^2)$.



Min. delay $J_i \xrightarrow{\geq \ell} J_j$: $\text{start}(J_j) \geq \ell + \text{end}(J_i)$.



Min. delay $J_i \xrightarrow{\geq \ell} J_j$: $\text{start}(J_j) \geq \ell + \text{end}(J_i)$.

Para-NP-hard (NP-hard when $\mu = 3$).

Going beyond NP-hardness

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
- P: deterministic time $poly(|\mathcal{I}|)$.
- NP: nondeterministic time $poly(|\mathcal{I}|)$.

Going beyond NP-hardness

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
  - Problem $(\mathcal{P}, k)$, instance $\mathcal{I}$, parameter $k : \mathcal{P} \to \mathbb{N}_0$.
- P: deterministic time $poly(|\mathcal{I}|)$.
  - FPT: deterministic time $f(k(\mathcal{I})) \cdot poly(|\mathcal{I}|)$.
- NP: nondeterministic time $poly(|\mathcal{I}|)$.

Going beyond NP-hardness

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
  - Problem $(\mathcal{P}, k)$, instance $\mathcal{I}$, parameter $k : \mathcal{P} \to \mathbb{N}_0$.
- P: deterministic time $poly(|\mathcal{I}|)$.
  - FPT: deterministic time $f(k(\mathcal{I})) \cdot poly(|\mathcal{I}|)$.
- NP: nondeterministic time $poly(|\mathcal{I}|)$.
  - para-NP: nondeterministic time $f(k(\mathcal{I})) \cdot poly(|\mathcal{I}|)$.

Going beyond NP-hardness

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
  - Problem $(\mathcal{P}, k)$, instance $\mathcal{I}$, parameter $k : \mathcal{P} \to \mathbb{N}_0$.
- P: deterministic time $poly(|\mathcal{I}|)$.
  - FPT: deterministic time $f(k(\mathcal{I})) \cdot poly(|\mathcal{I}|)$.
- NP: nondeterministic time $poly(|\mathcal{I}|)$.
  - para-NP: nondeterministic time $f(k(\mathcal{I})) \cdot poly(|\mathcal{I}|)$.

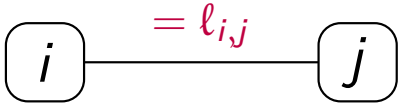
Fact [Flum, Grohe '06]

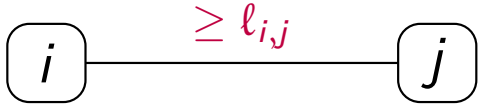
$(\mathcal{P}, k)$ is para-NP-hard if and only if $\mathcal{P}$ is NP-hard for some fixed value k_0 of parameter k.

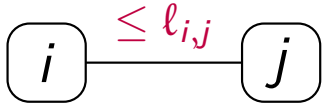
Precedence delay types

- Three precedence delay types studied: exact, minimum and maximum.

Precedence delay types

- Three precedence delay types studied: exact, minimum and maximum.
- If τ is a feasible schedule and $(i \xrightarrow{\ell_{i,j}} j)$:
  - Exact delay: 

  - Minimum delay: 

  - Maximum delay: 

Outline

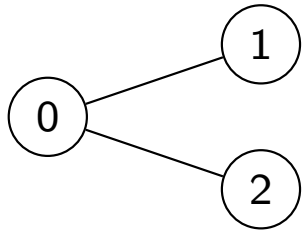
Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

- Reduction from 3-COLORING: $G = (V, E)$.

Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

- Reduction from 3-COLORING: $G = (V, E)$.

Example:



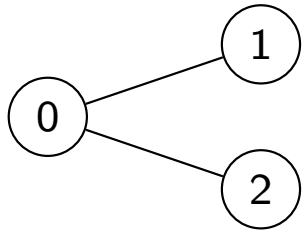
$\longrightarrow$

Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.

Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

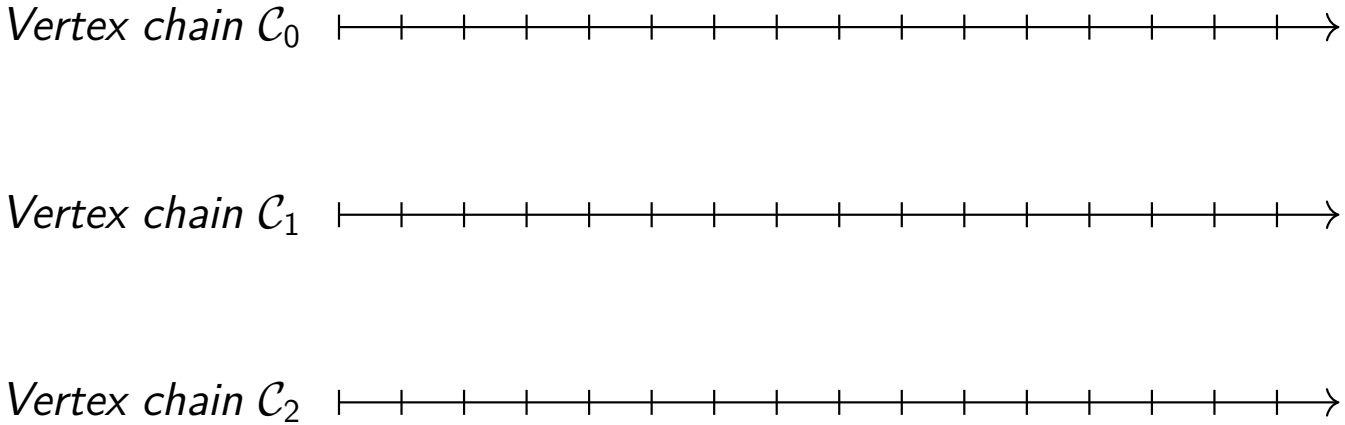
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

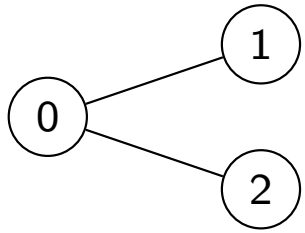
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

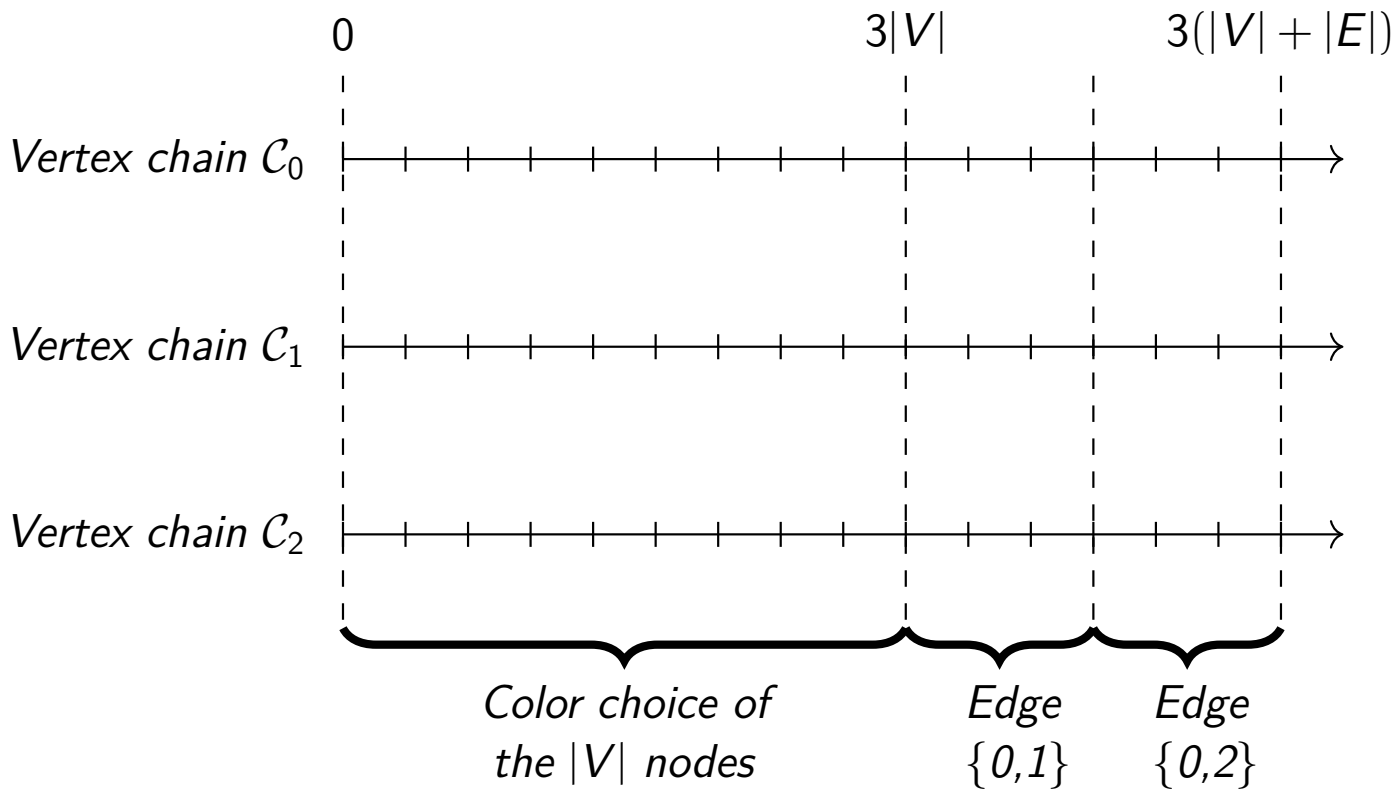
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

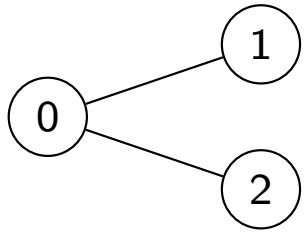
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

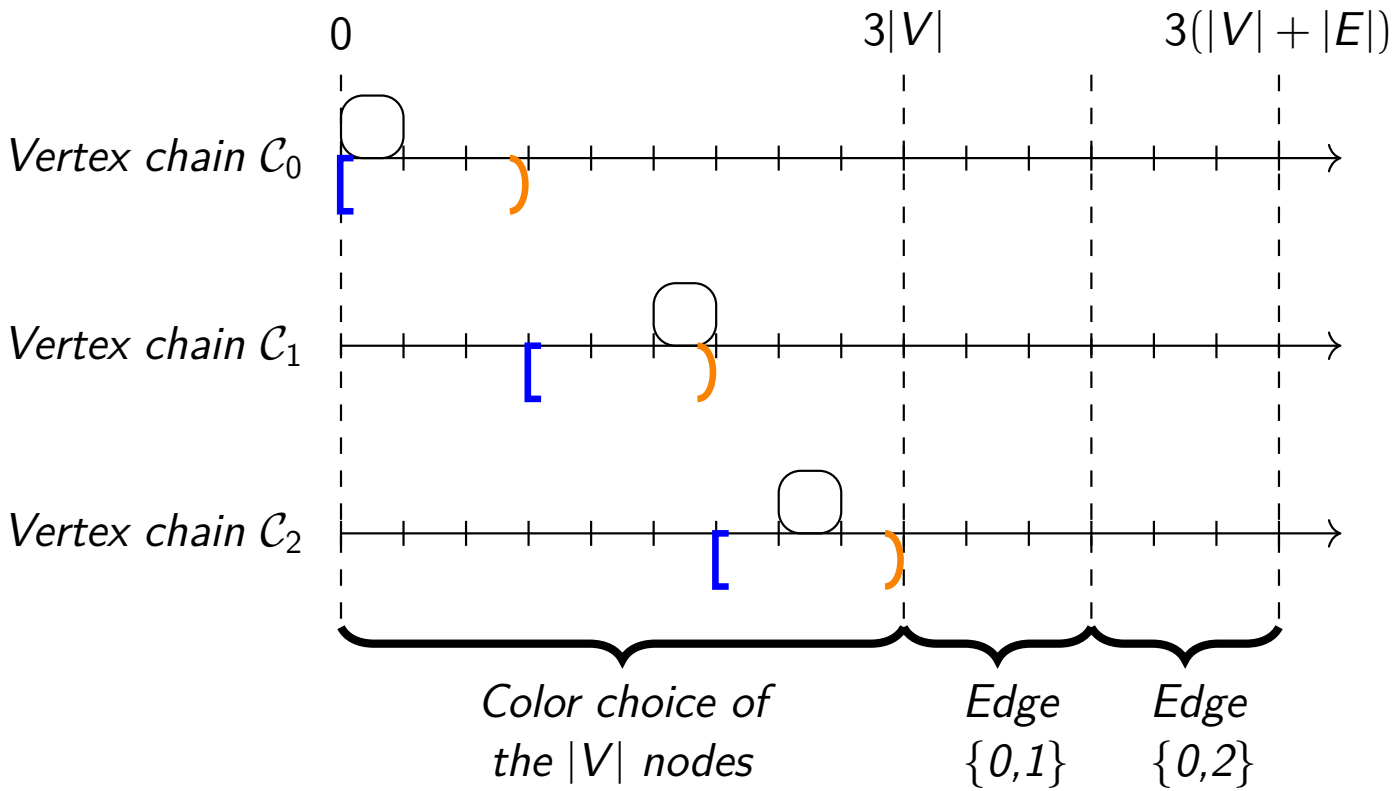
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

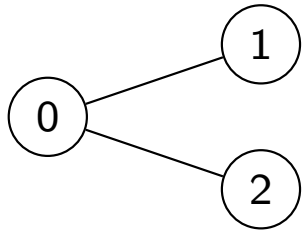
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

