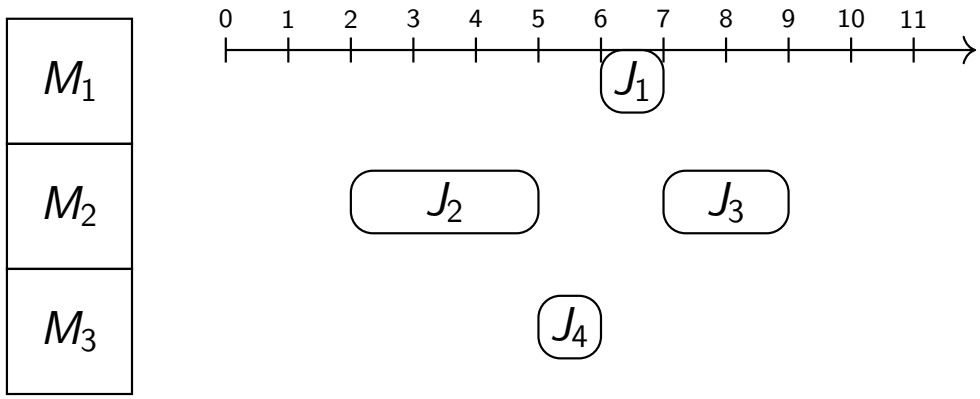
A New Structural Parameter on Single Machine Scheduling with Release Dates and Deadlines

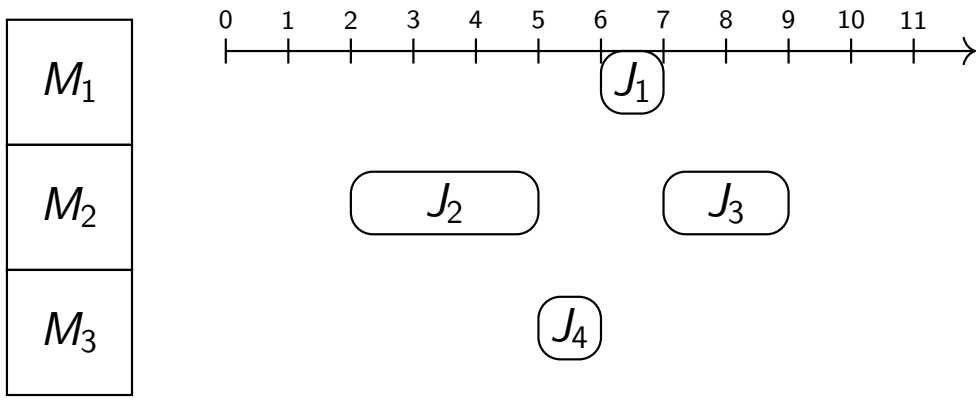
Maher Mallem, Claire Hanen, Alix Munier-Kordon

24 May 2024

Machine scheduling

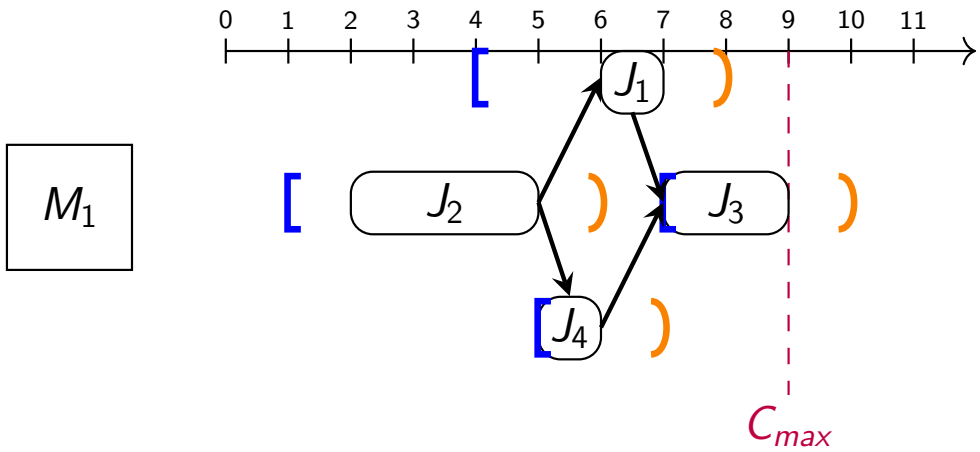


Machine scheduling



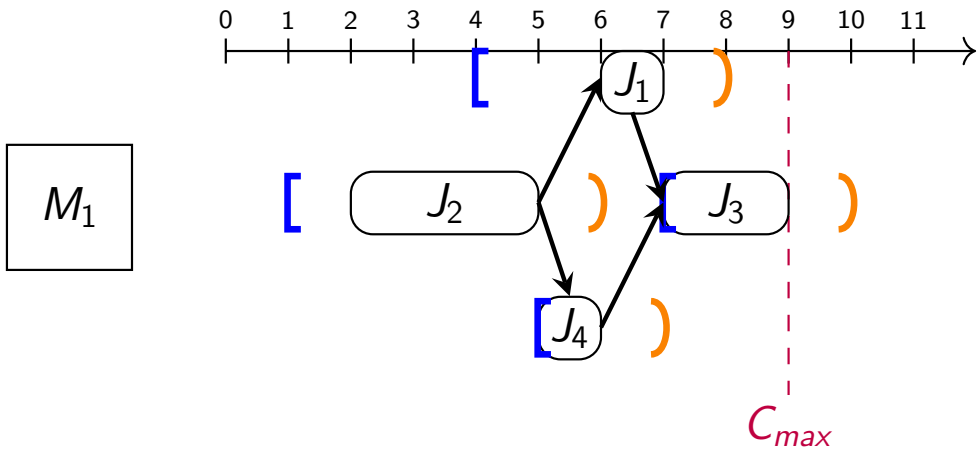
machines | job properties | optimization criterion

Machine scheduling



$1|prec, r_j, d_j|C_{max}$

Machine scheduling



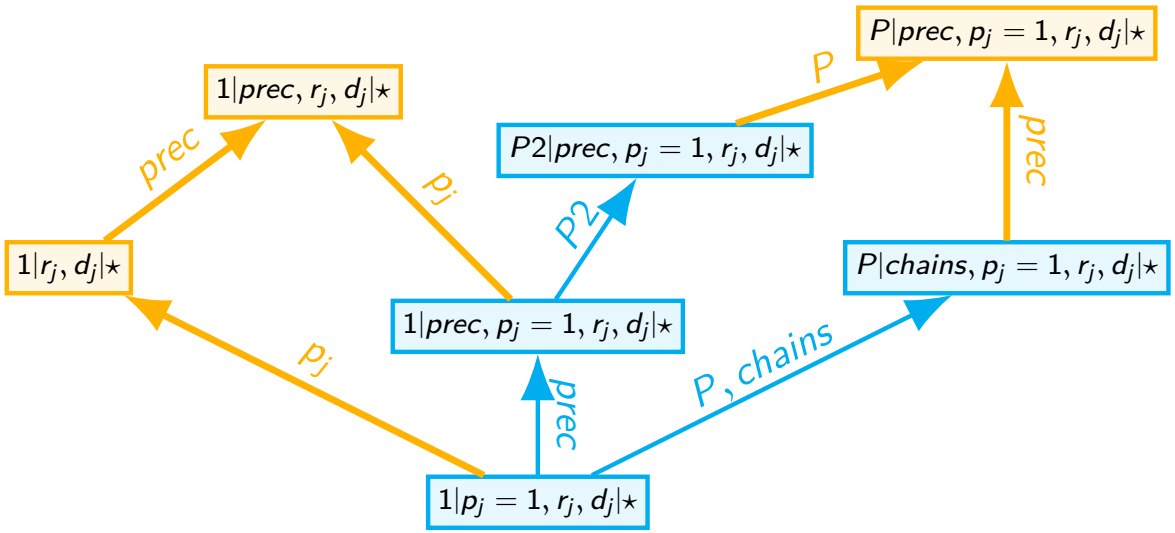
$1|prec, r_j, d_j|C_{max}$

$1|r_j, d_j|\star$ is strongly NP-hard. [Brucker, Kan, Lenstra 77]

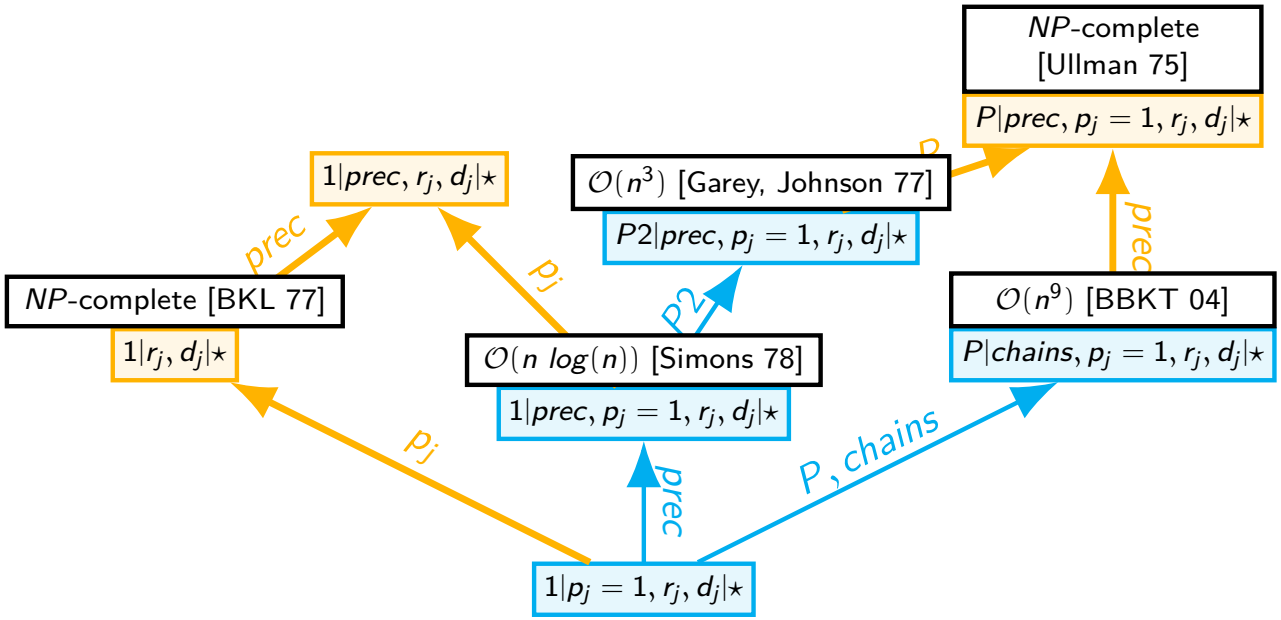
Classical complexity theory

$1|p_j = 1, r_j, d_j|\star$

Classical complexity theory



Classical complexity theory



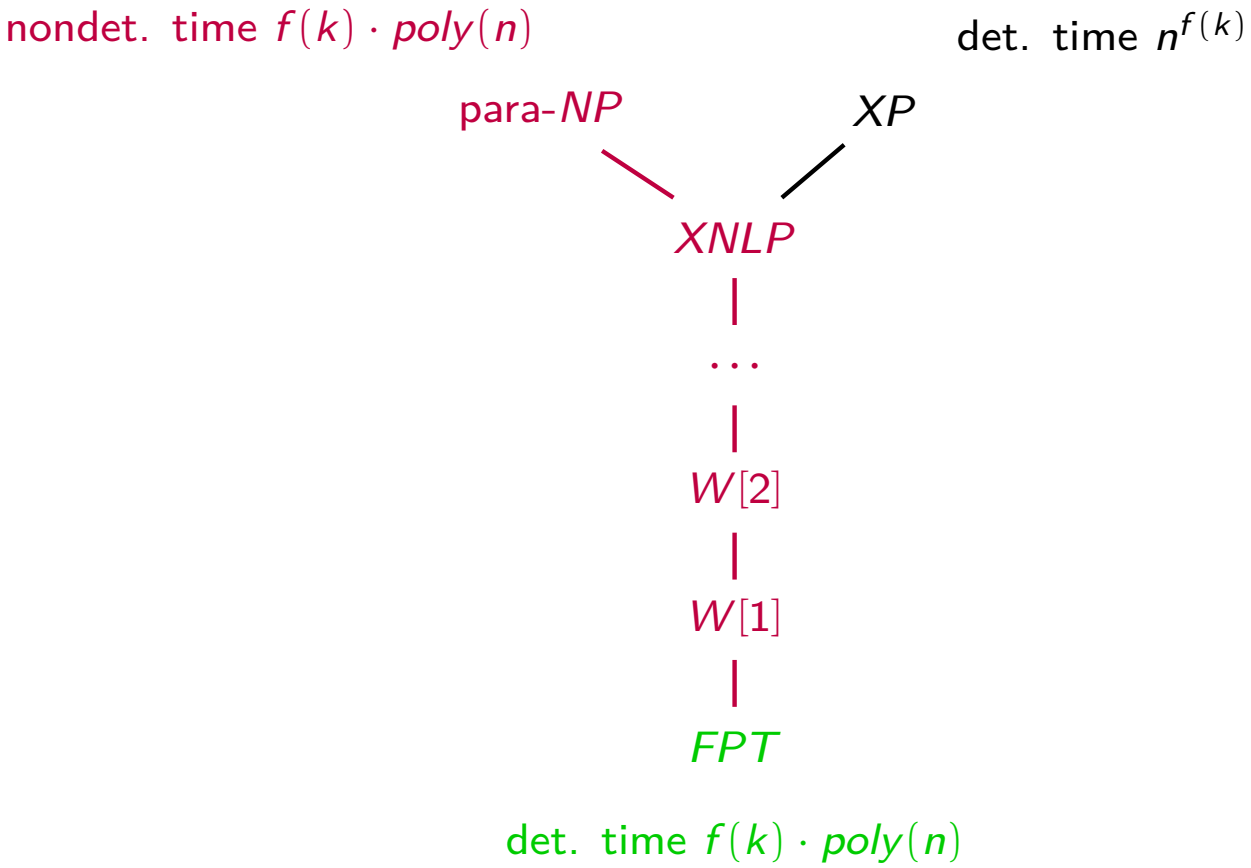
Going beyond NP-hardness

- Problem $\mathcal{P}$
- P: deterministic time $poly(n)$
- NP: nondeterministic time $poly(n)$

