Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

Parameterized Complexity of Single-machine Scheduling with Precedence, Release Dates and Deadlines

Claire Hanen, Maher Mallem, Alix Munier-Kordon

June 14th 2022

Going beyond NP-hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Single machine scheduling with precedence, release dates and deadlines
  - $1|prec, r_j, d_j|C_{max}$
  - $1|prec, r_j|L_{max}$

Going beyond NP-hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Single machine scheduling with precedence, release dates and deadlines
  - $1|prec, r_j, d_j|C_{max}$
  - $1|prec, r_j|L_{max}$
- Precedence + release dates
  - [Lawler 1973] $1|prec, r_j|C_{max}$ is in P.

Going beyond NP-hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Single machine scheduling with precedence, release dates and deadlines
  - $1|prec, r_j, d_j|C_{max}$
  - $1|prec, r_j|L_{max}$
- Precedence + release dates
  - [Lawler 1973] $1|prec, r_j|C_{max}$ is in P.
- Precedence + deadlines
  - [Lawler 1973] $1|prec|L_{max}$ is in P.

Going beyond NP-hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Single machine scheduling with precedence, release dates and deadlines
  - $1|prec, r_j, d_j|C_{max}$
  - $1|prec, r_j|L_{max}$
- Precedence + release dates
  - [Lawler 1973] $1|prec, r_j|C_{max}$ is in P.
- Precedence + deadlines
  - [Lawler 1973] $1|prec|L_{max}$ is in P.
- Release dates + deadlines
  - [Brucker et al. 1977] $1|r_j|L_{max}$ is strongly NP-hard.
  - $1|r_j, d_j|\star$ is strongly NP-hard.

FPT vs para-NP

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- P $\subseteq$ NP
- P: deterministic time $poly(n)$
- NP: nondeterministic time $poly(n)$

FPT vs para-NP

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- P $\subseteq$ NP
  - FPT $\subseteq$ para-NP, parameter k
- P: deterministic time $poly(n)$
  - FPT: deterministic time $f(k) \times poly(n)$
- NP: nondeterministic time $poly(n)$
  - para-NP: nondeterministic time $f(k) \times poly(n)$

FPT vs para-NP

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- P $\subseteq$ NP
  - FPT $\subseteq$ para-NP, parameter k
- P: deterministic time $poly(n)$
  - FPT: deterministic time $f(k) \times poly(n)$
- NP: nondeterministic time $poly(n)$
  - para-NP: nondeterministic time $f(k) \times poly(n)$
- P = NP iff $\exists k$, FPT = para-NP [Flum, Grohe 1998]

Parameterized example: SAT

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- SATISFIABILITY
  - INPUT: a set S of variables, a boolean formula $\mathcal{F}$ of size n based on S

    QUESTION: $\exists\ \mathcal{I} : S \rightarrow \{TRUE, FALSE\}$ s.t. $\mathcal{F}_{|\mathcal{I}} = TRUE$

Parameterized example: SAT

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- SATISFIABILITY
  - INPUT: a set S of variables, a boolean formula $\mathcal{F}$ of size n based on S

    QUESTION: $\exists\ \mathcal{I} : S \rightarrow \{TRUE, FALSE\}$ s.t. $\mathcal{F}_{|\mathcal{I}} = TRUE$
  - NP-complete

Parameterized example: SAT

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- SATISFIABILITY
  - INPUT: a set S of variables, a boolean formula $\mathcal{F}$ of size n based on S

    QUESTION: $\exists\ \mathcal{I} : S \rightarrow \{TRUE, FALSE\}$ s.t. $\mathcal{F}_{|\mathcal{I}} = TRUE$
  - NP-complete
- Parameterized SATISFIABILITY
  - INPUT: a set S of variables, a boolean formula $\mathcal{F}$ of size n based on S
    PARAMETER: the number k of distinct variables used in $\mathcal{F}$
    QUESTION: $\exists\ \mathcal{I} : S \rightarrow \{TRUE, FALSE\}$ s.t. $\mathcal{F}_{|\mathcal{I}} = TRUE$

Parameterized example: SAT

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- SATISFIABILITY
  - INPUT: a set S of variables, a boolean formula $\mathcal{F}$ of size n based on S

    QUESTION: $\exists\, \mathcal{I} : S \to \{TRUE, FALSE\}$ s.t. $\mathcal{F}_{|\mathcal{I}} = TRUE$
  - NP-complete
- Parameterized SATISFIABILITY
  - INPUT: a set S of variables, a boolean formula $\mathcal{F}$ of size n based on S
    PARAMETER: the number k of distinct variables used in $\mathcal{F}$
    QUESTION: $\exists\, \mathcal{I} : S \to \{TRUE, FALSE\}$ s.t. $\mathcal{F}_{|\mathcal{I}} = TRUE$
  - NP-complete but FPT algorithm in time $\mathcal{O}(2^k \times n)$

Parameterized example: $P||C_{max}$

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- $P||C_{max}$
  - INPUT: n jobs of processing times $p_1, \ldots, p_n$, m parallel machines

    OBJECTIVE: minimize C_{max}

Parameterized example: $P||C_{max}$

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- $P||C_{max}$
  - INPUT: n jobs of processing times $p_1, \ldots, p_n$, m parallel machines

    OBJECTIVE: minimize C_{max}
  - [Garey, Johnson 1978] NP-complete

Parameterized example: $P||C_{max}$

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- $P||C_{max}$
  - INPUT: n jobs of processing times $p_1, \ldots, p_n$, m parallel machines

    OBJECTIVE: minimize C_{max}
  - [Garey, Johnson 1978] NP-complete
- Parameterized $P||C_{max}$
  - INPUT: n jobs of processing times $p_1, \ldots, p_n$, m parallel machines
    PARAMETER: maximum processing time p_{max}
    OBJECTIVE: minimize C_{max}

Parameterized example: $P||C_{max}$

- $P||C_{max}$
  - INPUT: n jobs of processing times $p_1, \ldots, p_n$, m parallel machines

    OBJECTIVE: minimize C_{max}
  - [Garey, Johnson 1978] NP-complete
- Parameterized $P||C_{max}$
  - INPUT: n jobs of processing times $p_1, \ldots, p_n$, m parallel machines
    PARAMETER: maximum processing time p_{max}
    OBJECTIVE: minimize C_{max}
  - [Mnich, Wiese 2015] NP-complete but FPT algorithm in time $f(p_{max}) \times (n+m)^{\mathcal{O}(1)}$

Proving hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Proving para-NP-hardness

Proving hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Proving para-NP-hardness
  - Prove NP-hardness for a fixed value of the parameter [Flum, Grohe 1998]

Proving hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Proving para-NP-hardness
  - Prove NP-hardness for a fixed value of the parameter [Flum, Grohe 1998]
- W-hierarchy: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ para-NP

Proving hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Proving para-NP-hardness
  - Prove NP-hardness for a fixed value of the parameter [Flum, Grohe 1998]
- W-hierarchy: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ para-NP
  - W[1]-complete: k-INDEPENDENT SET, k-CLIQUE
  - W[2]-complete: k-DOMINATING SET
  - W[t]-complete, $t \geq 1$: WEIGHTED t-NORMALIZED SAT

Proving hardness

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Proving para-NP-hardness
  - Prove NP-hardness for a fixed value of the parameter [Flum, Grohe 1998]
- W-hierarchy: FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ para-NP
  - W[1]-complete: k-INDEPENDENT SET, k-CLIQUE
  - W[2]-complete: k-DOMINATING SET
  - W[t]-complete, $t \geq 1$: WEIGHTED t-NORMALIZED SAT
- Usually: "W[1]-hard $\Longrightarrow$ not FPT"

Some scheduling examples

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- [Bodlaender, Fellows 1995]
  - #machines m: $P|prec, p_j = 1|C_{max}$ is W[2]-hard.

Some scheduling examples

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- [Bodlaender, Fellows 1995]
  - #machines m: $P|prec, p_j = 1|C_{max}$ is W[2]-hard.
- [Mnich, Wiese 2015]
  - Maximum processing time p_{max}: $P||C_{max} \in$ FPT.

Some scheduling examples

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- [Bodlaender, Fellows 1995]
  - #machines m: $P|prec, p_j = 1|C_{max}$ is W[2]-hard.
- [Mnich, Wiese 2015]
  - Maximum processing time p_{max}: $P||C_{max} \in$ FPT.
- [Hermelin, Karhi, Pinedo, Shabtay 2018]
  - #distinct d_j, #distinct p_j, #distinct w_j: $1||\sum w_j U_j \in$ FPT with two of the three.

