Insight on the parameterized complexity of scheduling problems

Designing FPT algorithms with dynamic programming

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

[1]Sorbonne université, CNRS, LIP6, F-75005, Paris, France
[2] Université Paris Nanterre, 92000, Nanterre, France

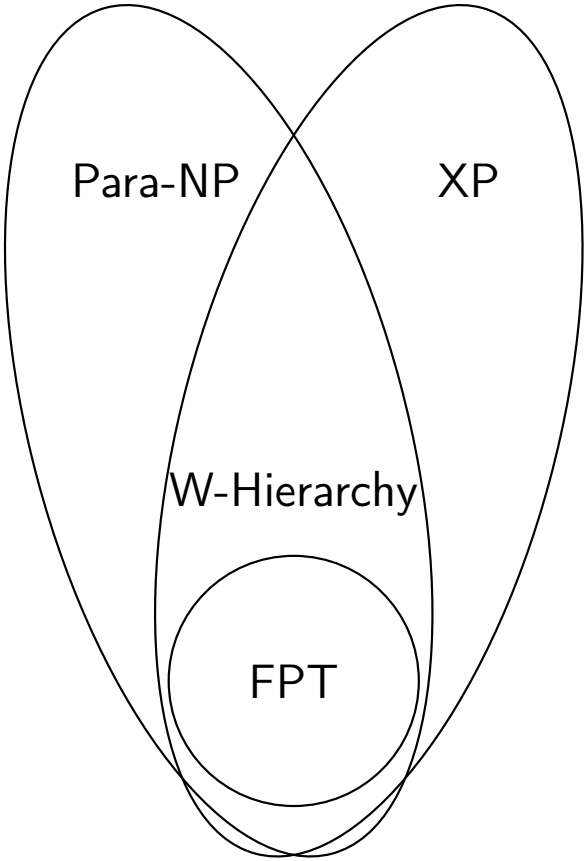
may 17, 2024

New challenges in scheduling Theory, Aussois 2024

Contents

1. Parameterized complexity
2. Dynamic programming for scheduling
3. Parameters
4. Single machine with pathwidth μ
5. Single machine with q-proper

Parameterized complexity classes



A parameterized problem of size n with parameter k :

Definition

FPT is the class of problems solvable by a fixed-parameter tractable algorithm with time complexity $\mathcal{O}(f(k) \times \text{poly}(n))$, where f is a computable function and poly(n) a polynome of n.

Challenges for scheduling problems

- Which problems can be shown FPT?
- Which relevant structural parameters for scheduling?
- How to design FPT algorithms?

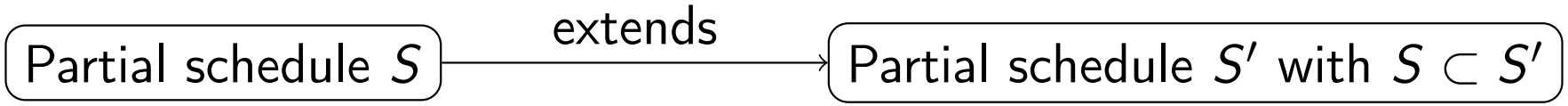
Challenges for scheduling problems

- Which problems can be shown FPT?
- Which relevant structural parameters for scheduling?
- How to design FPT algorithms?

Two main techniques:

1. ILP with special structure (for example Hermelin, Karhi, Pinedo,and Shabtay, 2021)
2. Dynamic programming

Dynamic programming for scheduling



- The second partial schedule extends the first one.
- A state records the minimum information needed from S to build feasible extensions S'
- Dynamic programming algorithm starts from an empty schedule and builds a state graph.