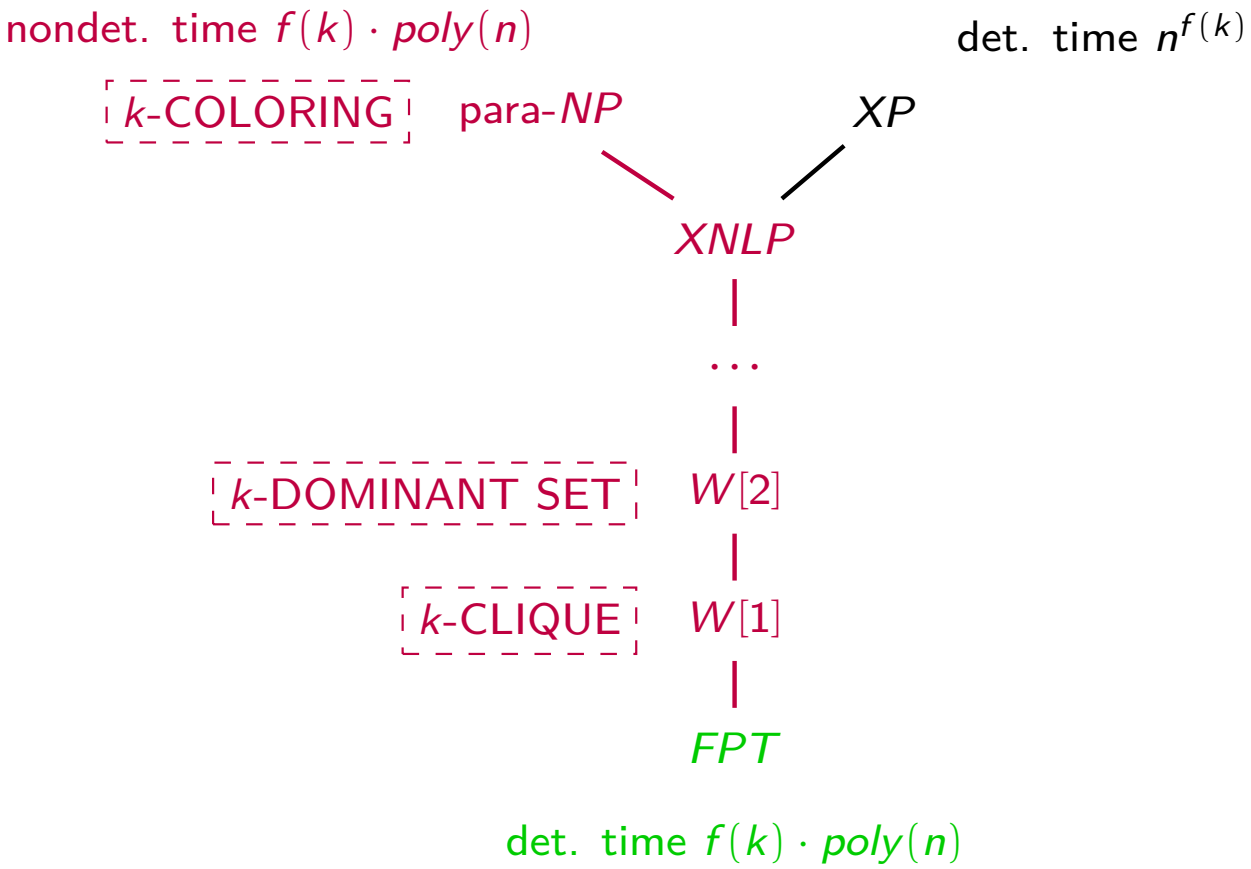
Going beyond NP-hardness

- Problem $\mathcal{P}$
  - Problem $(\mathcal{P}, k)$
- P: deterministic time $poly(n)$
  - FPT: deterministic time $f(k) \cdot poly(n)$
- NP: nondeterministic time $poly(n)$
  - para-NP: nondeterministic time $f(k) \cdot poly(n)$

Parameterized complexity classes



Parameterized complexity classes



Parameterized complexity in machine scheduling

Parameterized complexity in machine scheduling

- Negative results via *FPT* reductions

Parameterized complexity in machine scheduling

- Negative results via *FPT* reductions
  - [Bodlaender, Fellows 95] $(P|prec, p_j = 1|C_{max}, m)$ is W[2]-hard.

Parameterized complexity in machine scheduling

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

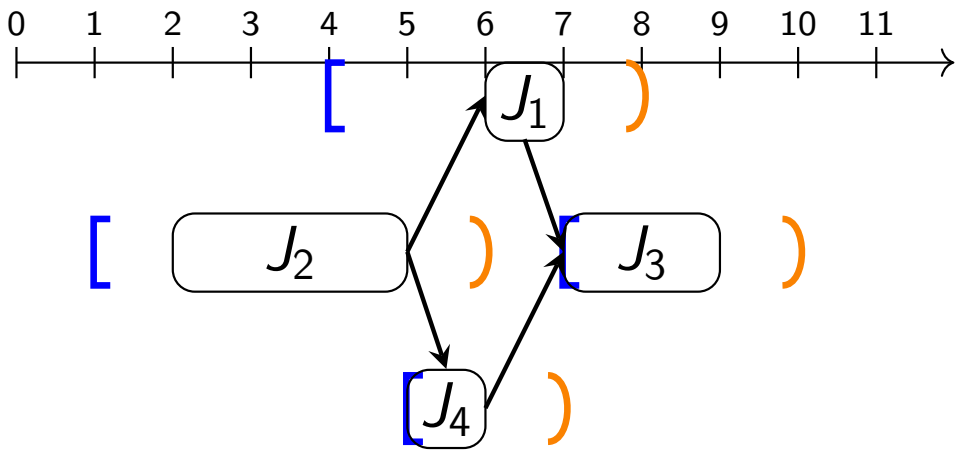
Parameterized complexity in machine scheduling

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

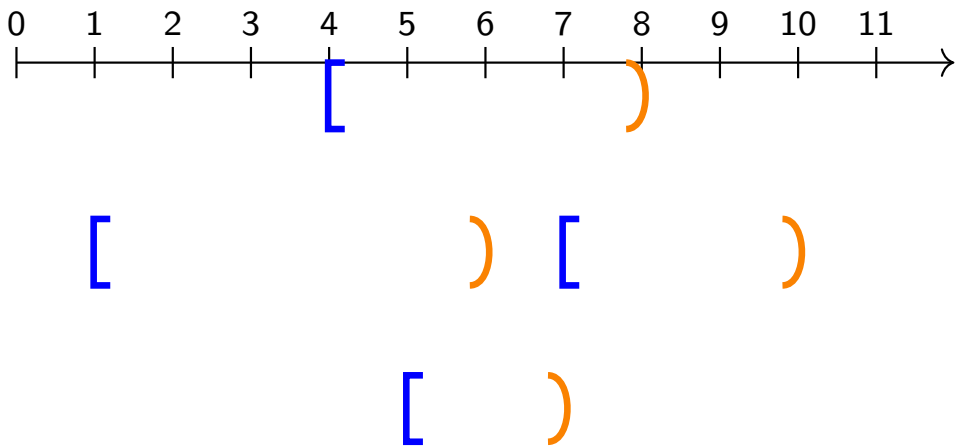
Parameterized complexity in machine scheduling

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

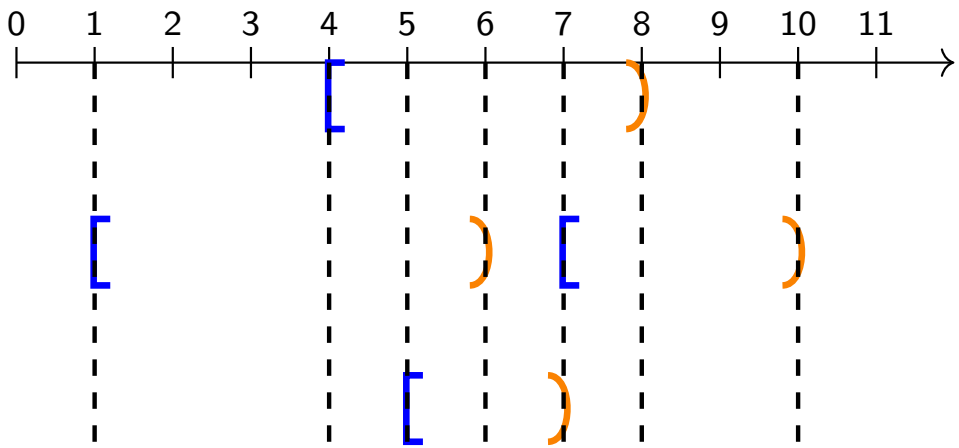
Pathwidth μ



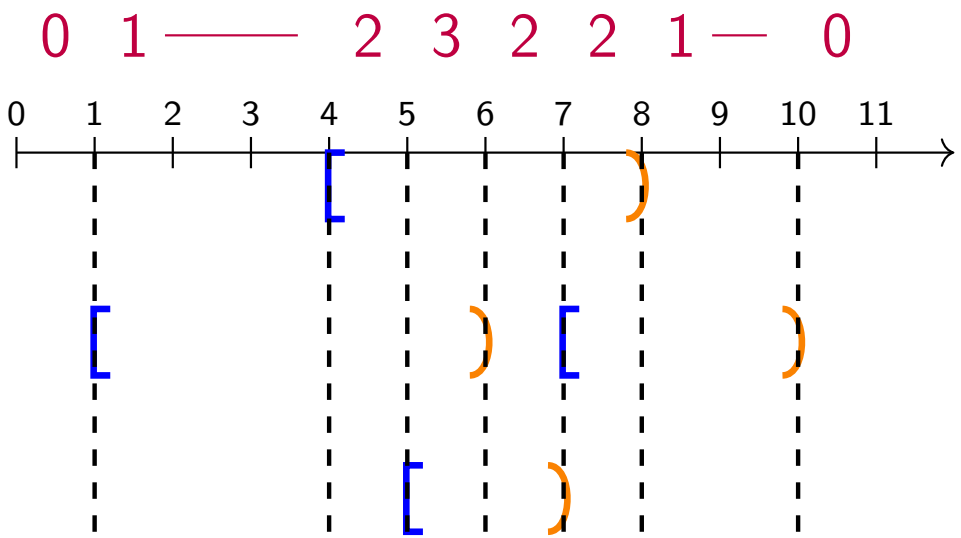
Pathwidth μ



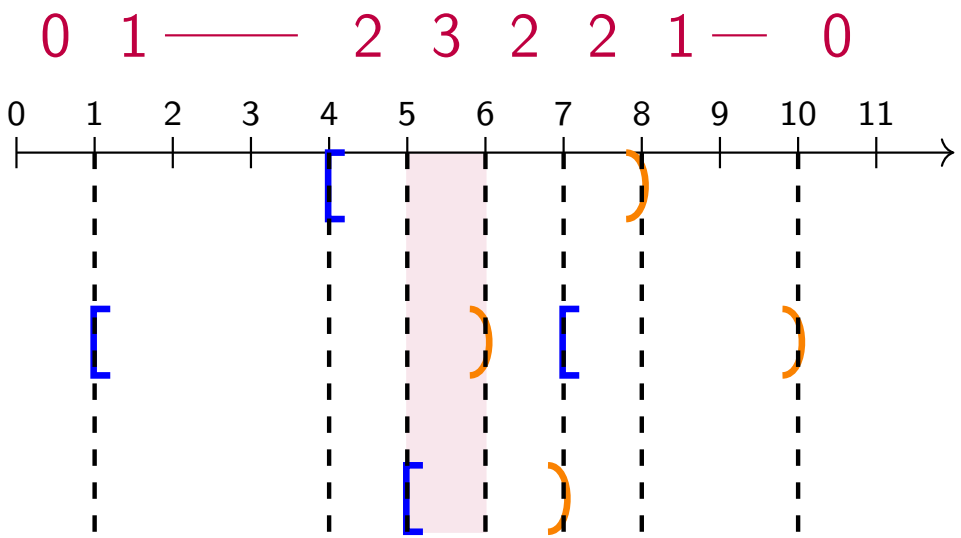
Pathwidth μ



Pathwidth μ

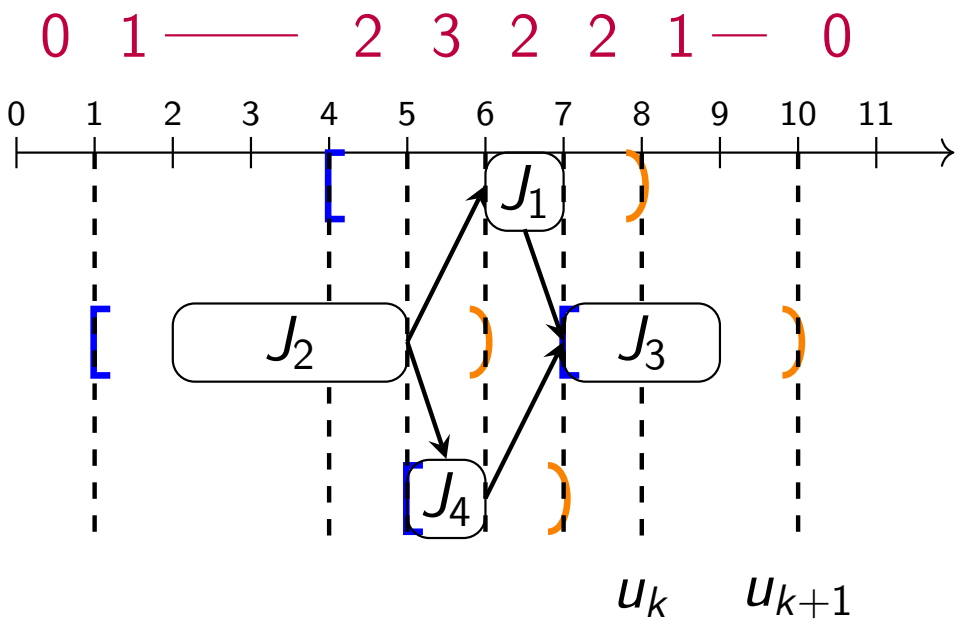


Pathwidth μ

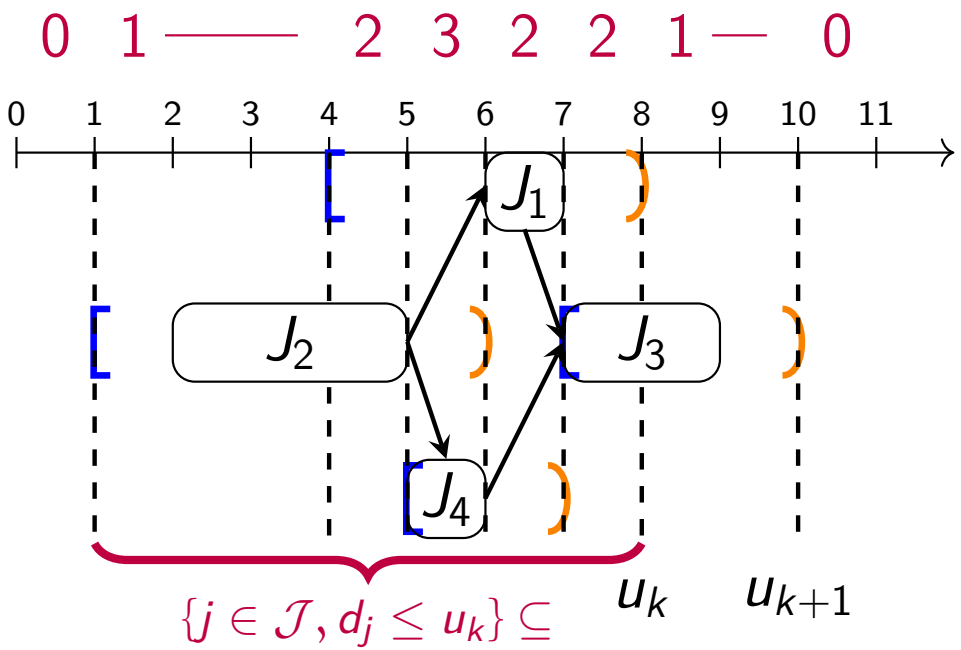


$$\mu = 3 - 1 = 2$$

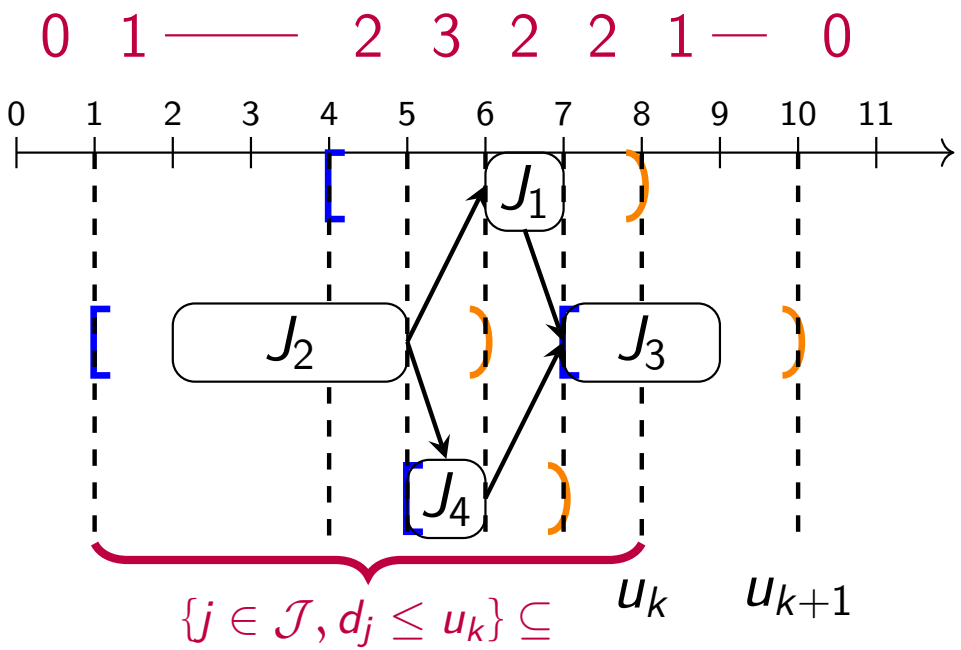
Pathwidth μ



Pathwidth μ



Pathwidth μ



$\leq 2^{\mu+1}$ states per time segment $[u_k, u_{k+1})$.