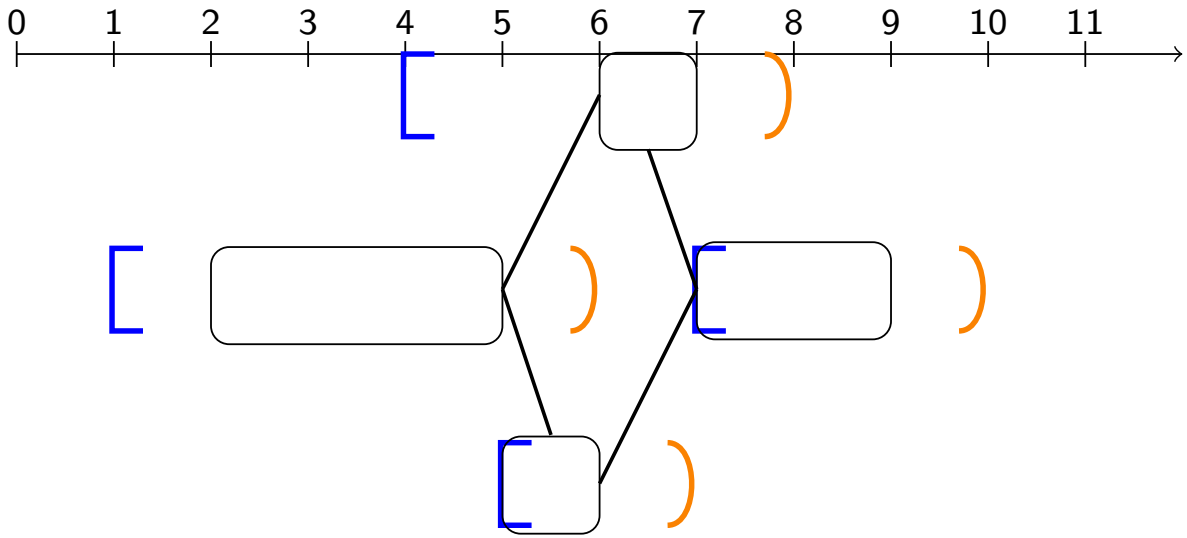
Some scheduling examples

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- [Bodlaender, Fellows 1995]
  - #machines m: $P|prec, p_j = 1|C_{max}$ is W[2]-hard.
- [Mnich, Wiese 2015]
  - Maximum processing time p_{max}: $P||C_{max} \in$ FPT.
- [Hermelin, Karhi, Pinedo, Shabtay 2018]
  - #distinct d_j, #distinct p_j, #distinct w_j: $1||\sum w_j U_j \in$ FPT with two of the three.
- Release dates + deadlines?
  - look for structural properties

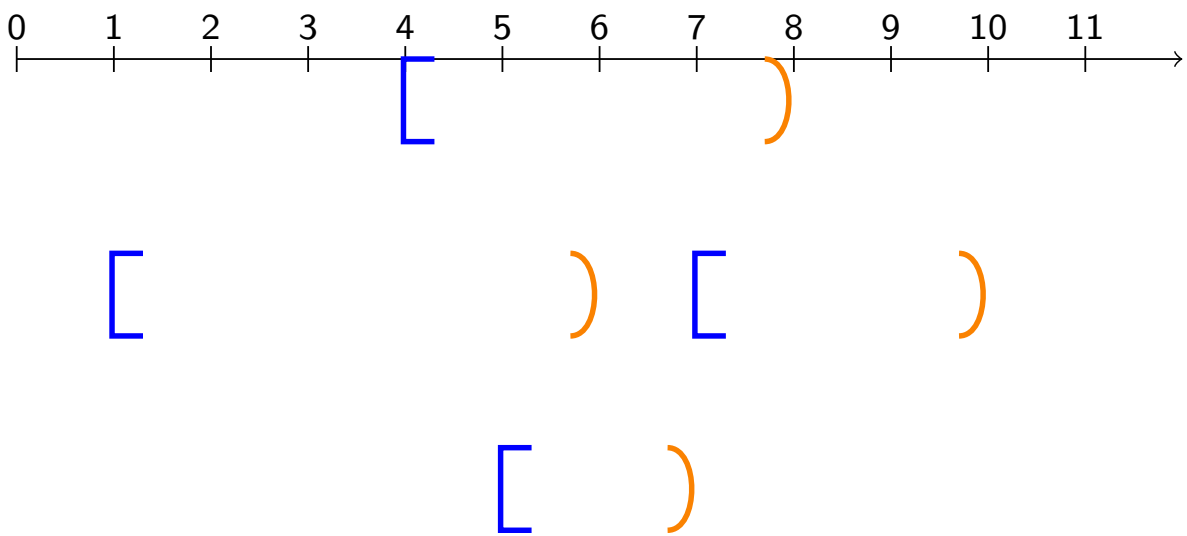
Pathwidth μ

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



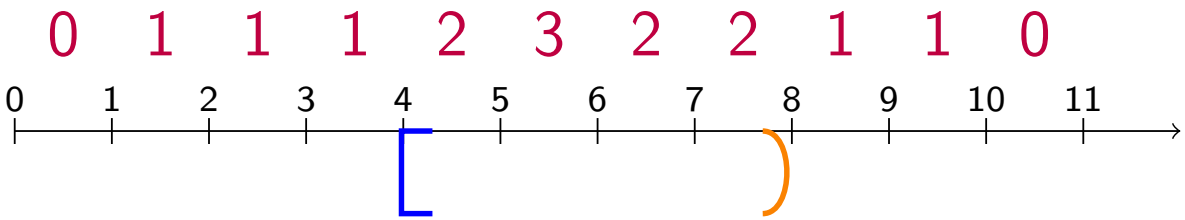
Pathwidth μ

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



Pathwidth μ

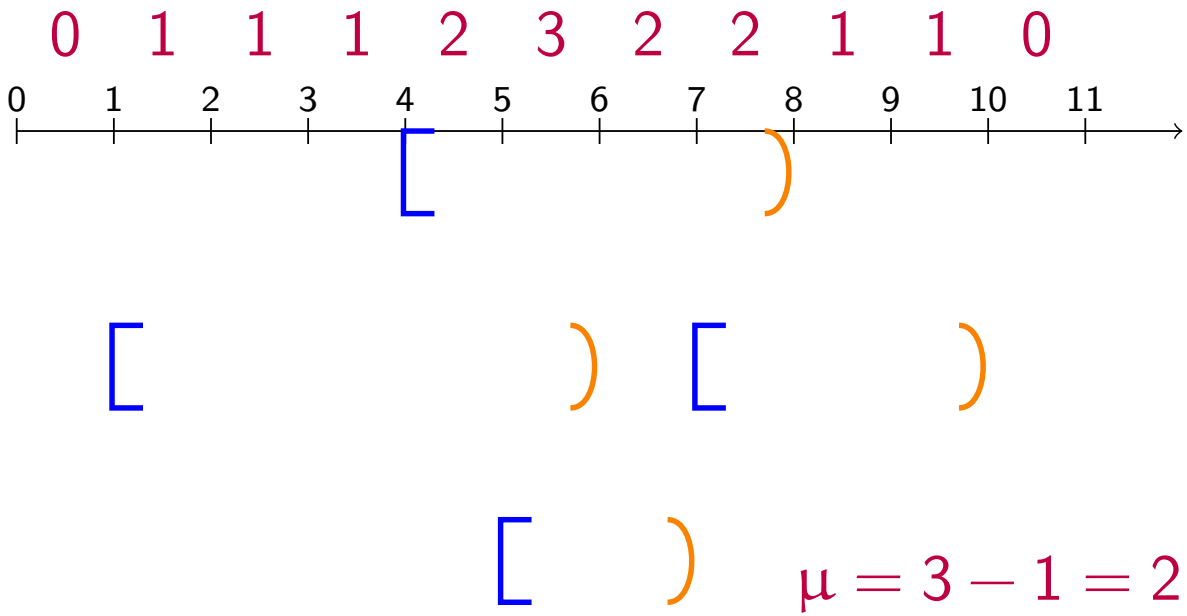
Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



$\mu = 3 - 1 = 2$

Pathwidth μ

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

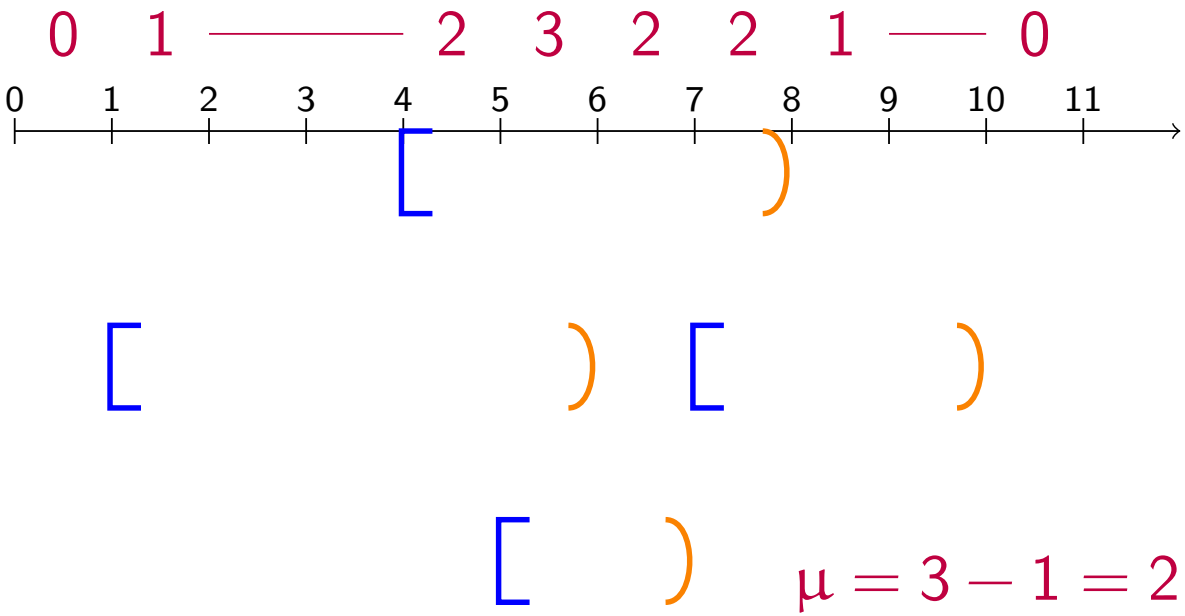


Definition: pathwidth

$$\mu \stackrel{\text{def}}{=} \max_{0 \leq t < \max_j d_j} |\{j \mid r_j \leq t < d_j\}| - 1$$

