Parameterized Complexity and New Efficient Enumerative Schemes for RCPSP

Maher Mallem

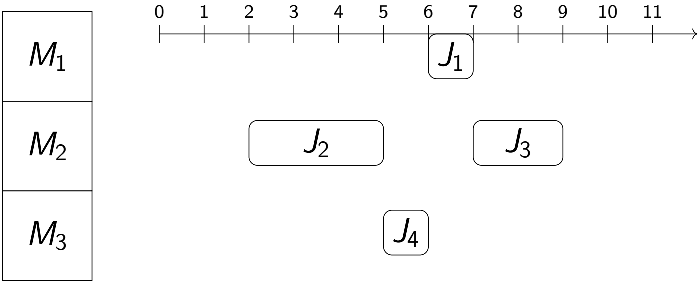
11 October 2024

Outline

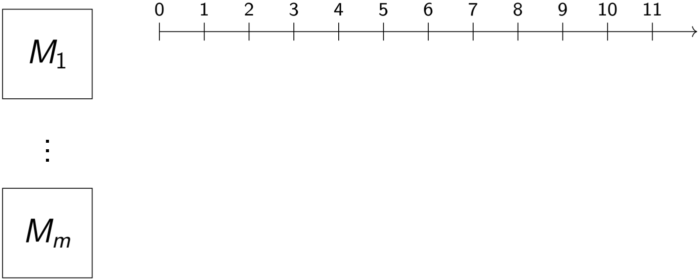
Outline

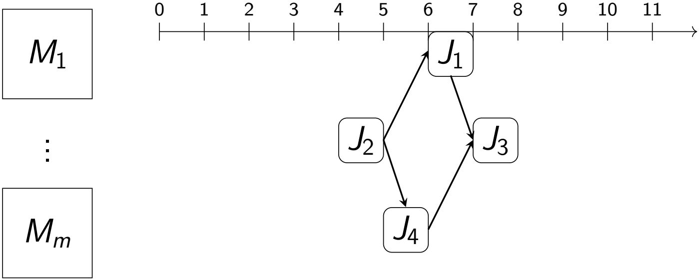
1. Background
   - Scheduling
   - Parameterized Complexity
2. Pathwidth
   - Motivation
   - Single machine FPT
   - With precedence delays
3. Maximum delay value
   - As a single parameter
   - Combined with pathwidth
4. Proper level
   - Motivation
   - Dominance
   - Summit-based enumeration
5. Conclusion

Resource-Constrained Project Scheduling Problem (RCPSP)

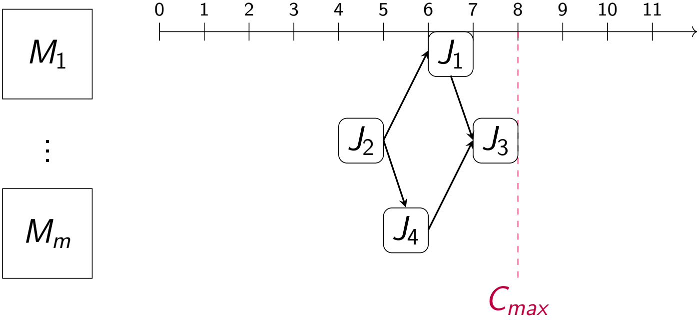


Resource-Constrained Project Scheduling Problem (RCPSP)

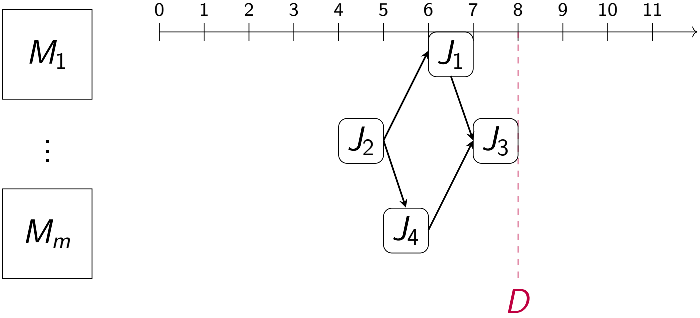




machines | job properties | optimization criterion

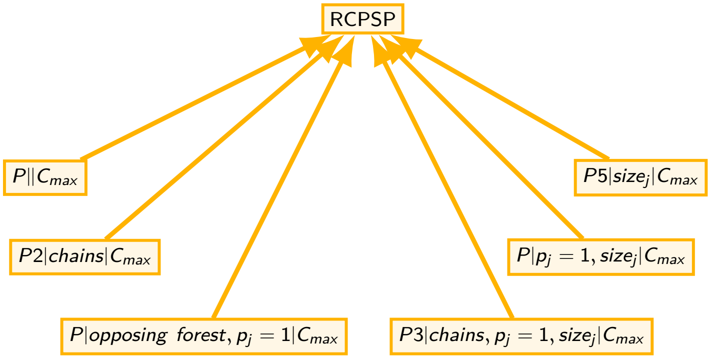


$$P|prec, p_j = 1|C_{max}$$

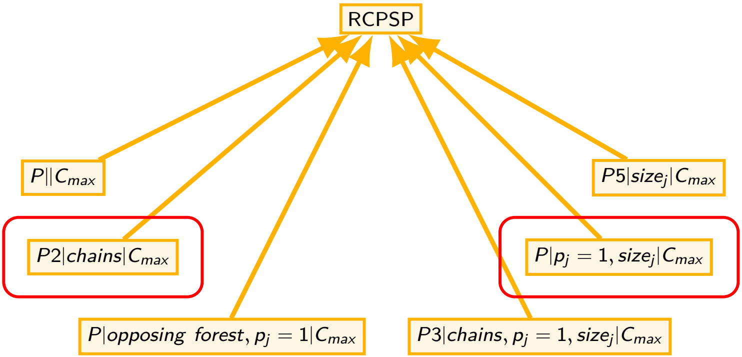


$$P|prec, p_j = 1|C_{max} \leq D$$

Problem map: RCPSP subproblems



Problem map: RCPSP subproblems



Going beyond NP-hardness

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

Going beyond NP-hardness

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
  - Problem $(\mathcal{P}, k)$, instance $\mathcal{I}$ with parameter value $\leq k$.
- P: deterministic time $poly(|\mathcal{I}|)$.
  - FPT: deterministic time $f(k) \cdot poly(|\mathcal{I}|)$.
- NP: nondeterministic time $poly(|\mathcal{I}|)$.
  - para-NP: nondeterministic time $f(k) \cdot poly(|\mathcal{I}|)$.

Parameterized complexity classes

nondet. time $f(k) \cdot poly(|\mathcal{I}|)$

det. time $|\mathcal{I}|^{f(k)}$

para-NP

XP

$XNLP$

$\dots$

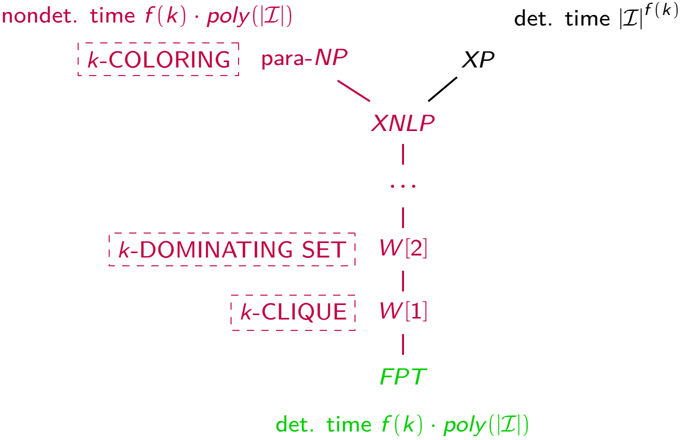
$W[2]$

$W[1]$

FPT

det. time $f(k) \cdot poly(|\mathcal{I}|)$

Parameterized complexity classes



Parameterized complexity in machine scheduling

- Negative results via *FPT* reductions

- Negative results via *FPT* reductions
  - [Bodlaender, Fellows 95] $P|prec, p_j = 1|C_{max}$ is W[2]-hard parameterized by the number m of machines.

- Negative results via *FPT* reductions
  - [Bodlaender, Fellows 95] $P|prec, p_j = 1|C_{max}$ is W[2]-hard parameterized by the number m of machines.