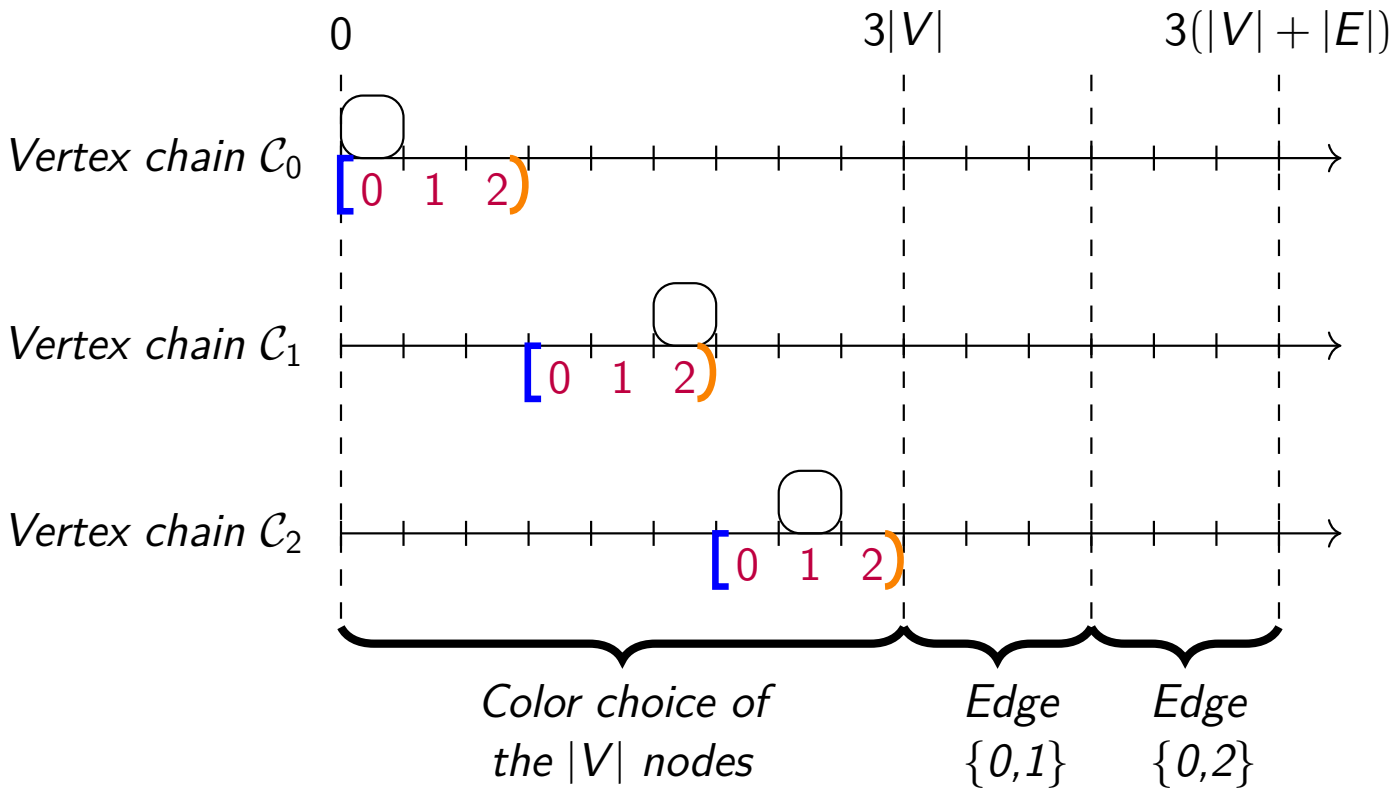
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

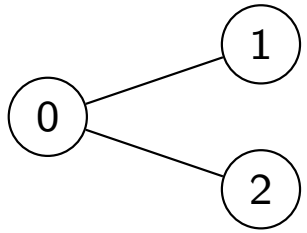
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

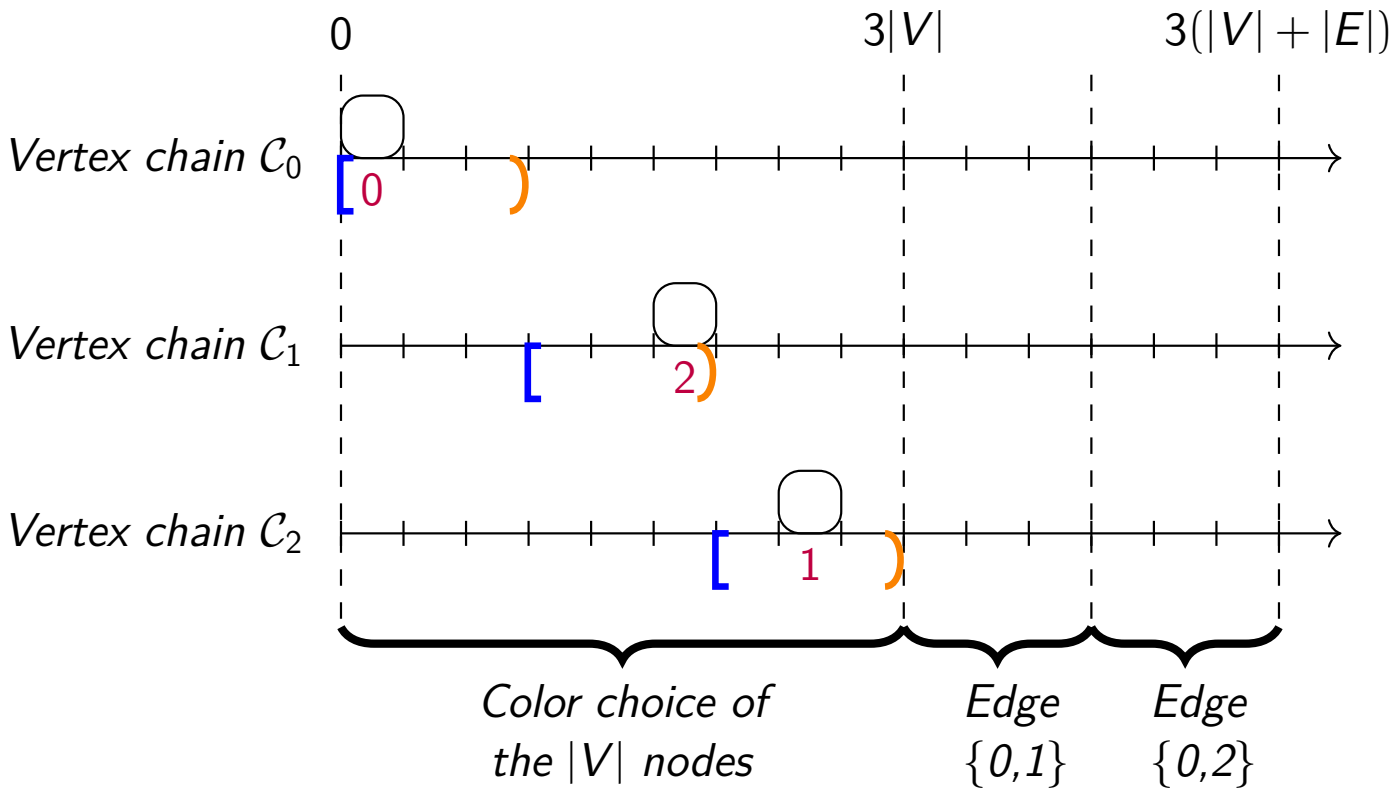
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

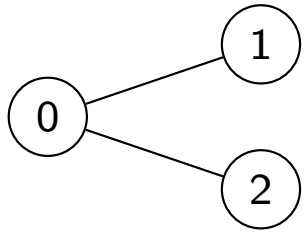
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

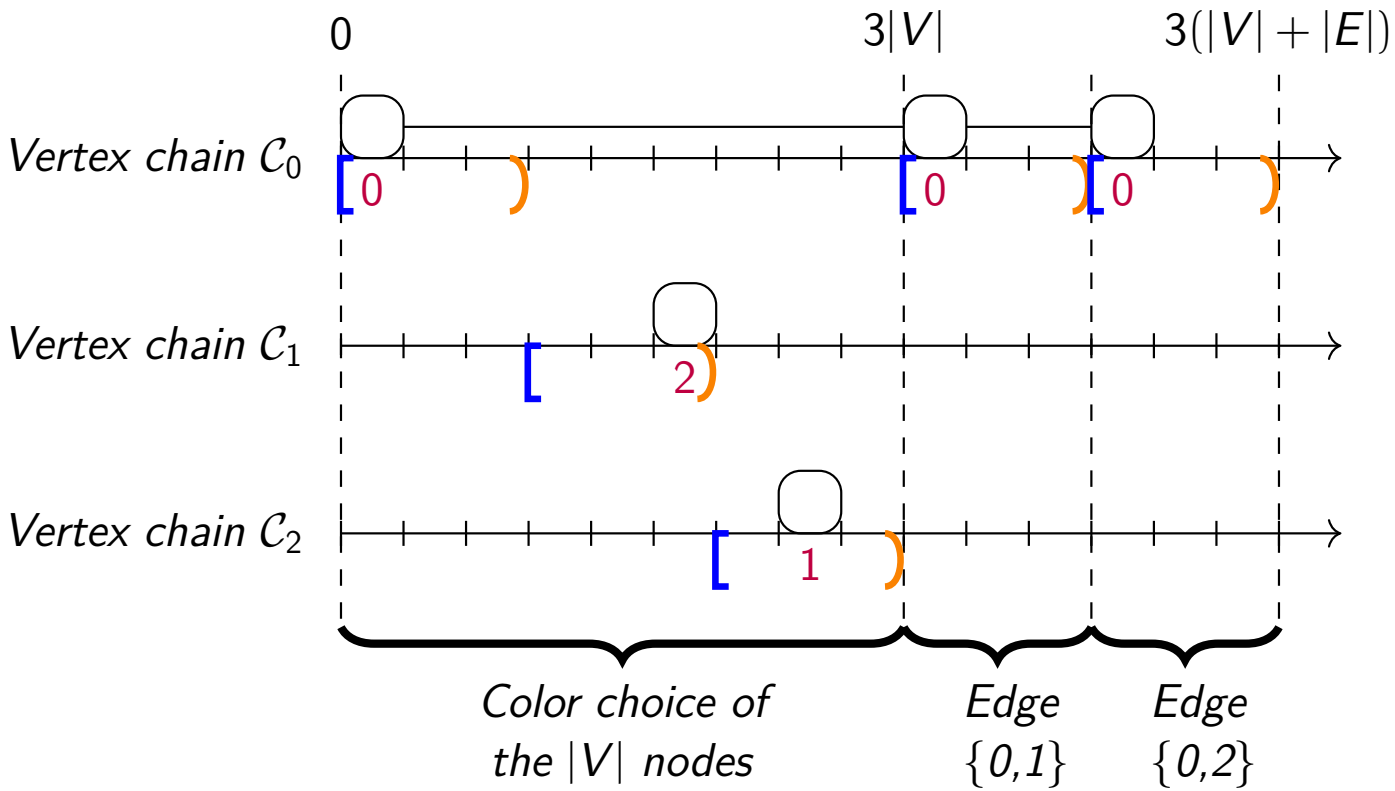
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

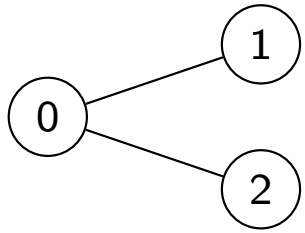
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

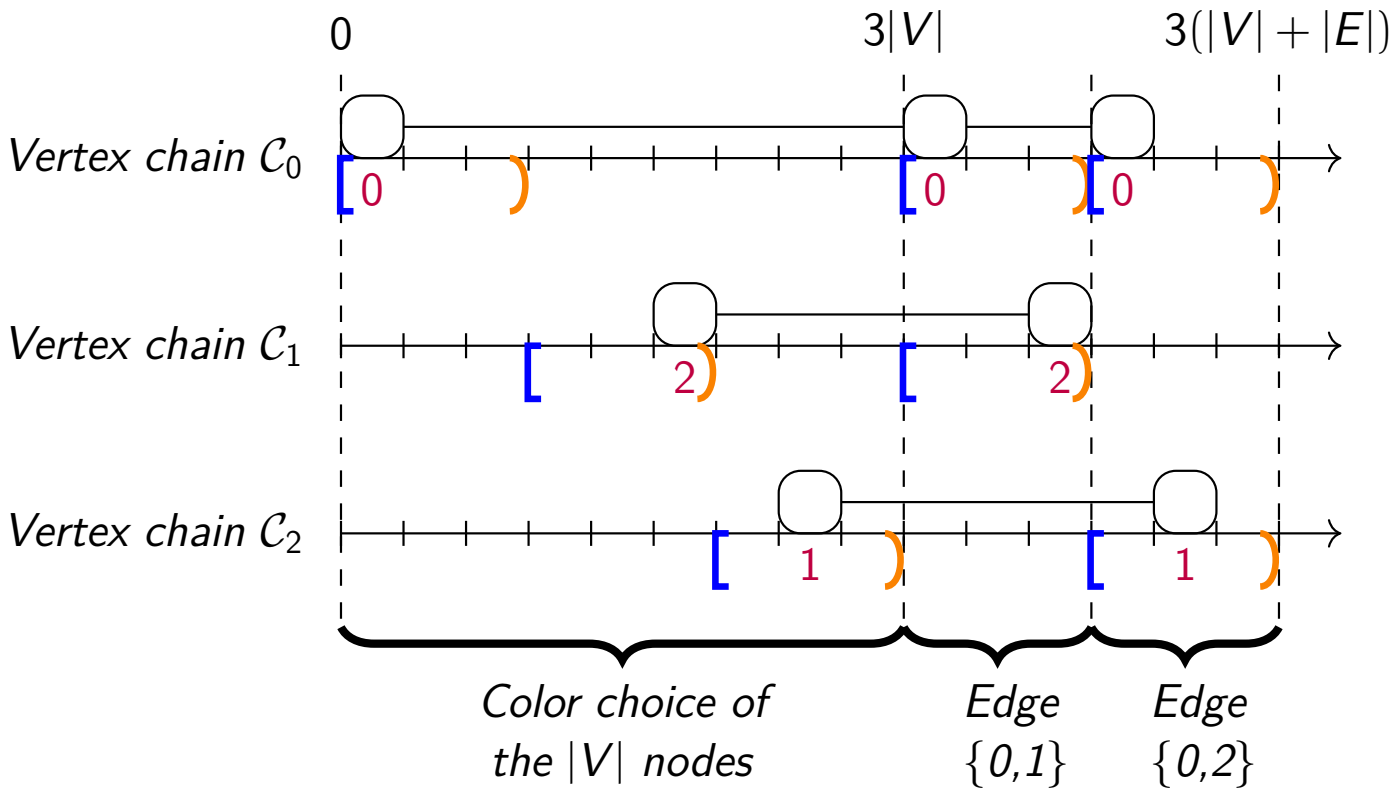
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