Pathwidth μ

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

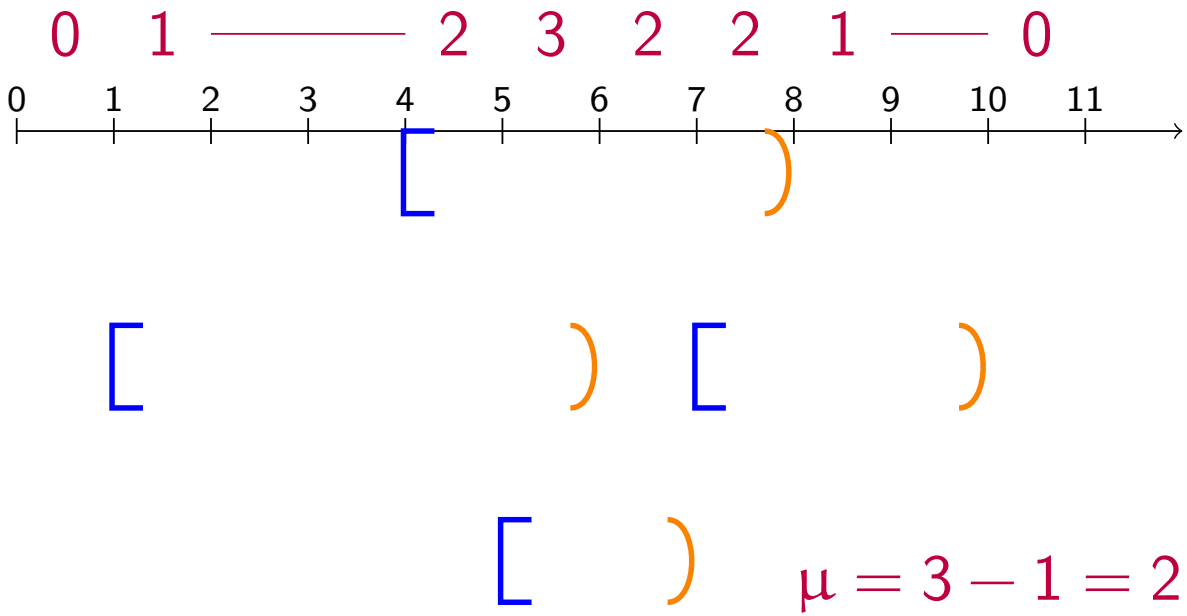


Definition: pathwidth

$$\mu \stackrel{\text{def}}{=} \max_{0 \leq t < \max_j d_j} |\{j \mid r_j \leq t < d_j\}| - 1$$

Pathwidth μ

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



Definition: pathwidth

$$\mu \stackrel{\text{def}}{=} \max_{0 \leq t < \max_j d_j} |\{j \mid r_j \leq t < d_j\}| - 1$$

- $=$ Pathwidth of the interval graph induced by the time windows
  - Nodes $=$ time windows
  - there is an edge if they intersect

Pathwidth μ (cont.)

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

Recent work:

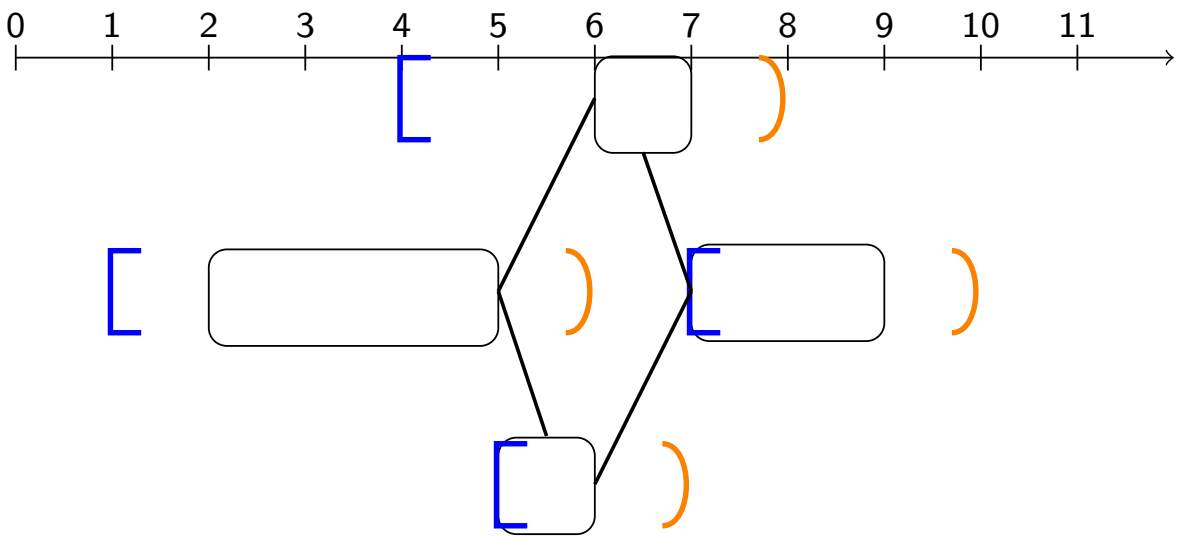
- [de Weerdt et al. 2020] $1|r_j, s_{jk}, reject, \tilde{d}_j| \sum_{j \notin R} v_j - \sum_{j \notin R} w_j T_j$ is FPT parameterized by (μ+slack).

- [Munier-Kordon, Tang 2021] $P|prec, p_j = 1, r_j|L_{max}$ is FPT parameterized by pathwidth μ.

$1|prec, r_j, d_j|C_{max}$ is FPT parameterized by μ

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

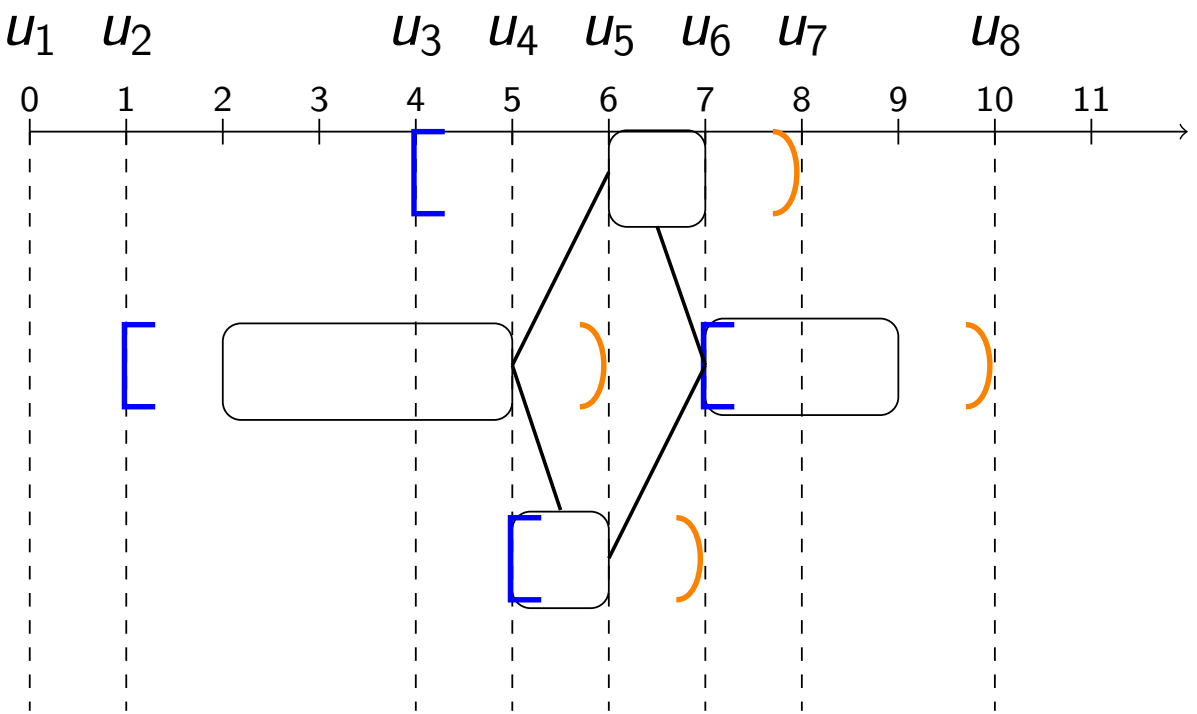
- Dynamic programming algorithm on active schedules



$1|prec, r_j, d_j|C_{max}$ is FPT parameterized by μ

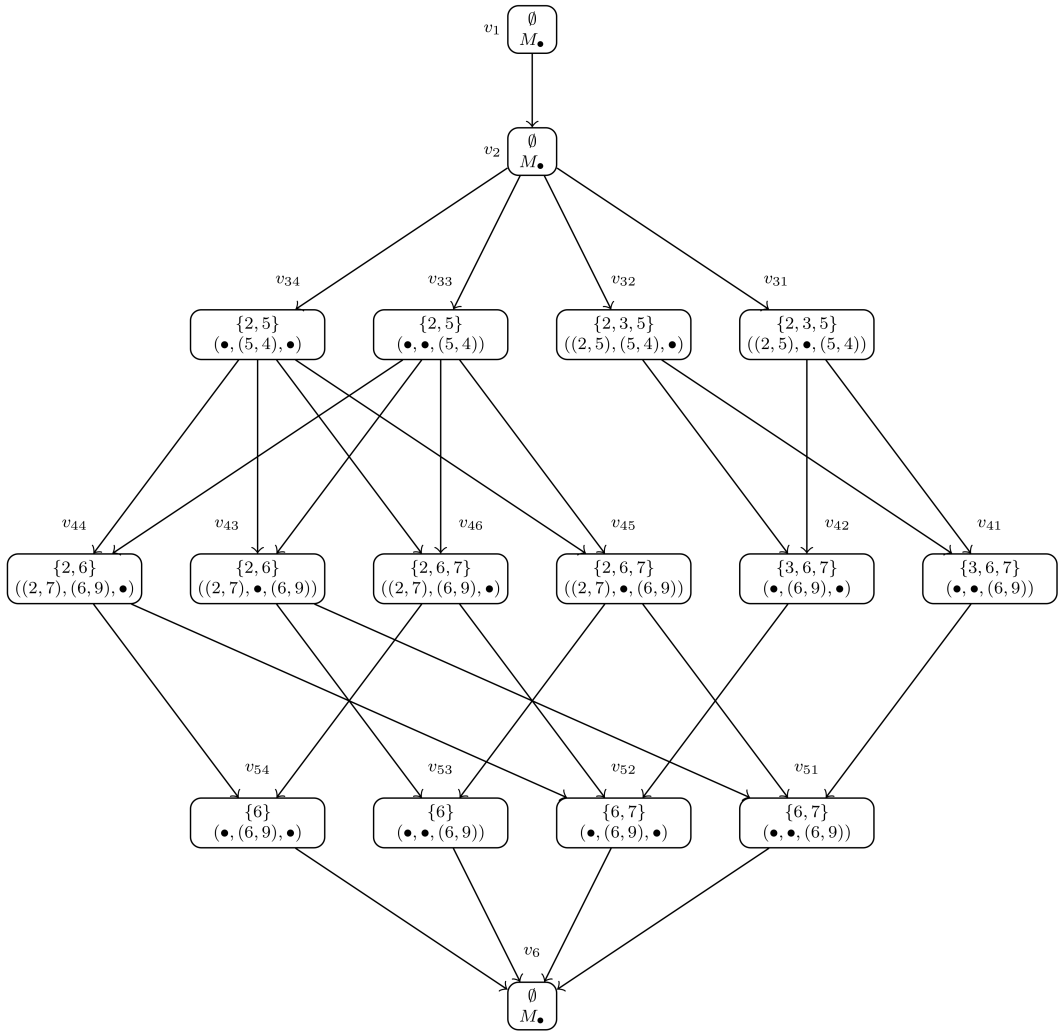
Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Dynamic programming algorithm on active schedules