[Mnich, Wiese 15]

- Negative results via *FPT* reductions
  - [Bodlaender, Fellows 95] $P|prec, p_j = 1|C_{max}$ is W[2]-hard parameterized by the number m of machines.
- MIP with a bounded number of integer variables

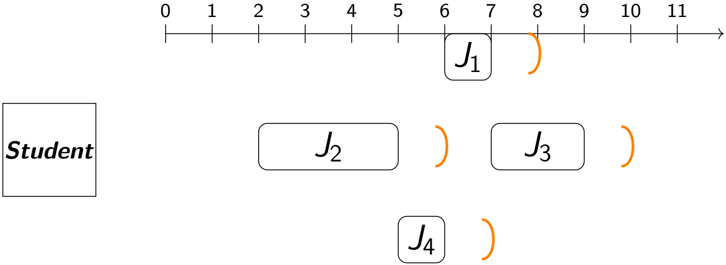
[Mnich, Wiese 15]

- Negative results via *FPT* reductions
  - [Bodlaender, Fellows 95] $P|prec, p_j = 1|C_{max}$ is W[2]-hard parameterized by the number m of machines.
- MIP with a bounded number of integer variables
  - [Lenstra 83] MILP: [Mnich, Wiese 15]
  - [Dadush et al. 11] MICP
  - [Knop et al. 19] MIMO

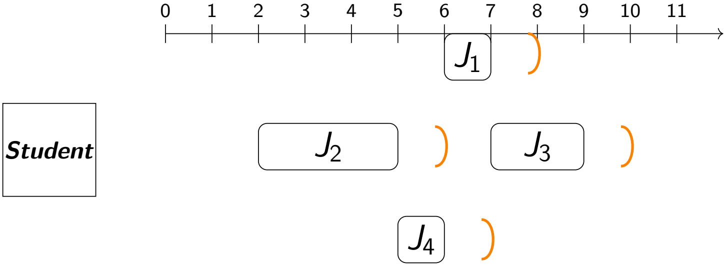
- Negative results via *FPT* reductions
  - [Bodlaender, Fellows 95] $P|prec, p_j = 1|C_{max}$ is W[2]-hard parameterized by the number m of machines.
- MIP with a bounded number of integer variables
  - [Lenstra 83] MILP: [Mnich, Wiese 15]
  - [Dadush et al. 11] MICP
  - [Knop et al. 19] MIMO
- Dynamic Programming

Outline

Adding job time windows $[r_j, \bar{d}_j)$

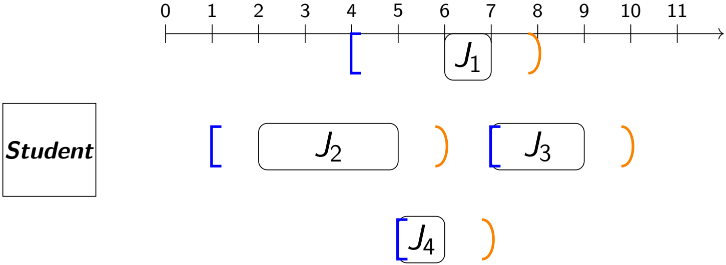


Adding job time windows $[r_j, \bar{d}_j)$



$$1|\bar{d}_j|\star \text{ in time } \mathcal{O}(n \cdot log(n)).$$

Adding job time windows $[r_j, \bar{d}_j)$



$1|r_j, \bar{d}_j|\star$ is strongly *NP*-hard.

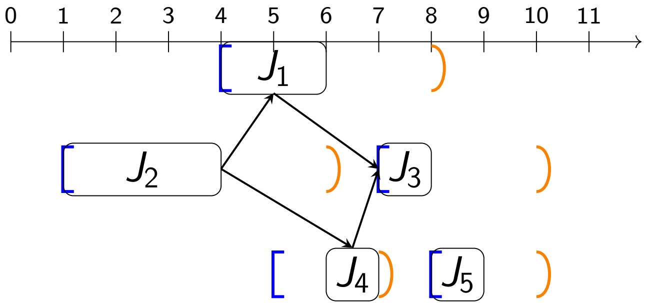
Results with parameter "pathwidth μ"

- [Munier 21]
  - $P|prec, p_j = 1, r_j, \bar{d}_j|C_{max}$
  - Exact resolution in time $\mathcal{O}(16^{\mu} \cdot n^4)$.
- [Baart, He, de Weerdt 21]
  - $1|r_j, s_{jk}, reject, \bar{d}_j| \sum_{j \notin R}(v_j - w_j T_j)$
  - $(1 - \epsilon)$-approximation in time $\mathcal{O}(\mu^2 \, 2^{\mu} \cdot \frac{n^3}{\epsilon^2})$.

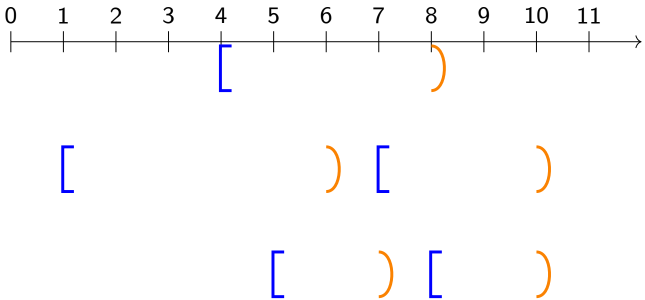
Results with parameter "pathwidth μ"

- [Munier 21]
  - $P|prec, p_j = 1, r_j, \bar{d}_j|C_{max}$
  - Exact resolution in time $\mathcal{O}(16^{\mu} \cdot n^4)$.
- [Baart, He, de Weerdt 21]
  - $1|r_j, s_{jk}, reject, \bar{d}_j| \sum_{j \notin R}(v_j - w_j T_j)$
  - $(1-\epsilon)$-approximation in time $\mathcal{O}(\mu^2\ 2^{\mu} \cdot \frac{n^3}{\epsilon^2})$.
- [Mallem, Hanen, Munier 22]
  - $1|prec, r_j, \bar{d}_j|C_{max}$
  - Exact resolution in time $\mathcal{O}(\mu^2\ 4^{\mu} \cdot n + n^2)$.