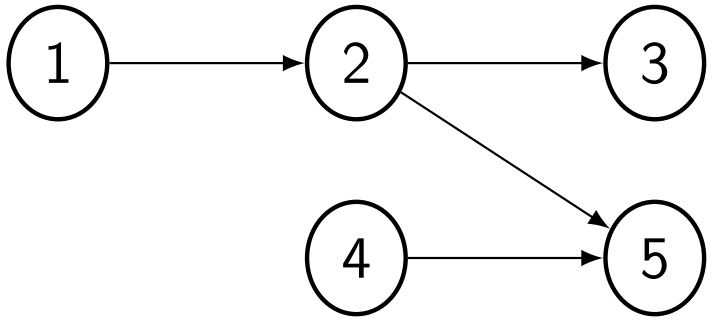
Key arguments

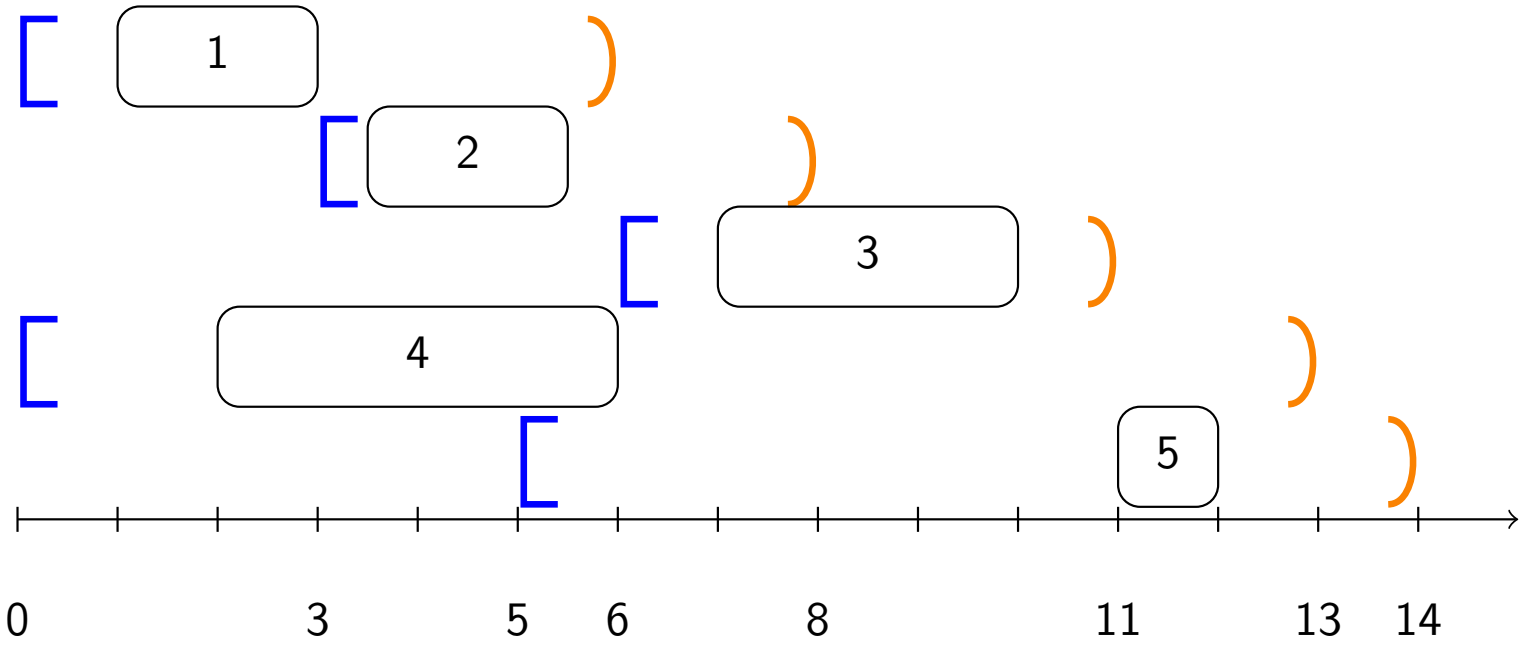
- Key argument 1: A schedule is defined by a path on the state graph.
- Feasibility: finding a path from a source to a sink
- Optimization: finding a minimum cost path from a source to a sink
- Key Argument 2 : Bounding the number of states.

Parameters: A single machine problem example

We consider $1|prec, r_i, \bar{d}_i|C_{max}$ usual parameters:

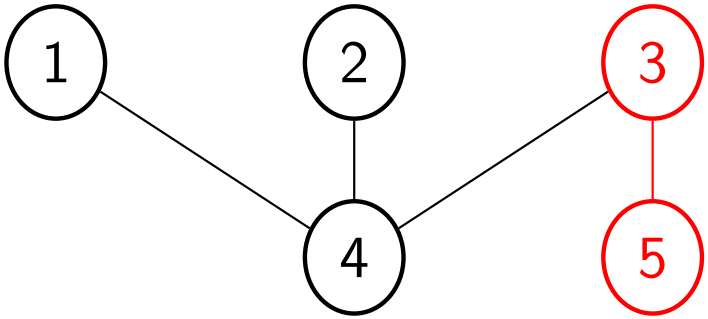
- C_{max}
- Slack $\sigma = \max_i(\bar{d}_i - r_i - p_i)$
- $p_{max} = \max_i p_i$





Compatibility graph

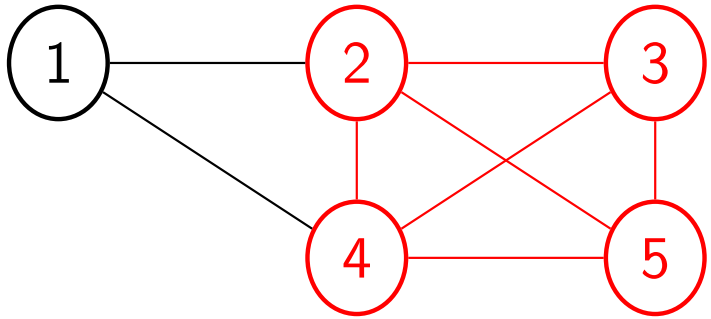
Compatibility graph of precedence constraints



$maxclique = w(G)$ (width of the precedence graph)

$P|prec, p_j = 1|C_{max} \leq D$ is XNLP-complete[Bodlaender et al, 2022]