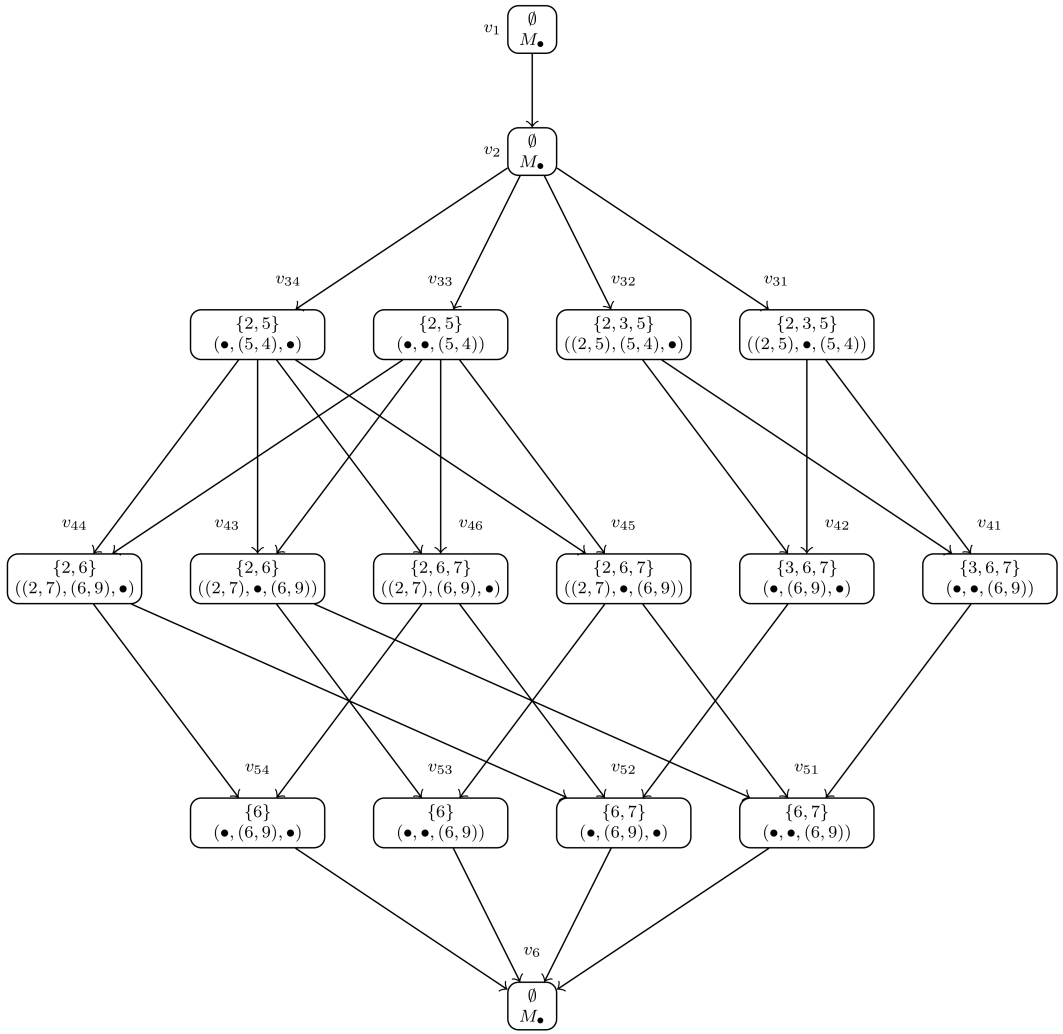
$1|prec, r_j, d_j|C_{max}$ is FPT parameterized by μ (cont.)

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



$1|prec, r_j, d_j|C_{max}$ is FPT parameterized by μ (cont.)

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



- FPT algorithm in time $\mathcal{O}(2^{2\mu} \times n^2)$

Pathwidth vs. precedence delays

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Delays added to precedence relations: what happens?
- If σ is a feasible schedule:
  - Exact delay $l^{ex}_{i,j}$: $J_i \prec J_j \implies \sigma(j) = \sigma(i) + p_i + l^{ex}_{i,j}$
  - Minimum delay $l^{min}_{i,j}$: $J_i \prec J_j \implies \sigma(j) \geq \sigma(i) + p_i + l^{min}_{i,j}$

Pathwidth vs. precedence delays

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Delays added to precedence relations: what happens?
- If σ is a feasible schedule:
  - Exact delay $l^{ex}_{i,j}$: $J_i \prec J_j \implies \sigma(j) = \sigma(i) + p_i + l^{ex}_{i,j}$
  - Minimum delay $l^{min}_{i,j}$: $J_i \prec J_j \implies \sigma(j) \geq \sigma(i) + p_i + l^{min}_{i,j}$
- [Bodlaender, van der Wegen 2020]
  $1|chains(l^{ex}_{i,j}), p_j = 1, r_C, d_C|\star$ and $1|chains(l^{min}_{i,j}), p_j = 1, r_C, d_C|\star$ with unary delays are W[t]-hard parameterized by #chains (or thickness) for all $t \geq 1$.

Pathwidth vs. precedence delays

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Delays added to precedence relations: what happens?
- If σ is a feasible schedule:
  - Exact delay $l^{ex}_{i,j}$: $J_i \prec J_j \implies \sigma(j) = \sigma(i) + p_i + l^{ex}_{i,j}$
  - Minimum delay $l^{min}_{i,j}$: $J_i \prec J_j \implies \sigma(j) \geq \sigma(i) + p_i + l^{min}_{i,j}$
- [Bodlaender, van der Wegen 2020]
  $1|chains(l^{ex}_{i,j}), p_j = 1, r_C, d_C|\star$ and $1|chains(l^{min}_{i,j}), p_j = 1, r_C, d_C|\star$ with unary delays are W[t]-hard parameterized by #chains (or thickness) for all $t \geq 1$.
- Feasibility problems
  - $1|chains(l^{ex}_{i,j}), p_j = 1, r_j, d_j|\star$ parameterized by μ
  - $1|chains(l^{min}_{i,j}), p_j = 1, r_j, d_j|\star$ parameterized by μ
- Result: para-NP-hardness in both cases

Pathwidth vs. precedence delays

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- Delays added to precedence relations: what happens?
- If σ is a feasible schedule:
  - Exact delay $l^{ex}_{i,j}$: $J_i \prec J_j \implies \sigma(j) = \sigma(i) + p_i + l^{ex}_{i,j}$
  - Minimum delay $l^{min}_{i,j}$: $J_i \prec J_j \implies \sigma(j) \geq \sigma(i) + p_i + l^{min}_{i,j}$
- [Bodlaender, van der Wegen 2020]
  $1|chains(l^{ex}_{i,j}), p_j = 1, r_C, d_C|\star$ and $1|chains(l^{min}_{i,j}), p_j = 1, r_C, d_C|\star$ with unary delays are W[t]-hard parameterized by #chains (or thickness) for all $t \geq 1$.
- Feasibility problems
  - $1|chains(l^{ex}_{i,j}), p_j = 1, r_j, d_j|\star$ parameterized by μ
  - $1|chains(l^{min}_{i,j}), p_j = 1, r_j, d_j|\star$ parameterized by μ
- Result: para-NP-hardness in both cases
  - prove NP-hardness for a fixed value of the parameter [Flum, Grohe 1998]

