Parameterized Complexity of Parallel Machine Scheduling with Time Windows

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

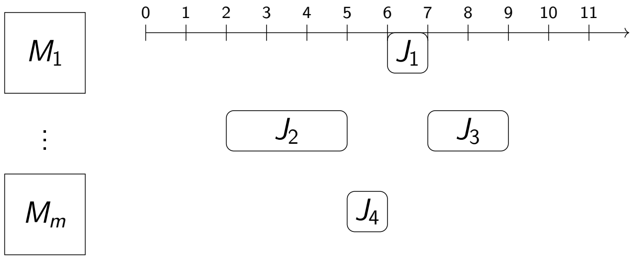
[1]Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France,
[2]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France,
[3]Université Paris Nanterre, F-92000 Nanterre, France,

16 May 2025

Outline

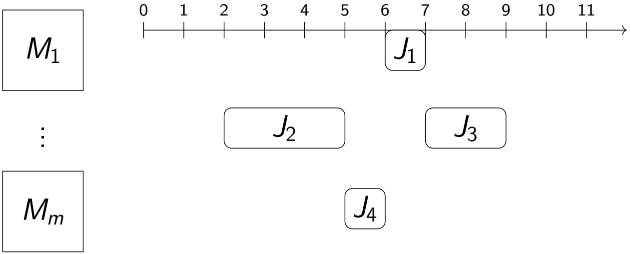
Outline

Scheduling



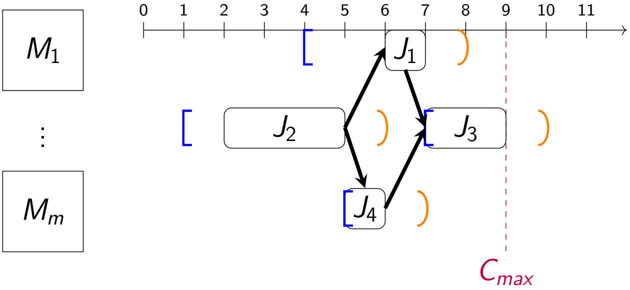
Set $\mathcal{J}$ of n jobs to schedule on m machines $M_1, \ldots, M_m$.

Scheduling



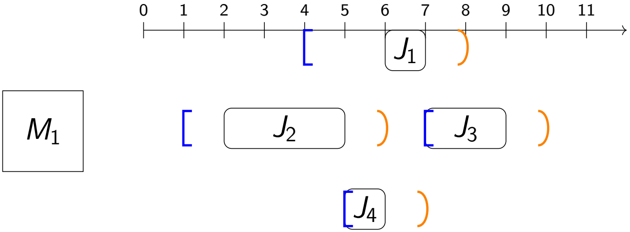
machines | job properties | optimization criterion

Scheduling



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

Scheduling



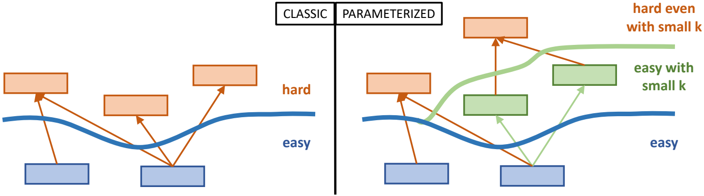
$1|r_j, d_j|C_{max}$ is strongly NP-hard [Lenstra et al. '77].

Motivation

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

Complexité paramétrée et nouveaux schémas énumératifs efficaces pour le RCPSP

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



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

Going beyond NP-hardness

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

Going beyond NP-hardness

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

Fact [Flum, Grohe '06]

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

- Hardness results via reductions

- Hardness results via reductions
  - [Lenstra, Rinnooy Kan '78] $P|prec, p_j = 1|C_{max}$ is para-NP-hard parameterized by C_{max} (with $C_{max} = 3$).

- Hardness results via reductions
  - [Lenstra, Rinnooy Kan '78] $P|prec, p_j = 1|C_{max}$ is para-NP-hard parameterized by C_{max} (with $C_{max} = 3$).

[Mnich, Wiese '15]

- Hardness results via reductions
  - [Lenstra, Rinnooy Kan '78] $P|prec, p_j = 1|C_{max}$ is para-NP-hard parameterized by C_{max} (with $C_{max} = 3$).
- MIP with a bounded number of integer variables

[Mnich, Wiese '15]

- Hardness results via reductions
  - [Lenstra, Rinnooy Kan '78] $P|prec, p_j = 1|C_{max}$ is para-NP-hard parameterized by C_{max} (with $C_{max} = 3$).
- MIP with a bounded number of integer variables
  - [Lenstra '83] MILP: [Mnich, Wiese '15]
  - [Dadush et al. '11] MICP
  - [Knop et al. '19] MIMO

- Hardness results via reductions
  - [Lenstra, Rinnooy Kan '78] $P|prec, p_j = 1|C_{max}$ is para-NP-hard parameterized by C_{max} (with $C_{max} = 3$).
- 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

High-multiplicity setting

Definition

Input of $P|HM, r_j, d_j|X$:

- integers m and d,
- integer vector $(n_i, p_i, r_i, d_i)_{1 \leq i \leq d}$.

High-multiplicity setting

Definition

Input of $P|HM, r_j, d_j|X$:

- integers m and d,
- integer vector $(n_i, p_i, r_i, d_i)_{1 \leq i \leq d}$.