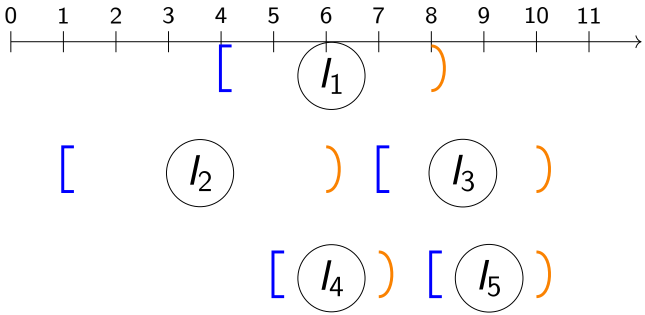
Pathwidth μ



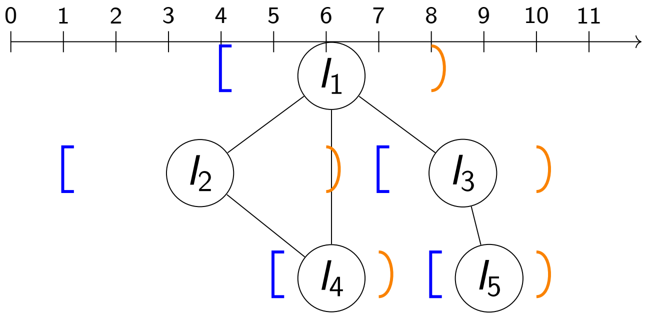
Pathwidth μ



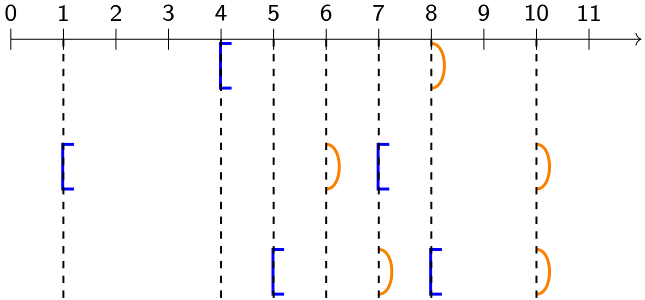
Pathwidth μ



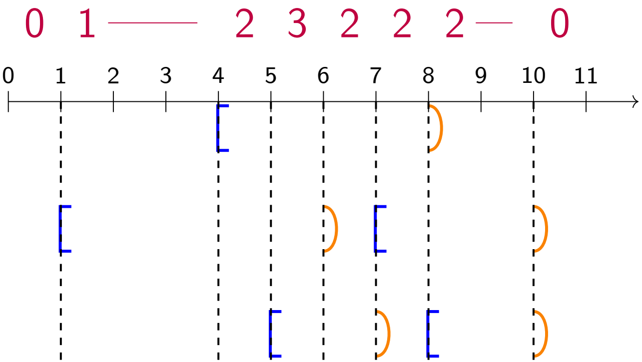
Pathwidth μ



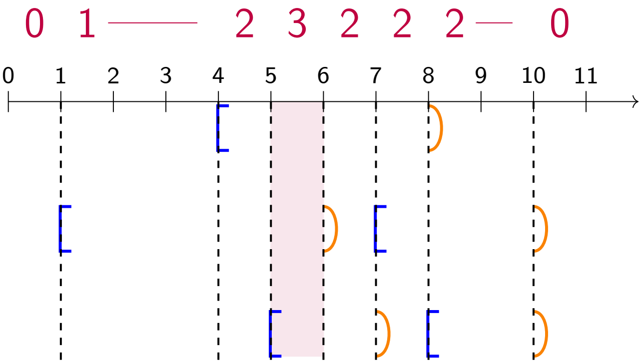
Pathwidth μ



Pathwidth μ

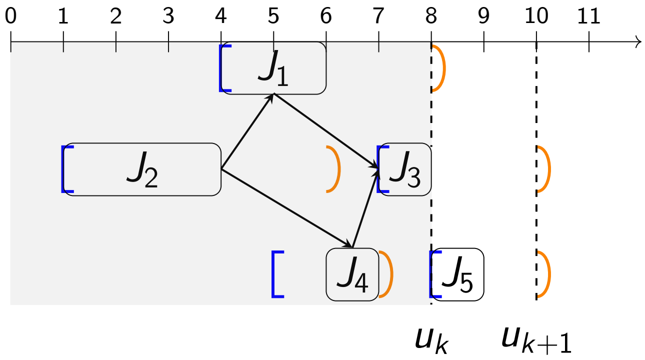


Pathwidth μ

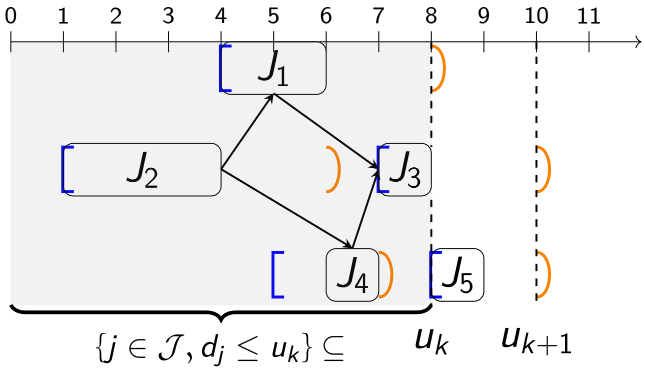


$$\mu = 3 - 1 = 2$$

Pathwidth μ

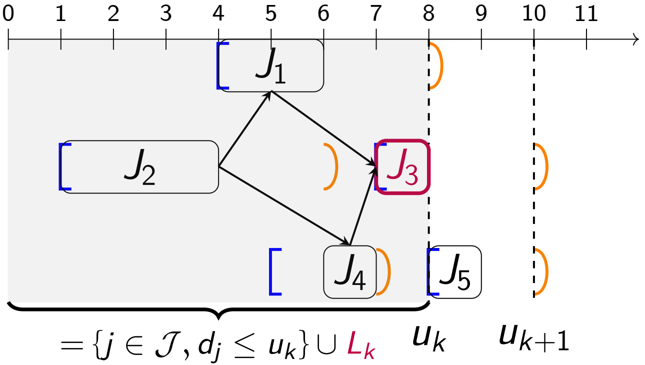


Pathwidth μ



0 1 2 3 4 5 6 7 8 9 10 11
J_1
J_2
J_3
J_4
J_5
$\{j \in \mathcal{J}, d_j \leq u_k\} \subseteq$
u_k
u_{k+1}

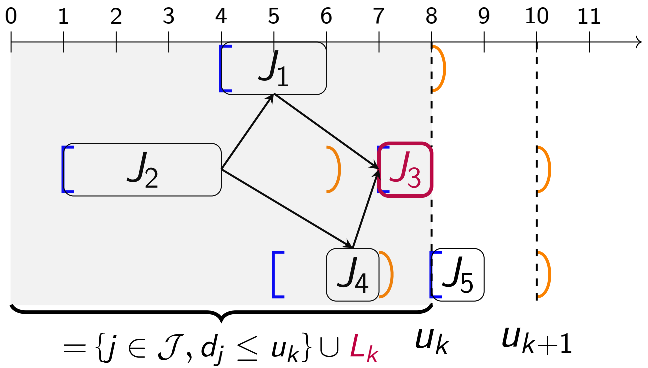
Pathwidth μ



$= \{j \in \mathcal{J}, d_j \leq u_k\} \cup L_k$

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

Pathwidth μ



$$= \underbrace{\{j \in \mathcal{J}, d_j \leq u_k\}}_{= Z_k} \cup L_k$$

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

Dynamic programming with pathwidth μ

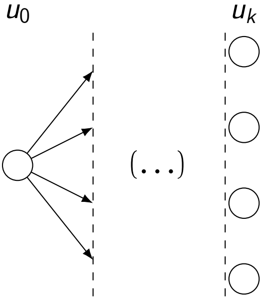
Dynamic programming state (k, L_k)

- 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)

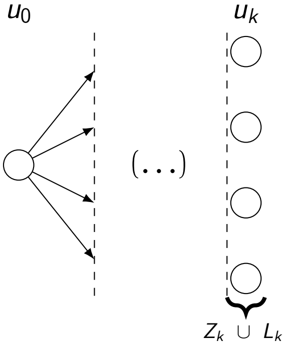
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

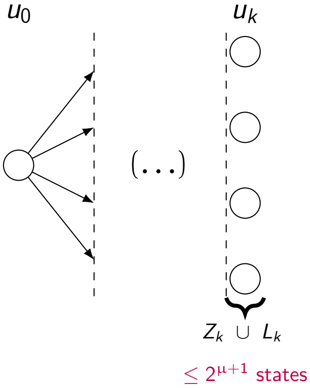
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

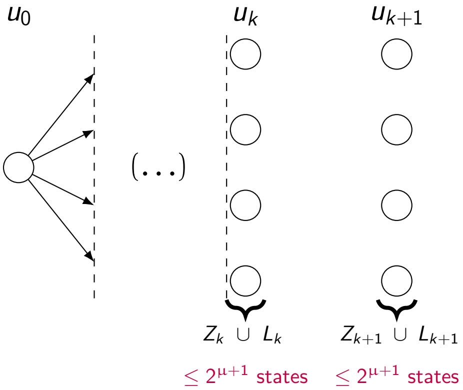
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

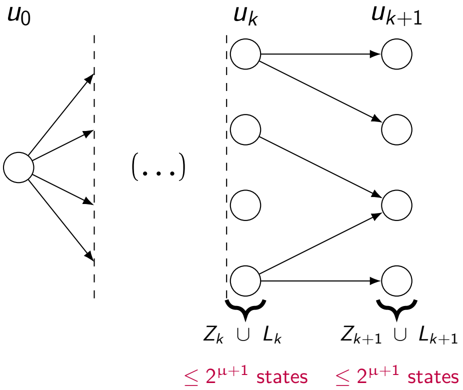
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

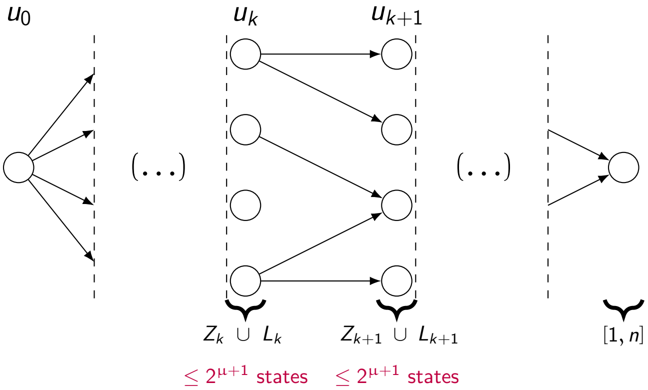
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



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.



Main result

Theorem [Mallem, Hanen, Munier 22]

$1|r_j, \bar{d}_j|C_{max}$ can be solved in time:

$$\mathcal{O}([(\mu+1) \cdot log(\mu+1) \cdot 4^{\mu}] \cdot n + n^2).$$

Main result

Theorem [Mallem, Hanen, Munier 22]

$1|r_j, \bar{d}_j|C_{max}$ can be solved in time:

$$\mathcal{O}([(\mu + 1) \cdot log(\mu + 1) \cdot 4^{\mu}] \cdot n + n^2).$$

- No need to bound p_{max}.