Pathwidth vs. precedence delays

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

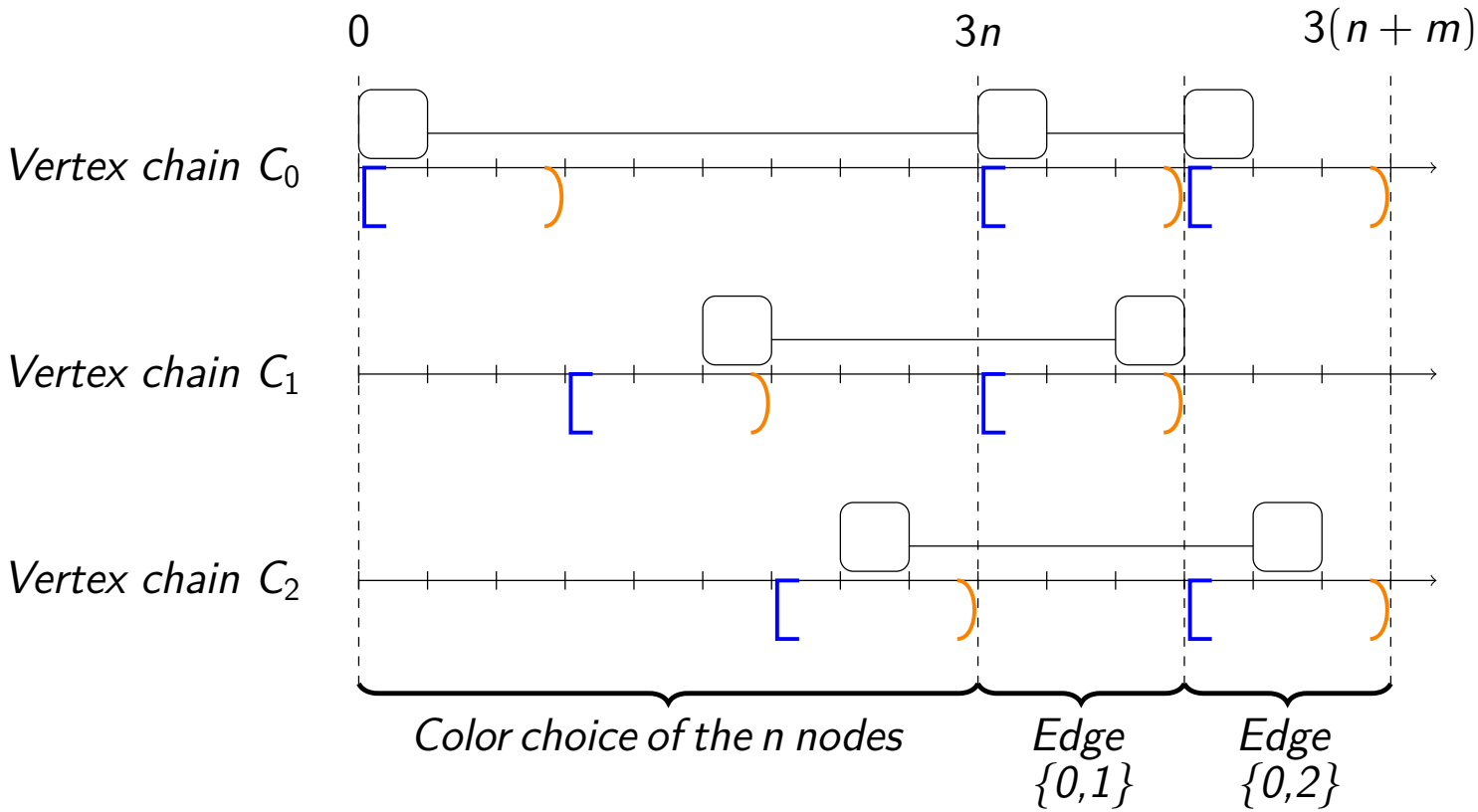
- Delays added to precedence relations: what happens?
- If σ is a feasible schedule:
  - Exact delay $l^{ex}_{i,j}$: $J_i \prec J_j \implies \sigma(j) = \sigma(i) + p_i + l^{ex}_{i,j}$
  - Minimum delay $l^{min}_{i,j}$: $J_i \prec J_j \implies \sigma(j) \geq \sigma(i) + p_i + l^{min}_{i,j}$
- [Bodlaender, van der Wegen 2020]
  $1|chains(l^{ex}_{i,j}), p_j = 1, r_C, d_C|\star$ and $1|chains(l^{min}_{i,j}), p_j = 1, r_C, d_C|\star$ with unary delays are W[t]-hard parameterized by #chains (or thickness) for all $t \geq 1$.
- Feasibility problems
  - $1|chains(l^{ex}_{i,j}), p_j = 1, r_j, d_j|\star$ parameterized by μ
  - $1|chains(l^{min}_{i,j}), p_j = 1, r_j, d_j|\star$ parameterized by μ
- Result: para-NP-hardness in both cases
  - prove NP-hardness for a fixed value of the parameter [Flum, Grohe 1998]
  - NP-hardness of $1|chains(l^{ex}_{i,j}), p_j = 1, r_j, d_j|\star$ with $\mu = 1$.