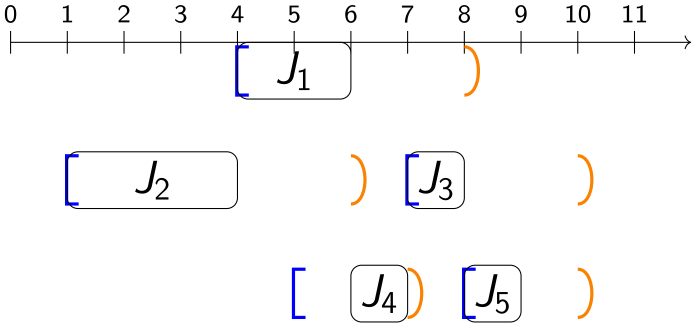
Theorem [Knop et al. '19]

Let $\mathcal{X}_{lin} = \{C_{max}, C_{min}, F_{max}, L_{max}, \sum w_j U_j\}$.
Let $\mathcal{X}_{poly} = \{\sum w_j C_j, \sum w_j F_j, \sum w_j T_j, \ell_p - norm\}$.

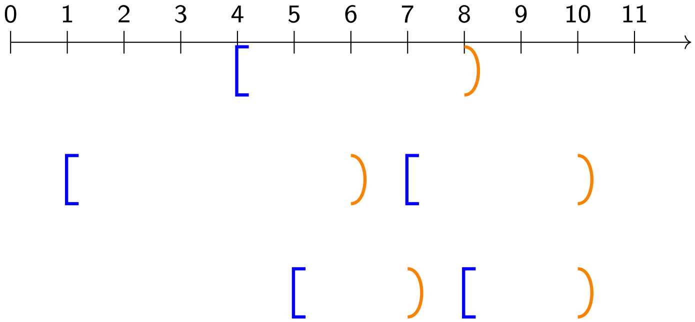
$P|HM, r_j, d_j|X$ is FPT parameterized by:

- $d + p_{max}$ if $X \in \mathcal{X}_{lin} \cup \mathcal{X}_{poly}$,
- $d + m$ if $X \in \mathcal{X}_{lin}$.