Main result

Theorem [Mallem, Hanen, Munier 22]

$1|r_j, \bar{d}_j|C_{max}$ can be solved in time:

$$\mathcal{O}([(\mu+1)\cdot log(\mu+1)\cdot 4^{\mu}]\cdot n + n^2).$$

- No need to bound p_{max}.
- *prec* can be added (almost) for free: $[(\mu+1)^2 \cdot 4^{\mu}]$.

Main result

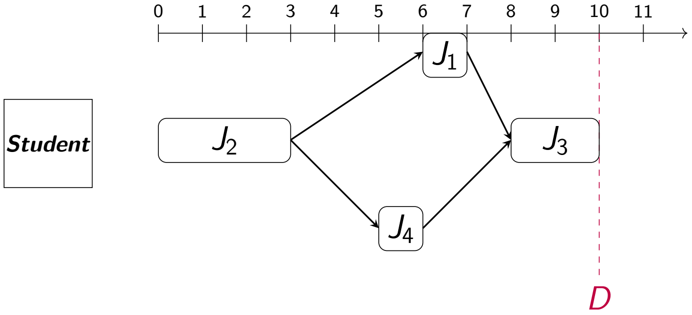
Theorem [Mallem, Hanen, Munier 22]

$1|r_j, \bar{d}_j|C_{max}$ can be solved in time:

$$\mathcal{O}([(\mu+1)\cdot log(\mu+1)\cdot 4^{\mu}]\cdot n + n^2).$$

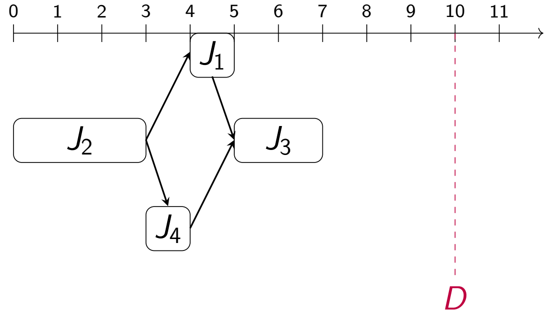
- No need to bound p_{max}.
- *prec* can be added (almost) for free: $[(\mu+1)^2 \cdot 4^{\mu}]$.
- What about precedence delays?

Adding precedence delays $\ell_{i,j}$



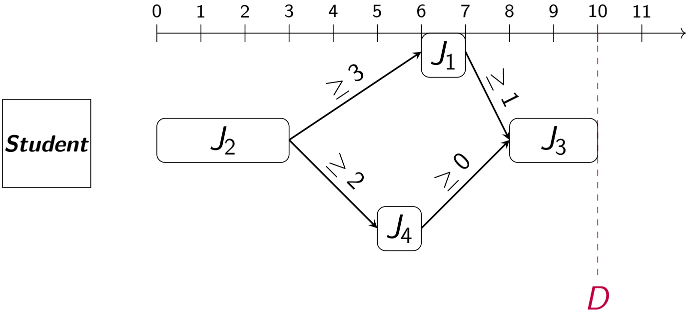
Adding precedence delays $\ell_{i,j}$

Student

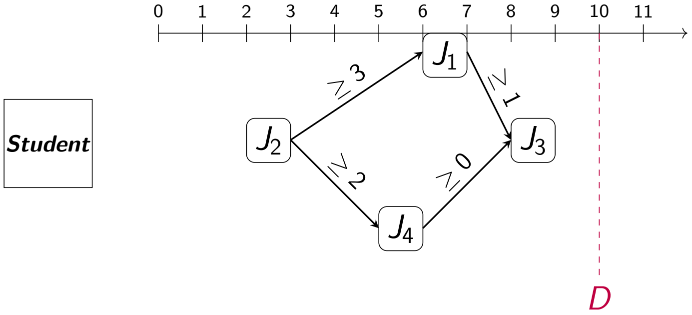


$$1|prec|C_{max} \leq D \text{ in time } \mathcal{O}(n).$$

Adding precedence delays $\ell_{i,j}$



$1|prec(\ell_{i,j})|C_{max} \leq D$ is strongly NP-hard.



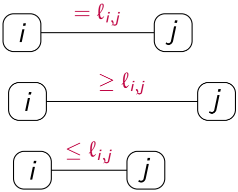
$1|prec(\ell_{i,j}), p_j = 1|C_{max} \leq D$ is strongly NP-hard.

Precedence delay types

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

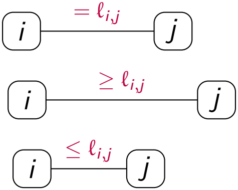
Precedence delay types

- 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:



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:



- [Bodlaender, van der Wegen 2020]

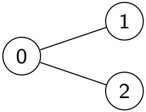
  $1|chains(\ell_{i,j}), p_j = 1, r_C, d_C|\star$ is $W[t]$-hard parameterized by #chains or thickness for all $t \geq 1$ with exact or minimum delays.

$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-NP-hard.

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:



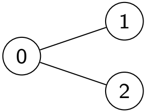
$\longrightarrow$

Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.

$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-*NP*-hard.

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:

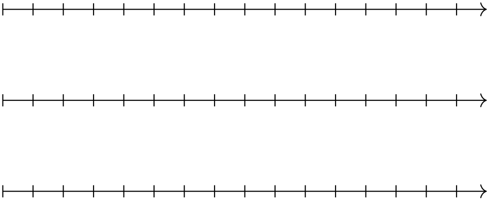


$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.

Vertex chain C_0

Vertex chain C_1

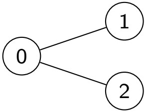
Vertex chain C_2



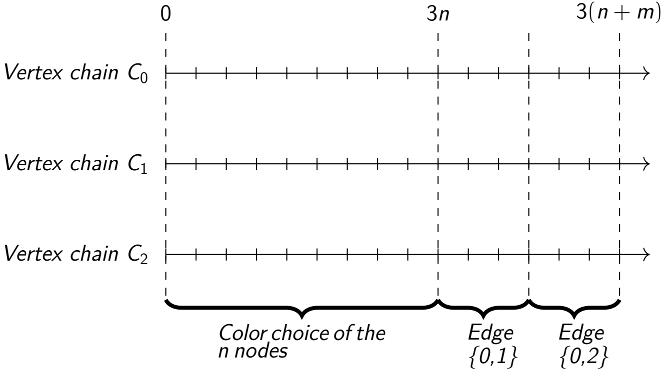
$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-NP-hard.

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:



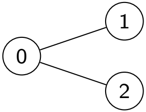
$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.



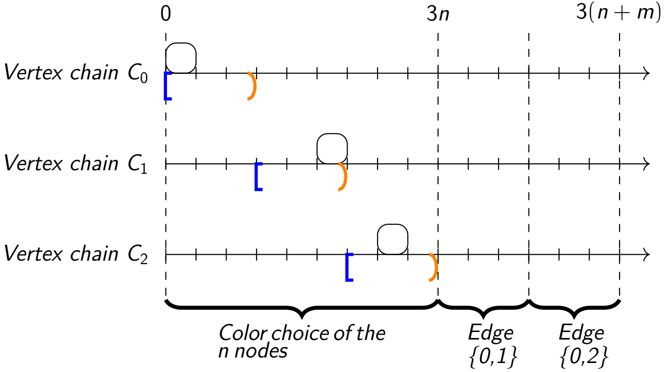
$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-*NP*-hard.

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:



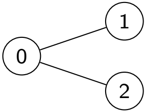
$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.



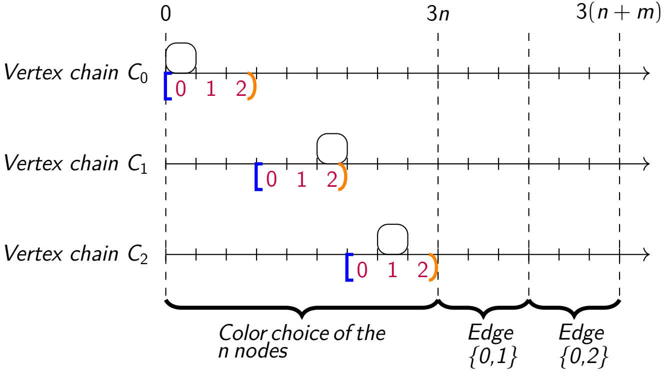
$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-*NP*-hard.

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:

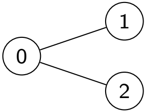


$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.