$1|chains(l_{i,j}^{ex}), p_j = 1, r_j, d_j|\star$, pathwidth 1

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

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

Example: $V = \{0, 1, 2\}, E = (\{0, 1\}, \{0, 2\})$

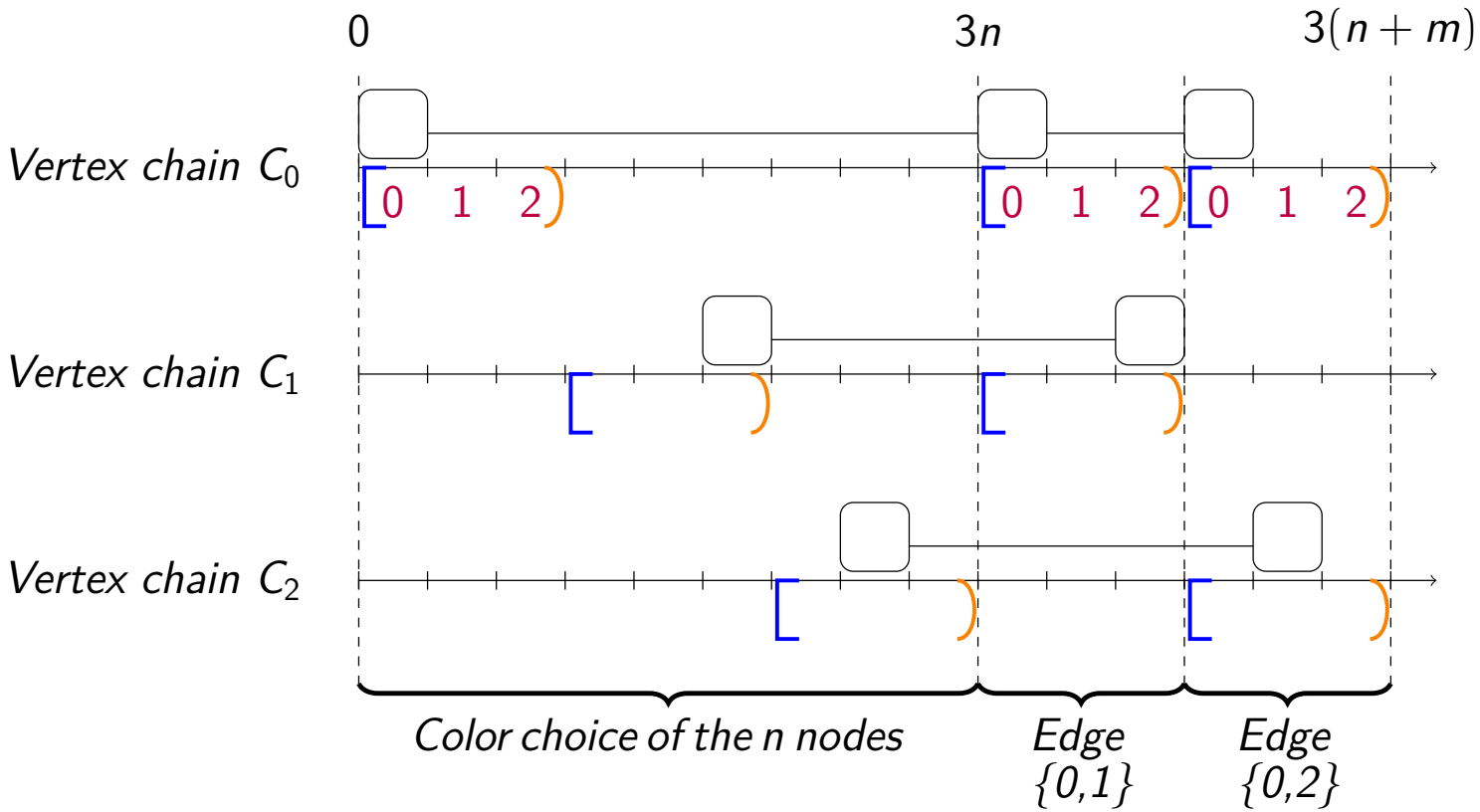


$1|chains(l_{i,j}^{ex}), p_j = 1, r_j, d_j|\star$, pathwidth 1

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

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

Example: $V = \{0, 1, 2\}, E = (\{0, 1\}, \{0, 2\})$

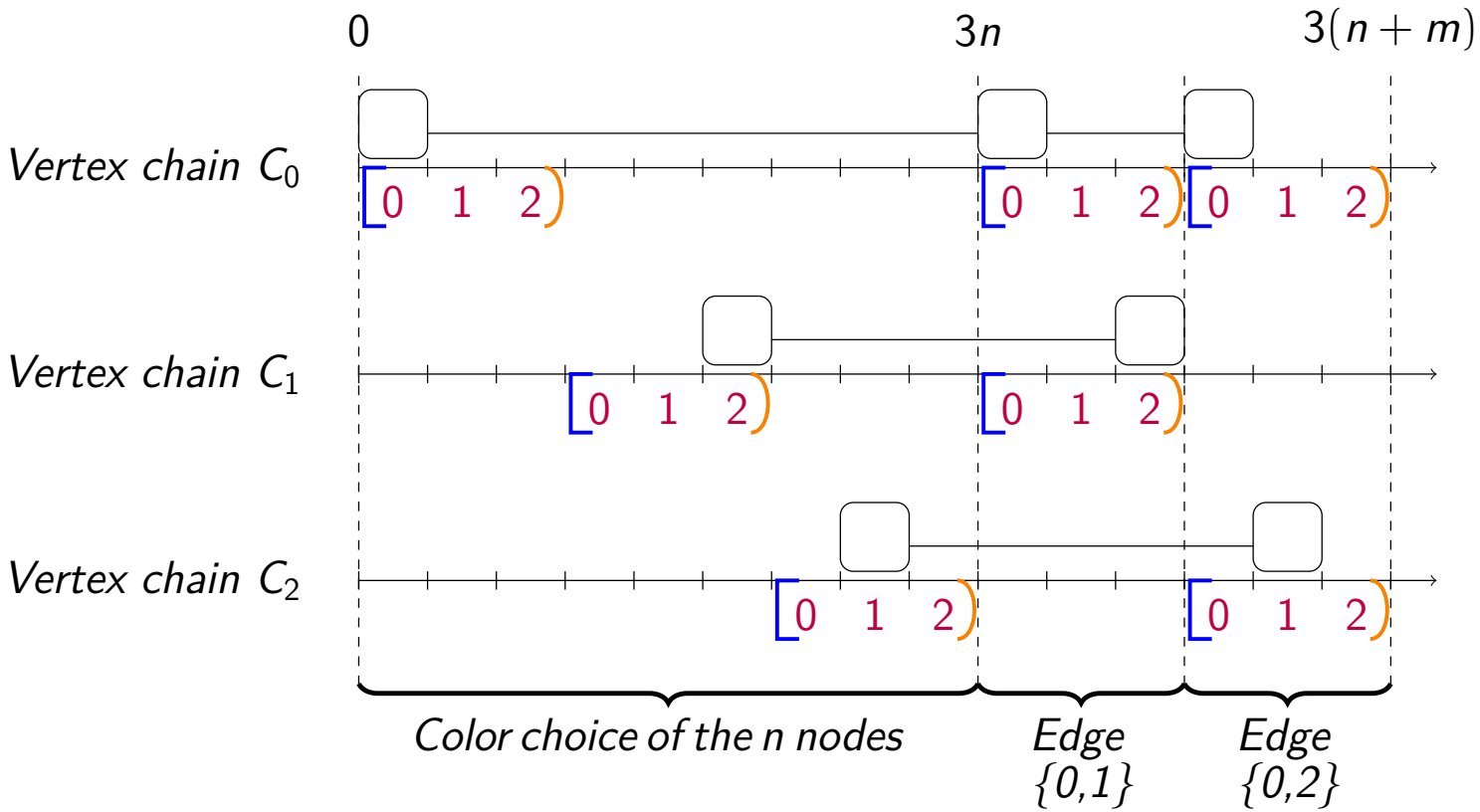


$1|chains(l_{i,j}^{ex}), p_j = 1, r_j, d_j|\star$, pathwidth 1

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

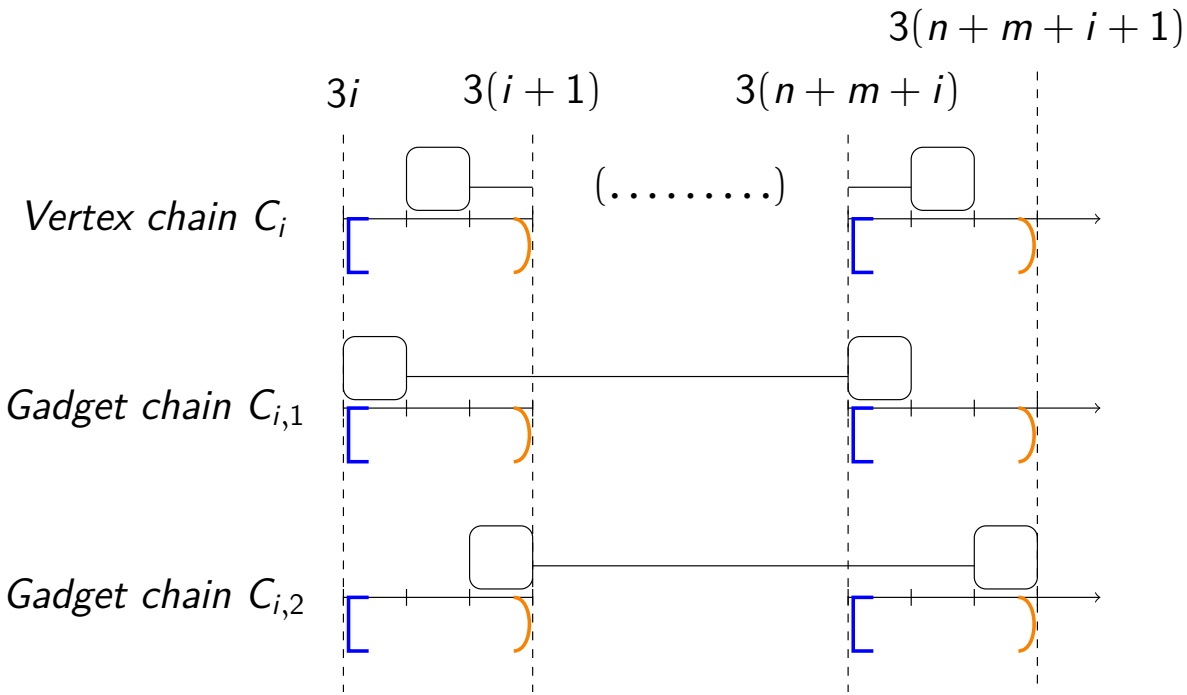
- Reduction from 3-COLORING: $G = (V, E), n = |V|, m = |E|$

Example: $V = \{0, 1, 2\}, E = (\{0, 1\}, \{0, 2\})$



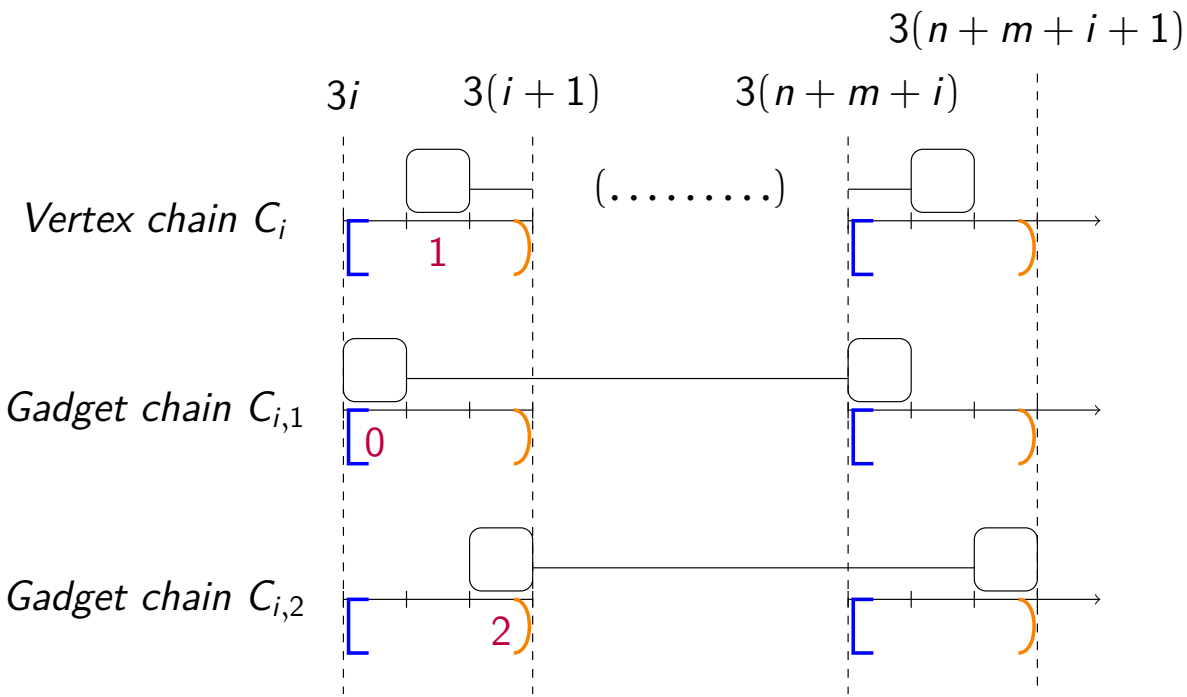
$1|chains(l_{i,j}^{min}), p_j = 1, r_j, d_j|\star$, pathwidth 2

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



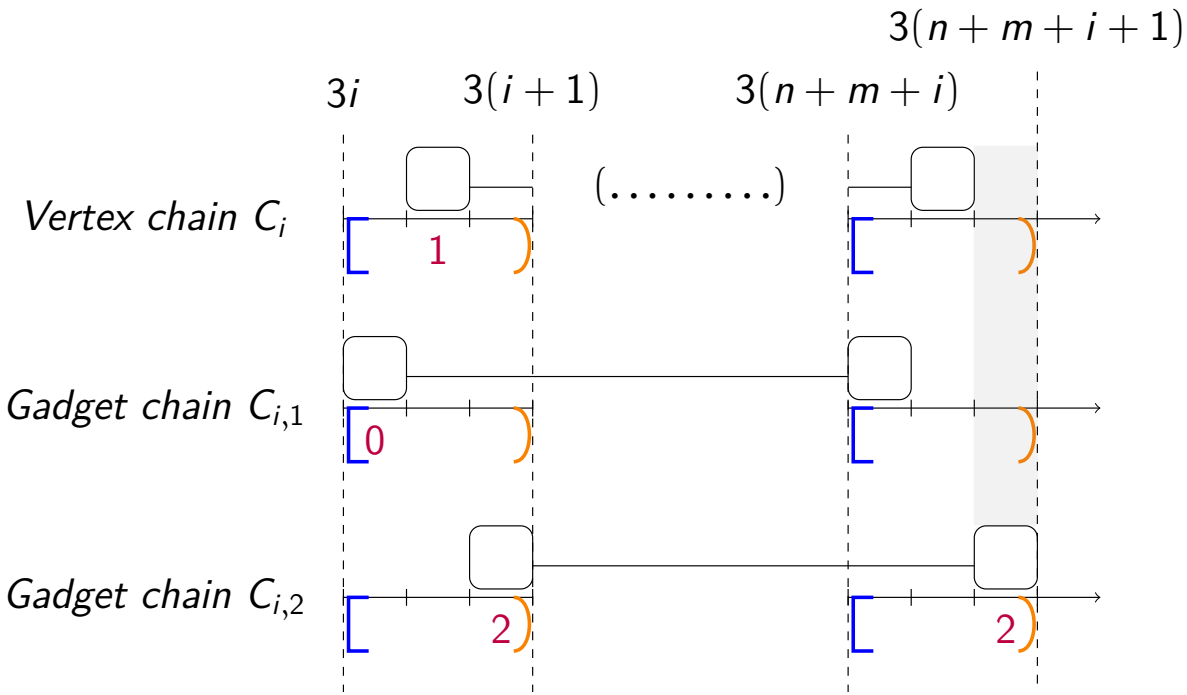
$1|chains(l_{i,j}^{min}), p_j = 1, r_j, d_j|\star$, pathwidth 2