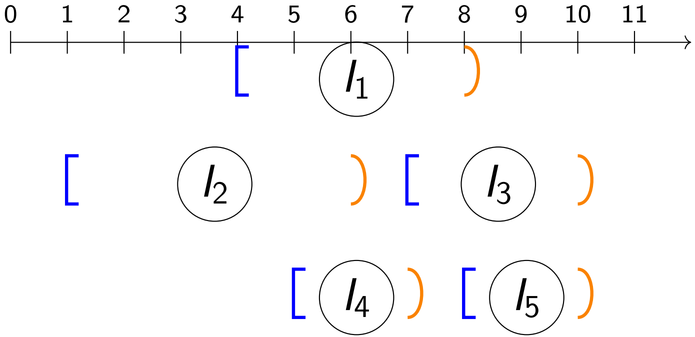
Outline


0
1
2
3
4
5
6
7
8
9
10
11
J_1
J_2
J_3
J_4
J_5

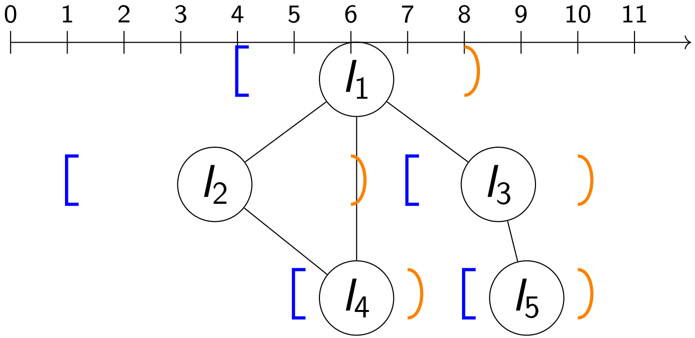
Pathwidth μ



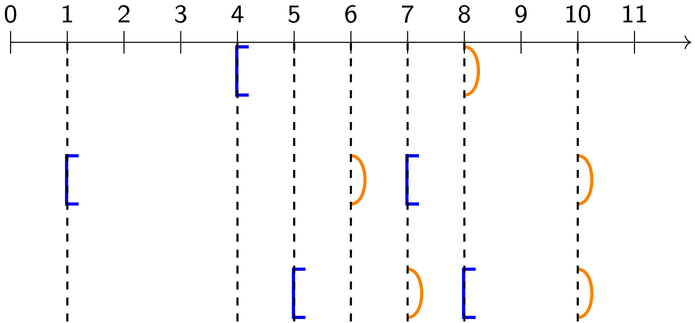
Pathwidth μ



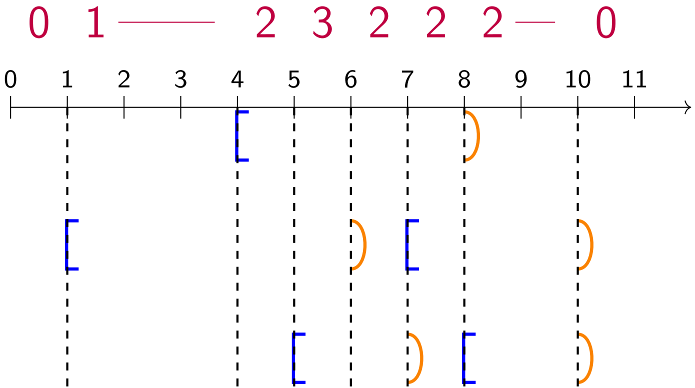
Pathwidth μ



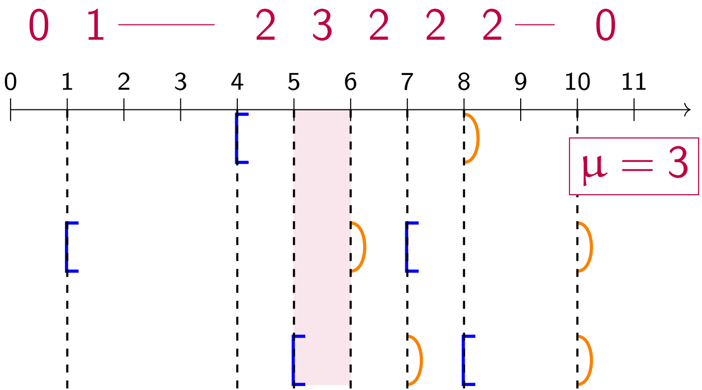
Pathwidth μ

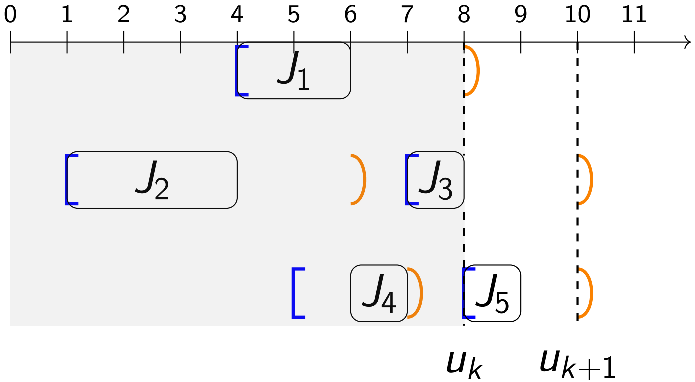


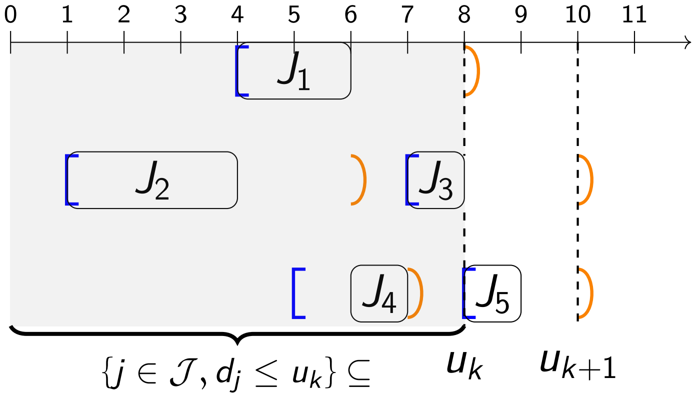
Pathwidth μ



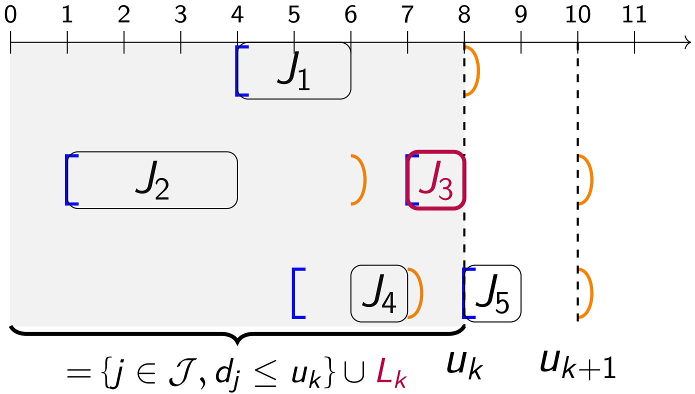
Pathwidth μ




0
1
2
3
4
5
6
7
8
9
10
11
J_1
J_2
J_3
J_4
J_5
u_k
u_{k+1}

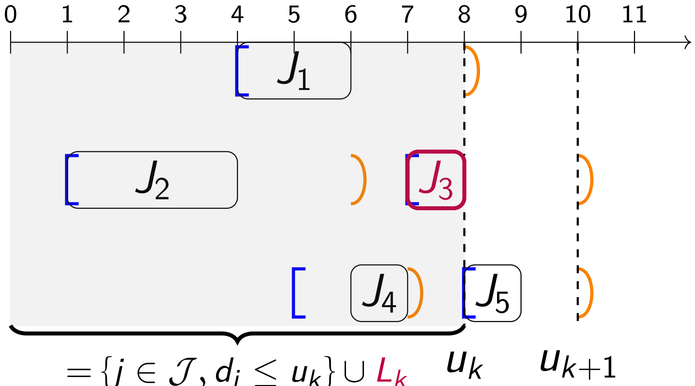

0
1
2
3
4
5
6
7
8
9
10
11
J_1
J_2
J_3
J_4
J_5
$\{j \in \mathcal{J}, d_j \leq u_k\} \subseteq$
u_k
u_{k+1}

Pathwidth μ



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

Pathwidth μ



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

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

Dynamic programming with pathwidth μ

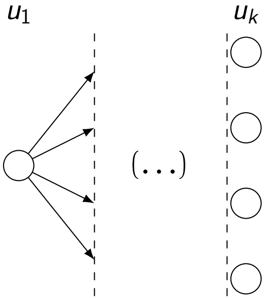
Dynamic programming state (k, L_k)