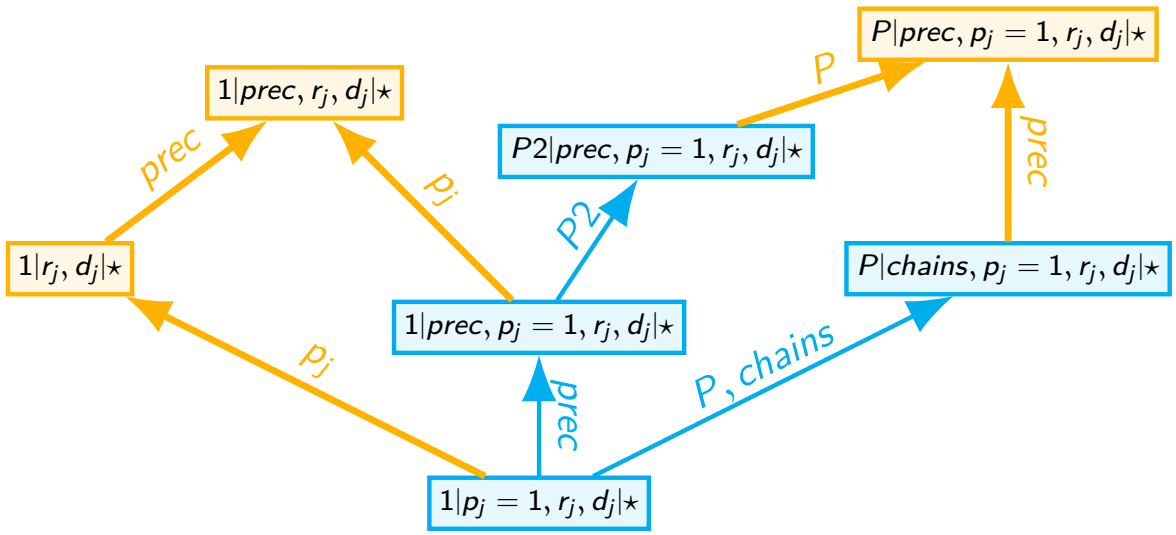
Results with parameter μ

- [Munier 21]
  - $P|prec, p_j = 1, r_j, d_j|C_{max}$
  - 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 $\mathcal{O}(\mu^2\ 2^{\mu} \cdot \frac{n^3}{\epsilon^2})$.

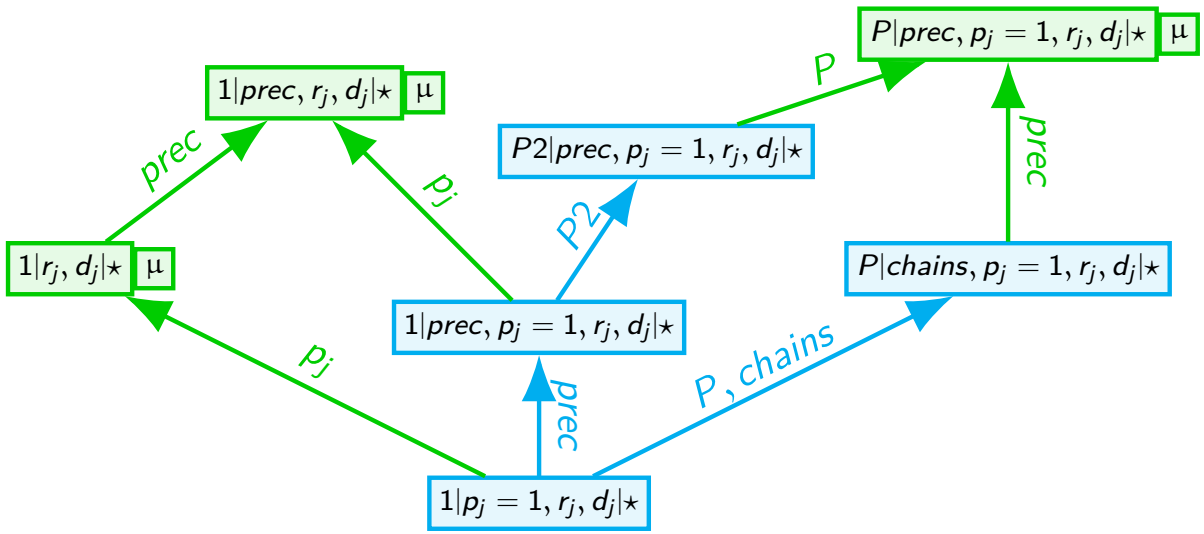
Results with parameter μ

- [Munier 21]
  - $P|prec, p_j = 1, r_j, d_j|C_{max}$
  - 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 $\mathcal{O}(\mu^2\ 2^{\mu} \cdot \frac{n^3}{\epsilon^2})$.
- [M., Hanen, Munier 22]
  - $1|prec, r_j, d_j|C_{max}$
  - Exact resolution in time $\mathcal{O}(\mu^2\ 4^{\mu} \cdot n + n^2)$.

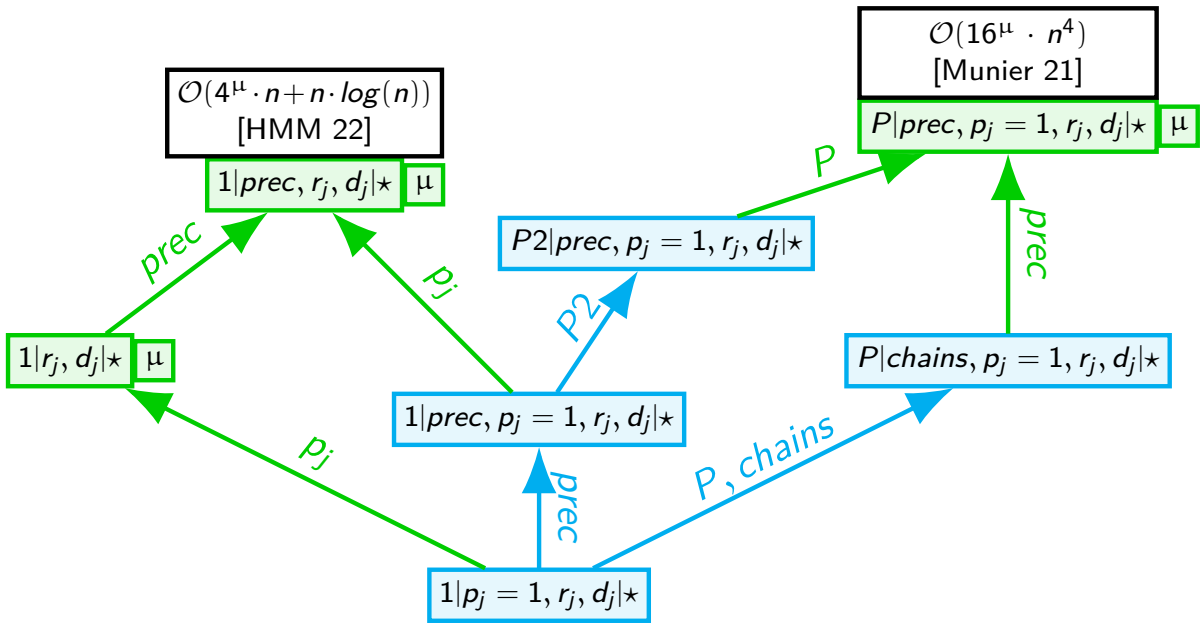
Parameter μ



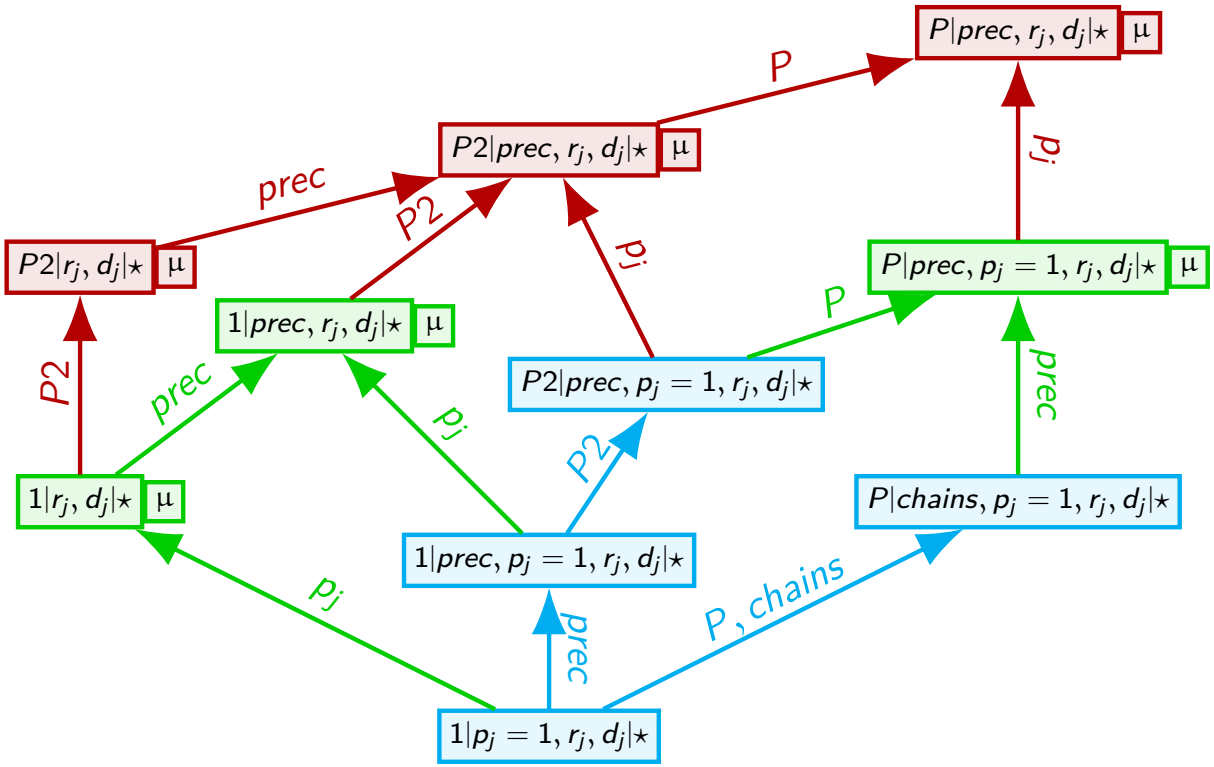
Parameter μ



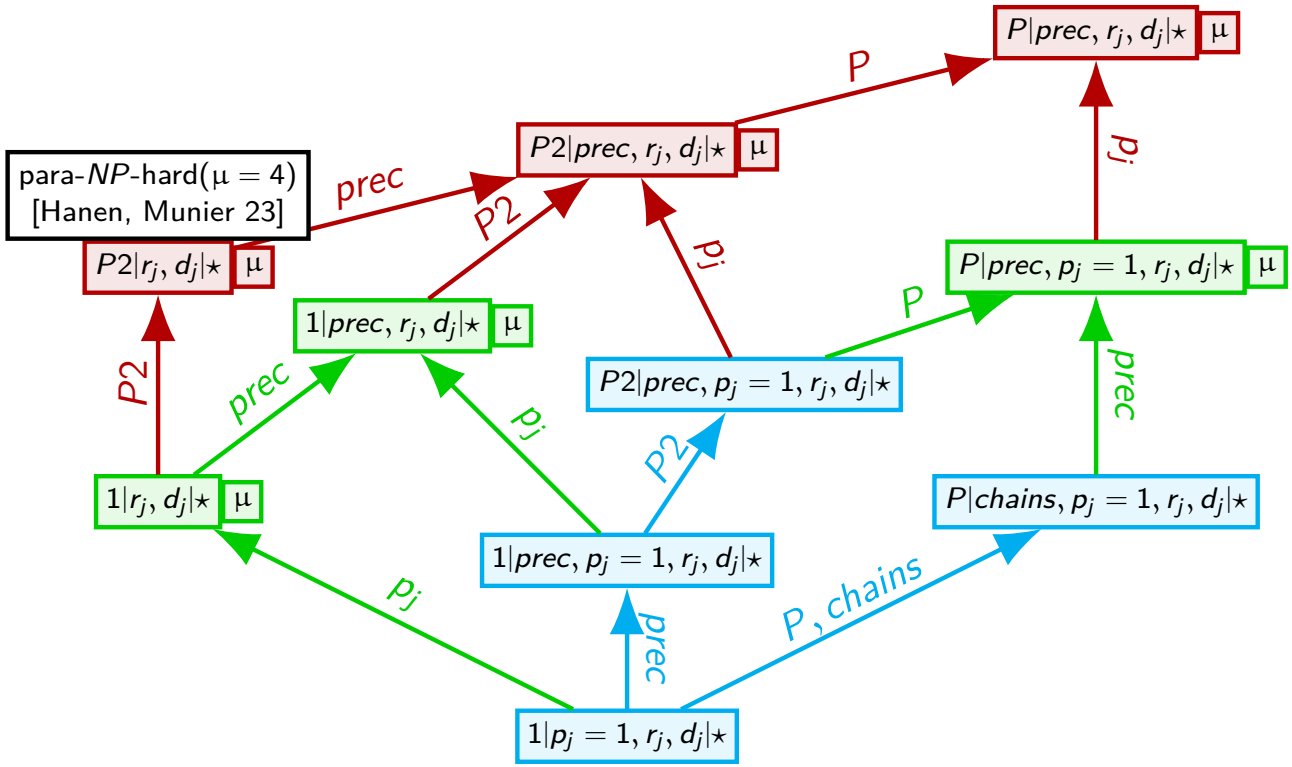
Parameter μ



Parameter μ

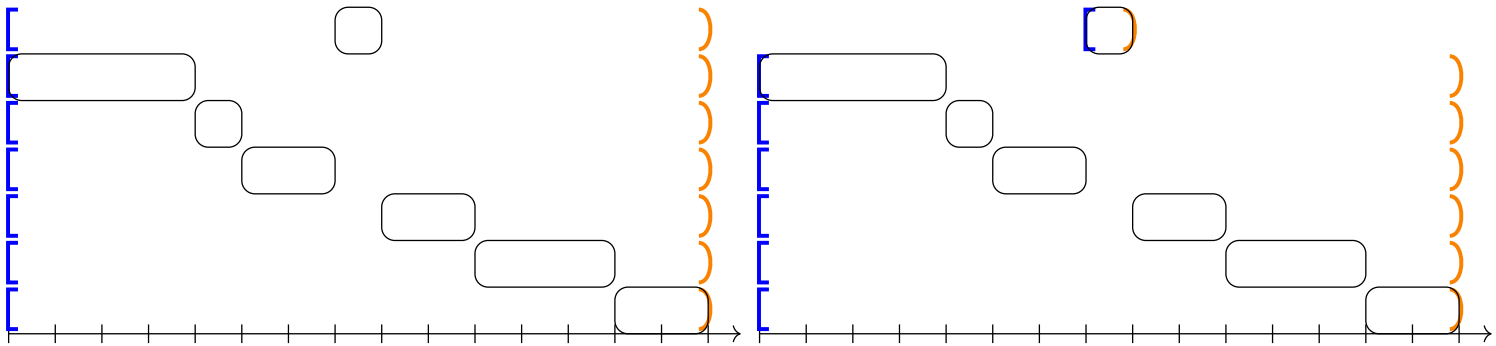


Parameter μ



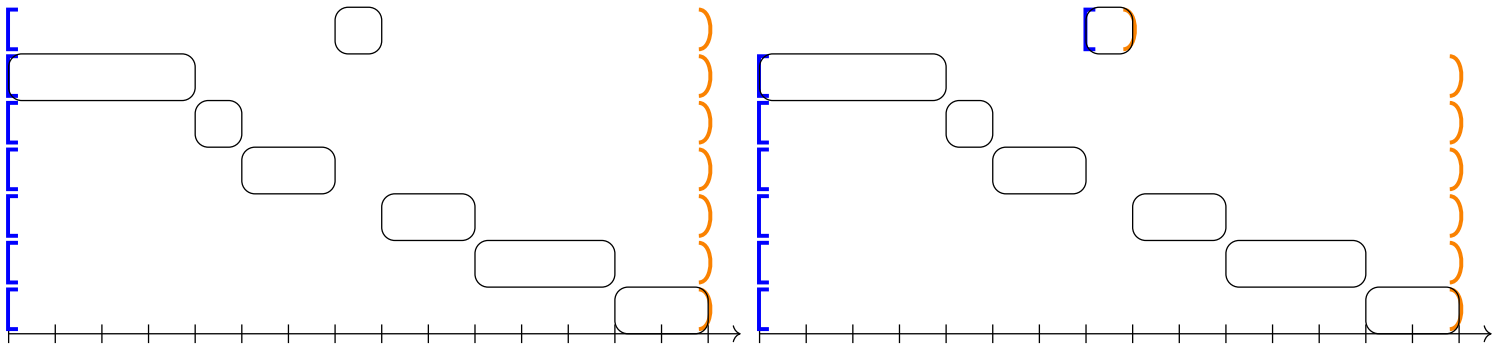
Pathwidth limitation

- Two instances of $1|r_j, d_j|\star$.