Compatibility graph of time windows



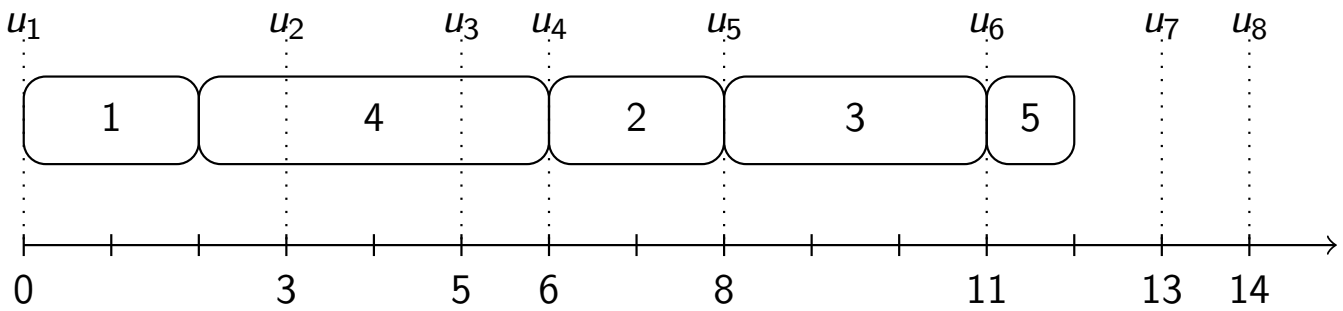
maxclique is the nb or overlaping intervals

$Pathwidth = maxclique - 1 = 3$

$P|prec, p_i = 1, r_i, d_i|\star$ FPT [Munier Kordon 2021]