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

Dynamic programming with pathwidth μ

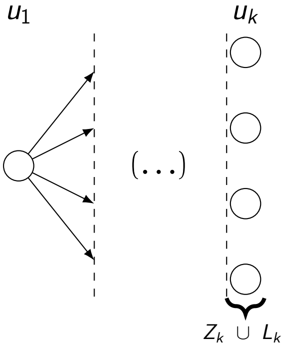
Dynamic programming state (k, L_k)

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



Dynamic programming state (k, L_k)

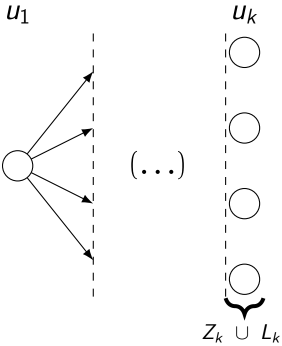
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

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

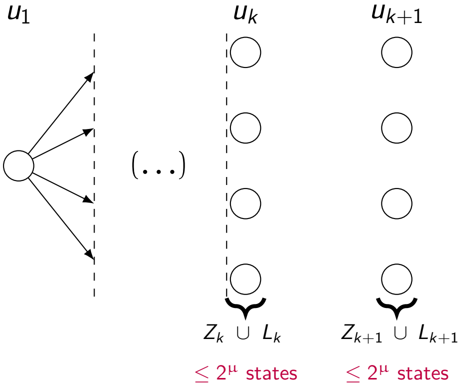


$\leq 2^{\mu}$ states

Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

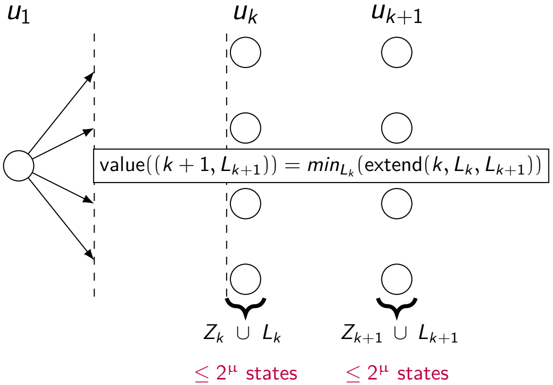
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

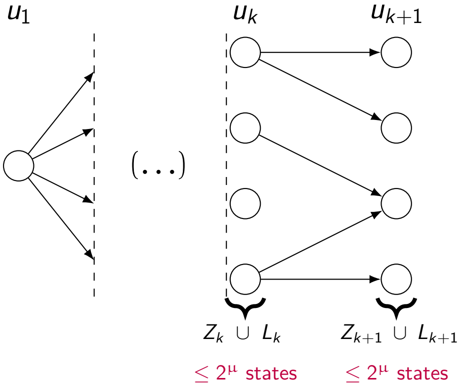
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

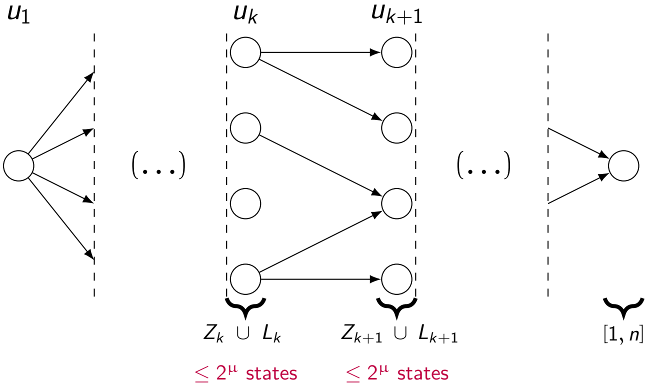
- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = set of jobs starting before u_k.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth μ

Dynamic programming state (k, L_k)

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



Algorithm 1: main()

▷ Solving $1|prec, r_j, d_j|C_{max}$.

preprocessing() ▷ Compute sequence $(u_k)_{1\leq k\leq K}$. Compute associated job subsets.
$u_{K+1} \leftarrow \infty$
Create table T with K columns (1 to K) and 2^{μ} slots in each column k.
Initialize T with ∞ everywhere.
$T[1, \emptyset] \leftarrow 0$
for $k = 1$ to $K - 1$ **do**
  for each (L_k, L_{k+1}) **do**
    $T[k+1, L_{k+1}] \leftarrow \min(T[k+1, L_{k+1}], \text{extend}(k, L_k, L_{k+1}))$
return $T[K, \emptyset]$

Algorithm 2: extend(k, L_k, L_{k+1})

$Cmax \leftarrow \max(u_k, T[k, L_k])$
if $Cmax = \infty$ **then return** ∞
▷ Ensure that all jobs in L_k with a deadline higher than u_{k+1} are still in L_{k+1}.
if $(L_k - Z_{k+1}) \not\subseteq L_{k+1}$ **then return** ∞
▷ Add jobs to the existing partial schedule.
$mandatory \leftarrow Z_{k+1} \cap \{J_j \in \mathcal{J}, [u_k, u_{k+1}) \subseteq [r_j, d_j)\}$
$add \leftarrow (mandatory \cup L_{k+1}) - L_k$
sort$(add, (prec, \text{nondecreasing } d_j))$
for j in add (in order) **do**
    if appending J_j would violate a precedence constraint **then return** ∞
    $Cmax \leftarrow Cmax + p_j$
    if $Cmax > d_j$ **then return** ∞
▷ Ensure that all the added jobs start earlier than u_{k+1}.
$last \leftarrow$ last job in add.
if $Cmax - p_{last} \geq u_{k+1}$ **then return** ∞
return $Cmax$