Pathwidth limitation

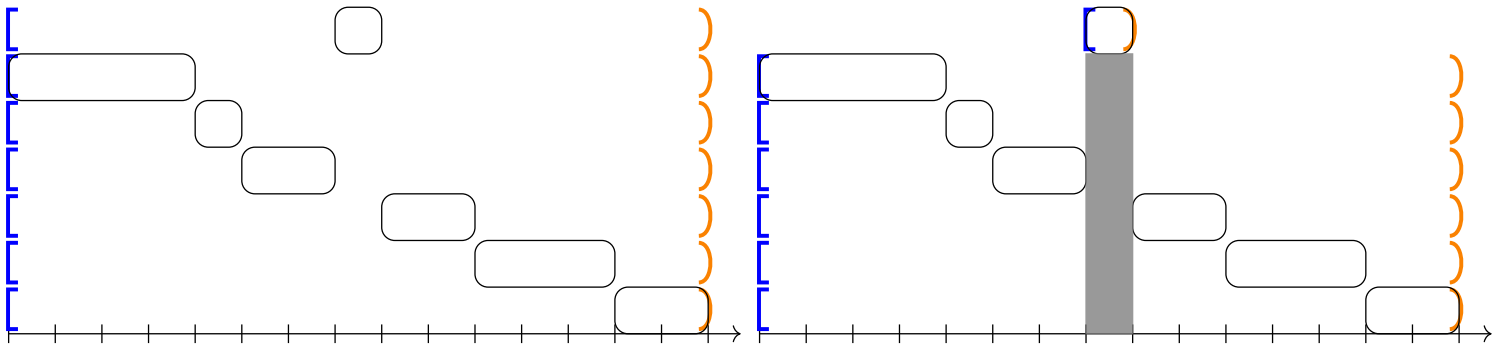
- Two instances of $1|r_j, d_j|\star$.



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

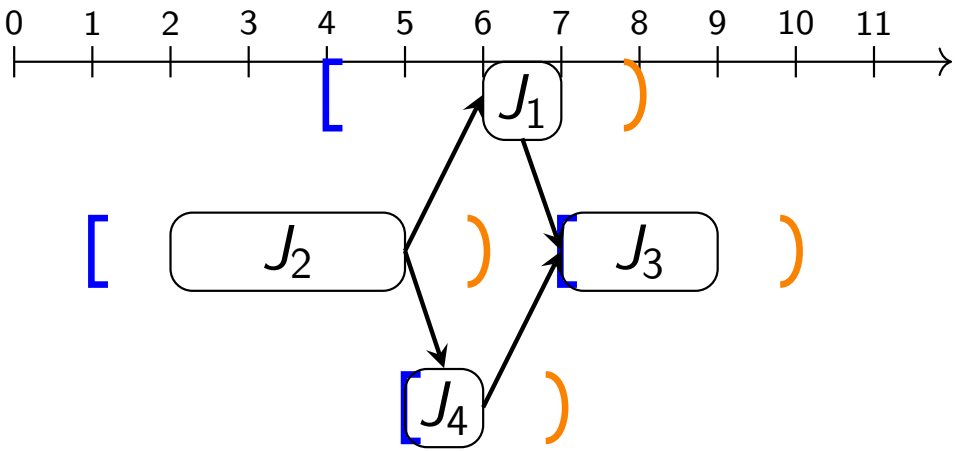
Pathwidth limitation

- Two instances of $1|r_j, d_j|\star$.

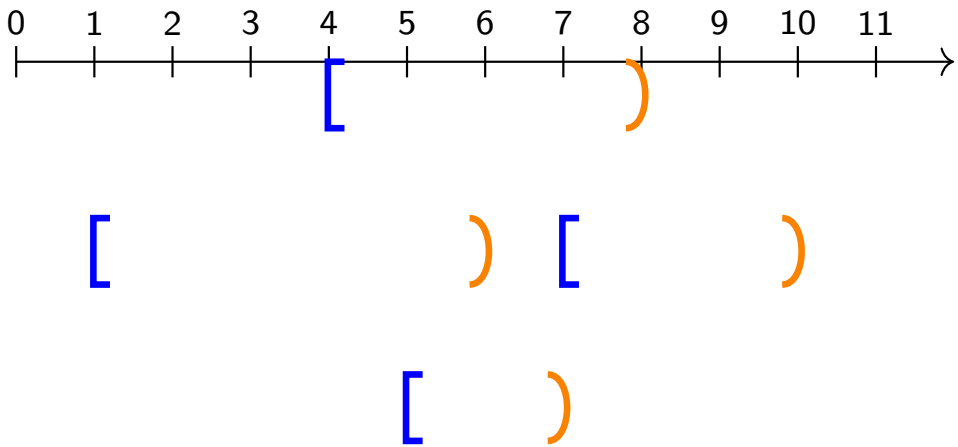


$\mu = n - 1$ $\mu = n - 1$

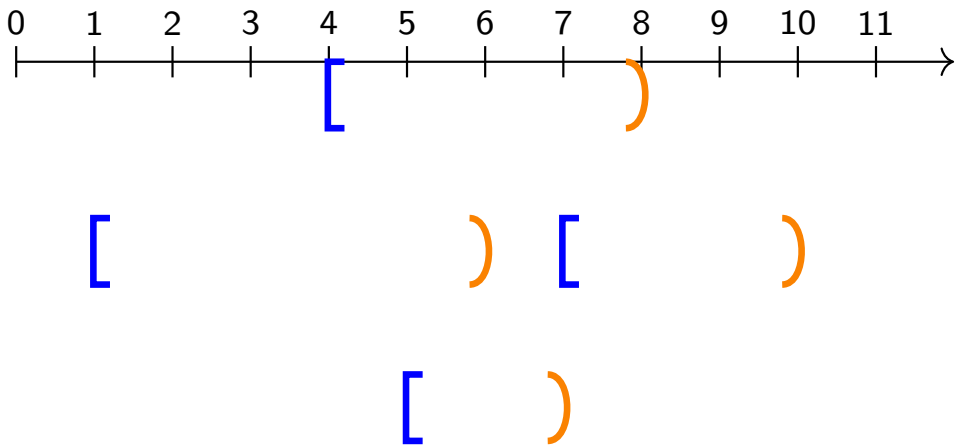
Proper level q



Proper level q



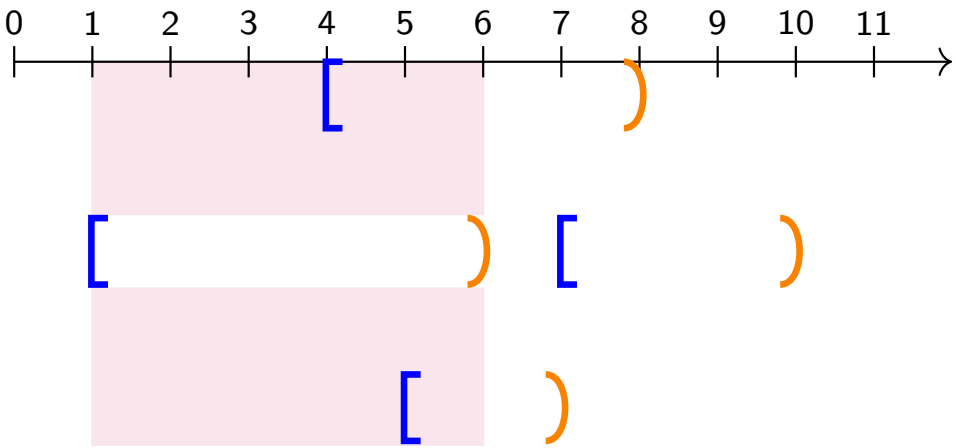
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

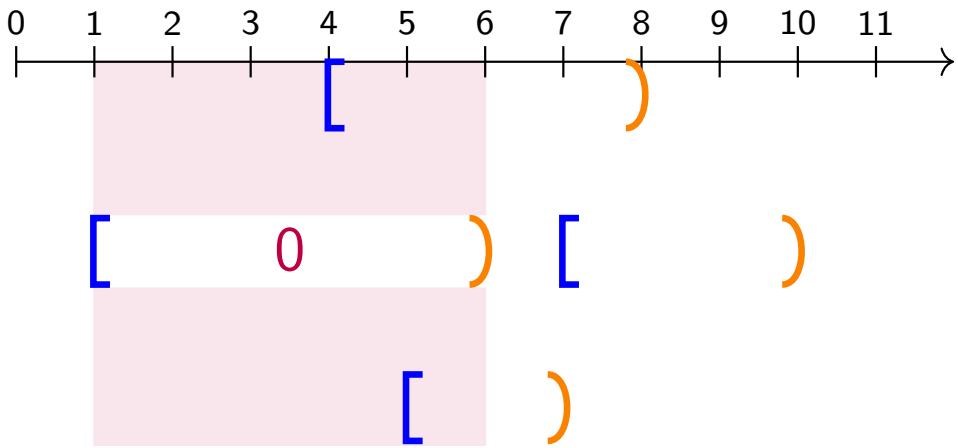
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

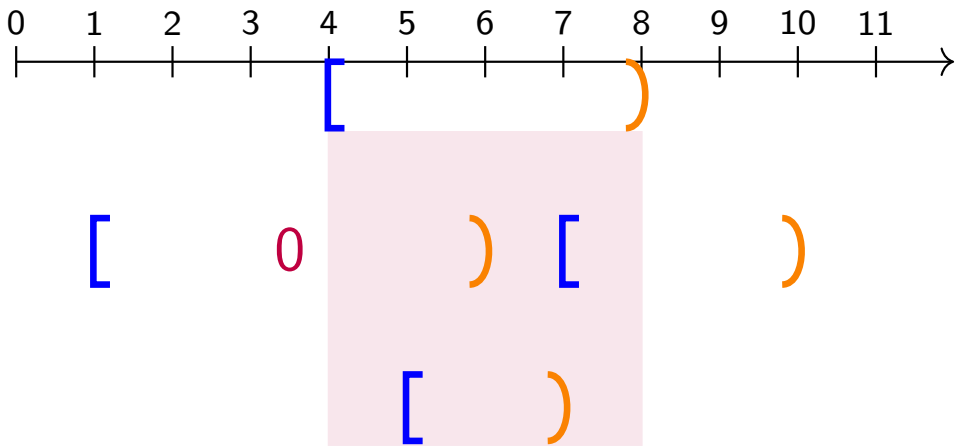
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

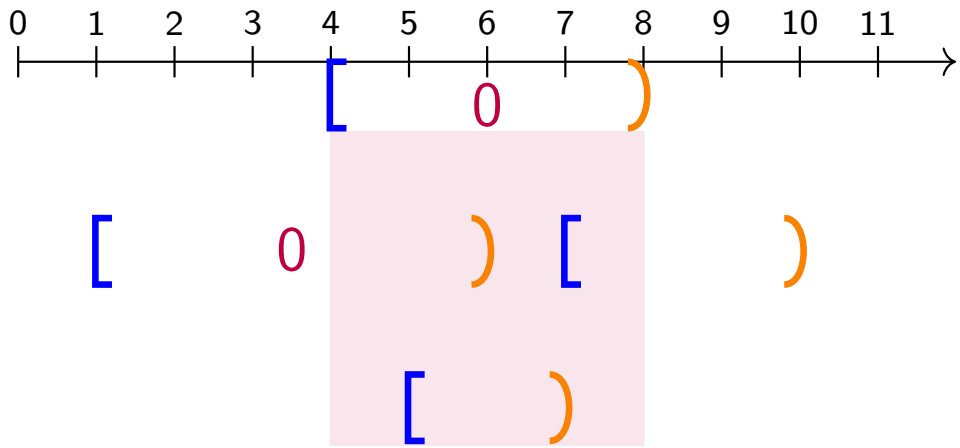
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

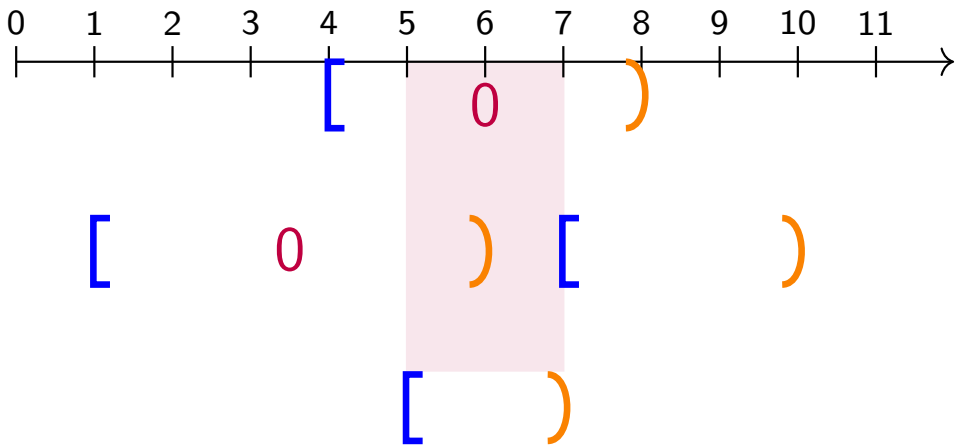
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

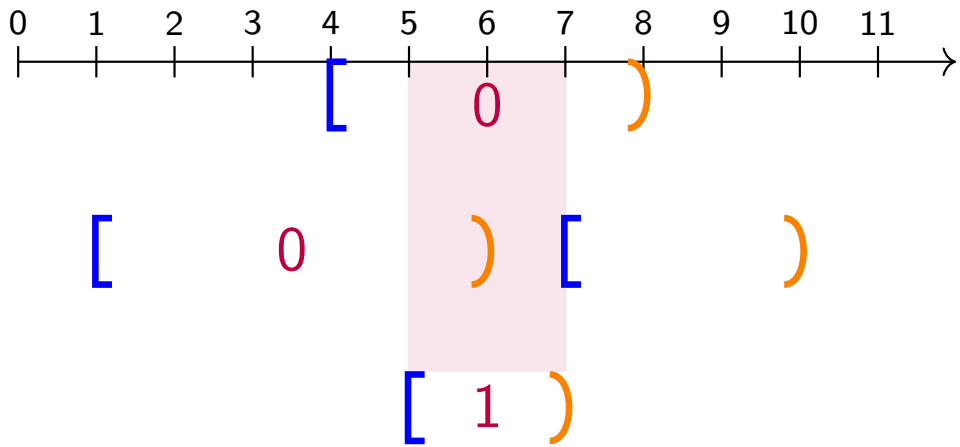
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