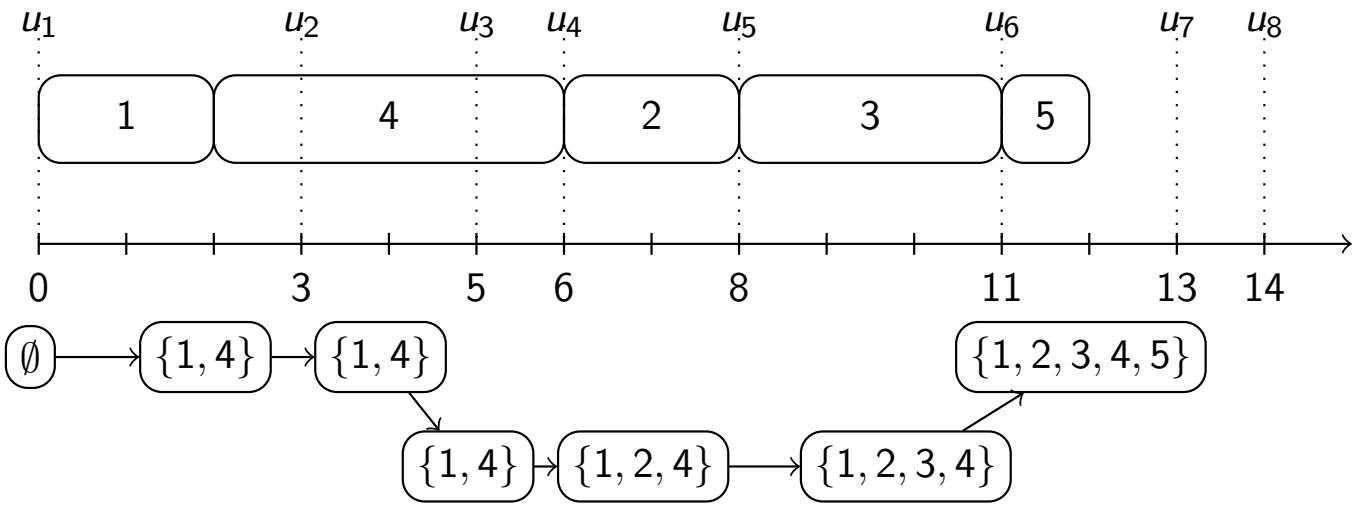
Single machine with pathwidth μ

Sorting the end points of time windows $u_1, \ldots, u_K$



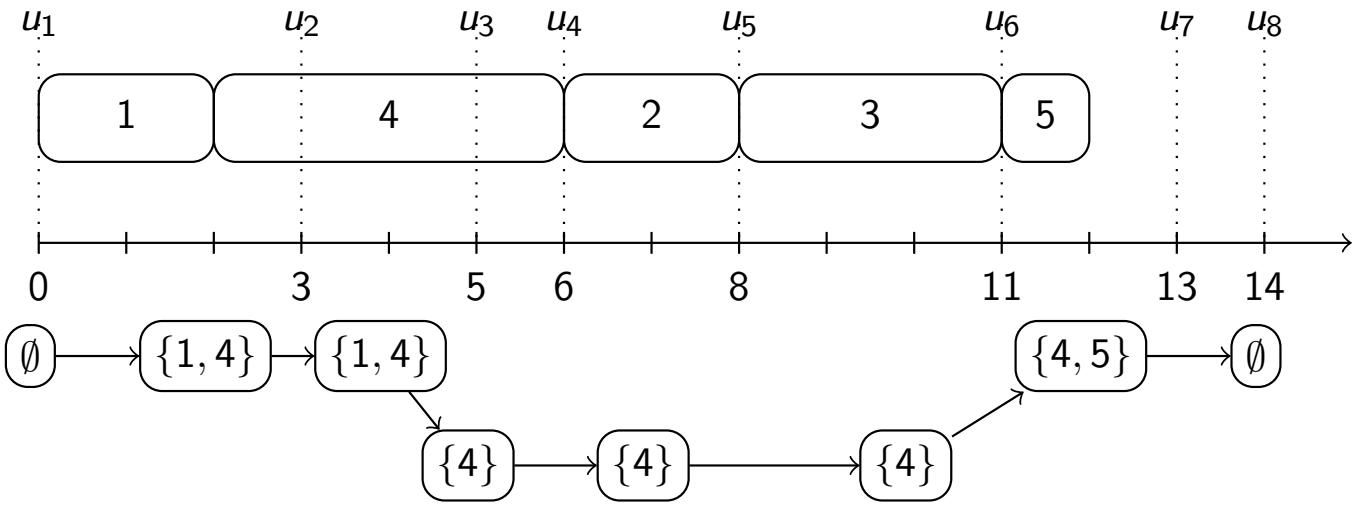
Single machine with pathwidth μ

Recording the set of scheduled jobs before each u_{k+1}

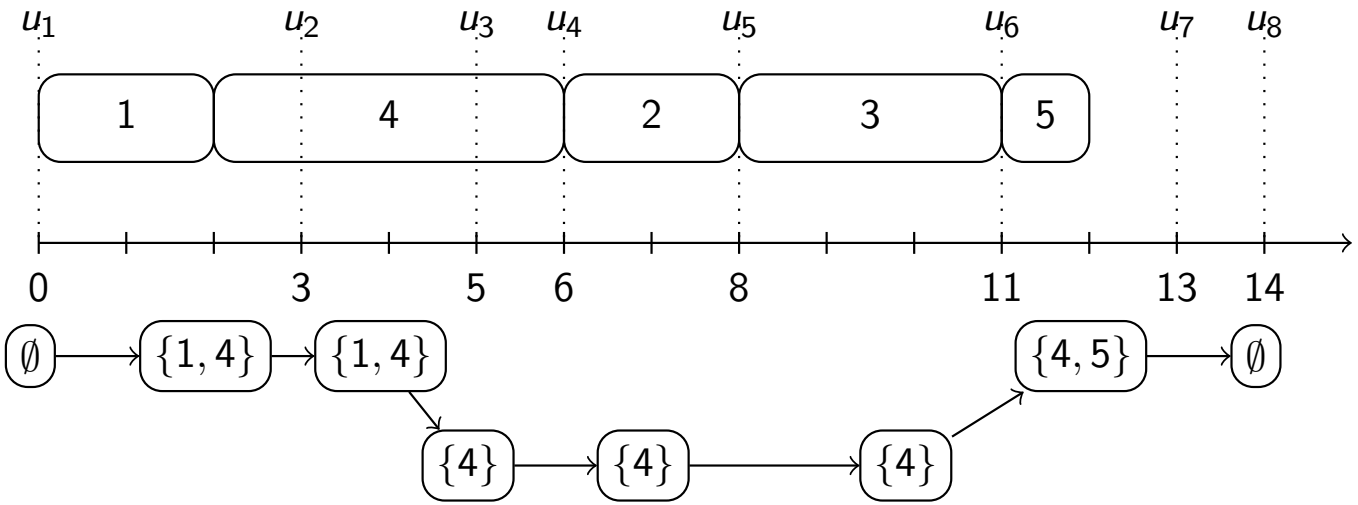


Single machine with pathwidth μ

Removing jobs that must end before u_{k+1}



Single machine with pathwidth μ



State L_k of stage k

L_k Set of jobs started before u_{k+1} and with deadline $> u_{k+1}$ Evaluation of a state: minimum makespan of a feasible schedule of L_k plus jobs with deadlines $\leq u_{k+1}$.

Results

Remark: nb of jobs i with $r_i \leq u_k < u_{k+1} < d_i$ is bounded by $\mu + 1$
In a state L_k of stage k, L_k is a subset of a set of size at most $\mu + 1$.

Theorem

- *The number of states is* $\mathcal{O}(n \cdot 2^{\mu+2})$.
- *The number of arcs is* $\mathcal{O}(n \cdot 2^{2(\mu+2)})$
- *Finding the optimal schedule is FPT with respect to* μ