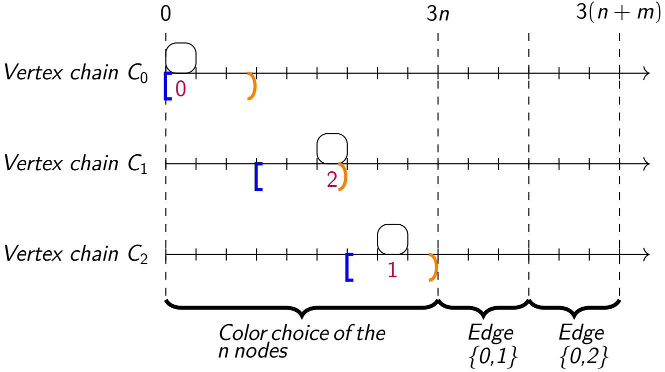


- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:



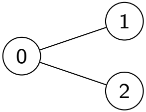
$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.



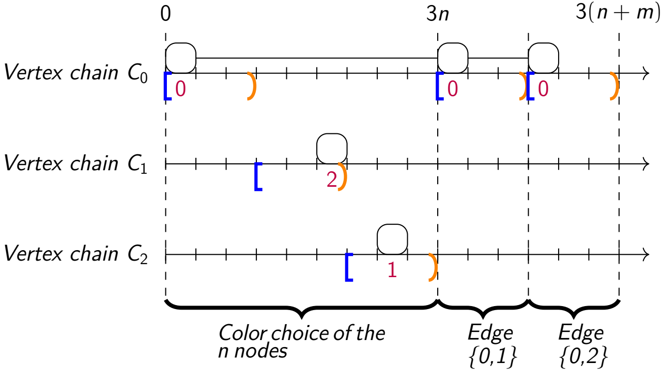
$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-*NP*-hard.

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:



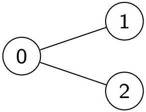
$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.



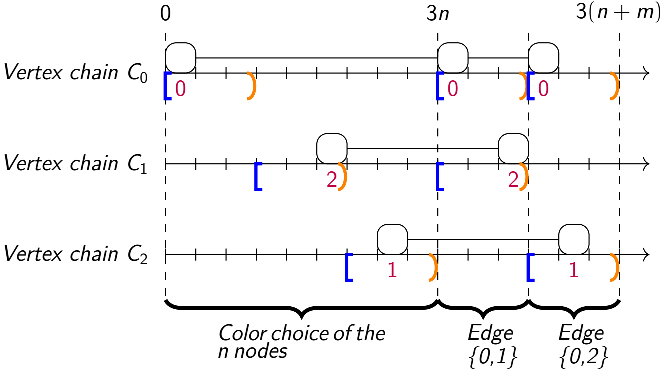
$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-*NP*-hard.

- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:



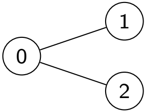
$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.



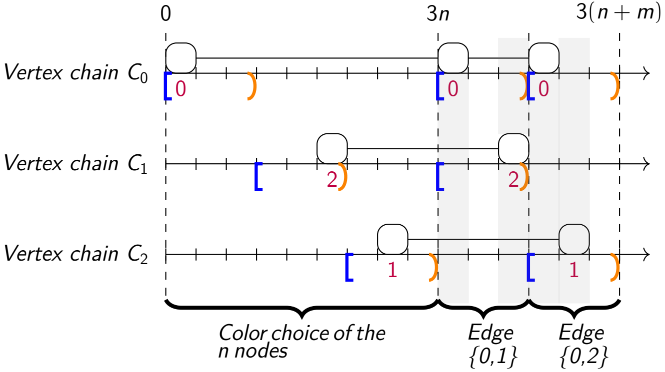
$(1|chains(\ell_{i,j}), p_j = 1, r_j, d_j|\star, \mu)$ is para-*NP*-hard.

- Reduction from 3-COLORING: $G = (V, E)$, $n = |V|$, $m = |E|$

Example:

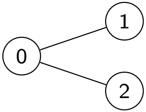


$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.

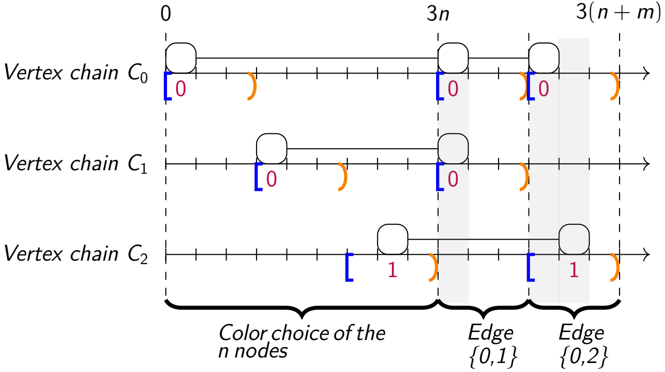


- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example:



$\longrightarrow$ Scheduling instance with $3n + 2m$ jobs, exact delays and $\mu = 1$.



Outline

Results with parameter "maximum delay value ℓ_{max}"

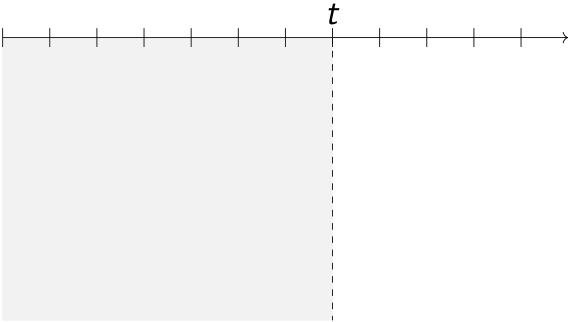
- [Mallem, Hanen, Munier 24]
  - $1|prec(\ell), p_j = 1|C_{max} \leq D$ is $XNLP$-hard parameterized by ℓ.
  - Reduction from $P|prec, p_j = 1|C_{max} \leq D$.
    - Proved $XNLP$-hard in [Bodlaender et al. 24].

Results with parameter "maximum delay value ℓ_{max}"

- [Mallem, Hanen, Munier 24]
  - $1|prec(\ell), p_j = 1|C_{max} \leq D$ is *XNLP*-hard parameterized by ℓ.
  - Reduction from $P|prec, p_j = 1|C_{max} \leq D$.
    - Proved *XNLP*-hard in [Bodlaender et al. 24].
- [Mallem, Hanen, Munier 24]
  - $1|chains(0, \ell), p_j = 1|C_{max} \leq D$ is $W[2]$-hard parameterized by ℓ.
  - Reduction from DOMINATING SET.
    - Similar to the $W[2]$-hardness proof of problem $P|prec, p_j = 1|C_{max} \leq D$ in [Bodlaender, Fellows 95].

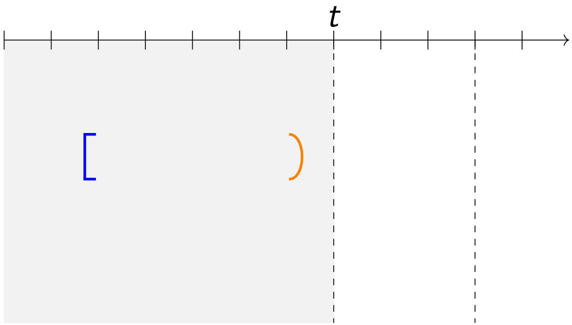
$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

- ...only with minimum delays.



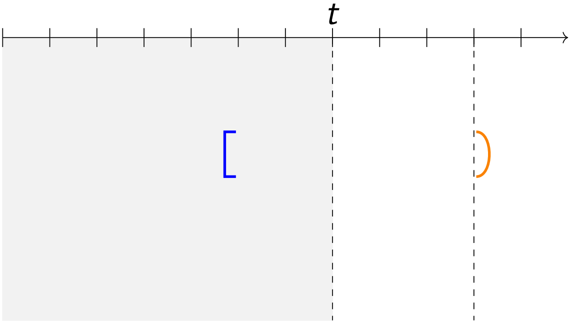
$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

- ...only with minimum delays.



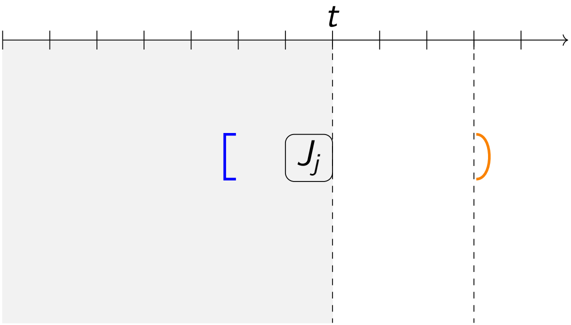
$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

- ...only with minimum delays.