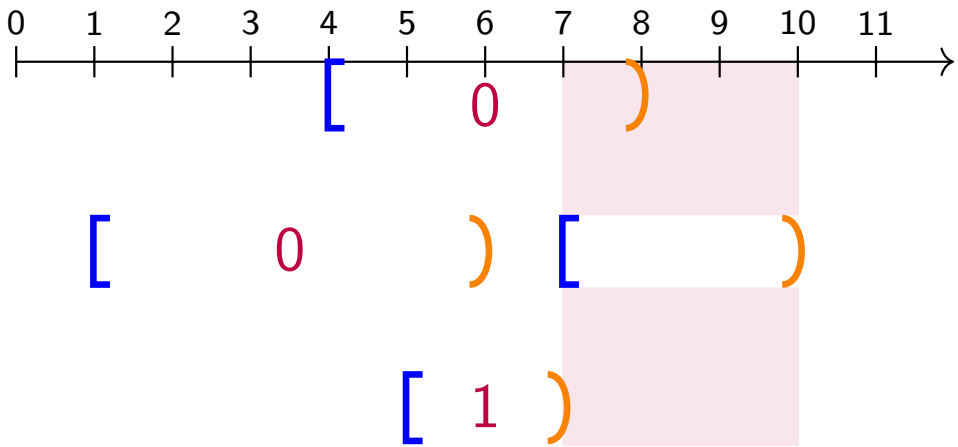
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

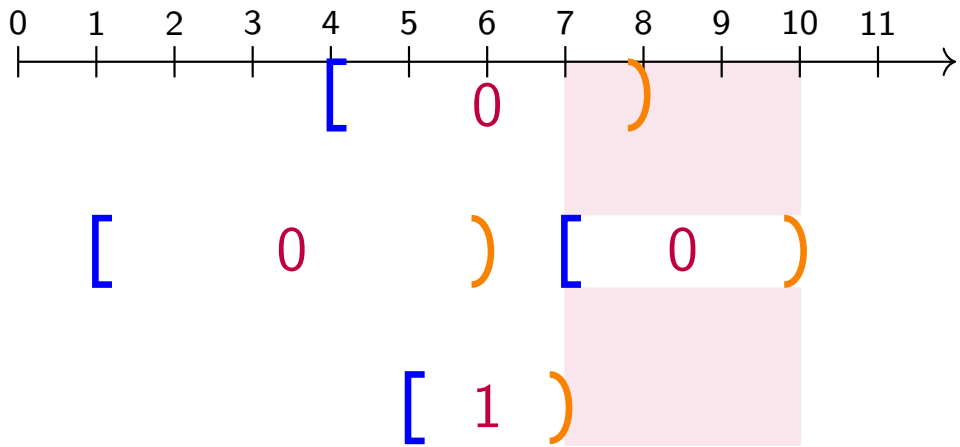
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

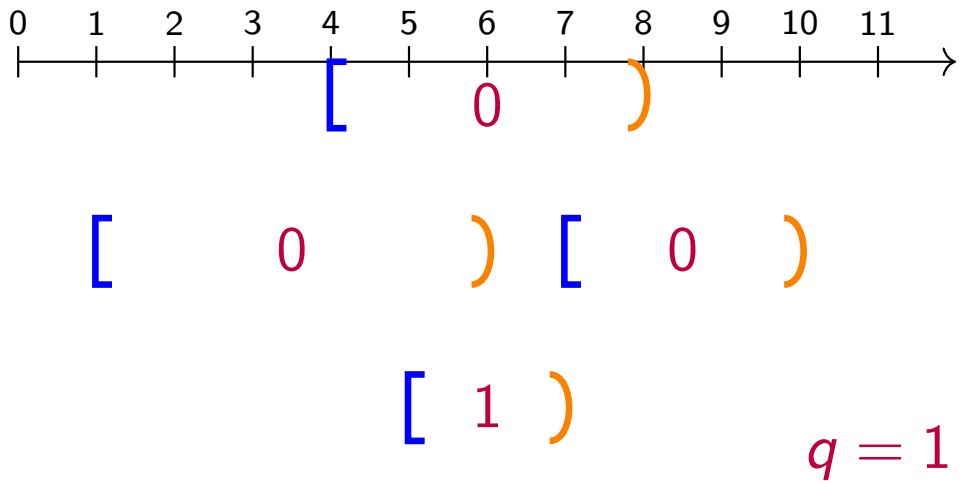
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

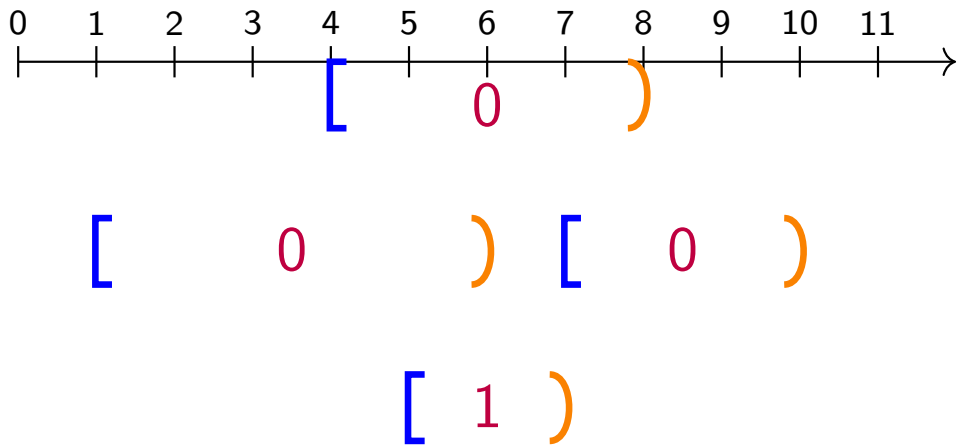
Proper level q



Definition: Proper sets and proper level

- $\forall i \in \mathcal{J}, \Pi_i = \{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $q = max_{i \in \mathcal{J}} |\Pi_i|$

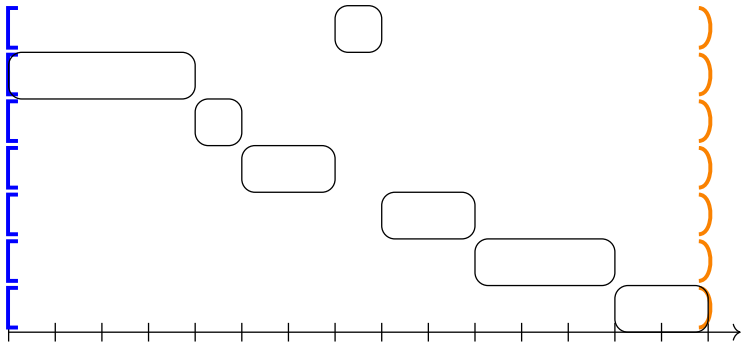
Proper level q



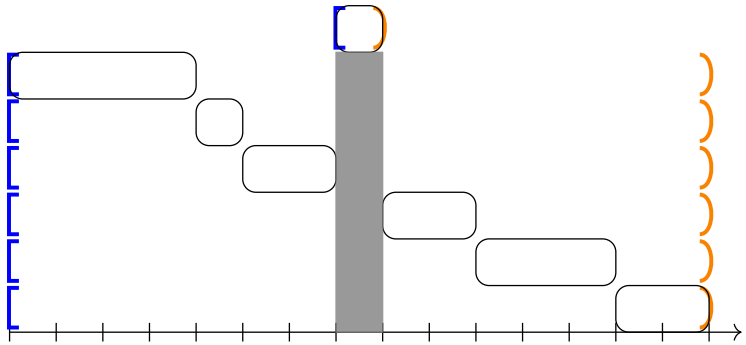
Definition: Proper sets and proper level

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

Pathwidth vs proper level

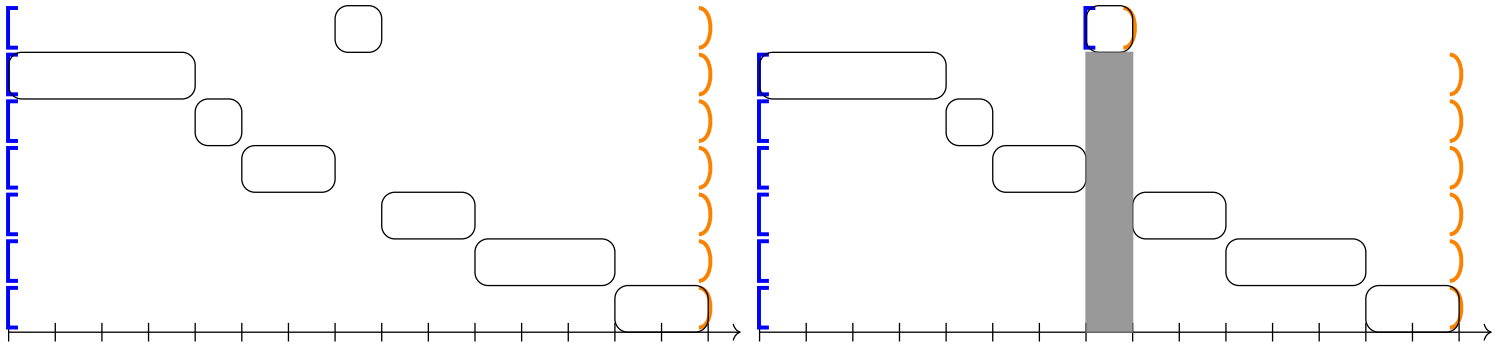


$\mu = n - 1$



$\mu = n - 1$

Pathwidth vs proper level



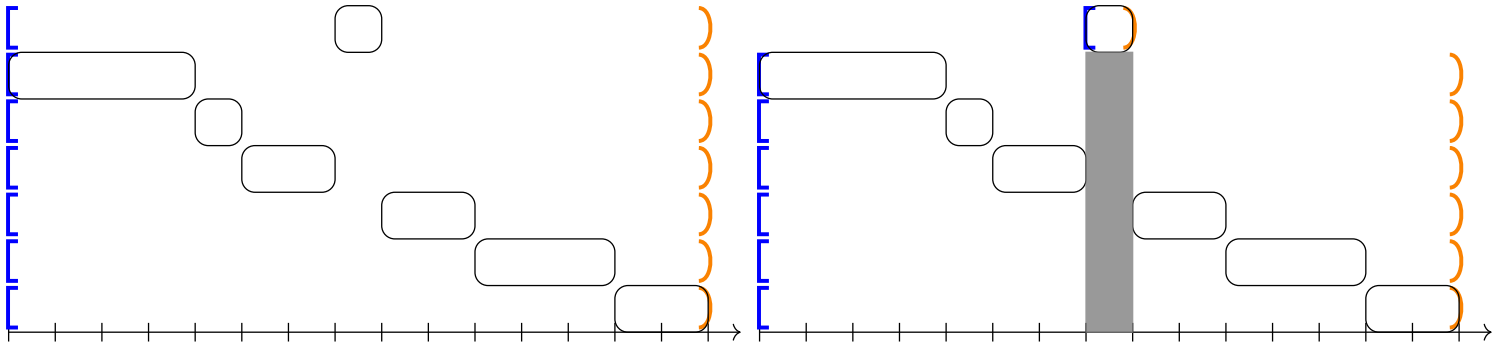
$\mu = n - 1$

$q = 0$

$\mu = n - 1$

$q = n - 1$

Pathwidth vs proper level



$\mu = n - 1$ $\quad$ $\mu = n - 1$

$q = 0$ $\quad$ $q = n - 1$

Result

$$q \leq \mu$$

Extending the Earliest Deadline rule

- σ optimal schedule on $1|r_j, d_j|C_{max}$ ($=$ job ordering).
- $\prec$ $=$ lexicographic order (non decreasing d_j, non decreasing r_j) on $\mathcal{J}$.

Extending the Earliest Deadline rule

- σ optimal schedule on $1|r_j, d_j|C_{max}$ ($=$ job ordering).
- $\prec$ $=$ lexicographic order (non decreasing d_j, non decreasing r_j) on $\mathcal{J}$.

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

Schedule σ follows the (ED) rule if:

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

Extending the Earliest Deadline rule

- σ optimal schedule on $1|r_j, d_j|C_{max}$ (= job ordering).
- $\prec$ = lexicographic order (non decreasing d_j, non decreasing r_j) on $\mathcal{J}$.

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

Schedule σ follows the (ED) rule if:

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

Definition: weak Earliest Deadline rule (wED)

Schedule σ follows the (wED) rule if:

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

Extending the Earliest Deadline rule

- σ optimal schedule on $1|r_j, d_j|C_{max}$ (= job ordering).
- $\prec$ = lexicographic order (non decreasing d_j, non decreasing r_j) on $\mathcal{J}$.

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

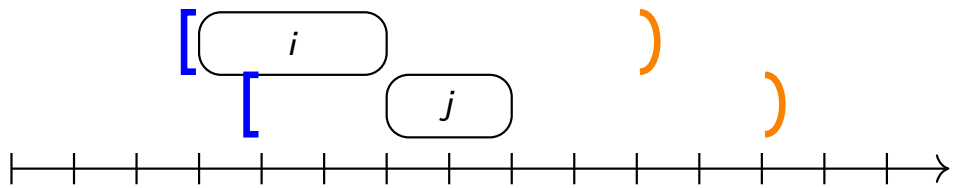
Schedule σ follows the (ED) rule if:

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

Definition: weak Earliest Deadline rule (wED)

Schedule σ follows the (wED) rule if:

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



Extending the Earliest Deadline rule

- σ optimal schedule on $1|r_j, d_j|C_{max}$ (= job ordering).
- $\prec$ = lexicographic order (non decreasing d_j, non decreasing r_j) on $\mathcal{J}$.

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

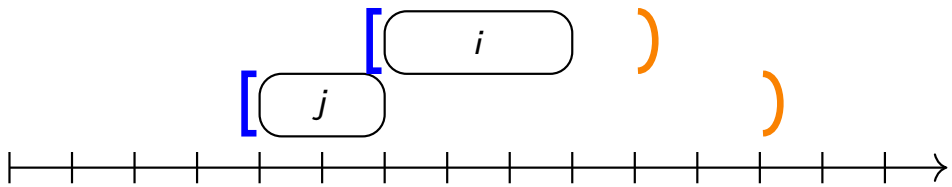
Schedule σ follows the (ED) rule if:

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

Definition: weak Earliest Deadline rule (wED)

Schedule σ follows the (wED) rule if:

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



Extending the Earliest Deadline rule

- σ optimal schedule on $1|r_j, d_j|C_{max}$ (= job ordering).
- $\prec$ = lexicographic order (non decreasing d_j, non decreasing r_j) on $\mathcal{J}$.

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

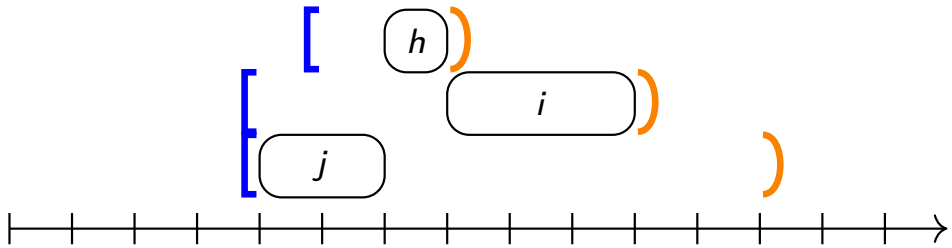
Schedule σ follows the (ED) rule if:

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

Definition: weak Earliest Deadline rule (wED)

Schedule σ follows the (wED) rule if:

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



Extending the Earliest Deadline rule (cont.)

Definition: weak Earliest Deadline rule (wED)

Schedule σ follows the (wED) rule if:

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

Extending the Earliest Deadline rule (cont.)

Definition: weak Earliest Deadline rule (wED)

Schedule σ follows the (wED) rule if:

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

Proposition

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

Extending the Earliest Deadline rule (cont.)