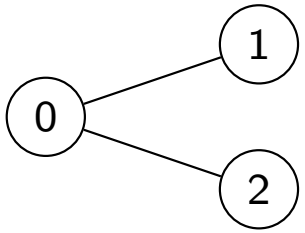
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

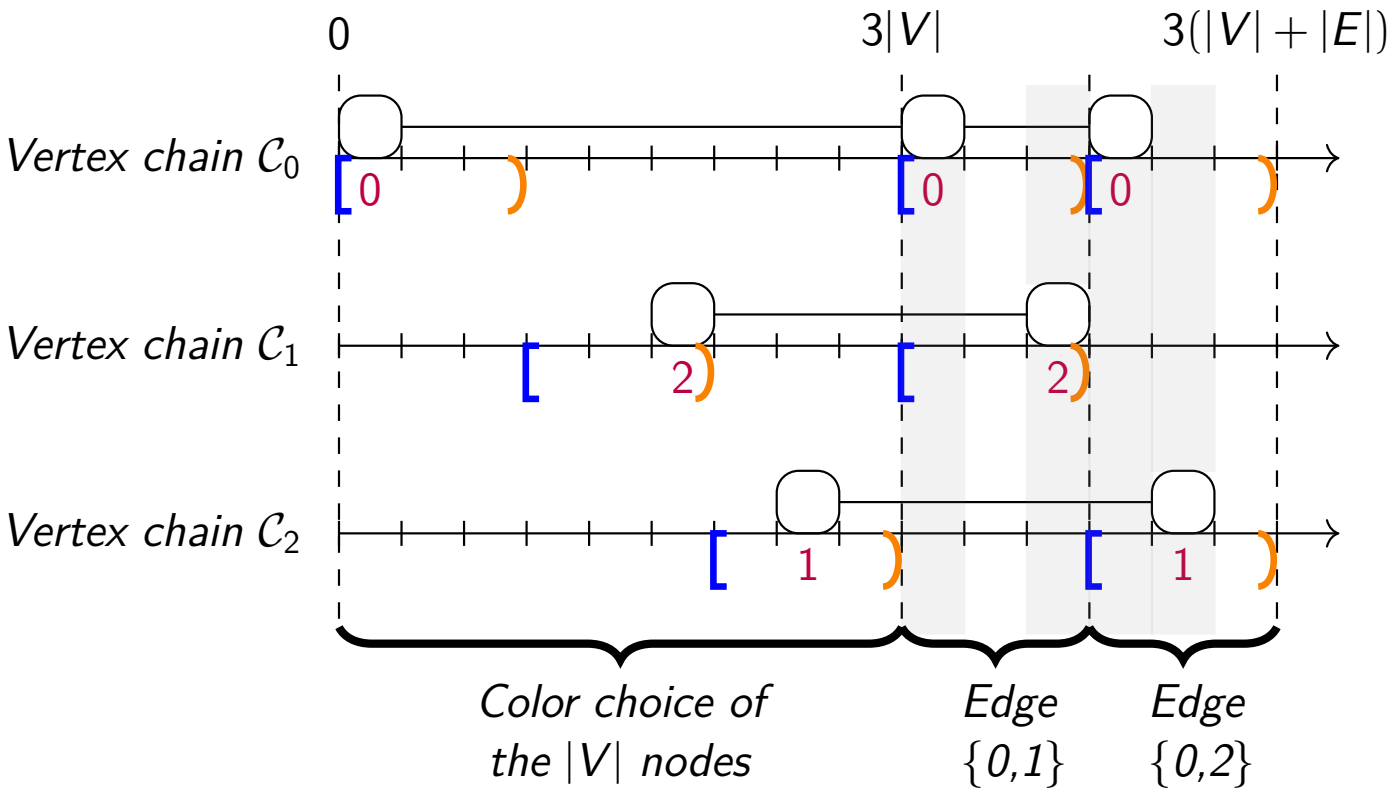
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

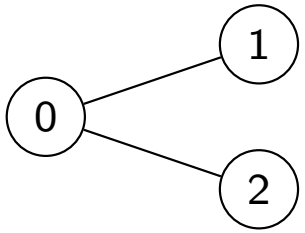
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

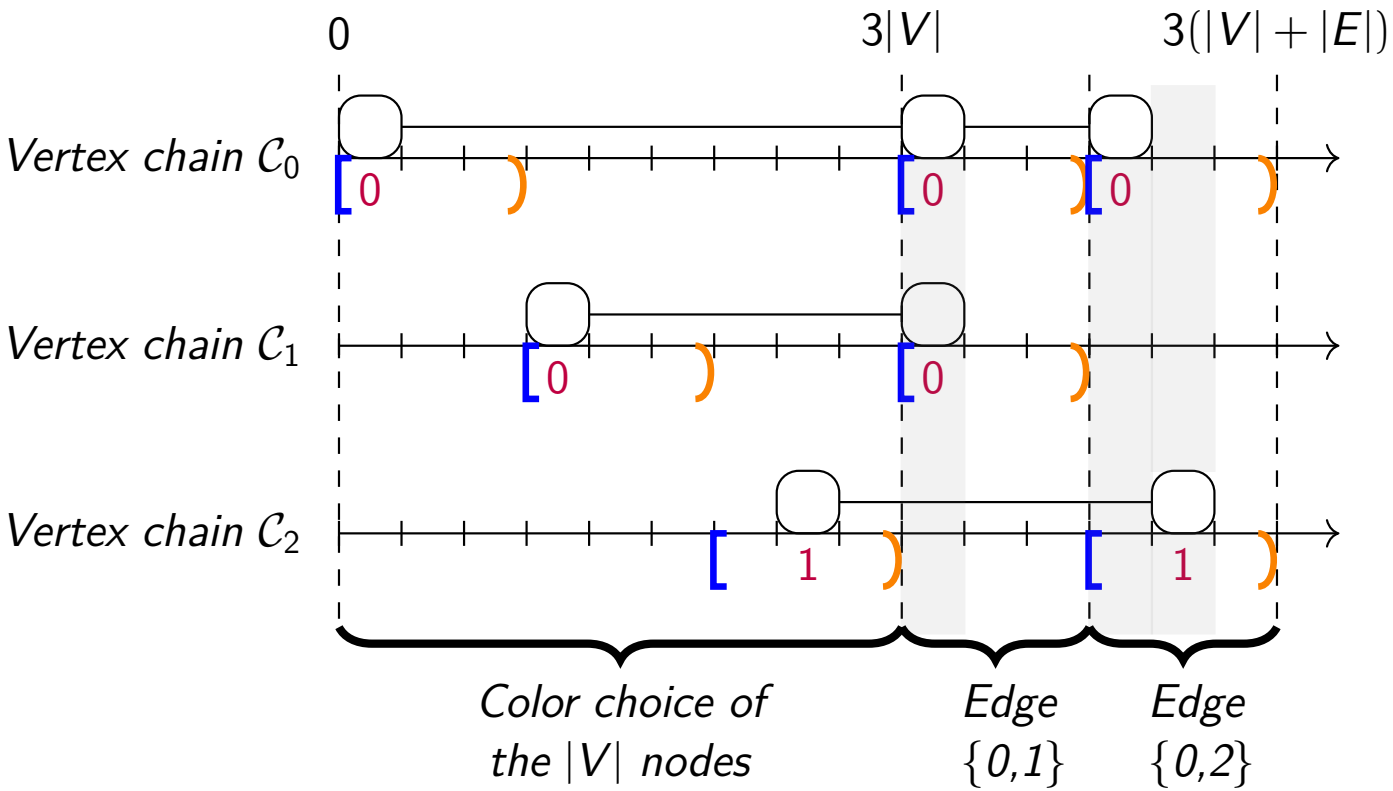
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

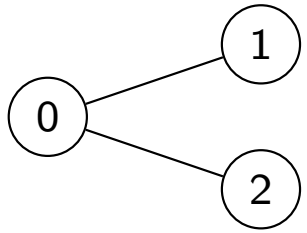
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



Single machine scheduling with precedence delays and fixed pathwidth μ and slack σ is NP-hard.

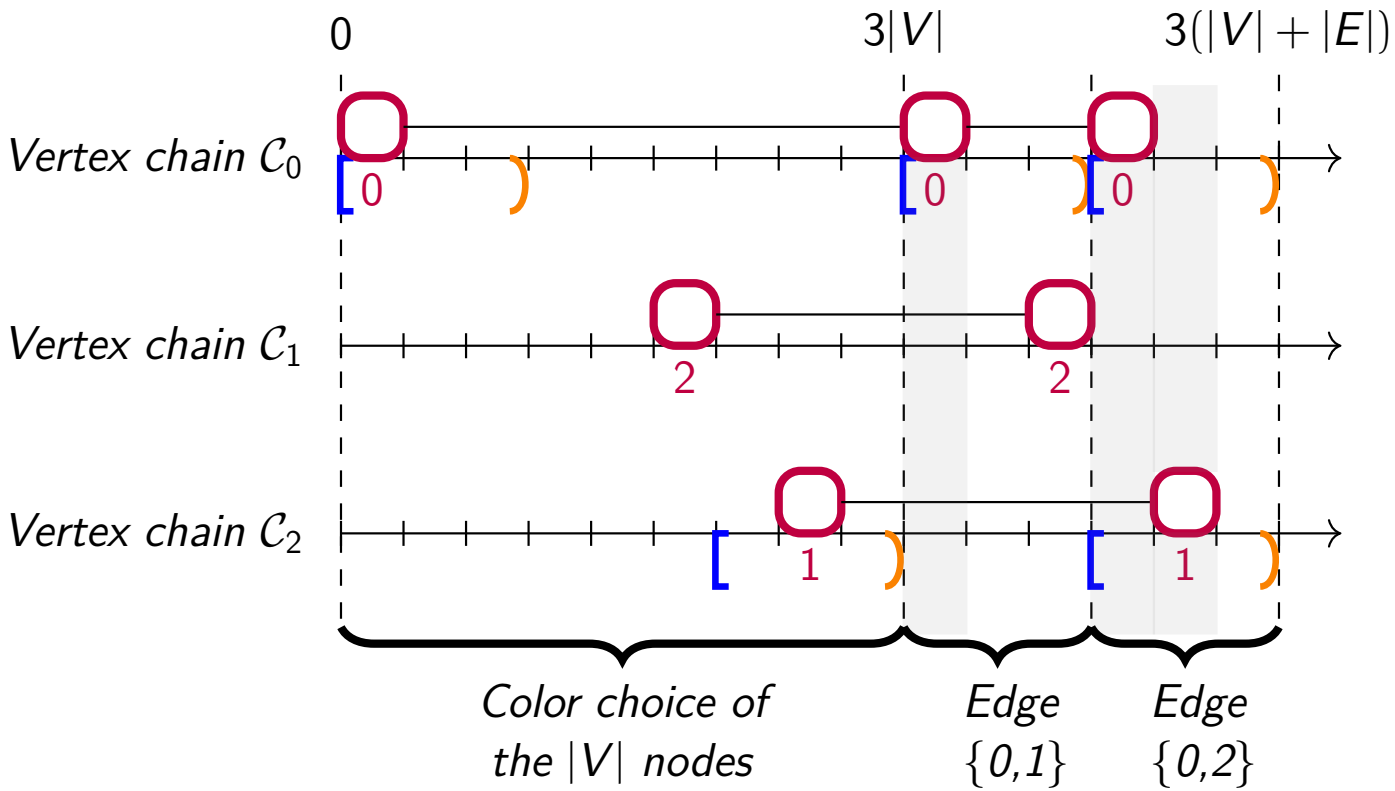
- Reduction from 3-COLORING: $G = (V, E)$.

Example:



$\longrightarrow$

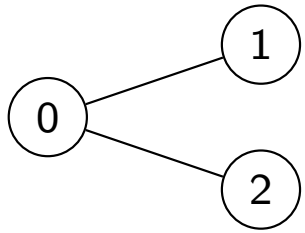
Scheduling instance with one machine, $3|V| + 2|E|$ jobs and exact delays.



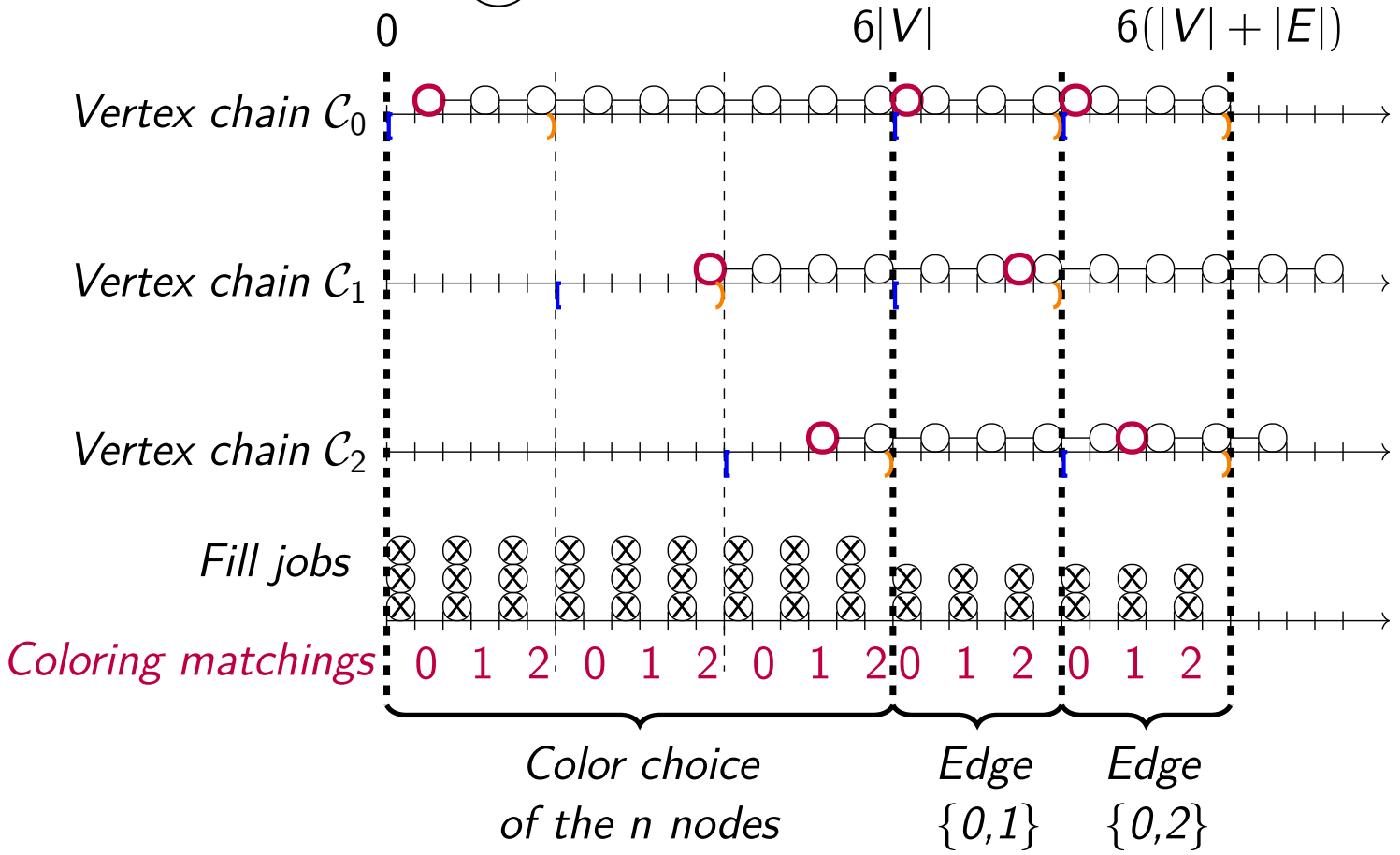
Parallel identical machine scheduling with precedence delays and fixed maximum delay value ℓ_{max} and slack σ is NP-hard.