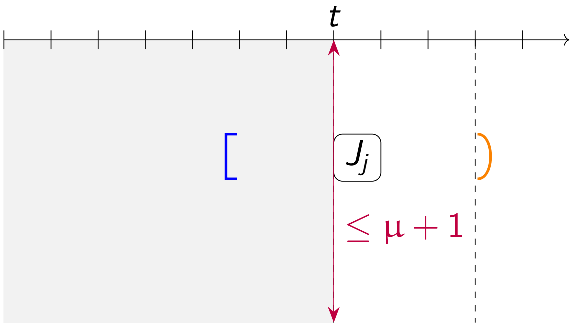
$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

- ...only with minimum delays.



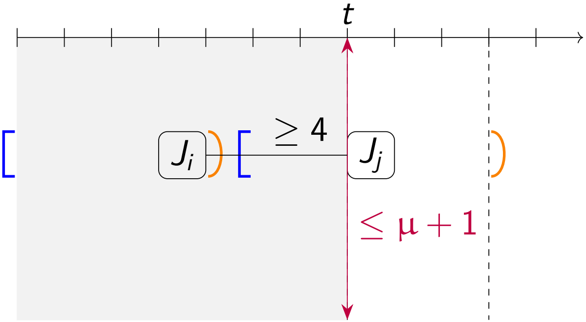
$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

- ...only with minimum delays.



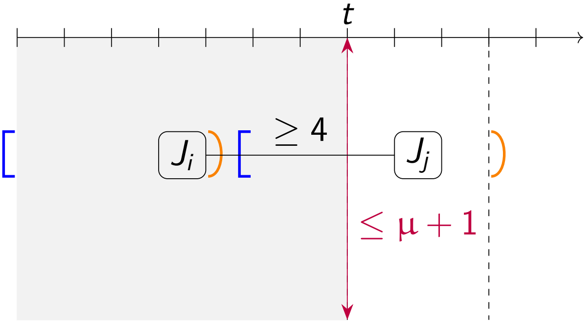
$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

- ...only with minimum delays.



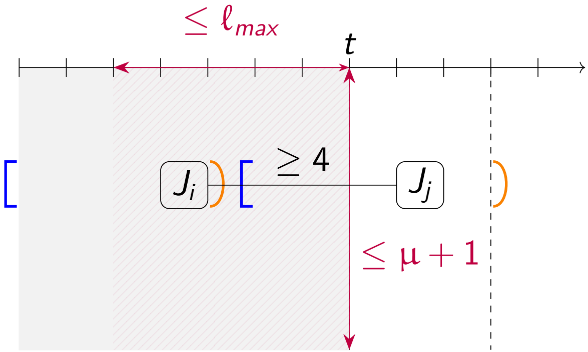
$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

- ...only with minimum delays.

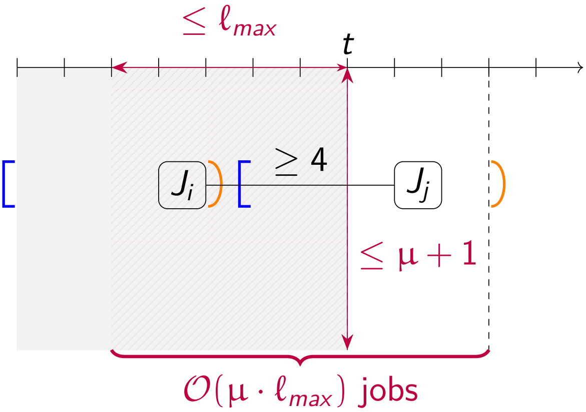


$P|prec(\ell_{i,j}), p_j = 1, r_j, d_j|\star$ is FPT parameterized by $\mu + \ell_{max}$...

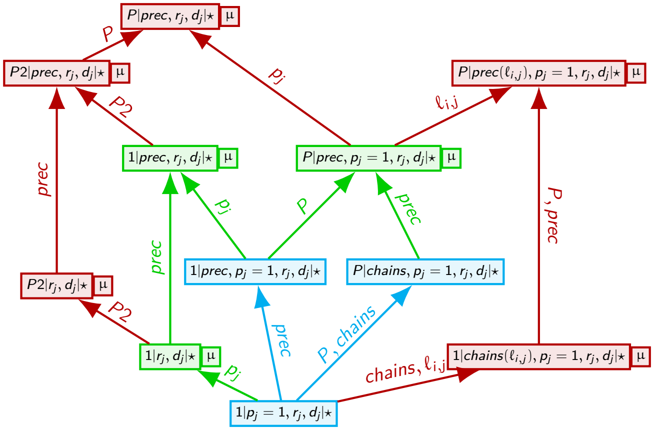
- ...only with minimum delays.



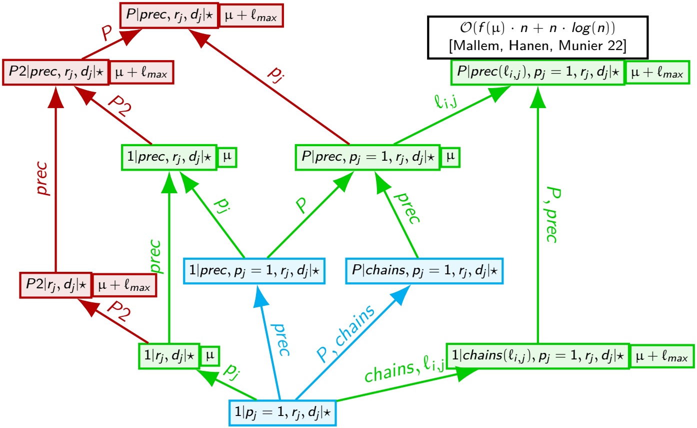
- ...only with minimum delays.



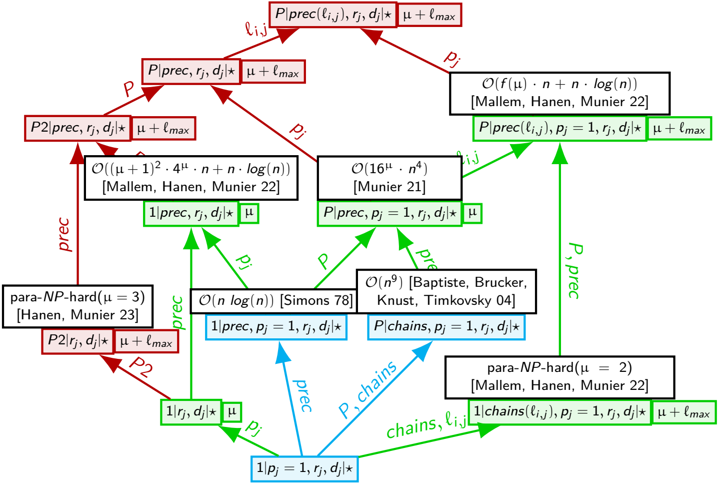
Parameter $\mu + \ell_{max}$



Parameter $\mu + \ell_{max}$



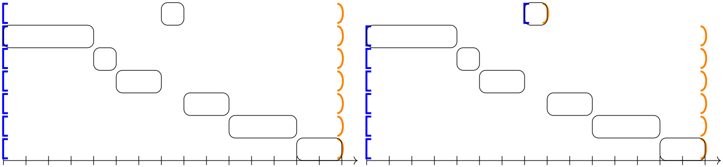
Parameter $\mu + \ell_{max}$



Outline

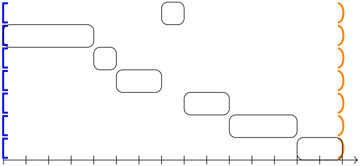
Pathwidth limitation

- Two instances of $1|r_j, \bar{d}_j|\star$.

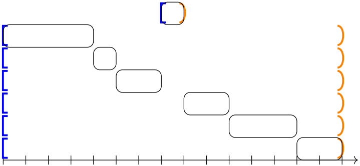


Pathwidth limitation

- Two instances of $1|r_j, \bar{d}_j|\star$.



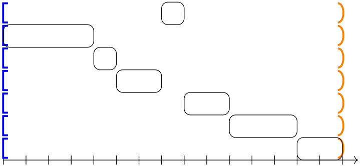
$\mu = n - 1$



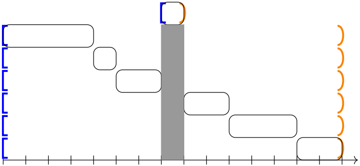
$\mu = n - 1$

Pathwidth limitation

- Two instances of $1|r_j, \bar{d}_j|\star$.