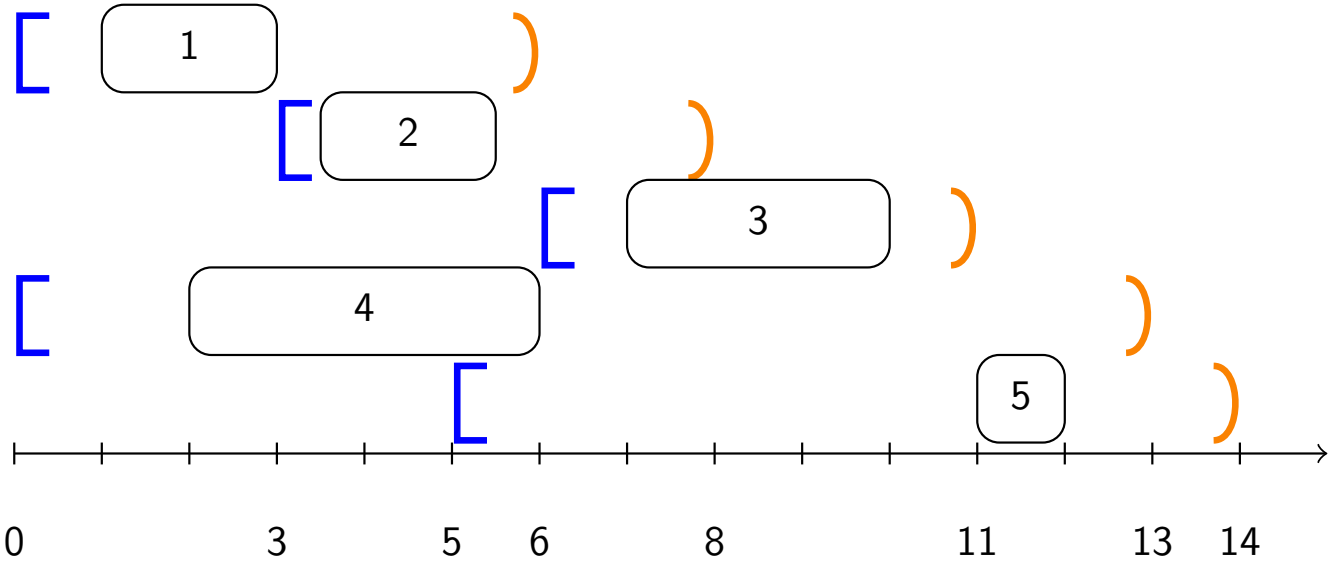
q-proper level parameter

Proper set Π_i of a job i

Set of jobs whose interval strictly encloses $[r_i, \bar{d}_i)$

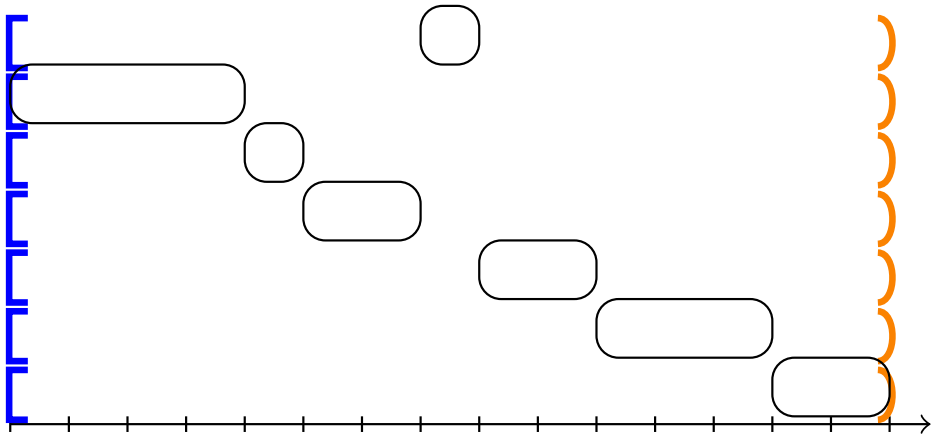
q-proper parameter

$q = \max_i |\Pi_i|$

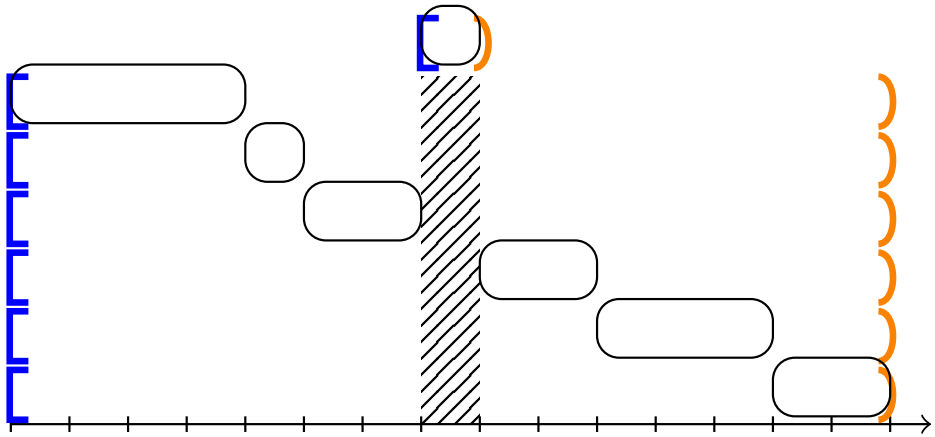


$\Pi_1 = \Pi_2 = \Pi_3 = \emptyset,$
$\Pi_2 = \{4\},$
$\Pi_3 = \{4, 5\},$
$q = 2$

q-proper level motivation



$\mu = n - 1$, $q = 0$, Easy problem



$\mu = n - 1$, $q = n - 1$, NP-hard problem

Assumptions

Consistency of time windows

We assume release times and deadlines consistent with precedence constraints: if $i \to j$ then $r_j \geq r_i + p_i \wedge \bar{d}_i \leq \bar{d}_j - p_j$