- Reduction from 3-COLORING: $G = (V, E)$.

Example:

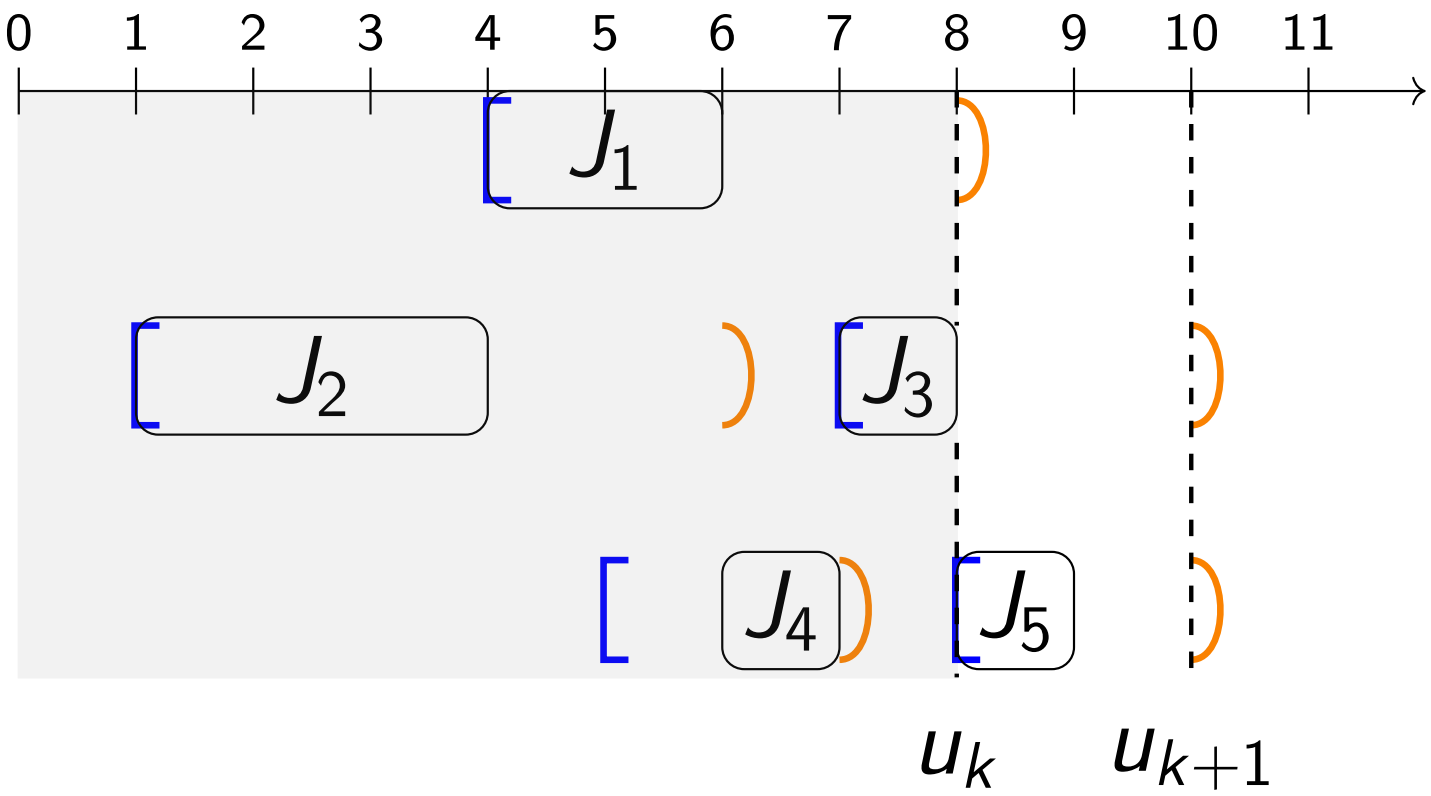


$\longrightarrow$ Scheduling instance with $|V|$ machines and exact delays.