Main result

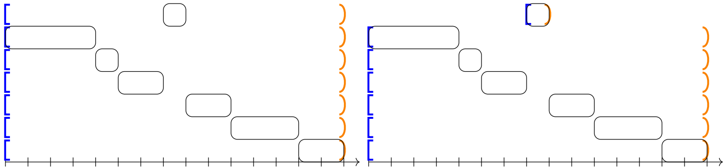
Theorem [Hanen, Mallem, Munier '22]

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

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

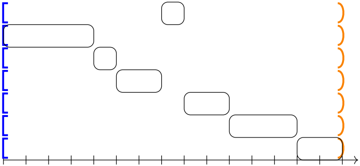
Improving on Pathwidth

- Two single machine instances with time windows.

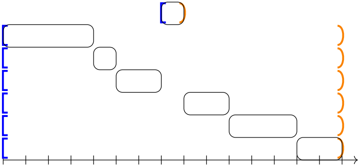


Improving on Pathwidth

- Two single machine instances with time windows.



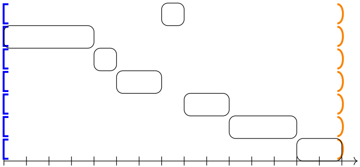
$\mu = n$



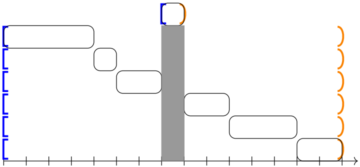
$\mu = n$

Improving on Pathwidth

- Two single machine instances with time windows.



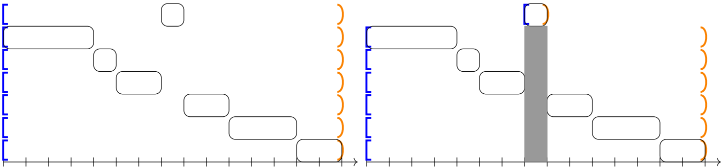
$\mu = n$



$\mu = n$

Improving on Pathwidth

- Two single machine instances with time windows.



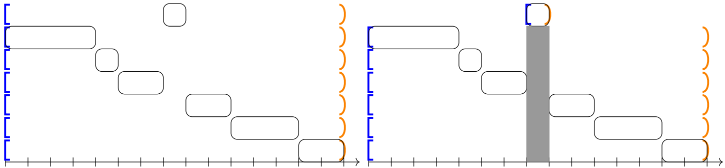
$\mu = n$ $\qquad\qquad$ $\mu = n$

Definition: Proper sets and proper level

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

Improving on Pathwidth

- Two single machine instances with time windows.



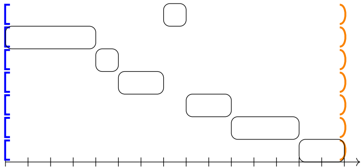
$\mu = n$ $\mu = n$

Definition: Proper sets and proper level

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

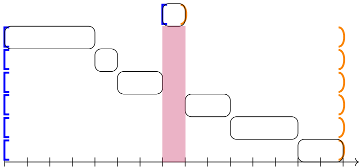
Improving on Pathwidth

- Two single machine instances with time windows.



$\mu = n$

$q = 0$



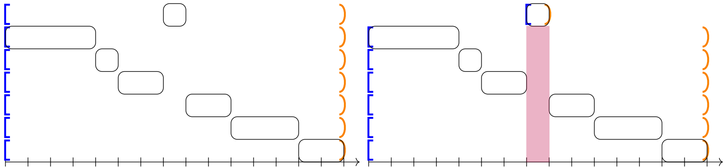
$\mu = n$

$q = n - 1$

Definition: Proper sets and proper level

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

- Two single machine instances with time windows.



$\mu = n$ $\quad$ $\mu = n$

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

Definition: Proper sets and proper level

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

Theorem [Mallem, Hanen, Munier '24]

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

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

Outline

Theorem [Hanen, Munier '23]

$P2|r_j, d_j|C_{max}$ is para-NP-hard parameterized by μ (with $\mu = 4$).

Pathwidth is not enough

Theorem [Hanen, Munier '23]

$P2|r_j, d_j|C_{max}$ is para-NP-hard parameterized by μ (with $\mu = 4$).

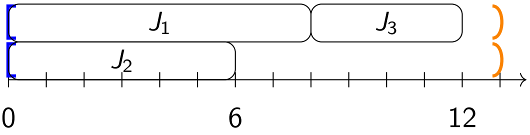
- For each job subset: Pareto front of makespan tuples.

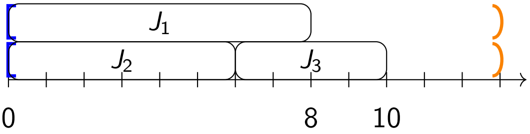
Pathwidth is not enough

Theorem [Hanen, Munier '23]

$P2|r_j, d_j|C_{max}$ is para-NP-hard parameterized by μ (with $\mu = 4$).

- For each job subset: Pareto front of makespan tuples.
  - Example: $(6, 12)$ vs. $(8, 10)$.