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.

Extending the Earliest Deadline rule (cont.)

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.

- $s_1 \prec \ldots \prec s_N$ summits of the instance ($N \leq n$).

Extending the Earliest Deadline rule (cont.)

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.

- $s_1 \prec \ldots \prec s_N$ summits of the instance ($N \leq n$).

Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

Extending the Earliest Deadline rule (cont.)

Definition: Summit

A summit is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.

- $s_1 \prec \ldots \prec s_N$ summits of the instance ($N \leq n$).

Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(i) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(ii) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

- Rename jobs in $[1, n]$ following their lexicographic order $\prec$.

- Schedule σ up to summit s_k: set $[1, s_k] \cup L_k$ with $L_k \subseteq \Pi_{s_k}$.
- σ of the form $T_1 s_1 \ldots s_k T_{k+1} s_{k+1} \ldots s_N T_{N+1}$.

Summit-based dominance

Lemma 2: Summit-based decomposition

Any feasible schedule σ which follows the (wED) rule is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$ where every job in sequence T_{k+1} is in set $\Pi_{s_k} \cup \Pi_{s_{k+1}}$.

Summit-based dominance

Lemma 2: Summit-based decomposition

Any feasible schedule σ which follows the (wED) rule is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$ where every job in sequence T_{k+1} is in set $\Pi_{s_k} \cup \Pi_{s_{k+1}}$.

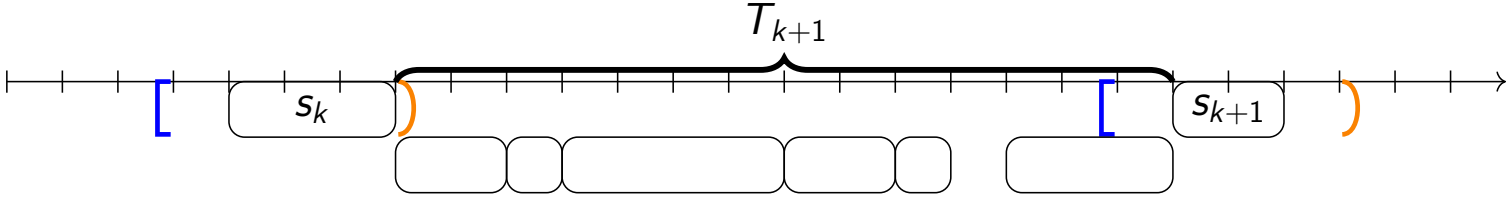
- $2^{2q} \cdot (2q)!$ choices for sequence T_{k+1}.

Summit-based dominance

Lemma 2: Summit-based decomposition

Any feasible schedule σ which follows the (wED) rule is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$ where every job in sequence T_{k+1} is in set $\Pi_{s_k} \cup \Pi_{s_{k+1}}$.

- $2^{2q} \cdot (2q)!$ choices for sequence T_{k+1}.

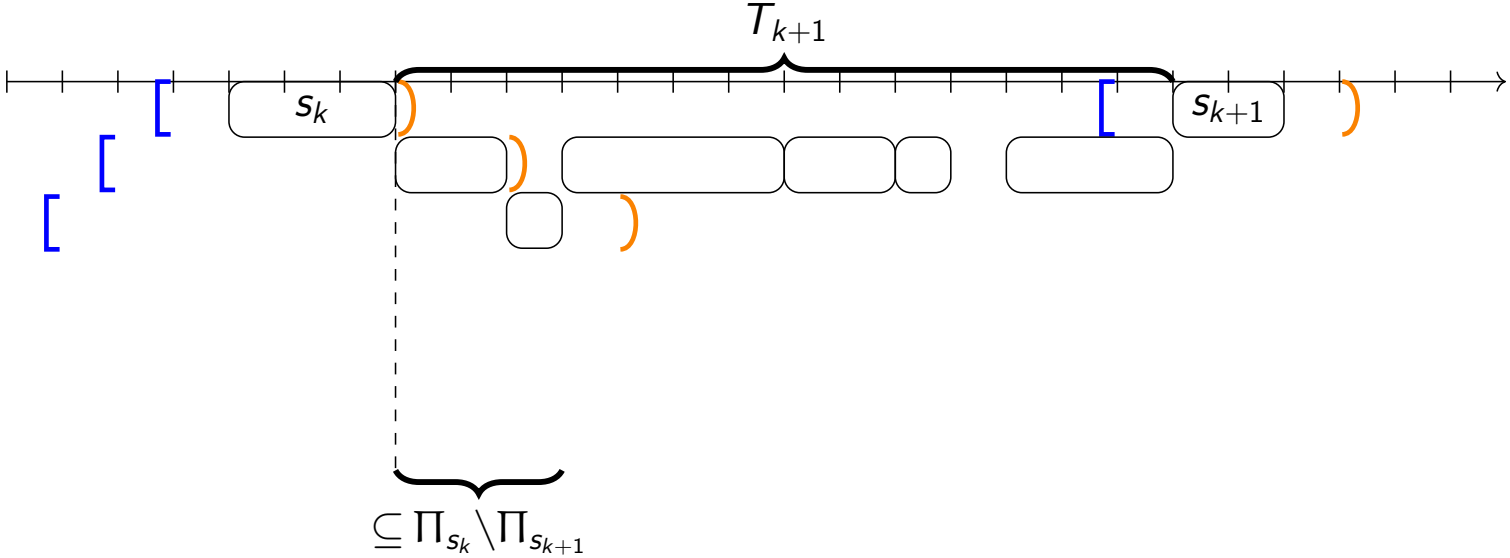


Summit-based dominance

Lemma 2: Summit-based decomposition

Any feasible schedule σ which follows the (wED) rule is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$ where every job in sequence T_{k+1} is in set $\Pi_{s_k} \cup \Pi_{s_{k+1}}$.

- $2^{2q} \cdot (2q)!$ choices for sequence T_{k+1}.

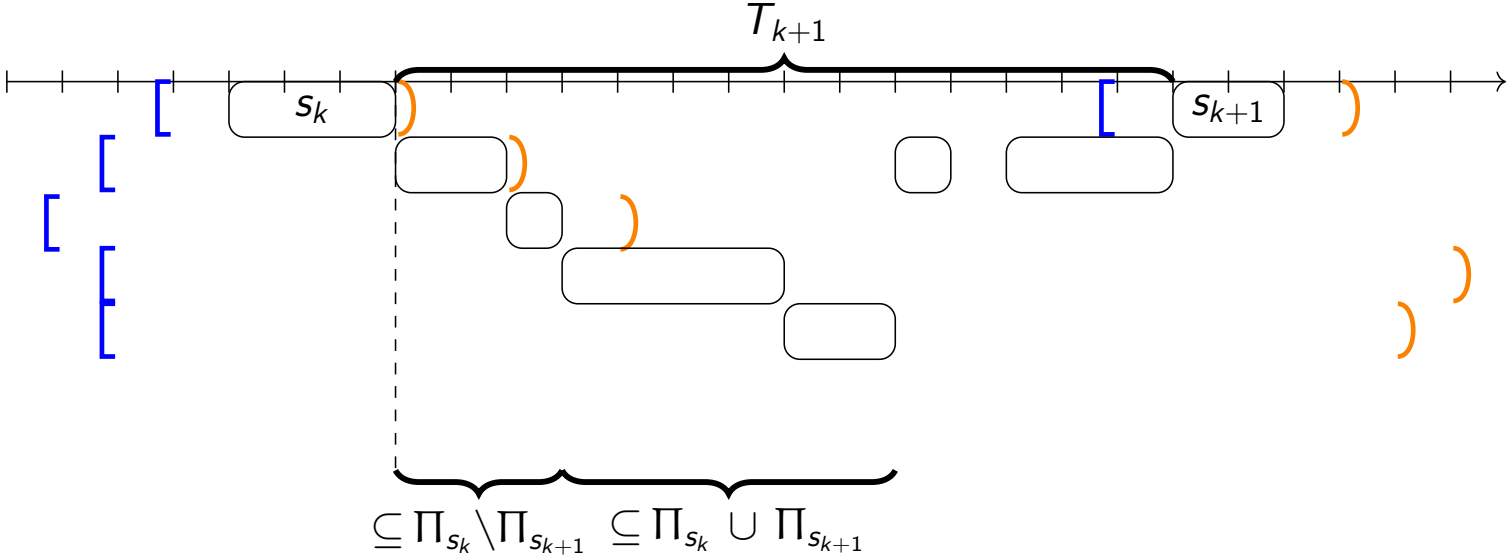


Summit-based dominance

Lemma 2: Summit-based decomposition

Any feasible schedule σ which follows the (wED) rule is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$ where every job in sequence T_{k+1} is in set $\Pi_{s_k} \cup \Pi_{s_{k+1}}$.

- $2^{2q} \cdot (2q)!$ choices for sequence T_{k+1}.

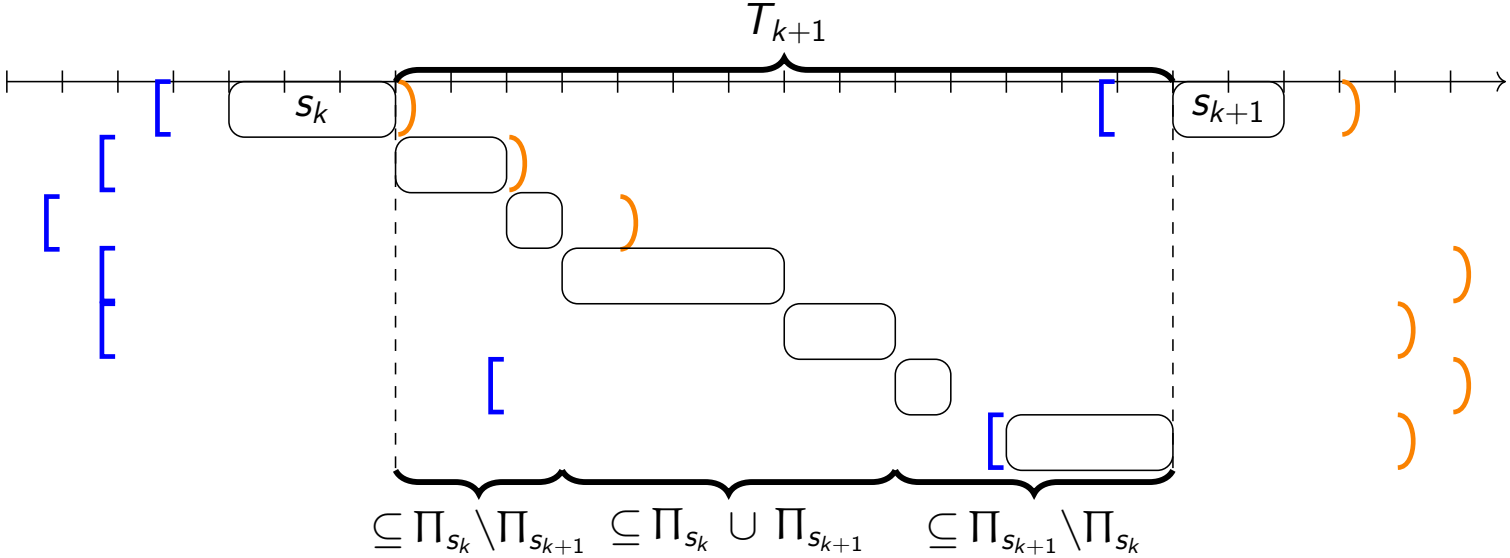


Summit-based dominance

Lemma 2: Summit-based decomposition

Any feasible schedule σ which follows the (wED) rule is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$ where every job in sequence T_{k+1} is in set $\Pi_{s_k} \cup \Pi_{s_{k+1}}$.

- $2^{2q} \cdot (2q)!$ choices for sequence T_{k+1}.

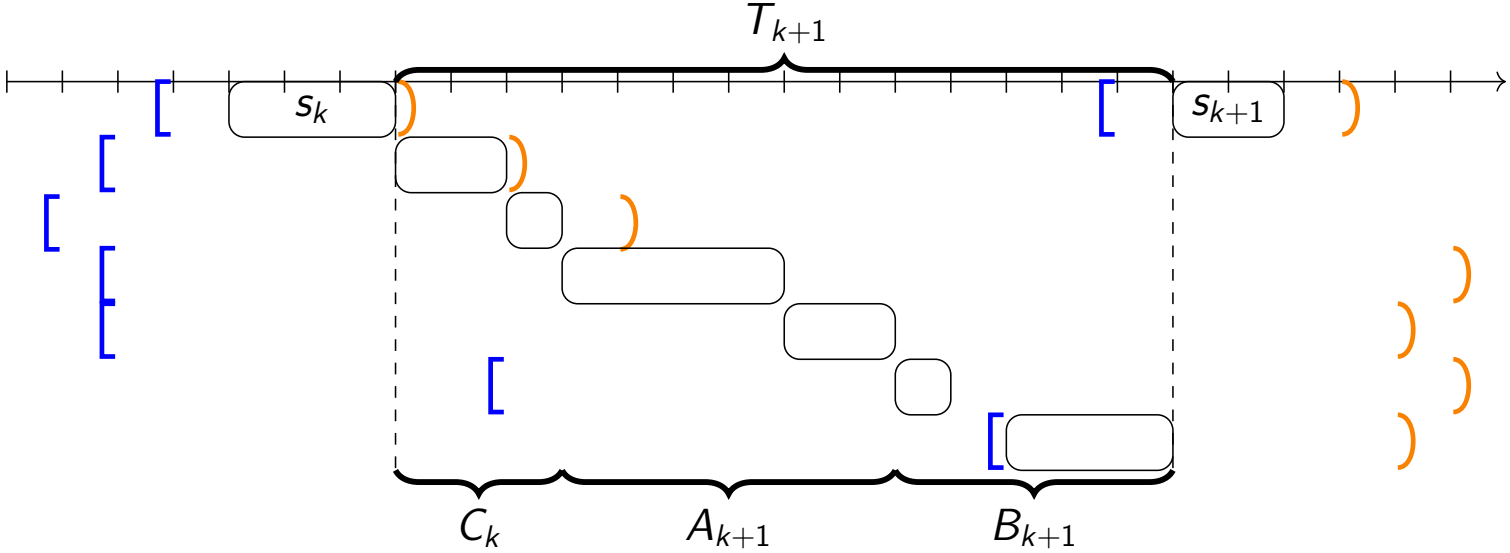


Summit-based dominance

Lemma 2: Summit-based decomposition

Any feasible schedule σ which follows the (wED) rule is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$ where every job in sequence T_{k+1} is in set $\Pi_{s_k} \cup \Pi_{s_{k+1}}$.

- $2^{2q} \cdot (2q)!$ choices for sequence T_{k+1}.



Summit-based dominance (cont.)

Definition: Summit-ordered schedule

A feasible schedule σ is called summit-ordered if it follows the (wED) rule and is of the form $B_1 s_1 \dots s_k C_k A_{k+1} B_{k+1} s_{k+1} \dots s_N C_N$ where:

Summit-based dominance (cont.)

Definition: Summit-ordered schedule

A feasible schedule σ is called summit-ordered if it follows the (wED) rule and is of the form $B_1 s_1 \dots s_k C_k A_{k+1} B_{k+1} s_{k+1} \dots s_N C_N$ where:

- all jobs in sequence C_k are in $\Pi_{s_k} \setminus \Pi_{s_{k+1}}$ and scheduled by nondecreasing d_j,

Summit-based dominance (cont.)

Definition: Summit-ordered schedule

A feasible schedule σ is called summit-ordered if it follows the (wED) rule and is of the form $B_1 s_1 \dots s_k C_k A_{k+1} B_{k+1} s_{k+1} \dots s_N C_N$ where:

- all jobs in sequence C_k are in $\Pi_{s_k} \setminus \Pi_{s_{k+1}}$ and scheduled by nondecreasing d_j,
- all jobs in sequence A_{k+1} are in $\Pi_{s_k} \cap \Pi_{s_{k+1}}$ and scheduled in their lexicographic order,

Summit-based dominance (cont.)