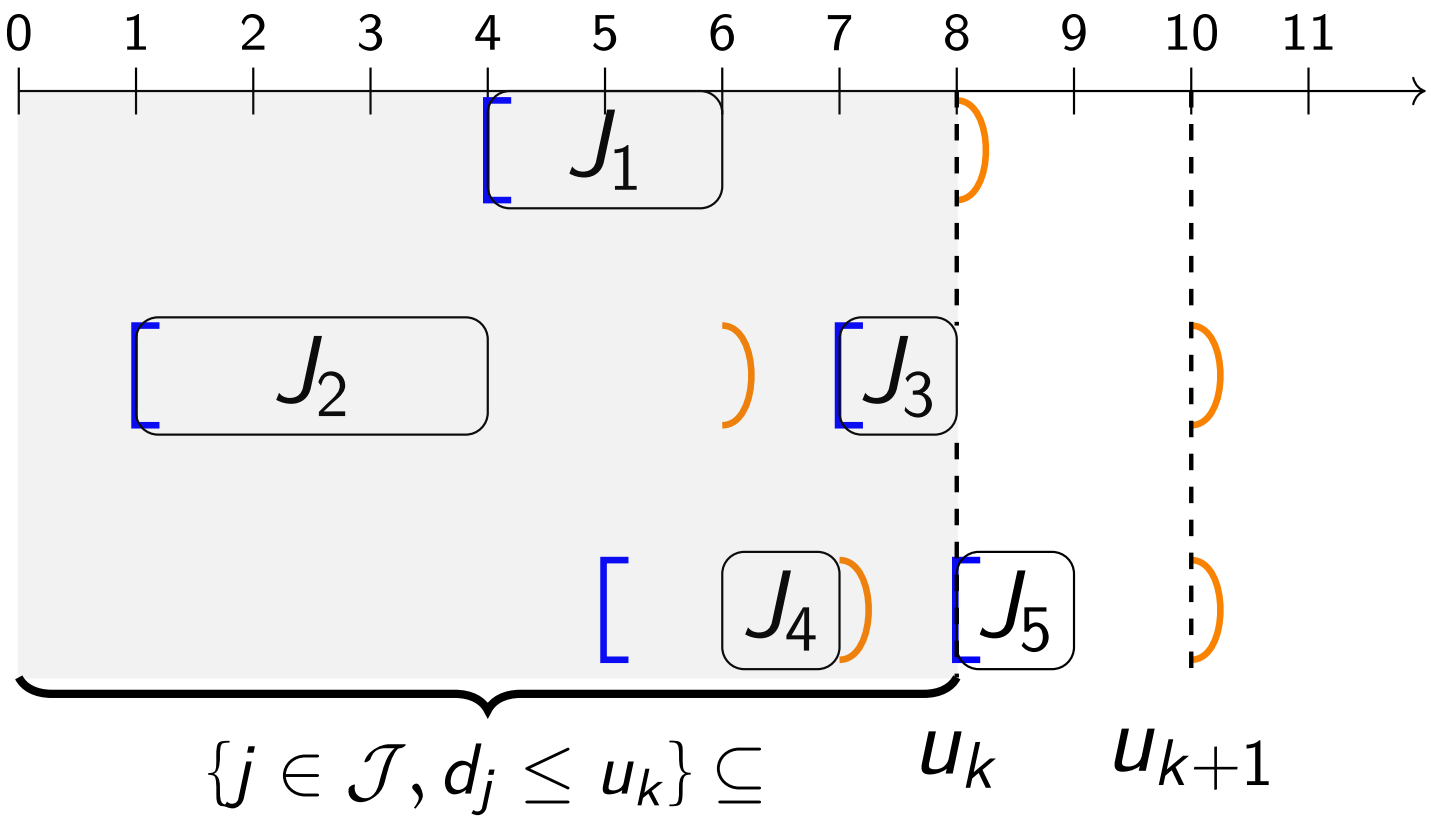


Outline

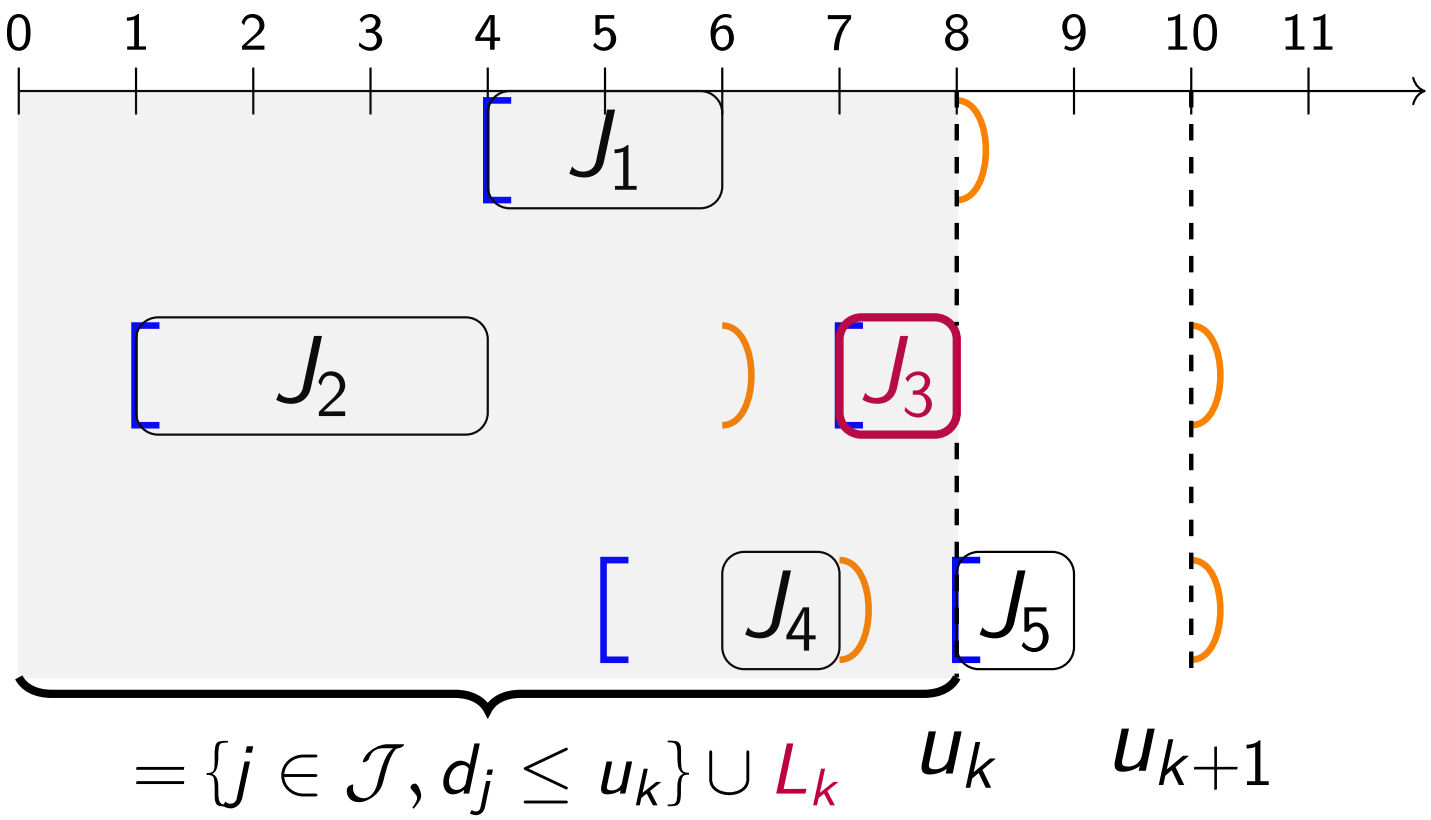
Pathwidth μ



Pathwidth μ

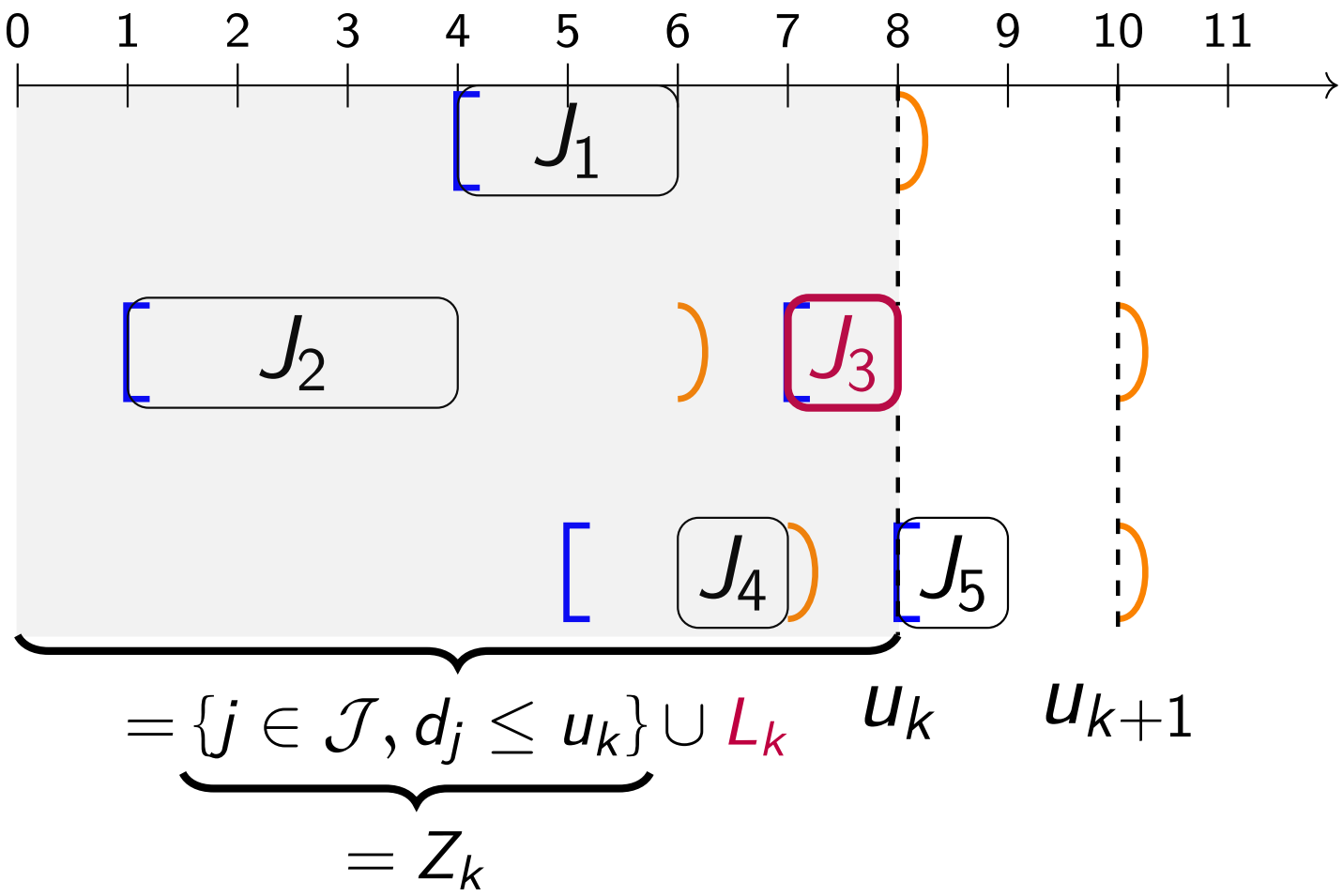


Pathwidth μ



- $J_j \in L_k \implies [u_k, u_{k+1}) \subseteq [r_j, d_j)$: at most μ such jobs.

Pathwidth μ



- $J_j \in L_k \implies [u_k, u_{k+1}) \subseteq [r_j, d_j)$: at most μ such jobs.

Dynamic programming with pathwidth μ

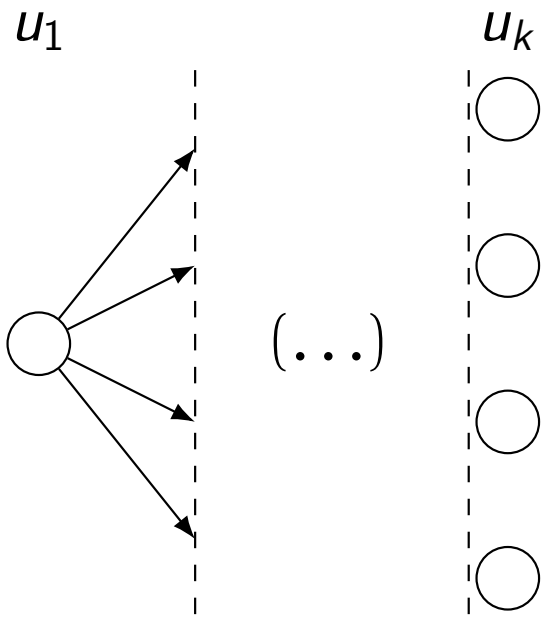
Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.

Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

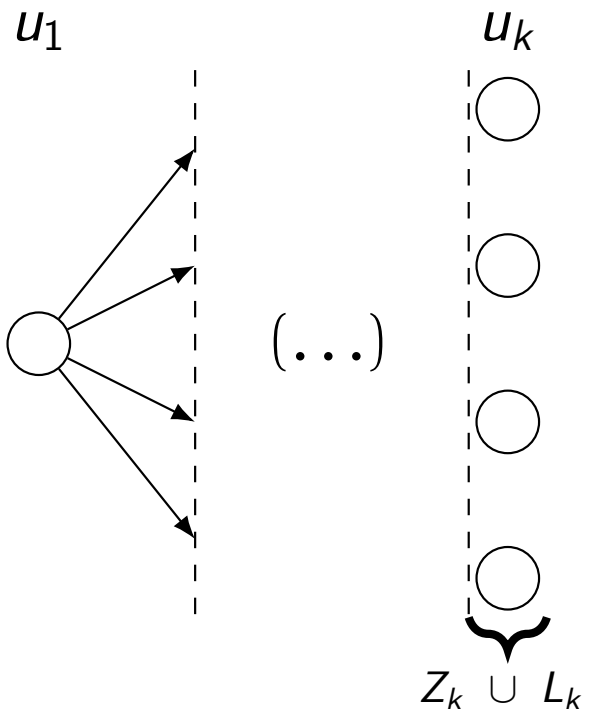
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

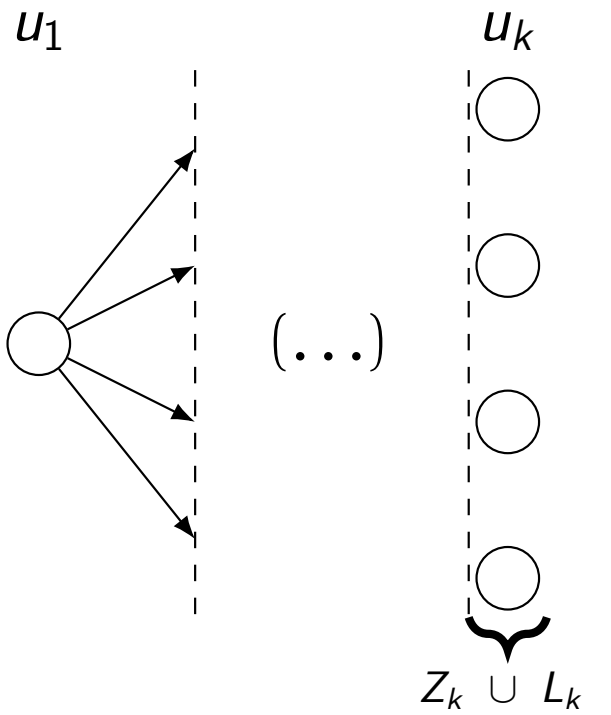
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.

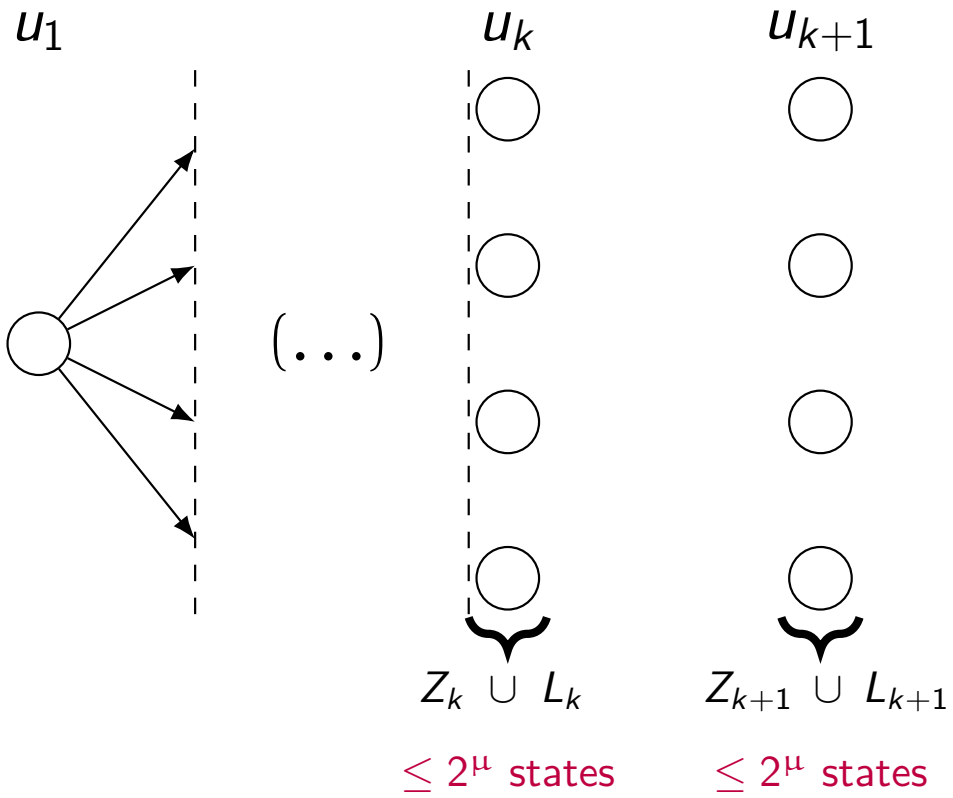


$\leq 2^{\mu}$ states

Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

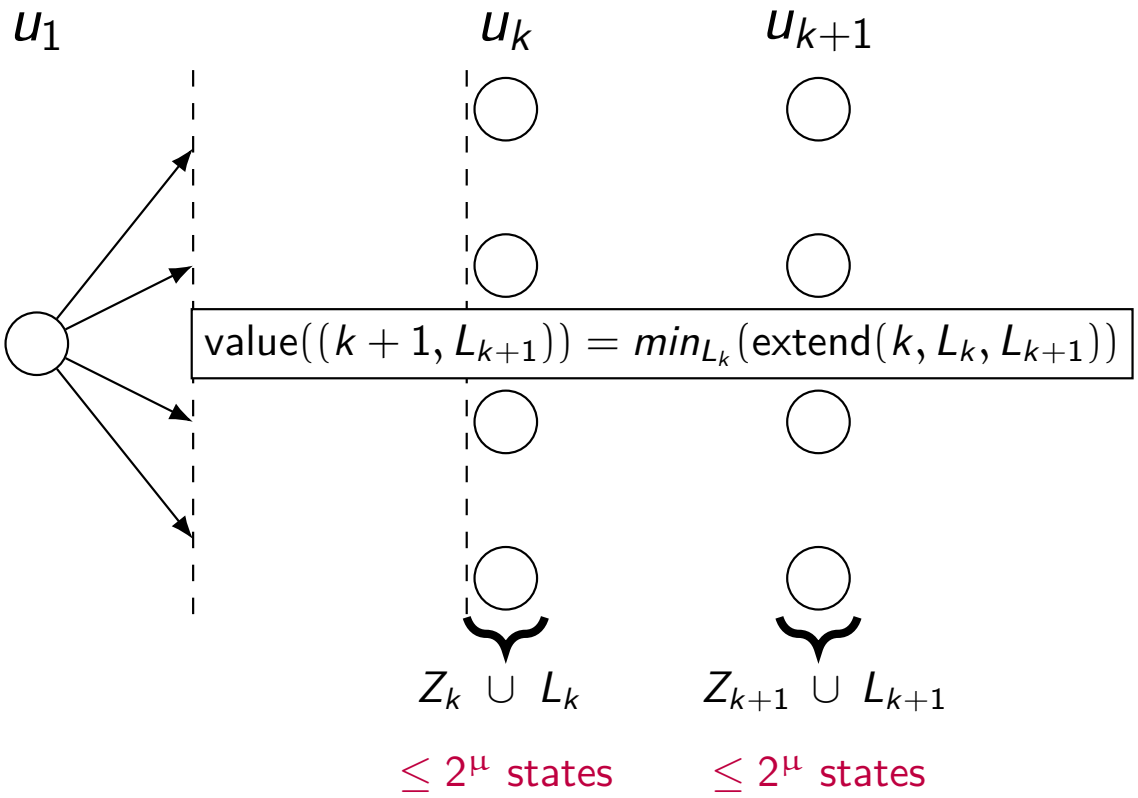
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

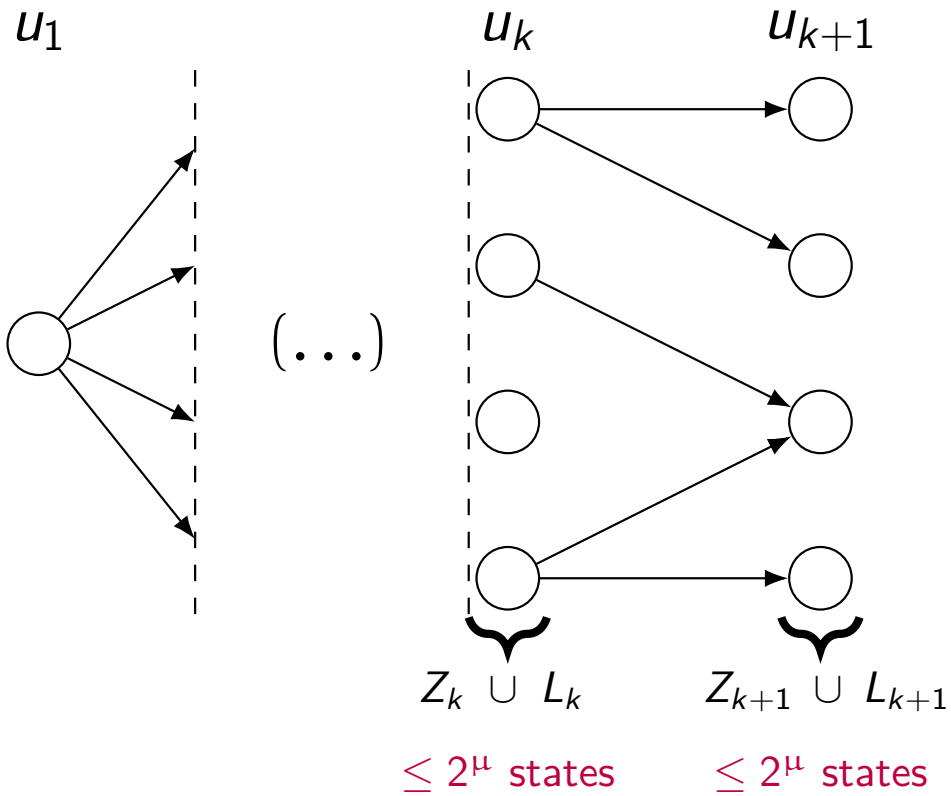
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