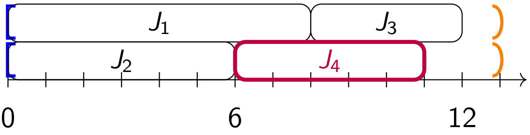


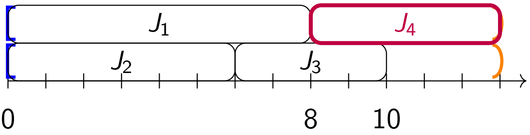


Theorem [Hanen, Munier '23]

$P2|r_j, d_j|C_{max}$ is para-NP-hard parameterized by μ (with $\mu = 4$).

- For each job subset: Pareto front of makespan tuples.
  - Example: $(6, 12)$ vs. $(8, 10)$.



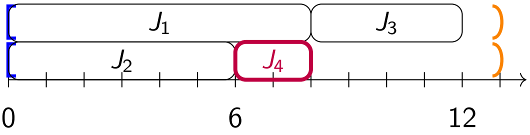


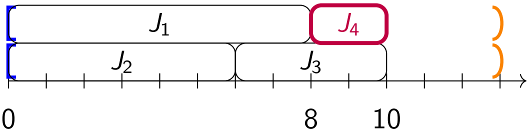
Pathwidth is not enough

Theorem [Hanen, Munier '23]

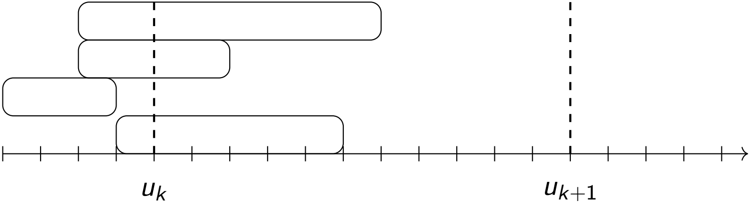
$P2|r_j, d_j|C_{max}$ is para-NP-hard parameterized by μ (with $\mu = 4$).

- For each job subset: Pareto front of makespan tuples.
  - Example: $(6, 12)$ vs. $(8, 10)$.

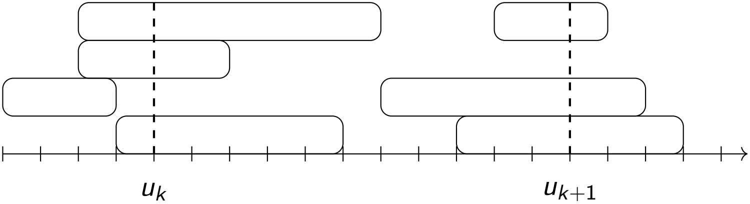




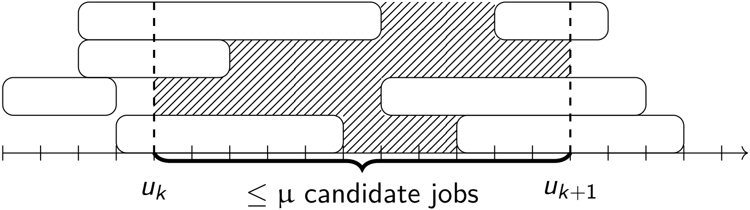
Bounding the number of border schedules



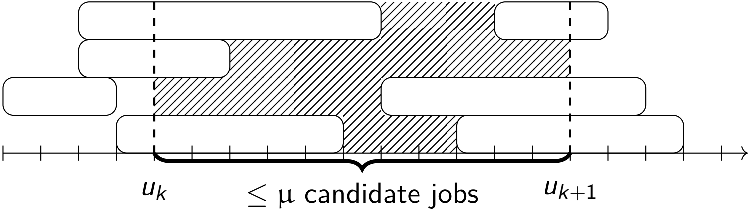
Bounding the number of border schedules



Bounding the number of border schedules



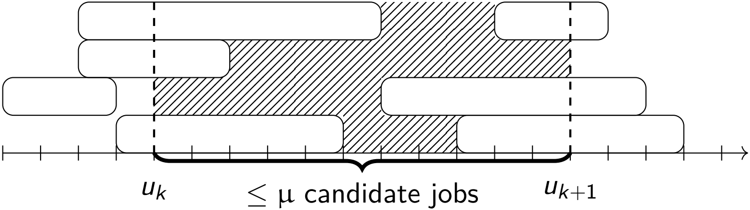
Bounding the number of border schedules



Parameter definitions

- Maximum processing time $p_{max} = max_{J_j \in \mathcal{J}}(p_j)$.

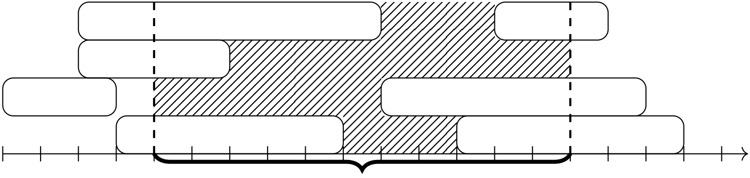
Bounding the number of border schedules



Parameter definitions

- Maximum processing time $p_{max} = max_{J_j \in \mathcal{J}}(p_j)$.
- Slack $\sigma = max_{J_j \in \mathcal{J}}(d_j - r_j - p_j)$.