Indexation

We assume jobs are indexed such that : if $i < j$ then

$$\bar{d}_i < \bar{d}_j \vee ((\bar{d}_i = \bar{d}_j) \wedge (r_i \leq r_j)$$

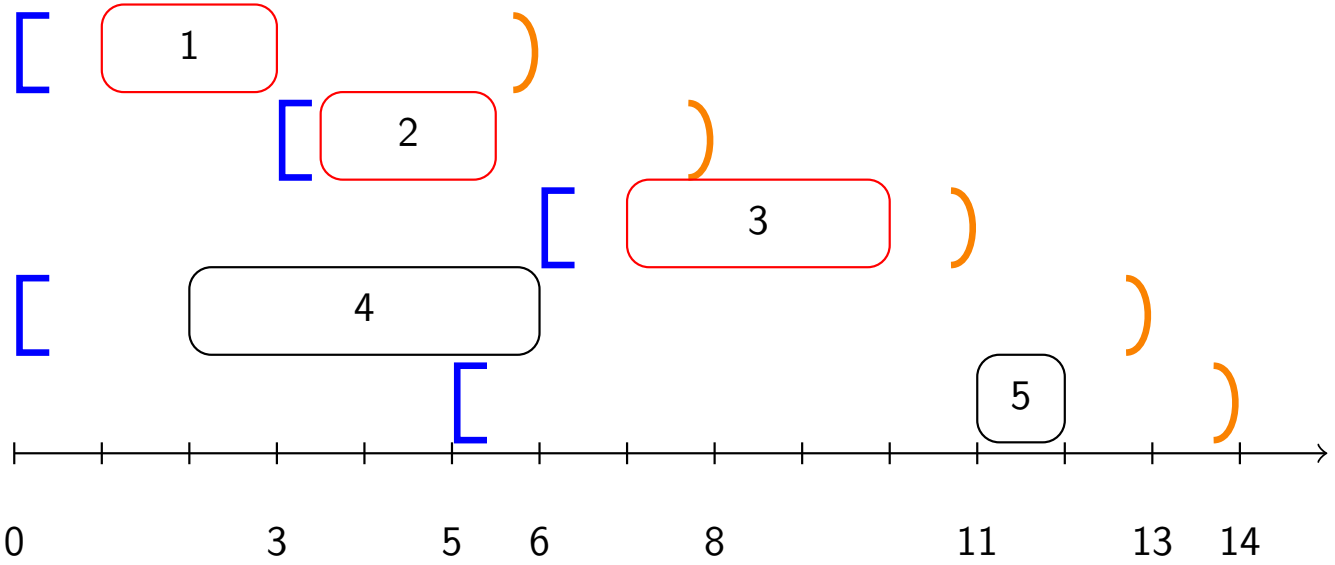
Summits

Definition

A summit is a job i such that the time window $[r_i, \bar{d}_i)$ does not stricly enclose another time window:

$$\forall j \neq i, i \notin \Pi_j$$

Increasing sequence $s_1 < \ldots < s_K$ of summits



Summits

$s_1 = 1$

$s_2 = 2$

$s_3 = 3$

Structure of a schedule : a dominance rule

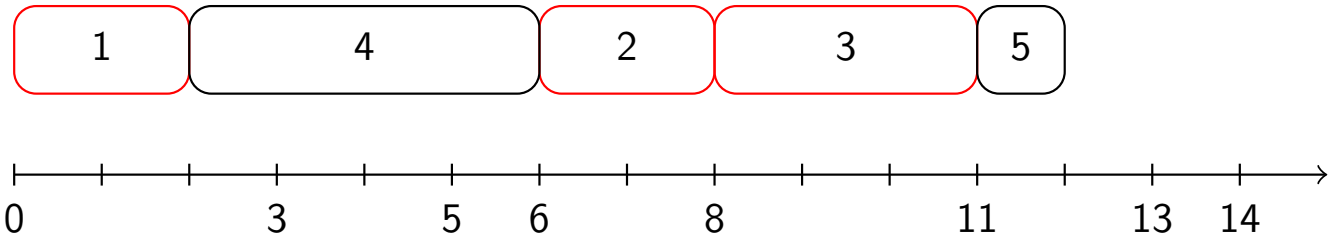
Dominance rule lemma

If there is a feasible schedule then there is a minimum makespan schedule σ of the form $T_1 s_1 \dots T_N s_N T_{N+1}$ where:

a. every job in sequence T_1 is in set Π_{s_1},

b. for all k in $[2, N]$ every job in sequence T_k is in set $\Pi_{s_{k-1}} \cup \Pi_{s_k}$, and

c. every job in sequence T_{N+1} is in set Π_{s_N}.



$T_1 = \emptyset, T_2 = \{4\}, T_3 = \emptyset, T_4 = \{5\}$

Dominance rule detailed

Lemma

In a dominant schedule $T_1 s_1 \dots T_N s_N T_{N+1}$,

- *All jobs with index* $< s_k$ *are performed before* s_k
- $T_k = C_{k-1} A_k B_k$

Where

- $C_{k-1} = T_k - \Pi_{s_k}$ *scheduled by increasing index*
- $A_k = T_k \cap \Pi_{s_{k-1}} \cap \Pi_{s_k}$ *scheduled by increasing index*
- $B_k = T_k - \Pi_{s_{k-1}}$ *scheduled by increasing* r_i

Inspired by a similar decomposition for problems without prec:

J. Erschler, Gérard Fontan, Colette Mercé, and François Roubellat. *A new dominance concept in scheduling n jobs on a single machine with ready times and due dates.* Oper. Res., 31(1):114–127, 1 (1983)

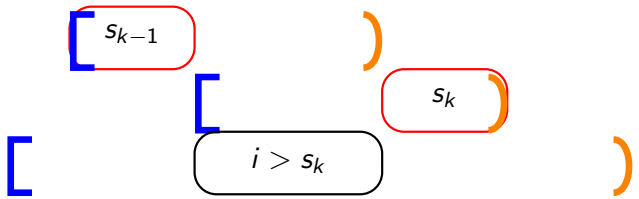
Dynamic programming

- A state of stage k records the minimum information about jobs performed before summit s_k
- Only jobs with index $> s_k$ are useful to store

Lemma

$$L_k = \bigcup_{1 \leq h \leq k} T_h \cap \{s_k + 1 \ldots, n\}$$

$$L_k \subset \Lambda_k = \bigcup_{1 \leq h \leq k} \Pi_{s_h} \cap \{s_k + 1 \ldots, n\} = \Pi_{s_k}$$



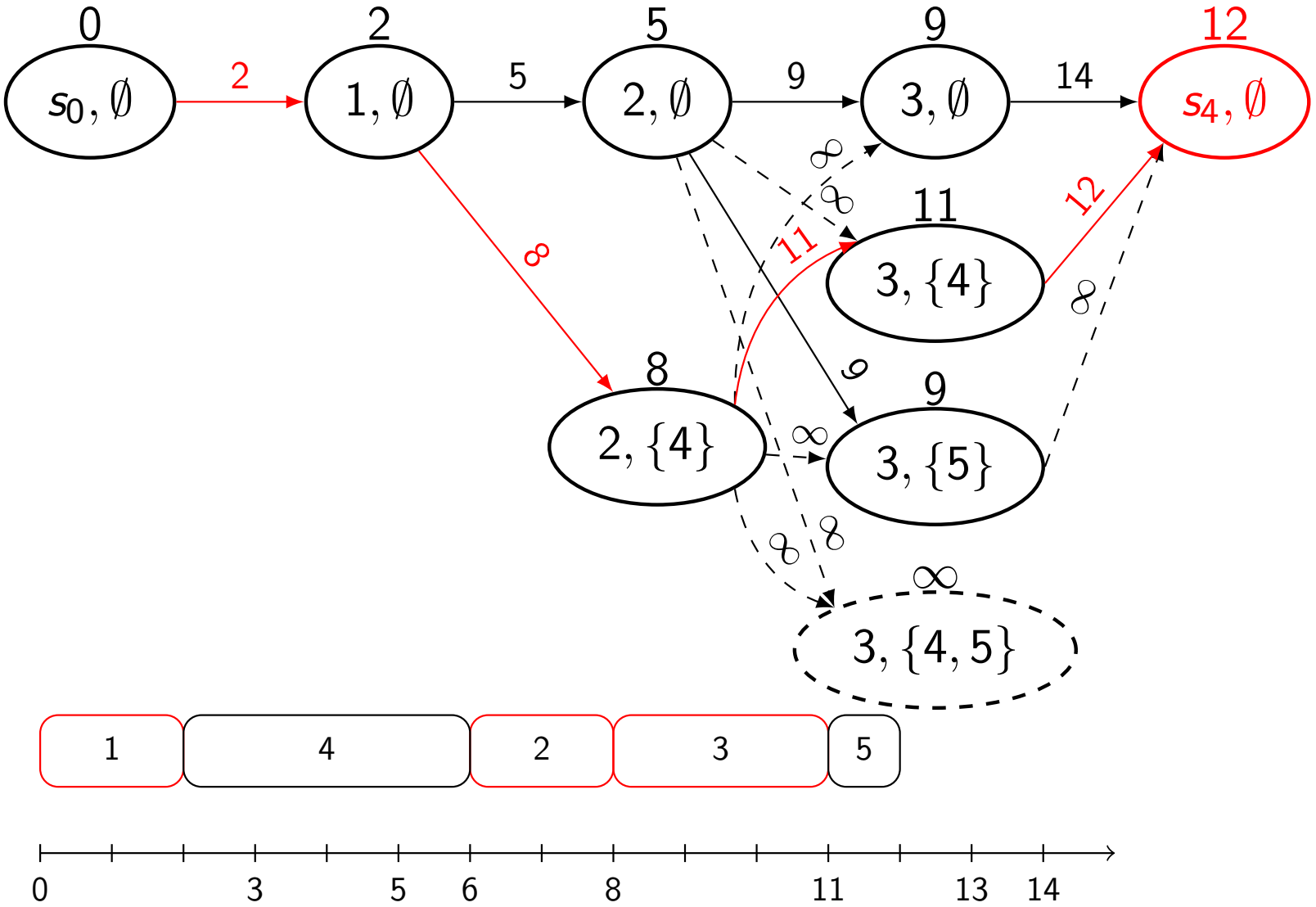
Results

- Cardinality of the set Λ_k bounded by q
- Evaluation of a state L_k of stage k : minimum makespan of a schedule of $\{1, \ldots, s_k\} \cup L_k$ ending by s_k.
- This evaluation can be performed knowing the evaluation of states of the lower stages.