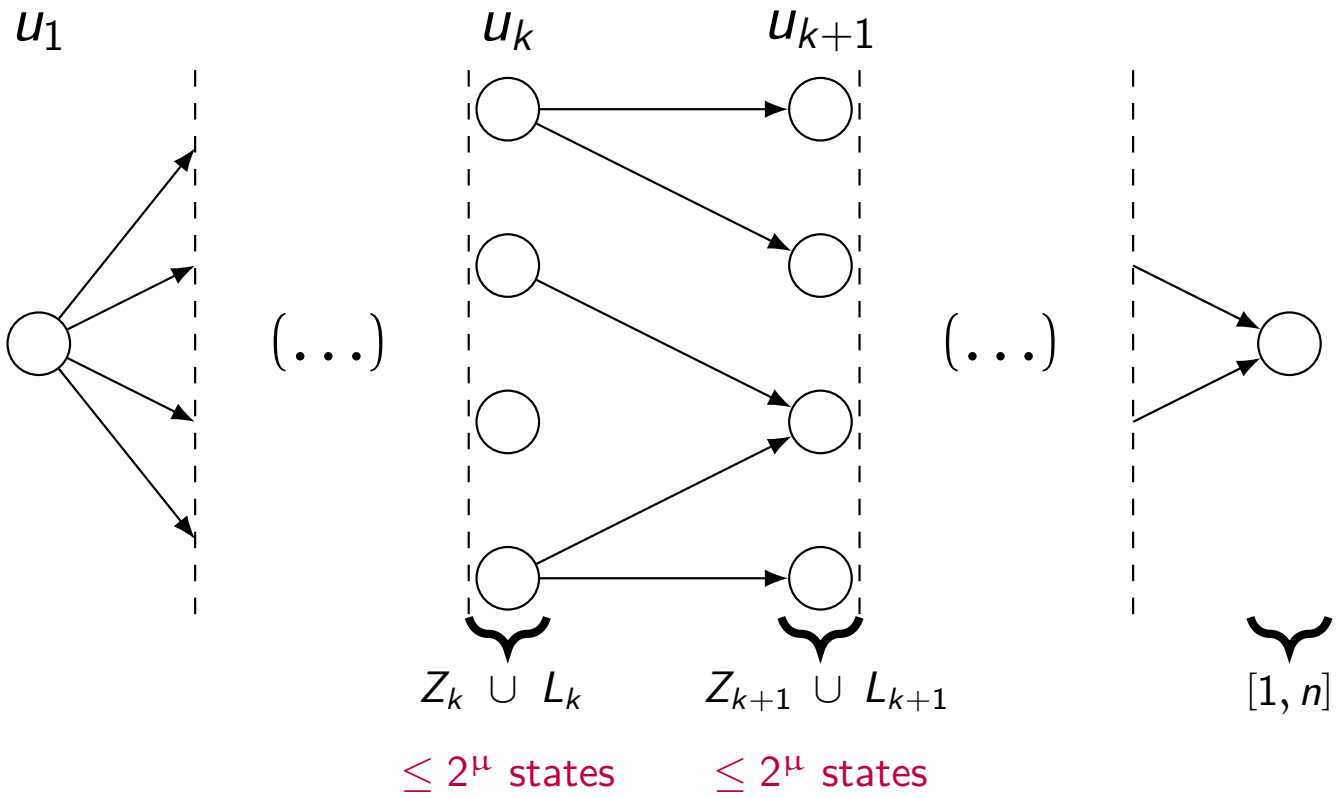
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



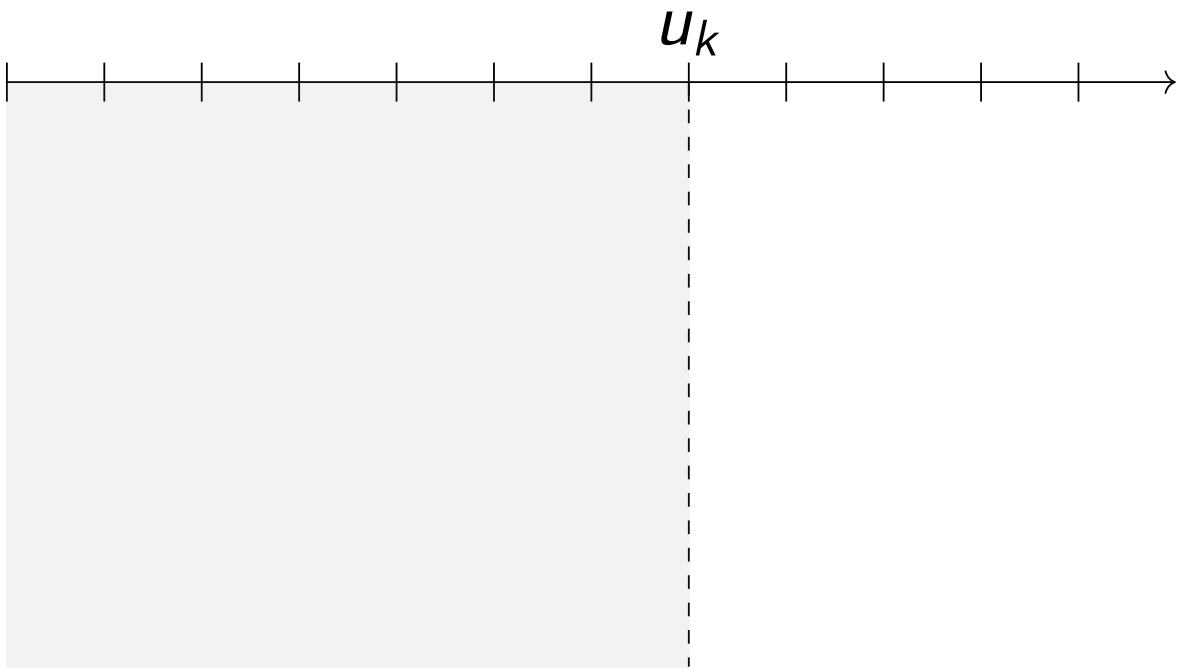
Result example

Theorem [Hanen, Mallem, Munier '22]

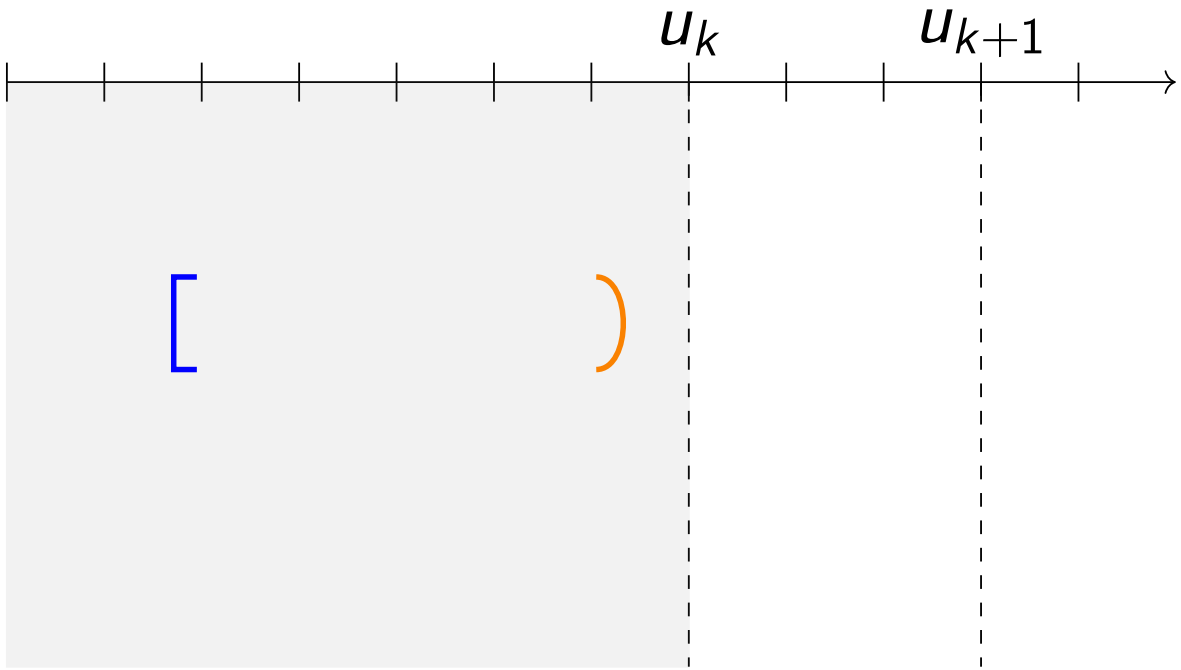
Single machine scheduling with time windows and precedence constraints can be solved in time:

$$\mathcal{O}(\mu^2 \cdot 4^{\mu} \cdot n + n^2).$$

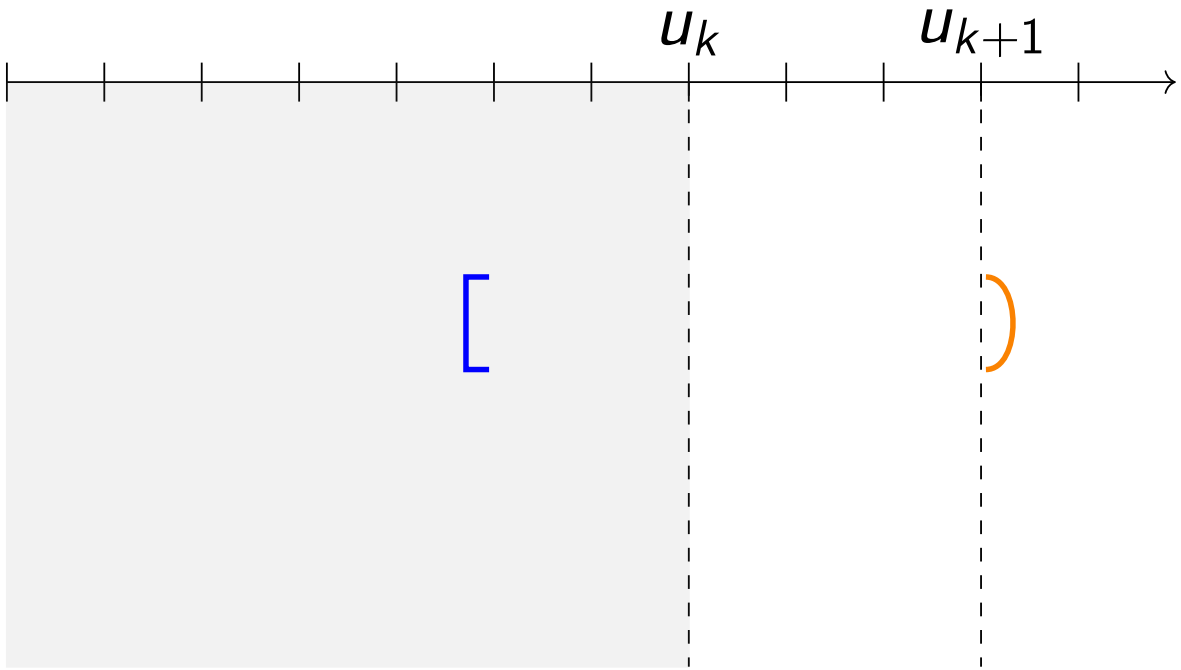
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



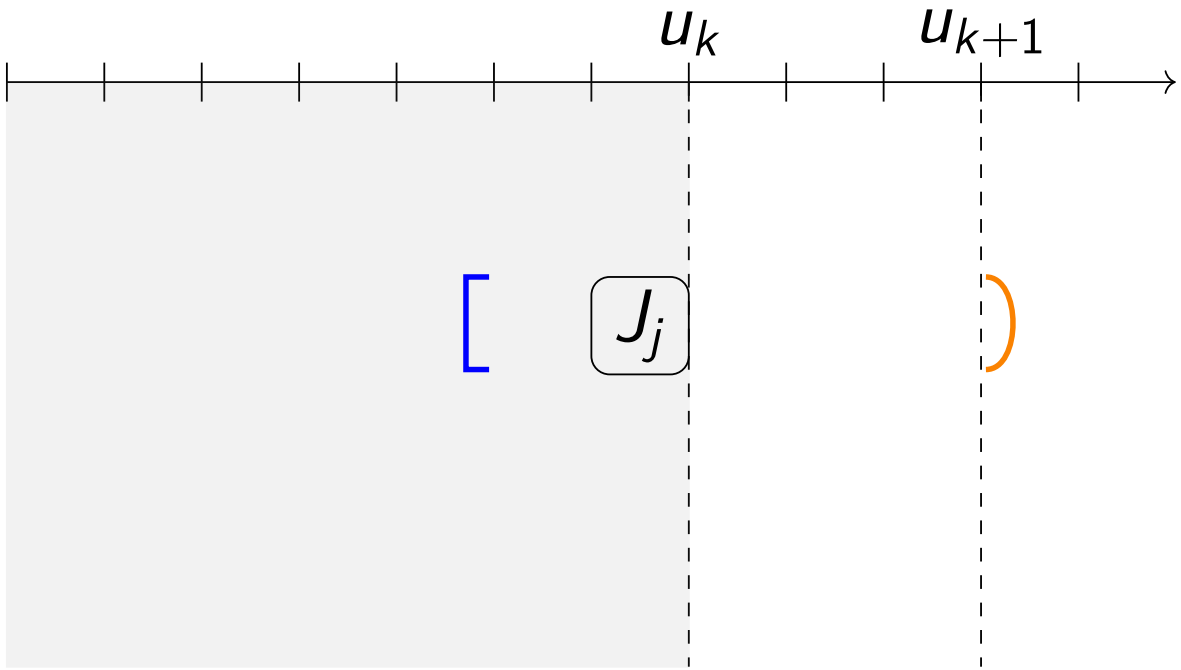
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