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



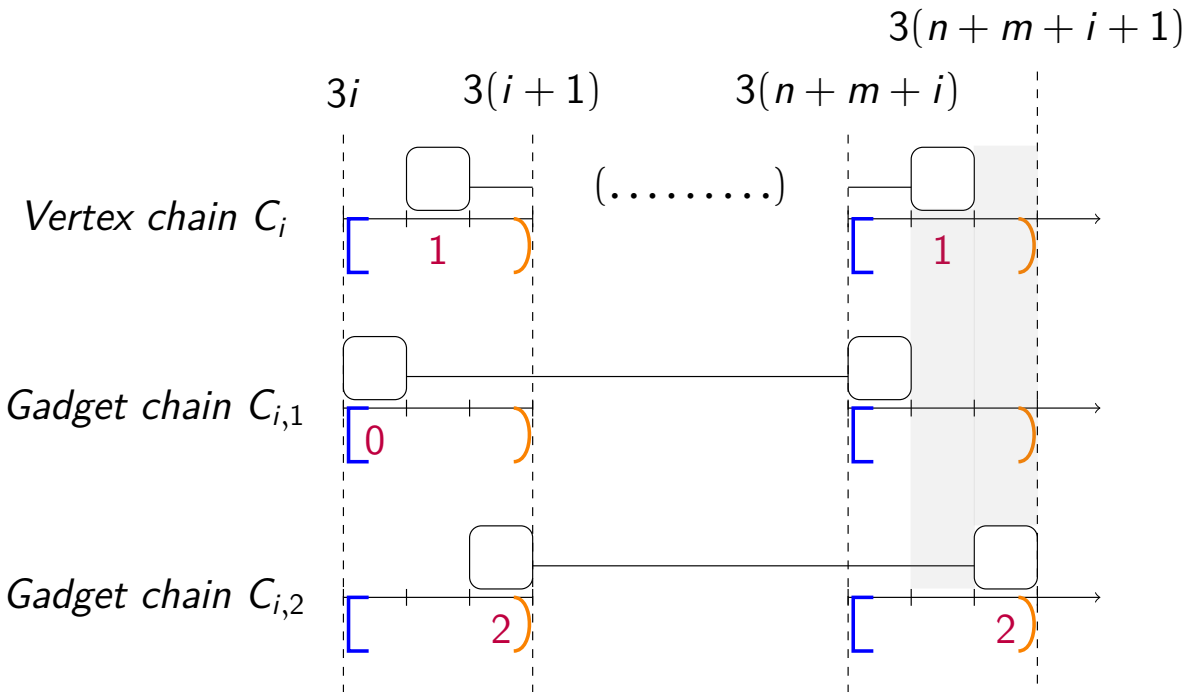
$1|chains(l_{i,j}^{min}), p_j = 1, r_j, d_j|\star$, pathwidth 2

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



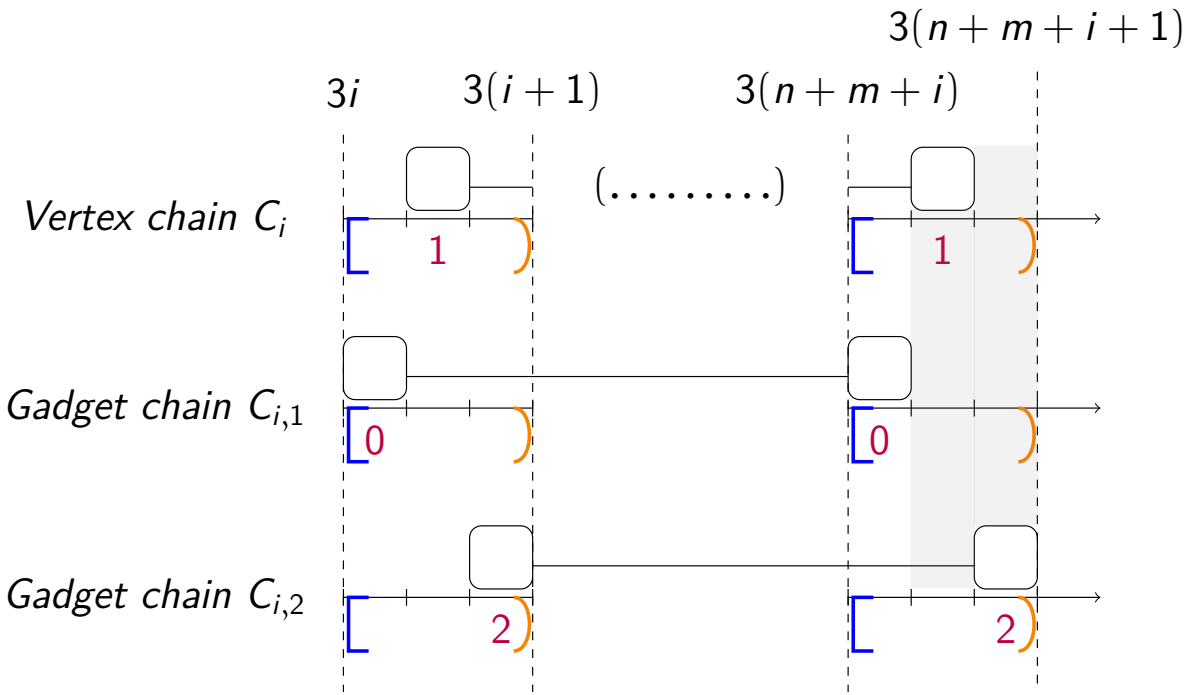
$1|chains(l_{i,j}^{min}), p_j = 1, r_j, d_j|\star$, pathwidth 2

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion



$1|chains(l_{i,j}^{min}), p_j = 1, r_j, d_j|\star$, pathwidth 2

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

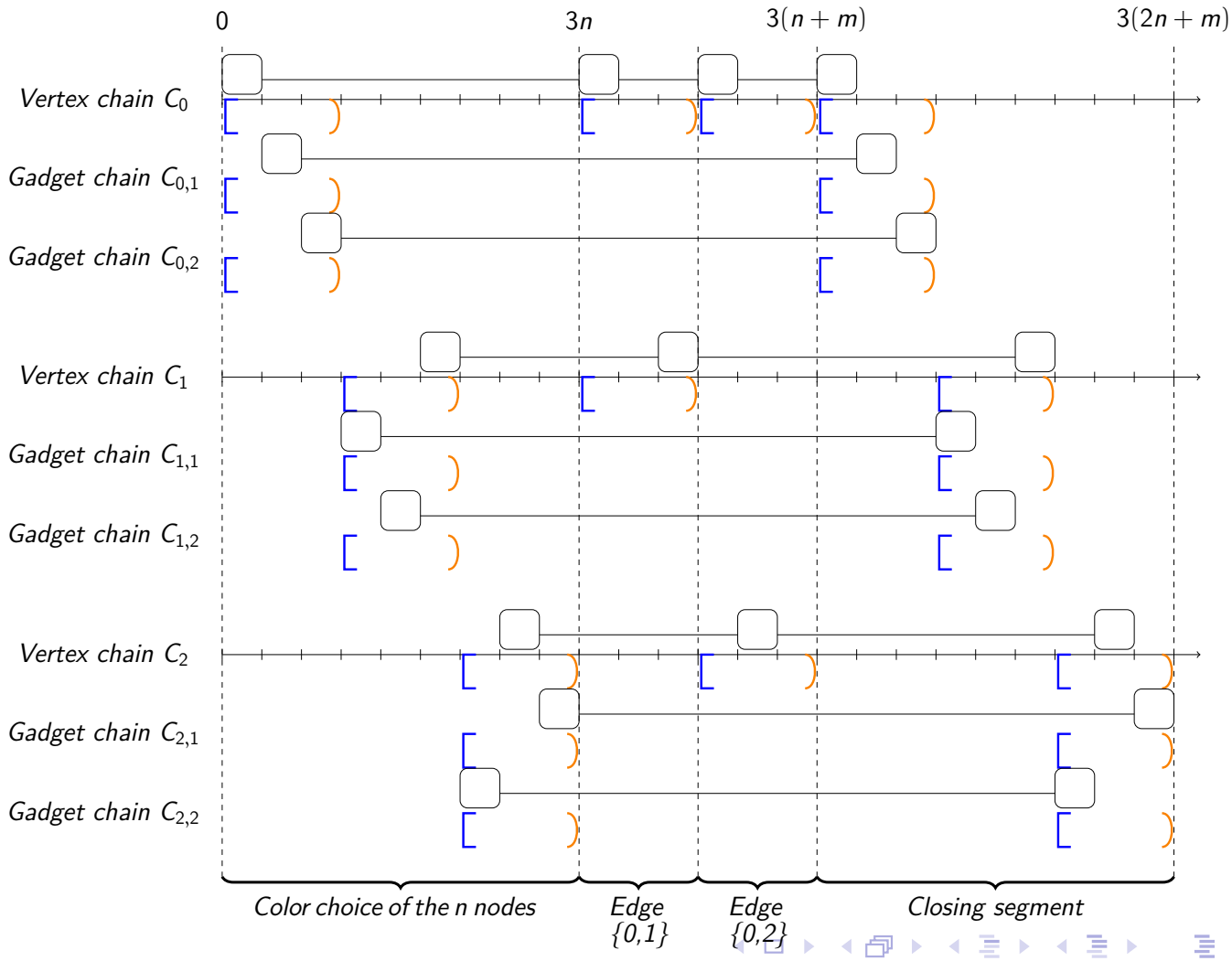


$1|chains(l_{i,j}^{min}), p_j = 1, r_j, d_j|\star$, pathwidth 2 (cont.)