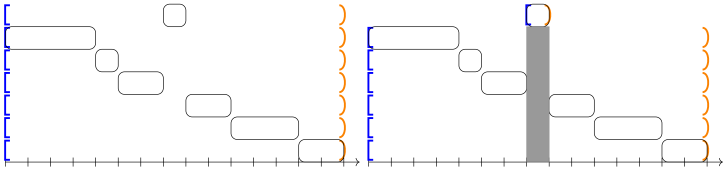
$\mu = n - 1$



$\mu = n - 1$

Pathwidth limitation

- Two instances of $1|r_j, \bar{d}_j|\star$.



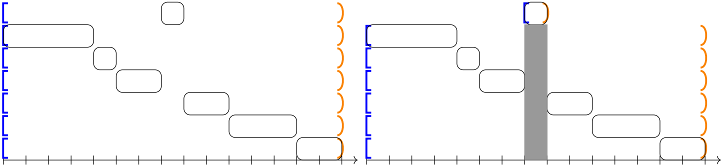
$\mu = n - 1$ $\qquad$ $\mu = n - 1$

Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge \bar{d}_i < \bar{d}_j\}$
- $q = max_{i \in \mathcal{J}} \ |\Pi_i|$

Pathwidth limitation

- Two instances of $1|r_j, \bar{d}_j|\star$.



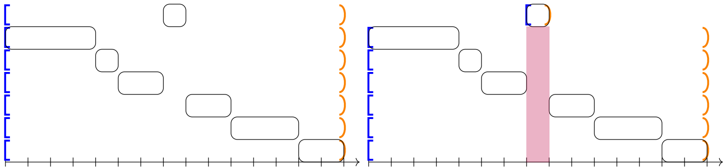
$\mu = n - 1$ $\qquad\qquad$ $\mu = n - 1$

Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge \bar{d}_i < \bar{d}_j\}$ [Erschler, Fontan, Merce 85]
- $q = max_{i \in \mathcal{J}} \ |\Pi_i|$ [Proskurowski, Telle 99]

Pathwidth limitation

- Two instances of $1|r_j, \bar{d}_j|\star$.



$\mu = n - 1$

$q = 0$

$\mu = n - 1$

$q = n - 1$

Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge \bar{d}_i < \bar{d}_j\}$ [Erschler, Fontan, Merce 85]
- $q = max_{i \in \mathcal{J}} \ |\Pi_i|$ [Proskurowski, Telle 99]

Extending the Earliest Deadline rule

- $\prec$ = lexicographic order (non decreasing $\bar{d}_j$, non decreasing r_j) on $\mathcal{J}$.

Definition: Earliest Deadline rule (ED) [Lawler, Moore 69]

Schedule τ follows the (ED) rule if:

$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies \tau(i) < \tau(j).$$

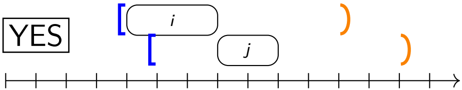
Extending the Earliest Deadline rule

- $\prec$ = lexicographic order (non decreasing $\bar{d}_j$, non decreasing r_j) on $\mathcal{J}$.

Definition: Earliest Deadline rule (ED) [Lawler, Moore 69]

Schedule τ follows the (ED) rule if:

$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies \tau(i) < \tau(j).$$



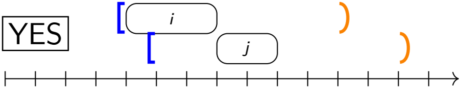
Extending the Earliest Deadline rule

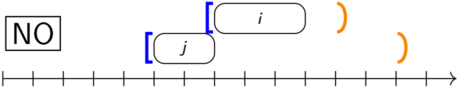
- $\prec$ = lexicographic order (non decreasing $\bar{d}_j$, non decreasing r_j) on $\mathcal{J}$.

Definition: Earliest Deadline rule (ED) [Lawler, Moore 69]

Schedule τ follows the (ED) rule if:

$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies \tau(i) < \tau(j).$$





Extending the Earliest Deadline rule (cont.)

Definition: weak Earliest Deadline rule (wED)

Schedule τ follows the (wED) rule if:

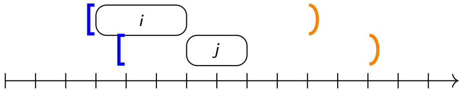
$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies [\tau(i) < \tau(j)] \vee [j \in \Pi_i] \vee [\exists h \prec i, \tau(j) < \tau(h) < \tau(i)].$$

Extending the Earliest Deadline rule (cont.)

Definition: weak Earliest Deadline rule (wED)

Schedule τ follows the (wED) rule if:

$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies [\tau(i) < \tau(j)] \vee [j \in \Pi_i] \vee [\exists h \prec i, \tau(j) < \tau(h) < \tau(i)].$$

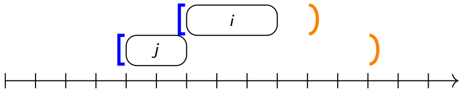


Extending the Earliest Deadline rule (cont.)

Definition: weak Earliest Deadline rule (wED)

Schedule τ follows the (wED) rule if:

$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies [\tau(i) < \tau(j)] \vee [j \in \Pi_i] \vee [\exists h \prec i, \tau(j) < \tau(h) < \tau(i)].$$

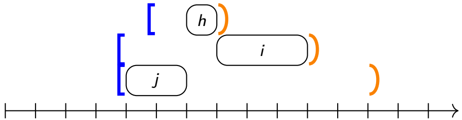


Extending the Earliest Deadline rule (cont.)

Definition: weak Earliest Deadline rule (wED)

Schedule τ follows the (wED) rule if:

$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies [\tau(i) < \tau(j)] \vee [j \in \Pi_i] \vee [\exists h \prec i, \tau(j) < \tau(h) < \tau(i)].$$

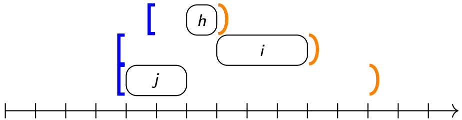


Extending the Earliest Deadline rule (cont.)

Definition: weak Earliest Deadline rule (wED)

Schedule τ follows the (wED) rule if:

$$\forall (i,j) \in \mathcal{J}^2, i \prec j \implies [\tau(i) < \tau(j)] \vee [j \in \Pi_i] \vee [\exists h \prec i, \tau(j) < \tau(h) < \tau(i)].$$



Proposition

On $1|r_j, d_j|C_{max}$ the (wED)-following schedules are dominant.

Summit-based schedule enumeration

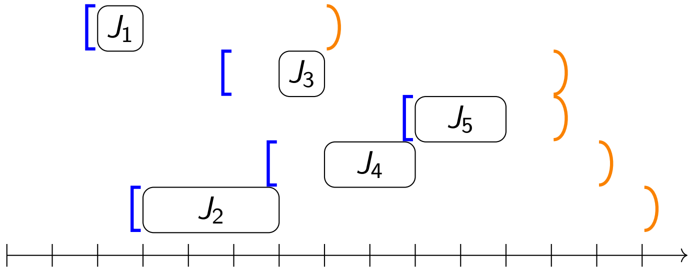
Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.
The summits of the instance are named $s_1 \prec \ldots \prec s_N$ $(N \leq n)$.

Summit-based schedule enumeration

Definition: Summit

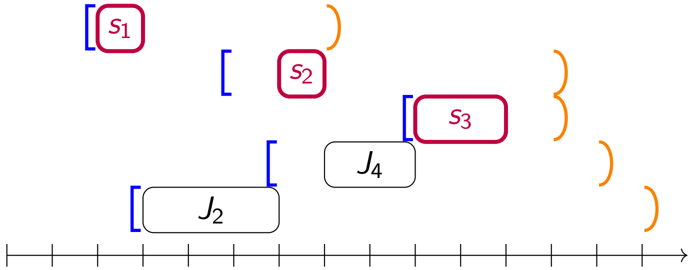
A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.
The summits of the instance are named $s_1 \prec \ldots \prec s_N$ $(N \leq n)$.



Summit-based schedule enumeration

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.
The summits of the instance are named $s_1 \prec \ldots \prec s_N$ $(N \leq n)$.



Summit-based schedule enumeration (cont.)

Definition: Summit

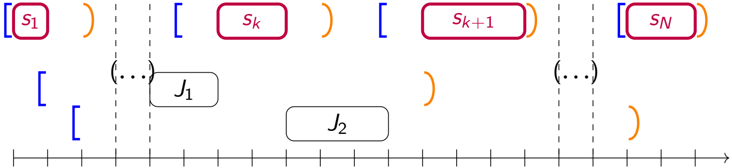
A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.
The summits of the instance are named $s_1 \prec \ldots \prec s_N$ $(N \leq n)$.