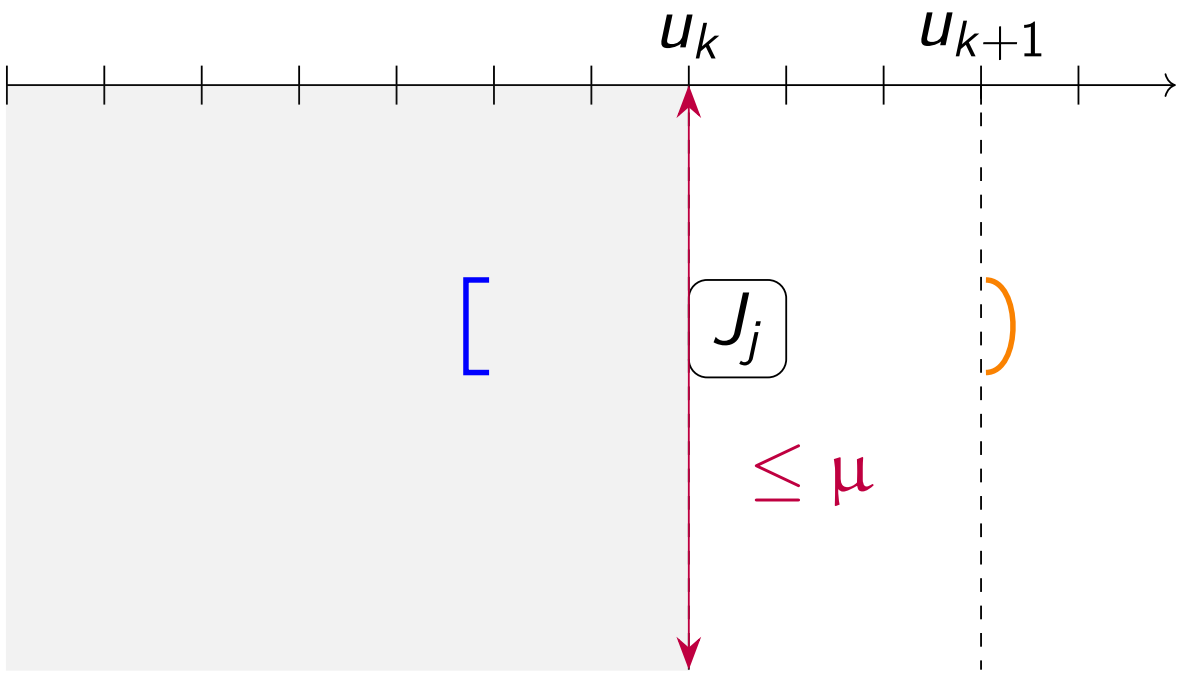
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



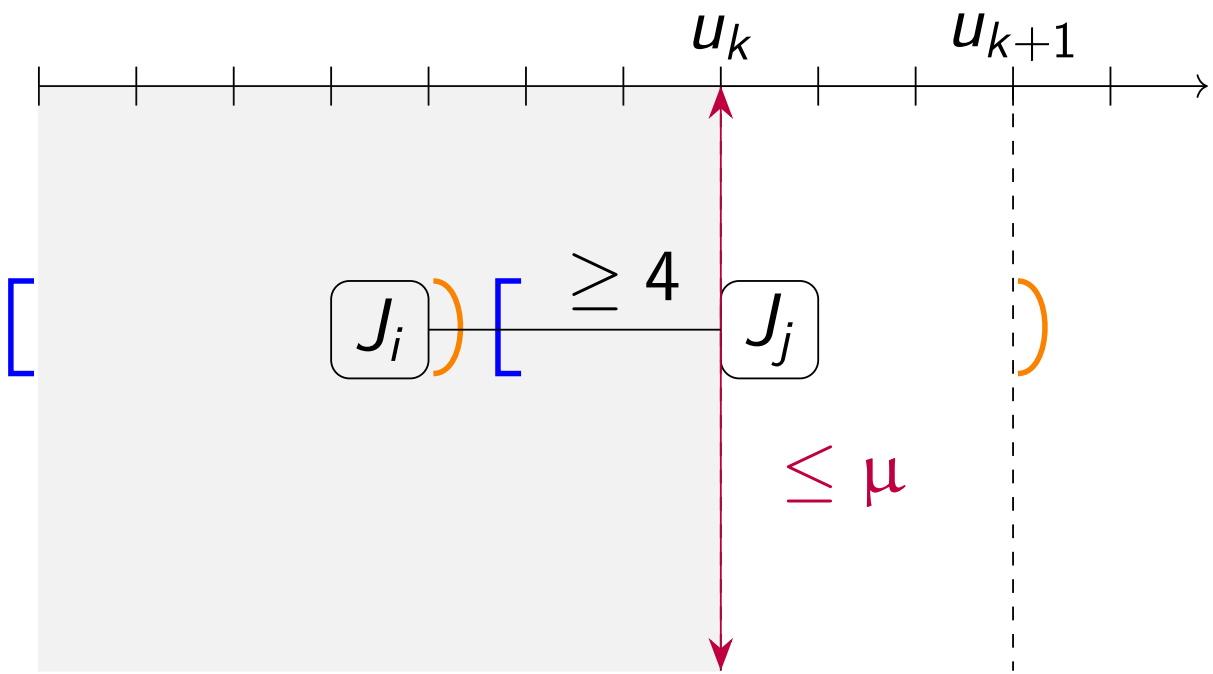
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



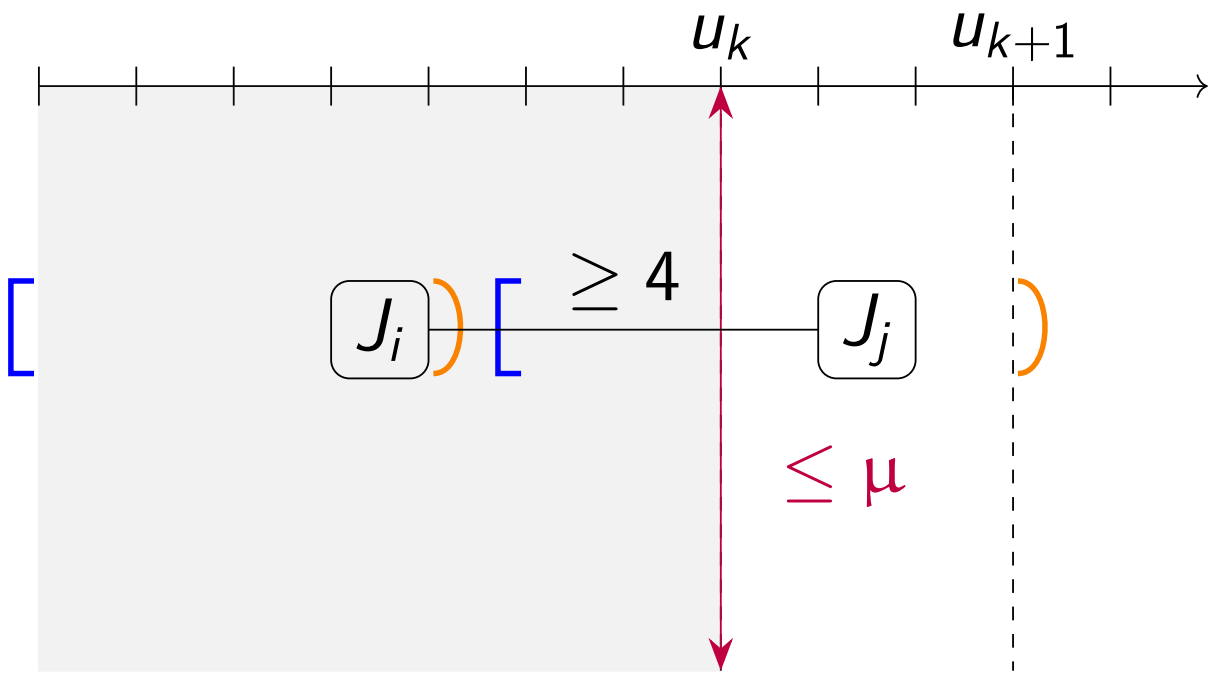
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



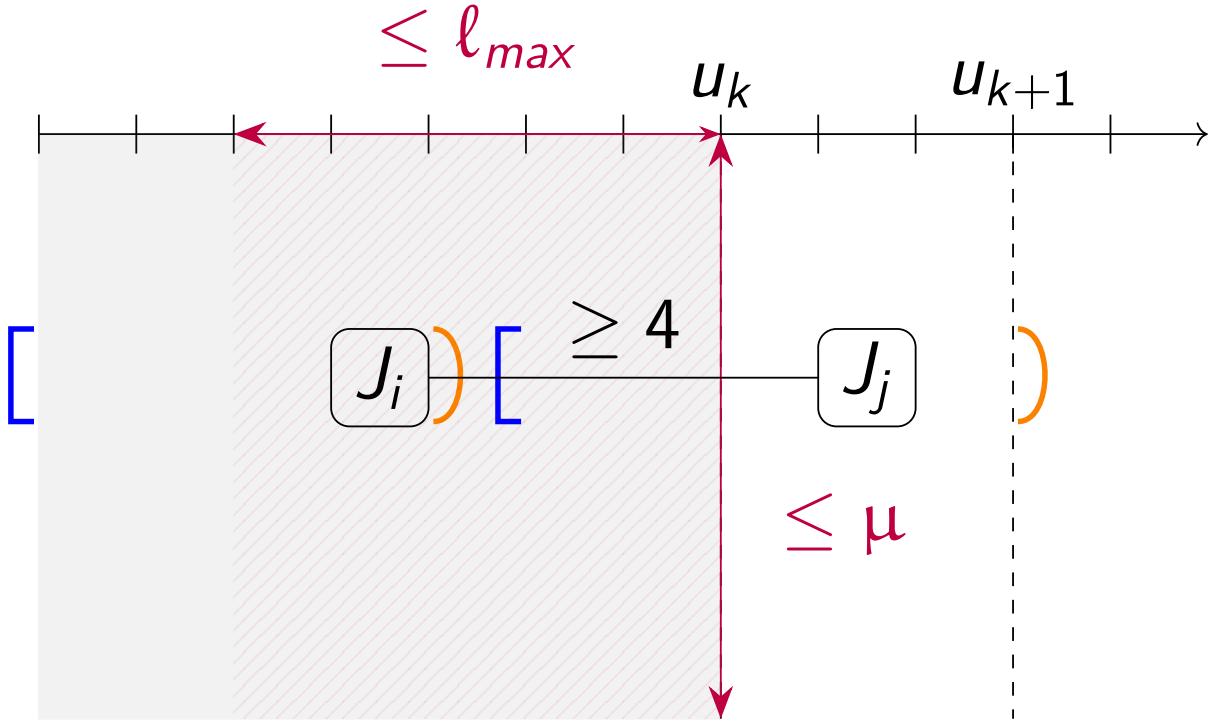
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



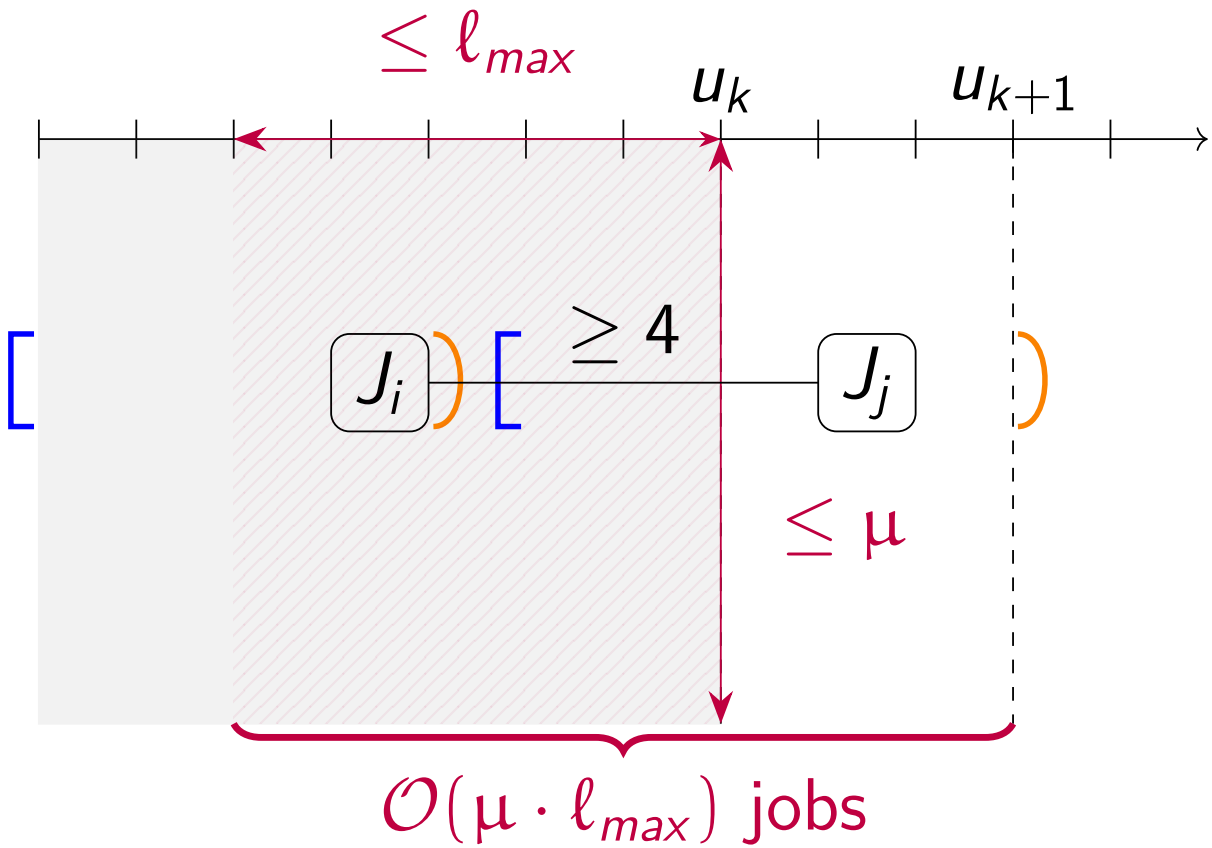
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