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

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

Example: $V = \{0, 1, 2\}, E = (\{0, 1\}, \{0, 2\})$

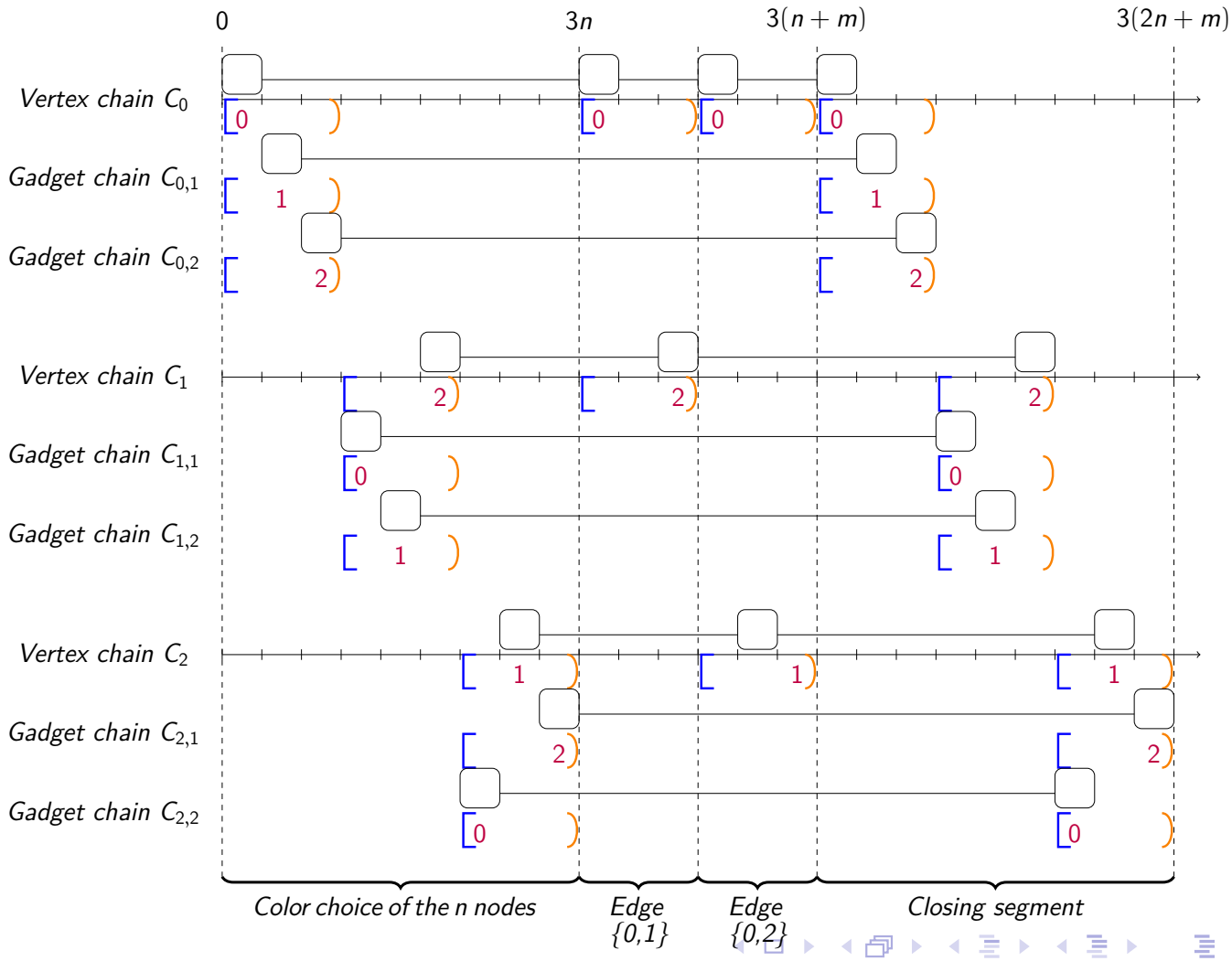


$1|chains(l_{i,j}^{min}), p_j = 1, r_j, d_j|\star$, pathwidth 2 (cont.)

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

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

Example: $V = \{0, 1, 2\}, E = (\{0, 1\}, \{0, 2\})$



Conclusion

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

Theorem

$1|prec, r_j, d_j|C_{max}$ is FPT parameterized by pathwidth μ.

Theorem

$1|chains(I^{ex}_{i,j}), p_j = 1, r_j, d_j|\star$ and $1|chains(I^{min}_{i,j}), p_j = 1, r_j, d_j|\star$ are para-NP-complete parameterized by pathwidth μ.

Conclusion

Theorem

$1|prec, r_j, d_j|C_{max}$ is FPT parameterized by pathwidth μ.

- [WIP] $P2|r_j, d_j|\star$ is para-NP-complete parameterized by pathwidth μ.

Theorem

$1|chains(I^{ex}_{i,j}), p_j = 1, r_j, d_j|\star$ and $1|chains(I^{min}_{i,j}), p_j = 1, r_j, d_j|\star$ are para-NP-complete parameterized by pathwidth μ.

Conclusion

Theorem

$1|prec, r_j, d_j|C_{max}$ is FPT parameterized by pathwidth μ.

- [WIP] $P2|r_j, d_j|\star$ is para-NP-complete parameterized by pathwidth μ.

Theorem

$1|chains(I^{ex}_{i,j}), p_j = 1, r_j, d_j|\star$ and $1|chains(I^{min}_{i,j}), p_j = 1, r_j, d_j|\star$ are para-NP-complete parameterized by pathwidth μ.

- Maximum delay l_{max} as a parameter
  - hard with parameter l_{max} alone?
  - FPT with parameter $\mu + l_{max}$?

Going further

Parameterized complexity
Definition
First examples
Proving hardness
Scheduling examples
Contribution
FPT result
Hardness reductions
Conclusion

- [Mnich, van Bevern 2018] Parameterized complexity of machine scheduling: 15 open problems

Parameterized complexity of machine scheduling: 15 open problems

Matthias Mnich[a,b], René van Bevern[c,d,*]

[a]Universität Bonn, Bonn, Germany
[b]Maastricht University, Maastricht, The Netherlands
[c]Department of Mechanics and Mathematics, Novosibirsk State University, Novosibirsk, Russian Federation
[d]Sobolev Institute of Mathematics of the Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russian Federation

Abstract

Machine scheduling problems are a long-time key domain of algorithms and complexity research. A novel approach to machine scheduling problems are fixed-parameter algorithms. To stimulate this thriving research direction, we propose 15 open questions in this area whose resolution we expect to lead to the discovery of new approaches and techniques both in scheduling and parameterized complexity theory.

Keywords: parallel machines, shop scheduling, makespan, total completion time, total tardiness, throughput, number of tardy jobs

1. Introduction

Algorithms for machine scheduling problems form one of the core applications of combinatorial optimization. In those problems, we are generally given a finite set J of jobs with certain characteristics, and we must find a *schedule* for processing the jobs on one or more machines, which also may have their individual specifications. Typical characteristics of a job j are its processing time $p_j \in \mathbb{N}$, its release date $r_j \in \mathbb{N}$, its due date $d_j \in \mathbb{N}$, or its importance reflected by a weight $w_j \in \mathbb{N}$. Jobs may be subject to precedence constraints enforcing some jobs to be completed before other jobs start. Also, jobs may be preempted, or may be required to be processed without preemption. Machine characteristics typically include their speed or whether they are capable of processing a certain type of job.

Usually, one is not only searching for a feasible schedule that respects all constraints, but additionally optimizes some objective function. Classical objectives include the minimization of the makespan or the sum of weighted completion times (which is equivalent to minimizing the weighted average completion time). Since the inception of the field in the 1950s, thousands of research papers have been devoted to understanding the complexity of scheduling problems. A significant portion of investigated problems turned out to be NP-hard. In consequence, algorithm designers proposed algorithms for these problems that yield approximate solutions in polynomial time. In 1999, Schuurman and Woeginger (1999) listed 10 of the most prominent open problems around polynomial-time approximation algorithms for NP-hard scheduling problems at that time.

Only recently, a different algorithmic approach has been put forward for solving NP-hard scheduling problems: *fixed-parameter algorithms*. The idea in fixed-parameter algorithms is to accept exponential running times, which are seemingly inevitable in solving NP-hard problems, but to restrict them to certain aspects of the problem, which are captured by *parameters* (Cygan et al., 2015; Flum and Grohe, 2006; Niedermeier, 2006). More formally, fixed-parameter algorithms solve an instance of size n with parameter k in $f(k) \cdot \text{poly}(n)$ time for some computable, typically superpolynomial, function f. Thus, fixed-parameter algorithms can solve even large instances of NP-hard scheduling problems if the parameter takes only small values, the function f grows only moderately, and the polynomial degree in n is small (van Bevern et al., 2015). Moreover, problems that are even difficult to approximate can be approximated efficiently and well in real-world instances using fixed-parameter *approximation* algorithms that exploit small parameters of real-world data (van Bevern et al., 2017a).

While fixed-parameter algorithms are now a well-investigated area of algorithmics, their systematic application to scheduling problems has gained momentum only recently (van Bevern and Pyatkin, 2016; Chen et al., 2017; Cieliebak et al., 2004; Halldórsson and Karlsson, 2006; Hermelin et al., 2015, 2018; Jansen et al., 2017; Marx and Schlotter, 2011, and more references below). This already led to the transfer of proof techniques from parameterized complexity to the world of scheduling, such as n-fold integer programming (Knop and Koutecký, 2017), color coding, problem kernelization (van Bevern et al., 2015), and W-hardness (Bentert et al., 2017; Bodlaender and Fellows, 1995; van Bevern et al., 2016, 2017b; Mnich and Wiese, 2015; Fellows and McCartin, 2003; Hermelin et al., 2017). It also led to the transfer of techniques from mathematical programming to parameterized complexity, such as convex integer programming (Mnich and Wiese, 2015) or parameterizing by structural properties of the integer feasible polytope of linear programs (Jansen and Klein, 2017).

In the following, we summarize known results and list open

*Corresponding author. René van Bevern, Department of Mechanics and Mathematics, Novosibirsk State University, ul. Pirogova 2, 630090 Novosibirsk, Russian Federation
Email addresses: mnich@uni-bonn.de (Matthias Mnich), rvb@nsu.ru (René van Bevern)

- [Flum, Grohe 2006] Parameterized Complexity Theory