Summit-based schedule enumeration (cont.)

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.
The summits of the instance are named $s_1 \prec \ldots \prec s_N$ $(N \leq n)$.

- Schedule τ follows the (wED) rule
  $\implies$ τ is of the form $T_1 s_1 \ldots s_k T_{k+1} s_{k+1} \ldots s_N T_{N+1}$.

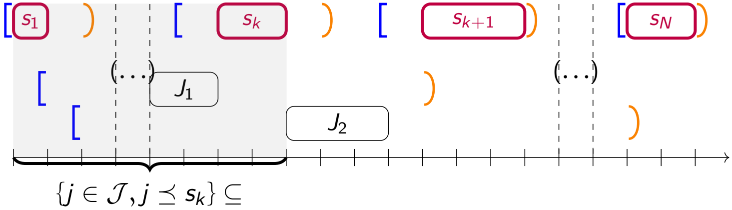


Summit-based schedule enumeration (cont.)

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.
The summits of the instance are named $s_1 \prec \ldots \prec s_N$ ($N \leq n$).

- Schedule τ follows the (wED) rule
  $\implies$ τ is of the form $T_1 s_1 \ldots s_k T_{k+1} s_{k+1} \ldots s_N T_{N+1}$.

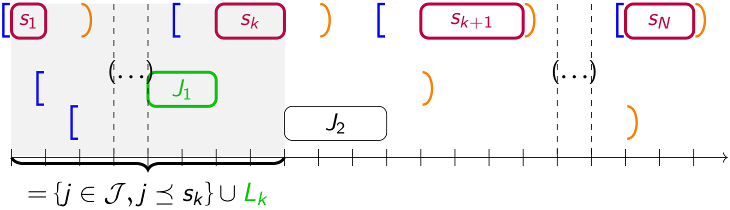


Summit-based schedule enumeration (cont.)

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.
The summits of the instance are named $s_1 \prec \ldots \prec s_N$ $(N \leq n)$.

- Schedule τ follows the (wED) rule
  $\implies$ τ is of the form $T_1 s_1 \ldots s_k T_{k+1} s_{k+1} \ldots s_N T_{N+1}$.



- $j \in L_k \implies j \in \Pi_{s_k}$: at most q such jobs j.

Outline

PhD results (1/2)

Parameter	Section	Problem / Setting	Result
ℓ_{max}	3.2	$1\|chains(\ell), p_j = 1, r_j, \bar{d}_j\|\star$	para-*NP*-hard with exact or maximum delays.
		$1\|chains(\ell), p_j = 1\|C_{max} < D$	$W[1]$-hard with exact delays.
		$1\|chains(\ell_{i,j}), p_j = 1\|C_{max} < D$	$\mathcal{O}(n)$ with maximum delays.
		$1\|chains(0, \ell), p_j = 1, r_j\|C_{max} < D$ $1\|chains(0, \ell), p_j = 1, \bar{d}_j\|\star$	*NP*-hard with maximum delays.
	3.3	$1\|prec(\ell), p_j = 1\|C_{max} < D$	*XNLP*-hard($\ell_{max} + w$) with minimum delays.
		$1\|chains(0, \ell), p_j = 1\|C_{max} < D$	$W[2]$-hard($\ell_{max} + \#$chains) with minimum delays.
		$P\|chains(0, \ell), p_j = 1, r_j, d_j\|\star$	para-*NP*-hard($\ell = 1$) with minimum delays.
μ	4.2	$1\|prec, r_j, \bar{d}_j\|C_{max}$	*FPT* in time $\mathcal{O}(\mu^2 \cdot 4^{\mu} \cdot n + n^2)$.
	4.3	$1\|chains(\ell_{ij}), p_j = 1, r_j, \bar{d}_j\|\star$	para-*NP*-hard for all three delay types.
$\mu + \ell_{max}$	4.4	$P\|prec(\ell_{i,j}), p_j = 1, r_j, \bar{d}_j\|\star$	*FPT* with minimum delays.
q	6.2	-	$q \leq \mu$
		$P2\|r_j, \bar{d}_j\|\star$ $1\|(1, \ell, 1), r_j, \bar{d}_j\|\star$	Inferred para-*NP*-hardness results.
	6.3	$1\|prec, r_j, \bar{d}_j\|C_{max}$	*FPT* in time $\mathcal{O}(max(1, q^2 \cdot 4^q) \cdot N + n^2)$.
	6.4	$P\|prec, p_j = 1, r_j, \bar{d}_j\|\star$	para-*NP*-hard($q + D$).
			$W[2]$-hard($q + w$).

PhD results (2/2)

Parameter	Section	Problem / Setting	Result
σ	5.2	Single machine	$\mu \leq 2\sigma$
		Pm	$\mu \leq 2(\sigma+1)\cdot m - 1$
		-	Several inferred *FPT* results.
	5.3	$1\|(1,\ell,1), r_j, \bar{d}_j\|\star$	para-*NP*-hard for all three delay types.
$\sigma + \ell_{max}$	5.4	$1\|prec(\ell_{i,j}), r_j, \bar{d}_j\|\star$	*FPT* with all three delay types combined.
μ	7.1	$1\|r_j, \bar{d}_j\|\star$	No polynomial kernel (unless $NP \subseteq coNP/poly$).
vc		$1\|prec, r_j, \bar{d}_j\|\star$	Polynomial kernel with both $(vc + p_{max})$ and vc.
		$P\|\mathcal{M}_j, r_j, \bar{d}_j\|\star$	$W[1]$-hard.
tww, $pthin$, cw	7.2	$1\|r_j, \bar{d}_j\|\star$ $P\|tree, p_j = 1, r_j, \bar{d}_j\|\star$	para-*NP*-hard.
μ_{avg}, σ_{avg}, q_{avg}	7.3	-	Several inferred *FPT* results.

PhD results (2/2)

Parameter	Section	Problem / Setting	Result
σ	5.2	Single machine	$\mu \leq 2\sigma$
		Pm	$\mu \leq 2(\sigma+1)\cdot m - 1$
		-	Several inferred *FPT* results.
	5.3	$1\|(1,\ell,1), r_j, \bar{d}_j\|\star$	para-*NP*-hard for all three delay types.
$\sigma + \ell_{max}$	5.4	$1\|prec(\ell_{i,j}), r_j, \bar{d}_j\|\star$	*FPT* with all three delay types combined.
μ	7.1	$1\|r_j, \bar{d}_j\|\star$	No polynomial kernel (unless $NP \subseteq coNP/poly$).
vc		$1\|prec, r_j, \bar{d}_j\|\star$	Polynomial kernel with both $(vc + p_{max})$ and vc.
		$P\|\mathcal{M}_j, r_j, \bar{d}_j\|\star$	$W[1]$-hard.
tww, $pthin$, cw	7.2	$1\|r_j, \bar{d}_j\|\star$ $P\|tree, p_j = 1, r_j, \bar{d}_j\|\star$	para-*NP*-hard.
μ_{avg}, σ_{avg}, q_{avg}	7.3	-	Several inferred *FPT* results.

Further work

- Consider other parameters based on the job time window interval graph.
- Find kernels associated to the *FPT* algorithms.
- Refine the approaches (bounds, heuristics).

Publications & Conferences

- **International Conferences with Program committees and Proceedings**

  ISCO 2024 [Mallem, Hanen, Munier 24]
  - Title: *A new structural parameter on single machine scheduling with release dates and deadlines.*
  - Contents: sections 6.2, 6.3 (proper level q).

  IPEC 2022 [Mallem, Hanen, Munier 22]
  - Title: *Parameterized Complexity of a Parallel Machine Scheduling Problem.*
  - Contents: sections 3.3, 4.3, 4.4 (pathwidth μ and max. delay value ℓ_{max}).
- **Other International Conferences**

  MAPSP 2022
  - Title: *Parameterized Complexity of a Single Machine Scheduling Problem with Precedence, Release Dates and Deadlines.*
  - Contents: sections 4.2, 4.3 (pathwidth μ: negative result and *FPT*).
- **National Conferences**: ROADEF 2022, ROADEF 2024
- **In Preparation**

  Theoretical Computer Science
  - Title: *Parameterized analysis of single machine scheduling with time windows and precedence delays.*
  - Contents: chapter 5 (slack).

  INFORMS Journal on Computing
  - Title: *Kernelization Algorithms on Machine Scheduling with Time Windows.*
  - Contents: section 7.1 (pathwidth μ proper level q and vertex cover vc).