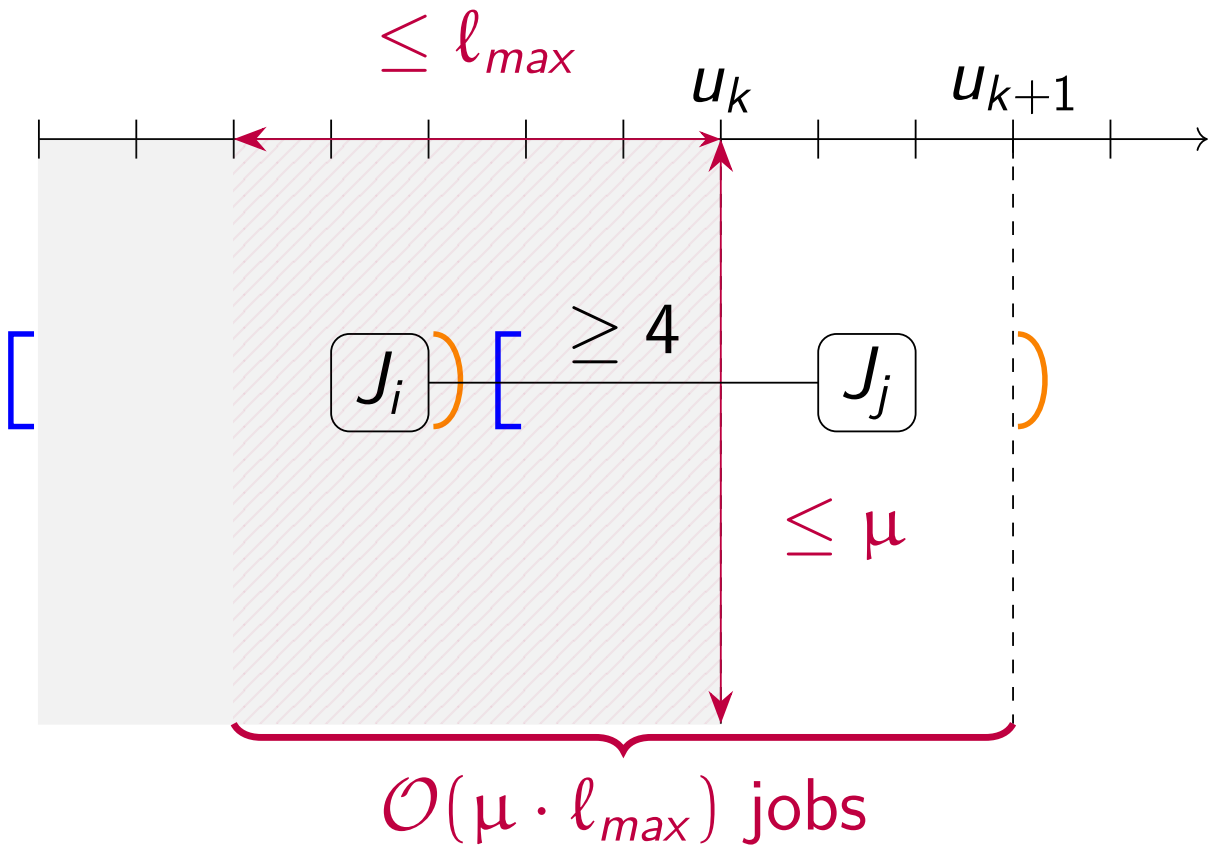
Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



Parallel identical machine scheduling of unit-time jobs with all three delay types combined is FPT w.r.t. $(\mu + \ell_{max})$.



- Extra requirement: $u_{k+1} - u_k \leq \ell_{max}$.

Outline

Conclusion

- Using parameterized complexity to understand better what makes scheduling problems difficult.
- Opportunities for new algorithm ideas.

Future work

- Improve the existing algorithms.
- Consider other parameters.
  - Width w of the precedence graph.
  - Slack σ.
- Investigate other scheduling settings with precedence delays.
  - Single machine, unbounded processing times.
    - No time windows.
    - Bounded chain lengths.

Conclusion

- Using parameterized complexity to understand better what makes scheduling problems difficult.
- Opportunities for new algorithm ideas.

Future work

- Improve the existing algorithms.
- Consider other parameters.
  - Width w of the precedence graph.
  - Slack σ.
- Investigate other scheduling settings with precedence delays.
  - Single machine, unbounded processing times.
  - No time windows.
  - Bounded chain lengths.

Conclusion

- Using parameterized complexity to understand better what makes scheduling problems difficult.
- Opportunities for new algorithm ideas.

Future work

- Improve the existing algorithms.
- Consider other parameters.
  - Width w of the precedence graph.
  - Slack σ.
- Investigate other scheduling settings with precedence delays.
  - Single machine, unbounded processing times.
  - No time windows.
  - Bounded chain lengths.

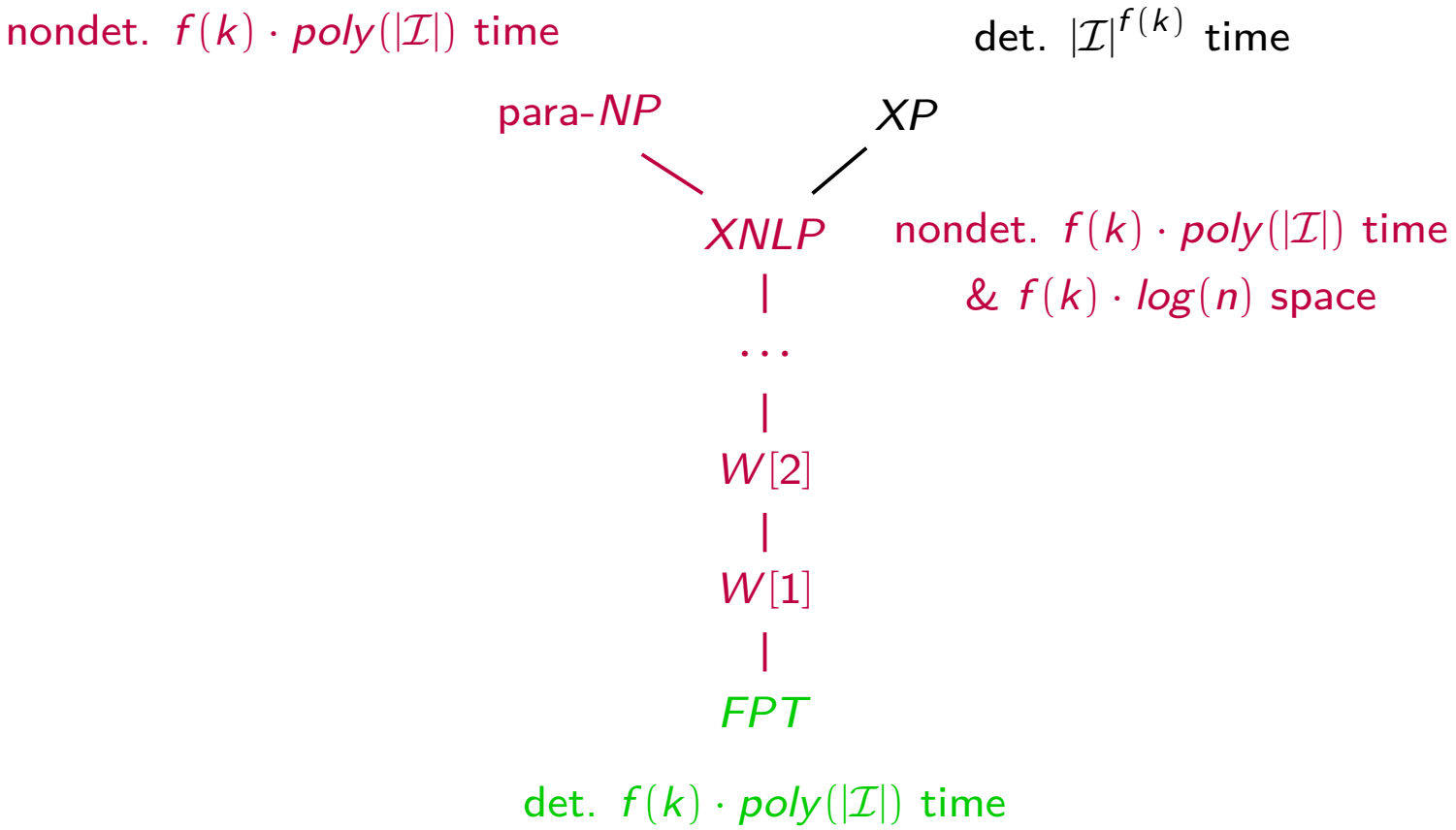
Conclusion

- Using parameterized complexity to understand better what makes scheduling problems difficult.
- Opportunities for new algorithm ideas.

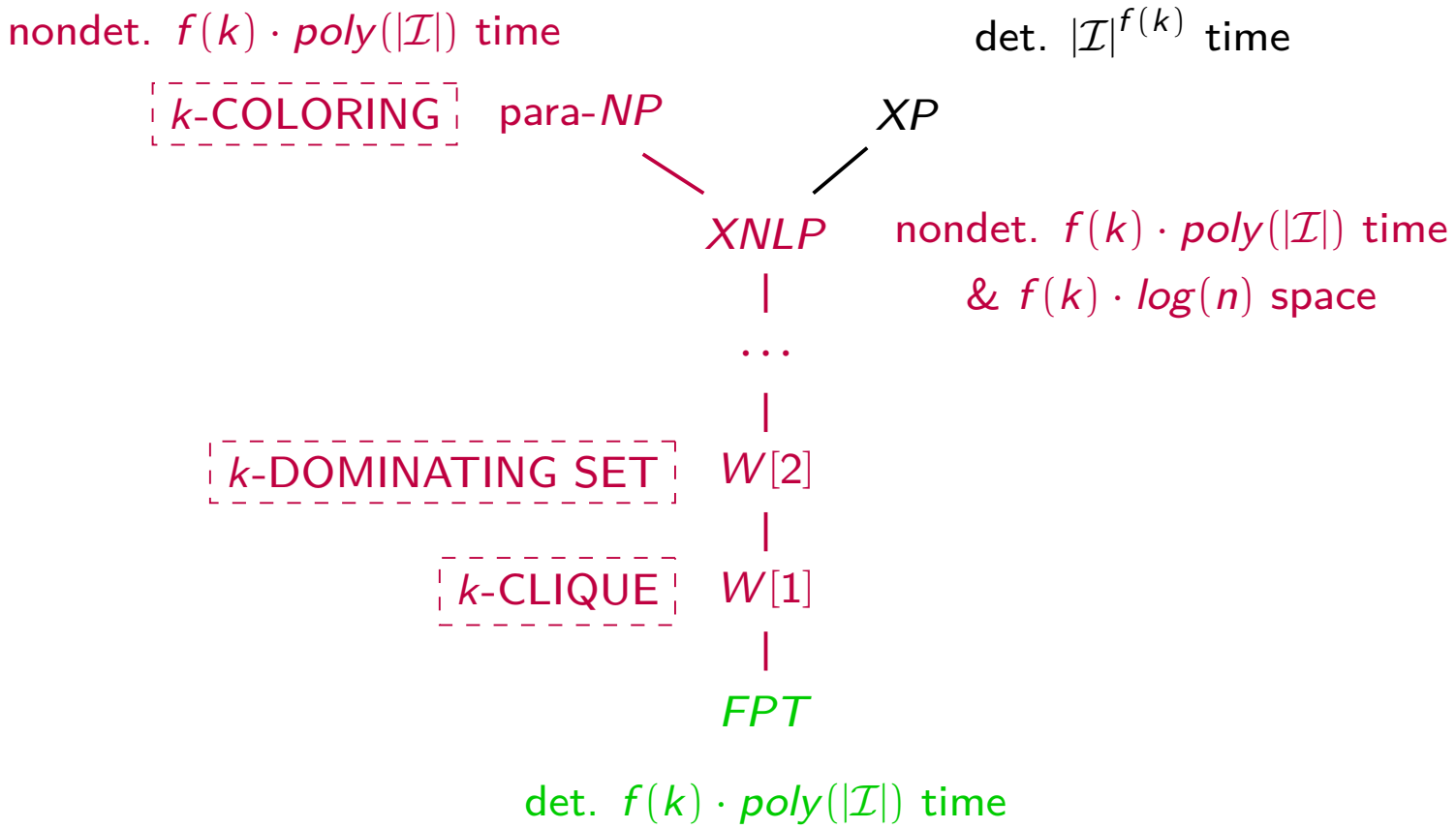
Future work

- Improve the existing algorithms.
- Consider other parameters.
  - Width w of the precedence graph.
  - Slack σ.
- Investigate other scheduling settings with precedence delays.
  - Single machine, unbounded processing times.
  - No time windows.
  - Bounded chain lengths.

Parameterized complexity classes



Parameterized complexity classes




P. Baptiste, P. Brucker, S. Knust, and Vadim Timkovsky.
Ten notes on equal-execution-time scheduling.
4OR, 2:111–127, January 2004.


P. Brucker, M.R. Garey, and D.S. Johnson.
Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness.
Math. Oper. Res., 2(3):275–284, 1977.


M. R. Garey and D. S. Johnson.
Two-processor scheduling with start-times and deadlines.
SIAM Journal on Computing, 6(3):416–426, September 1977.


Claire Hanen and Alix Munier Kordon.
Fixed-parameter tractability of scheduling dependent typed tasks subject to release times and deadlines.
Journal of Scheduling, pages 1–15, 2023.


Claire Hanen, Maher Mallem, and Alix Munier-Kordon.
Parameterized complexity of single-machine scheduling with precedence, release dates and deadlines.
In *15th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP)*, 2022.


J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker.

Complexity of machine scheduling problems.
Ann. of Discrete Math., 1:343–362, 1977.

Alix Munier Kordon.
A fixed-parameter algorithm for scheduling unit dependent tasks on parallel machines with time windows.
Discrete Applied Mathematics, 290:1–6, 2021.

Barbara Simons.
A fast algorithm for single processor scheduling.
In *19th Annual Symposium on Foundations of Computer Science (SFCS 1978)*, pages 246–252, October 1978.
ISSN: 0272-5428.

Barbara Simons.
Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines.
SIAM Journal on Computing, 12(2):294–299, 1983.