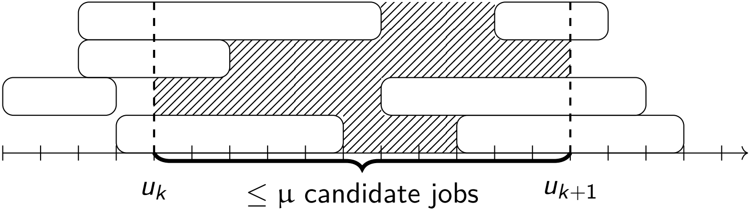
Bounding the number of border schedules



Parameter definitions

- Maximum processing time $p_{max} = max_{J_j \in \mathcal{J}}(p_j)$.
- Slack $\sigma = max_{J_j \in \mathcal{J}}(d_j - r_j - p_j)$.
- Border flexibility $\beta = max_{J_j \in \mathcal{J}}[min(p_j - 1, d_j - r_j - p_j)]$.

Bounding the number of border schedules

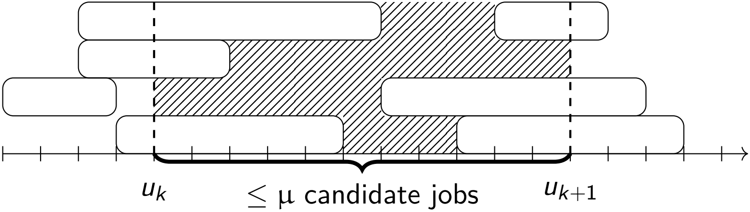


Parameter definitions

- Maximum processing time $p_{max} = max_{J_j \in \mathcal{J}}(p_j)$.
- Slack $\sigma = max_{J_j \in \mathcal{J}}(d_j - r_j - p_j)$.
- Border flexibility $\beta = max_{J_j \in \mathcal{J}}[min(p_j - 1, d_j - r_j - p_j)]$.

- $\mathcal{O}((\mu\beta + 1)^m)$ "border schedules" B_k associated to u_k.

Bounding the number of border schedules



Parameter definitions

- Maximum processing time $p_{max} = max_{J_j \in \mathcal{J}}(p_j)$.
- Slack $\sigma = max_{J_j \in \mathcal{J}}(d_j - r_j - p_j)$.
- Border flexibility $\beta = max_{J_j \in \mathcal{J}}[min(p_j - 1, d_j - r_j - p_j)]$.

- $\mathcal{O}((\mu\beta + 1)^m)$ "border schedules" B_k associated to u_k.
- $\mathcal{O}(2^\mu \cdot (\mu\beta + 1)^\mu)$ states (k, L_k, B_k) associated to u_k.

Results on parallel machines

Theorem [Tarhan et al. '23]

$P|prec, r_j, d_j|C_{max}$ is FPT parameterized by $\mu + p_{max}$.

Theorem [Tarhan et al. '23]

$P|prec, r_j, d_j|C_{max}$ is FPT parameterized by $\mu + p_{max}$.

Theorem [Hanen, Munier '23]

$P|\mathcal{M}_j(type), prec, r_j, d_j|C_{max}$ is FPT parameterized by $\mu + min(p_{max}, \sigma)$.

Theorem [Tarhan et al. '23]

$P|prec, r_j, d_j|C_{max}$ is FPT parameterized by $\mu + p_{max}$.

Theorem [Hanen, Munier '23]

$P|\mathcal{M}_j(type), prec, r_j, d_j|C_{max}$ is FPT parameterized by $\mu + min(p_{max}, \sigma)$.

Theorem [Munier '21]

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

Results on parallel machines

Theorem [Tarhan et al. '23]

$P|prec, r_j, d_j|C_{max}$ is FPT parameterized by $\mu + p_{max}$.

Theorem [Hanen, Munier '23]

$P|\mathcal{M}_j(type), prec, r_j, d_j|C_{max}$ is FPT parameterized by $\mu + min(p_{max}, \sigma)$.

Theorem [Munier '21]

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

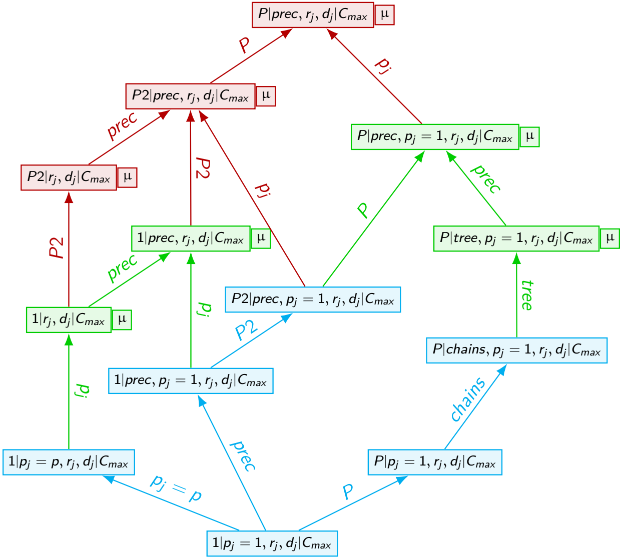
Theorem [Mallem, Hanen, Munier '25 (submitted)]

$P|prec, p_j = 1, r_j, d_j|C_{max}$ is para-NP-hard parameterized by $q + C_{max}$ (with $q = 0$ and $C_{max} = 3$).