Definition: Summit-ordered schedule

A feasible schedule σ is called summit-ordered if it follows the (wED) rule and is of the form $B_1 s_1 \dots s_k C_k A_{k+1} B_{k+1} s_{k+1} \dots s_N C_N$ where:

- all jobs in sequence C_k are in $\Pi_{s_k} \setminus \Pi_{s_{k+1}}$ and scheduled by nondecreasing d_j,
- all jobs in sequence A_{k+1} are in $\Pi_{s_k} \cap \Pi_{s_{k+1}}$ and scheduled in their lexicographic order,
- all jobs in sequence B_{k+1} are in $\Pi_{s_{k+1}} \setminus \Pi_{s_k}$ and scheduled by nondecreasing r_j.

Summit-based dominance (cont.)

Definition: Summit-ordered schedule

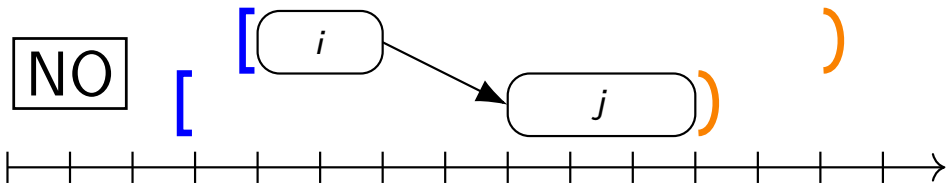
A feasible schedule σ is called summit-ordered if it follows the (wED) rule and is of the form $B_1 s_1 \dots s_k C_k A_{k+1} B_{k+1} s_{k+1} \dots s_N C_N$ where:

- all jobs in sequence C_k are in $\Pi_{s_k} \setminus \Pi_{s_{k+1}}$ and scheduled by nondecreasing d_j,
- all jobs in sequence A_{k+1} are in $\Pi_{s_k} \cap \Pi_{s_{k+1}}$ and scheduled in their lexicographic order,
- all jobs in sequence B_{k+1} are in $\Pi_{s_{k+1}} \setminus \Pi_{s_k}$ and scheduled by nondecreasing r_j.

Proposition [Erschler, Fontan, Merce 85]

On $1|r_j, d_j|C_{max}$ the summit-ordered schedules are dominant.

With precedence relations



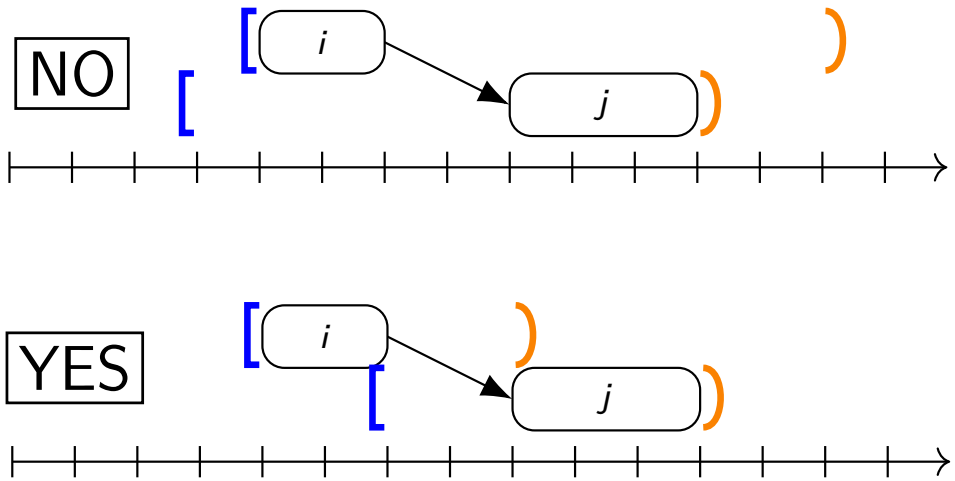
With precedence relations

- *prec* precedence graph of the instance: "$i \longrightarrow j$".

Definition: *prec*-consistency

An instance I of $1|prec, r_j, d_j|C_{max}$ is called *prec*-consistent when:

$$\forall (i,j) \in \mathcal{J}^2, i \rightarrow j \implies [r_i + p_i \leq r_j] \wedge [d_i \leq d_j + p_j].$$



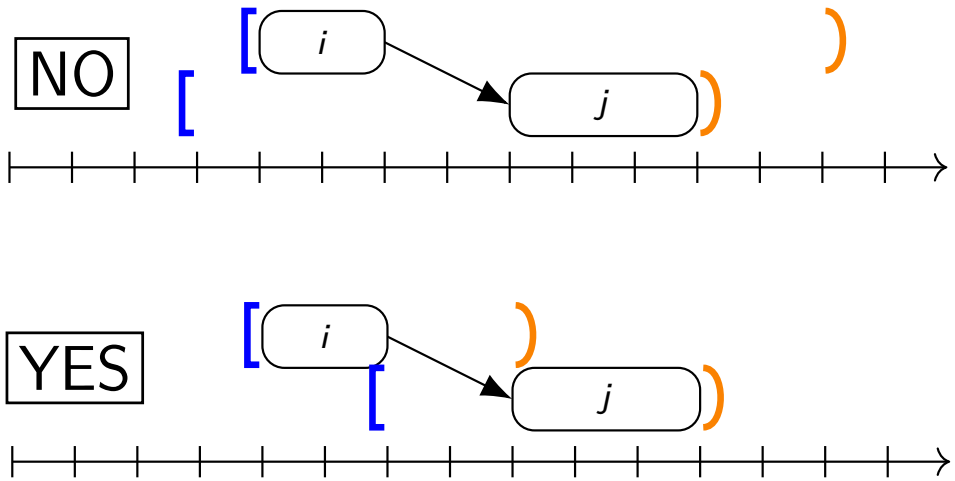
With precedence relations

- *prec* precedence graph of the instance: "$i \longrightarrow j$".

Definition: *prec*-consistency

An instance I of $1|prec, r_j, d_j|C_{max}$ is called *prec*-consistent when:

$$\forall (i,j) \in \mathcal{J}^2, i \longrightarrow j \implies [r_i + p_i \leq r_j] \wedge [d_i \leq d_j + p_j].$$

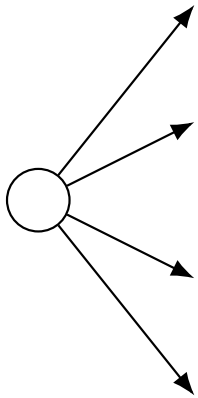


Proposition

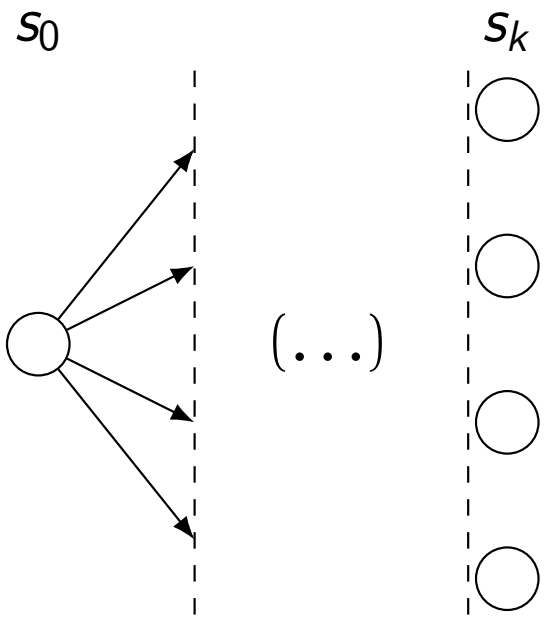
On the *prec*-consistent instances of $1|prec, r_j, d_j|C_{max}$ the summit-ordered schedules are dominant.

Dynamic programming with the summit-ordered schedules

s_0



Dynamic programming with the summit-ordered schedules



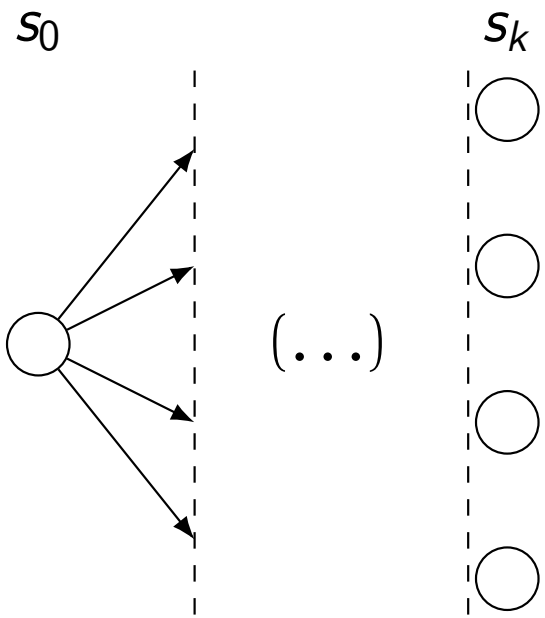
Dynamic programming with the summit-ordered schedules

Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.



Dynamic programming with the summit-ordered schedules

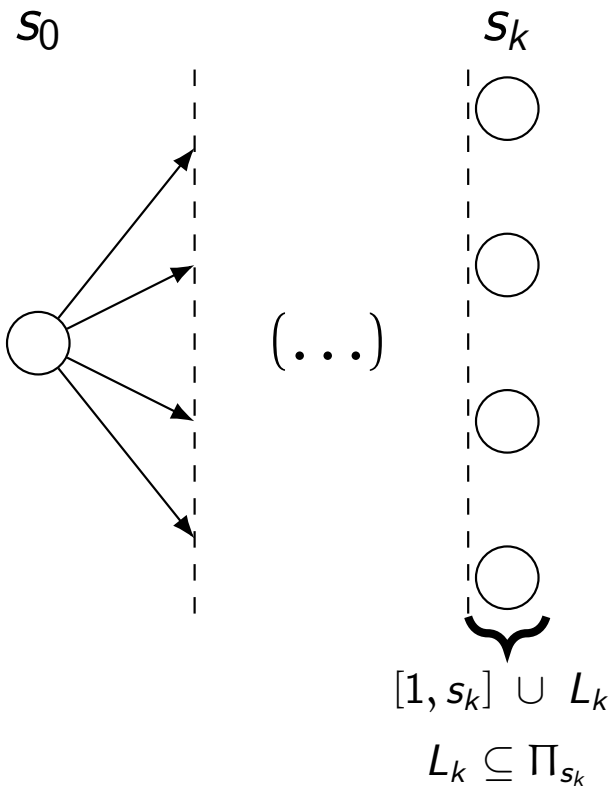
Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

- Rename jobs in $[1, n]$ following their lexicographic order $\prec$.



Dynamic programming with the summit-ordered schedules

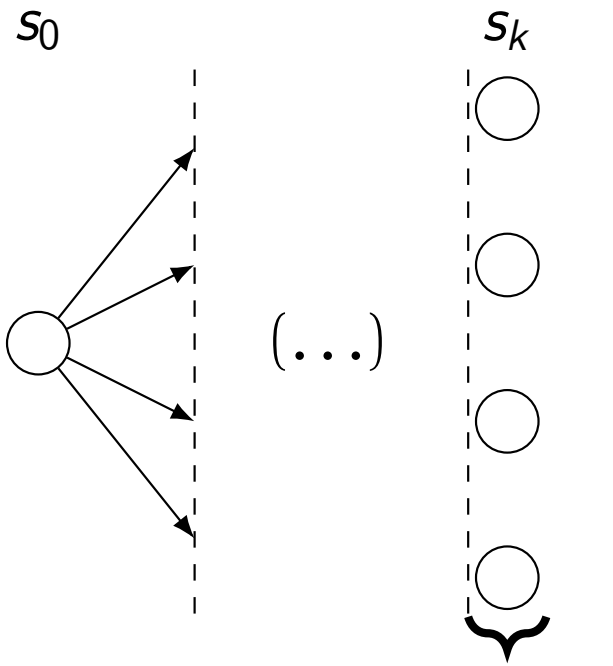
Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

- Rename jobs in $[1, n]$ following their lexicographic order $\prec$.



Dynamic programming with the summit-ordered schedules

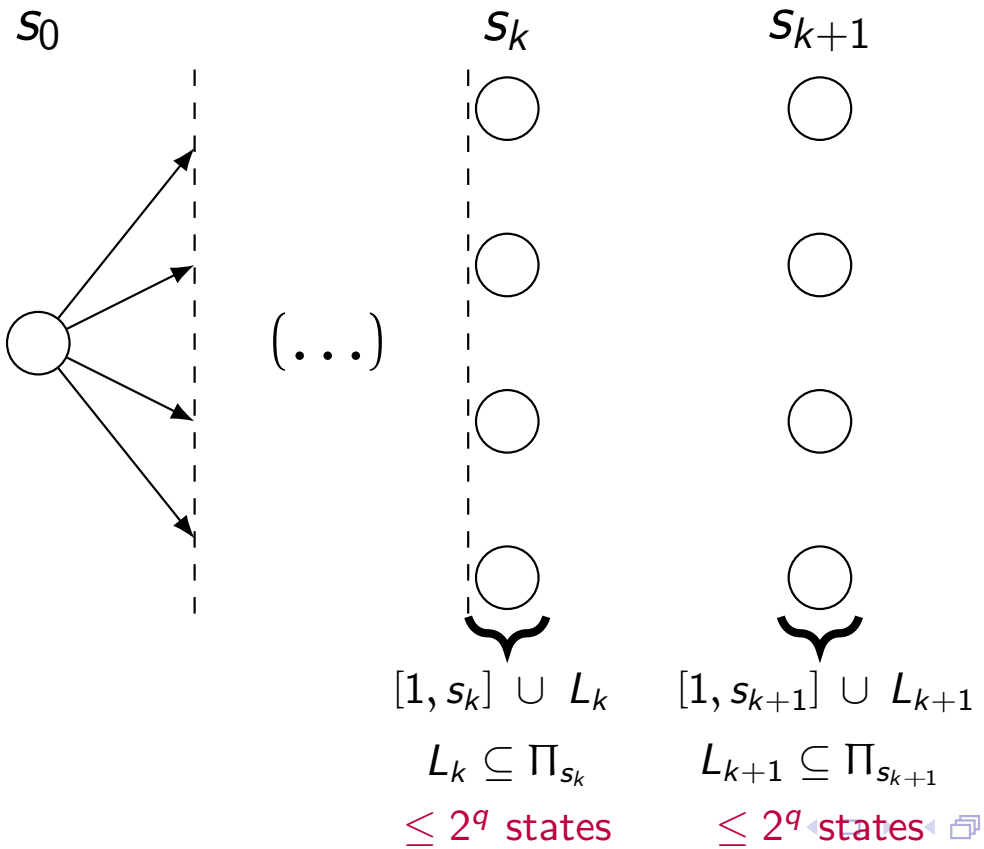
Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

- Rename jobs in $[1, n]$ following their lexicographic order $\prec$.



Dynamic programming with the summit-ordered schedules

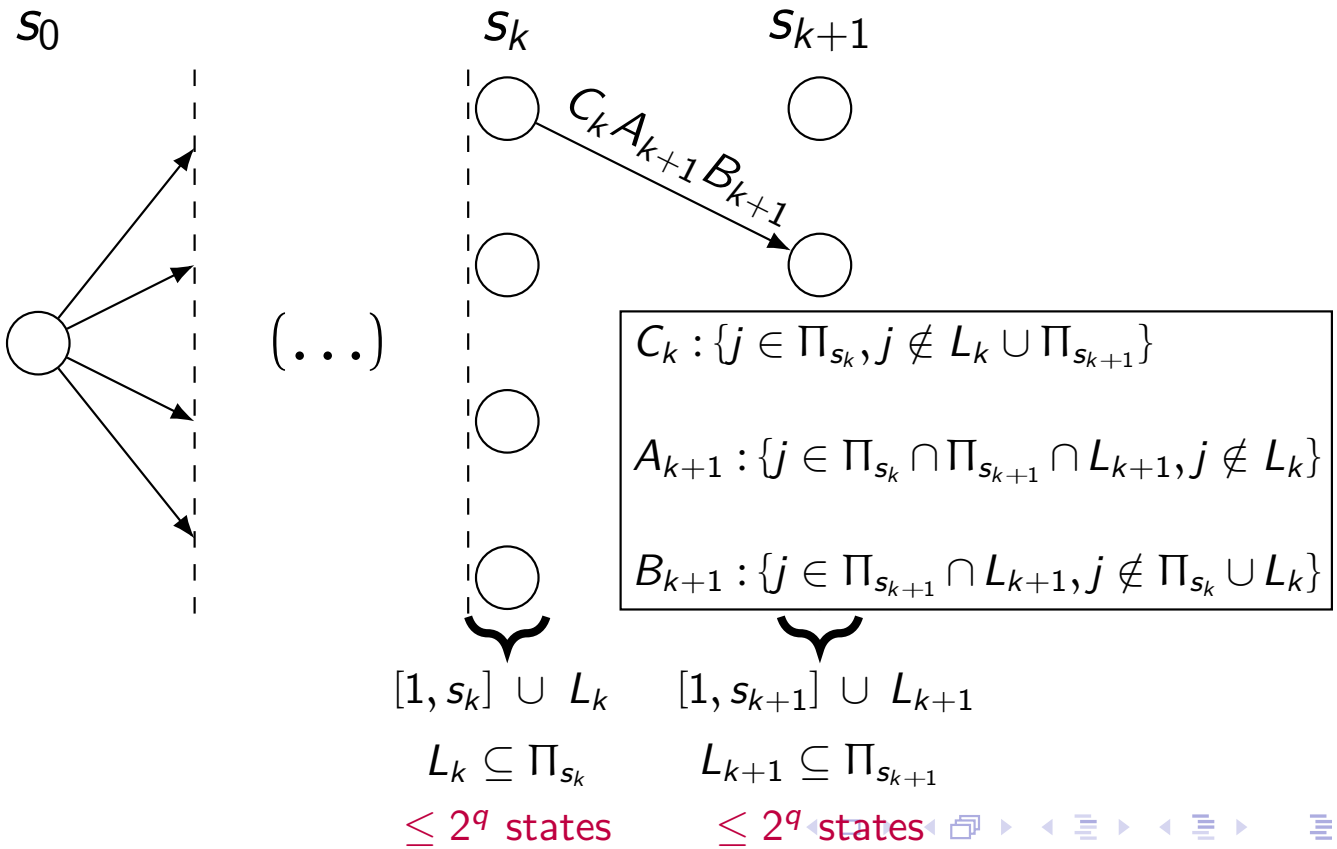
Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

- Rename jobs in $[1, n]$ following their lexicographic order $\prec$.



Dynamic programming with the summit-ordered schedules

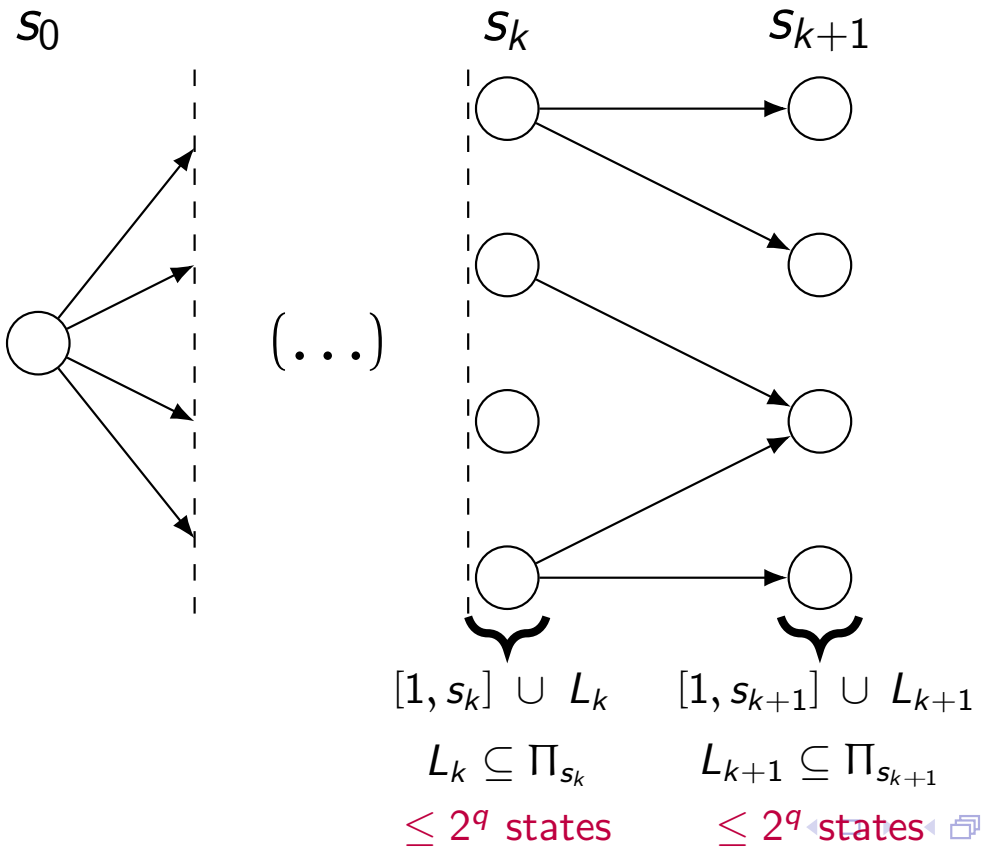
Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

- Rename jobs in $[1, n]$ following their lexicographic order $\prec$.



Dynamic programming with the summit-ordered schedules

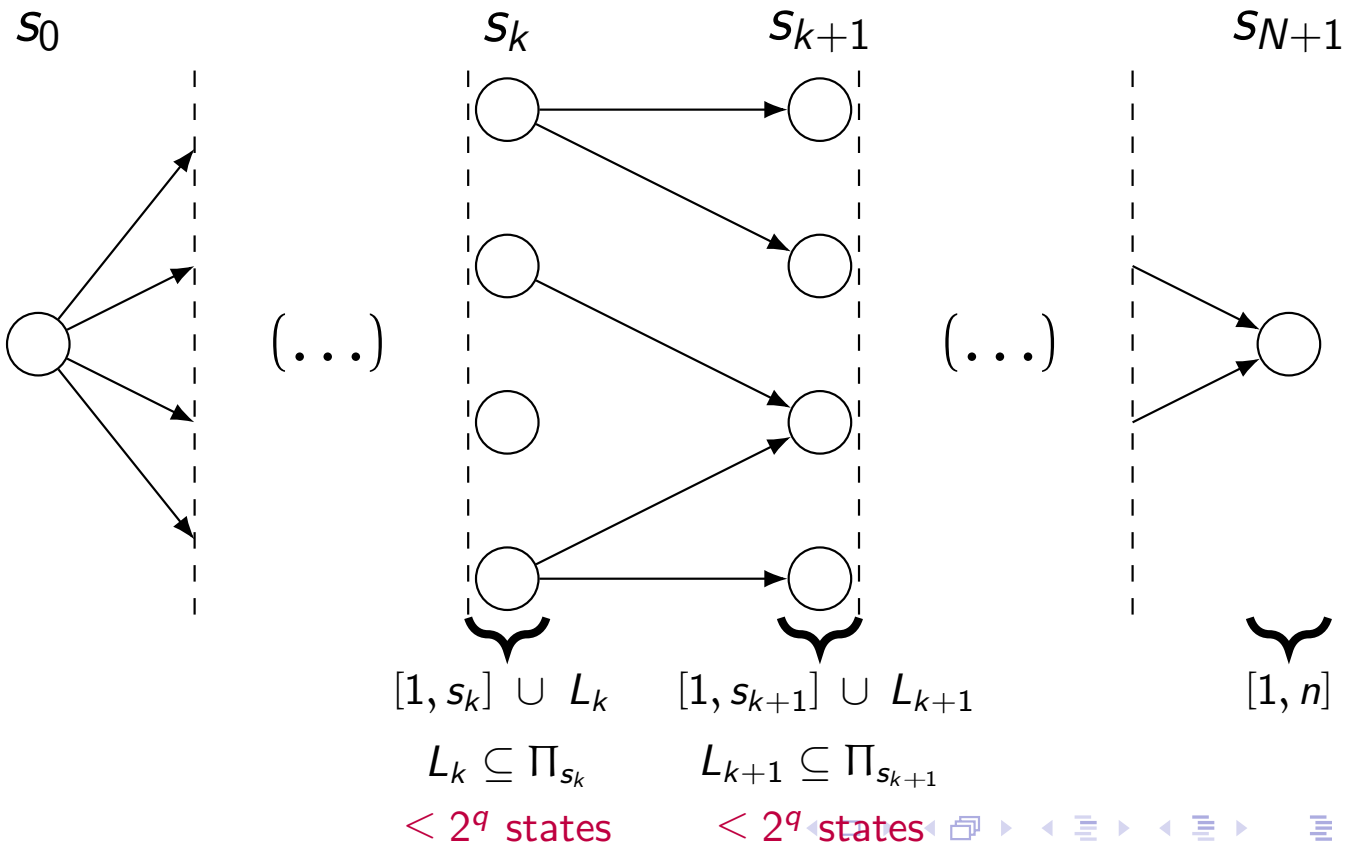
Lemma 1: Job order with summits

In any feasible schedule σ which follows the (wED) rule:

(*i*) for all k in $[1, N]$ and $i \prec s_k$ we have $\sigma(i) < \sigma(s_k)$,

(*ii*) for all k in $[1, N]$ and $j \succ s_k$ if $\sigma(j) < \sigma(s_k)$ then $j \in \Pi_{s_k}$.

- Rename jobs in $[1, n]$ following their lexicographic order $\prec$.



Main result

Theorem [M., Hanen, Munier 24]

$1|prec, r_j, d_j|C_{max}$ can be solved in time:

$$\mathcal{O}(max(1, q^2 \cdot 4^q) \cdot N + n^2)$$

where N is the number of summits ($\leq n$).

Main result

Theorem [M., Hanen, Munier 24]

$1|prec, r_j, d_j|C_{max}$ can be solved in time:

$$\mathcal{O}(max(1, q^2 \cdot 4^q) \cdot N + n^2)$$

where N is the number of summits ($\leq n$).

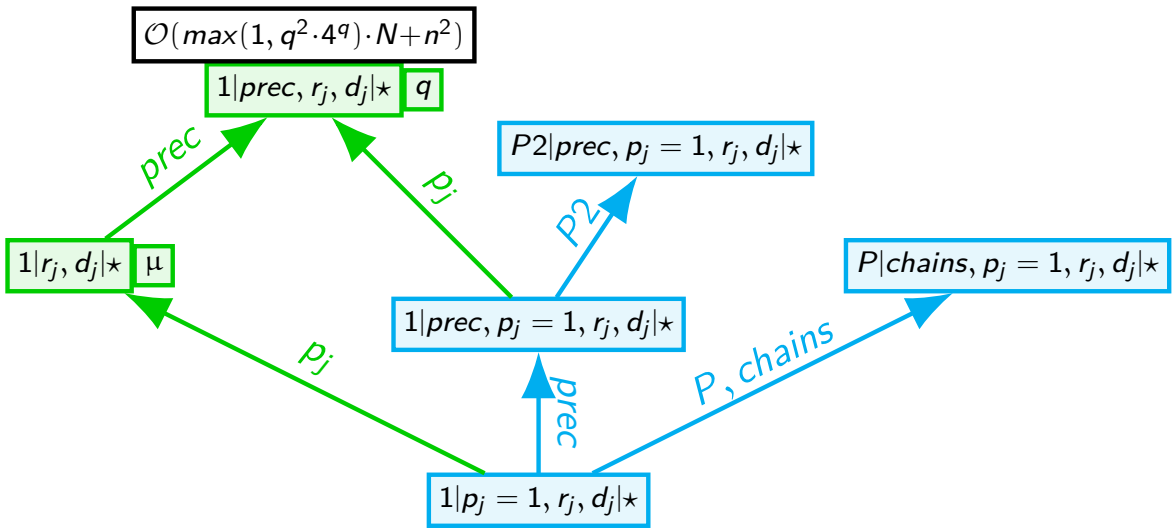
Corollary: Maximum lateness

$1|prec, r_j|L_{max}$ can be solved in time:

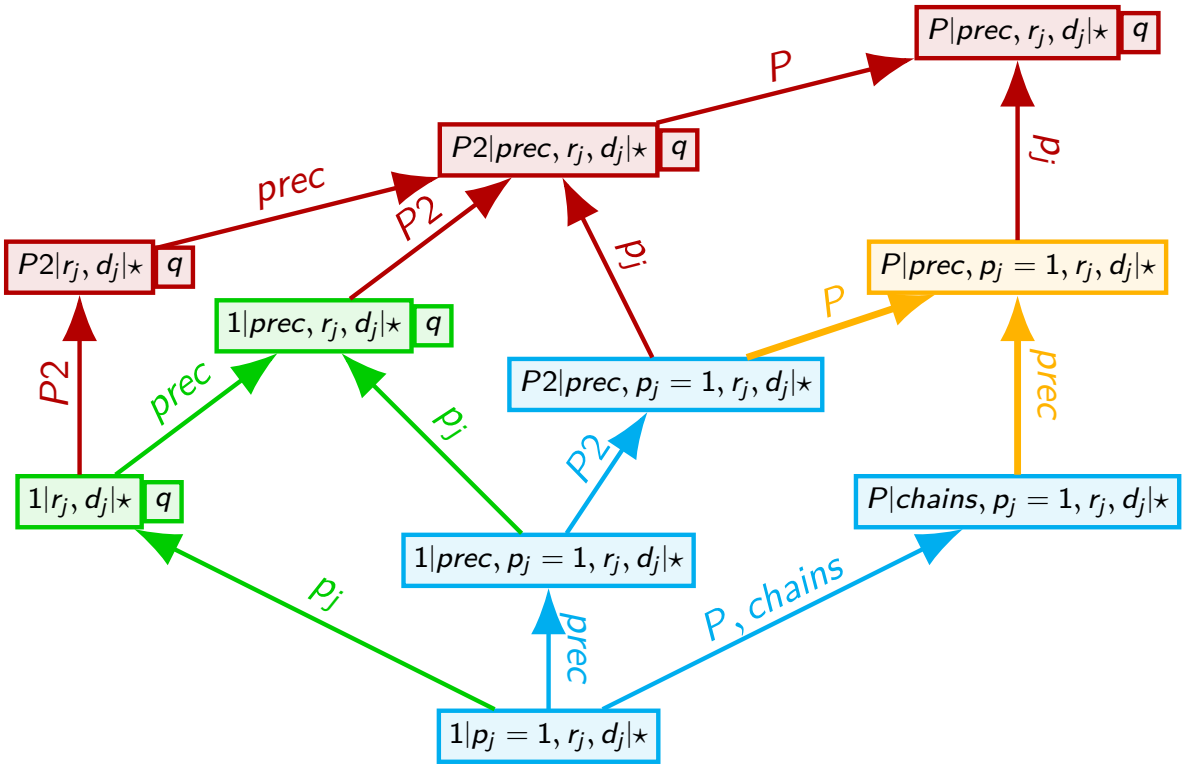
$$\mathcal{O}(max(1, q^2 \cdot 4^q) \cdot N \cdot log(D) + n^2)$$

where N is the number of summits ($\leq n$) and $D = (max_{j \in \mathcal{J}} r_j) + (\sum_{j \in \mathcal{J}} p_j)$.

Problem map



Problem map



Conclusion & future work

Conclusion

- New structural parameter on single machine scheduling with job time windows.
- *FPT* algorithm on $1|prec, r_j, d_j|C_{max}$ derived from a new dominance rule.

Conclusion & future work

Conclusion

- New structural parameter on single machine scheduling with job time windows.
- *FPT* algorithm on $1|prec, r_j, d_j|C_{max}$ derived from a new dominance rule.

Future work

- Consider other settings (like $P|prec, p_j = 1, r_j, d_j|C_{max}$).
- Refine the approach (bounds, heuristics).