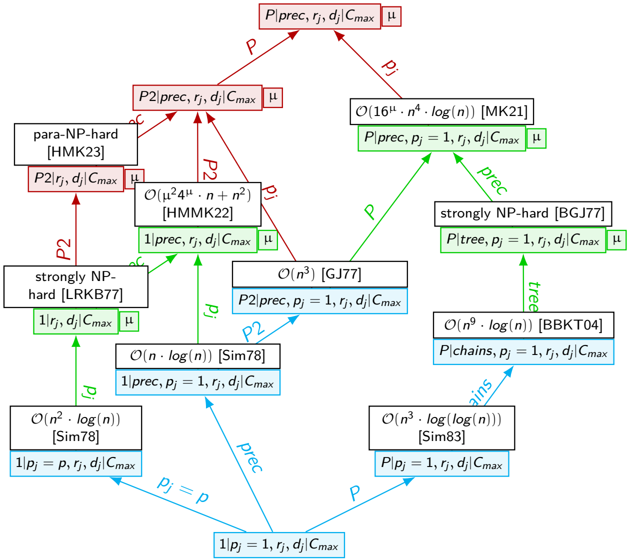
Outline

1. Introduction
   - The Problem
   - Parameterized Complexity
2. With a single machine
   - Parameter Pathwidth
   - Dynamic Programming
   - Improving the Pathwidth: Parameter Proper Level
3. With Parallel Identical Machines
   - Pathwidth is not Enough
   - Border Enumeration
4. Conclusion

Conclusion


$P|prec, r_j, d_j|C_{max}$ µ
P
p_j
$P2|prec, r_j, d_j|C_{max}$ µ
$prec$
$P2$
p_j
$P|prec, p_j = 1, r_j, d_j|C_{max}$ µ
$P2|r_j, d_j|C_{max}$ µ
P
$prec$
$P2$
$1|prec, r_j, d_j|C_{max}$ µ
$P|tree, p_j = 1, r_j, d_j|C_{max}$ µ
$prec$
$1|r_j, d_j|C_{max}$ µ
p_j
$P2|prec, p_j = 1, r_j, d_j|C_{max}$
$tree$
$P2$
$P|chains, p_j = 1, r_j, d_j|C_{max}$
$1|prec, p_j = 1, r_j, d_j|C_{max}$
p_j
$chains$
$prec$
$1|p_j = p, r_j, d_j|C_{max}$
$P|p_j = 1, r_j, d_j|C_{max}$
$p_j = p$
P
$1|p_j = 1, r_j, d_j|C_{max}$

Conclusion


$P|prec, r_j, d_j|C_{max}$ μ
P
pj
$P2|prec, r_j, d_j|C_{max}$ μ
$\mathcal{O}(16^{\mu} \cdot n^4 \cdot log(n))$ [MK21]
$P|prec, p_j = 1, r_j, d_j|C_{max}$ μ
para-NP-hard [HMK23]
$P2|r_j, d_j|C_{max}$ μ
P2
$\mathcal{O}(\mu^2 4^{\mu} \cdot n + n^2)$ [HMMK22]
$1|prec, r_j, d_j|C_{max}$ μ
prec
strongly NP-hard [BGJ77]
$P|tree, p_j = 1, r_j, d_j|C_{max}$ μ
strongly NP-hard [LRKB77]
$1|r_j, d_j|C_{max}$ μ
$\mathcal{O}(n^3)$ [GJ77]
$P2|prec, p_j = 1, r_j, d_j|C_{max}$
tree
$\mathcal{O}(n^9 \cdot log(n))$ [BBKT04]
$P|chains, p_j = 1, r_j, d_j|C_{max}$
$\mathcal{O}(n \cdot log(n))$ [Sim78]
$1|prec, p_j = 1, r_j, d_j|C_{max}$
$\mathcal{O}(n^2 \cdot log(n))$ [Sim78]
$1|p_j = p, r_j, d_j|C_{max}$
$\mathcal{O}(n^3 \cdot log(log(n)))$ [Sim83]
$P|p_j = 1, r_j, d_j|C_{max}$
$p_j = p$
P
$1|p_j = 1, r_j, d_j|C_{max}$

Future work

- Improve the existing algorithms.

Future work

- Improve the existing algorithms.
- Investigate other scheduling problems with time windows.
  - Postdoc: $1|periodic_L(r_j, d_j)|C_{max}$

Future work

- Improve the existing algorithms.
- Investigate other scheduling problems with time windows.
  - Postdoc: $1|periodic_L(r_j, d_j)|C_{max}$
- Consider FPT approximation algorithms.

  Example:

  [Baart et al. '21] $1|r_j, \bar{d}_j, s_{j,k}, reject|\sum_{J_j accepted}(v_j - w_j T_j)$

Future work

- Improve the existing algorithms.
- Investigate other scheduling problems with time windows.
  - Postdoc: $1|periodic_L(r_j, d_j)|C_{max}$
- Consider FPT approximation algorithms.

  Example:
  [Baart et al. '21] $1|r_j, \bar{d}_j, s_{j,k}, reject|\sum_{J_j accepted}(v_j - w_j T_j)$
  - FPT exact resolution with respect to $\mu + \sigma$: $\mathcal{O}(\sigma\mu^2\, 2^{\mu} \cdot n^2)$.
  - FPT *FPTAS* with respect to μ: $\mathcal{O}(\mu^2\, 2^{pw} \cdot \frac{n^3}{\varepsilon^2})$.

Parameterized complexity classes

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

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

para-NP

XP

$XNLP$

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

$\cdots$

$W[2]$

$W[1]$

FPT

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

Parameterized complexity classes

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

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

$\boxed{k\text{-COLORING}}$ para-NP

XP

$XNLP$

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

$\dots$

$\boxed{k\text{-DOMINATING SET}}$ $W[2]$

$\boxed{k\text{-CLIQUE}}$ $W[1]$

FPT

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

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

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

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

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

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

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

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

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

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

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