Theorem

- *The number of states is* $\mathcal{O}(n2^q)$
- *Solving* $1|prec, r_i, \bar{d}_i|C_{max}|$ *is FPT with respect to* q.

State graph


0
$s_0, \emptyset$
2
2
$1, \emptyset$
5
5
$2, \emptyset$
9
9
$3, \emptyset$
14
12
$s_4, \emptyset$
8
∞
∞
11
$3, \{4\}$
12
11
∞
8
$2, \{4\}$
∞
9
$3, \{5\}$
∞
∞
∞
$3, \{4, 5\}$
1
4
2
3
5
0
3
5
6
8
11
13
14

Relations between parameters

Lemma

In any instance of $1|prec, r_i, \bar{d}_i|*, \quad q \leq \mu.$

Corollary

A FPT algorithm for q there is a FPT algorithm for parameter μ. If a problem is para-NP-hard for parameter μ, it will be also for parameter q

Lemma

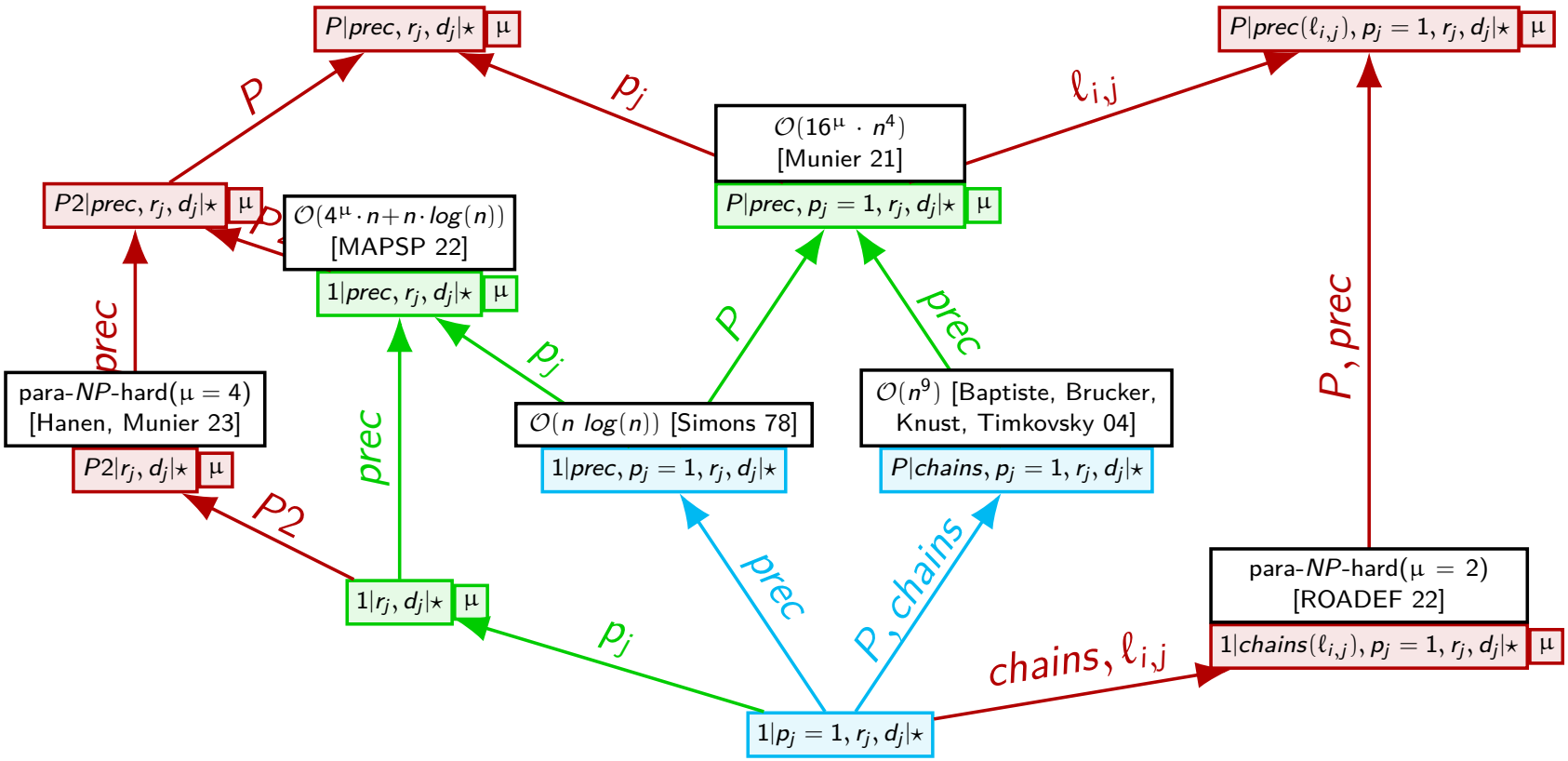
In any feasible instance of $1|prec, r_i, \bar{d}_i|* \ \mu \leq 2\sigma$ *(slack);*

Corollary

A FPT algorithm for μ is a FPT algorithm for parameter σ.

Map

Parameter μ



Concluding remarks

- We provided two FPT algorithms for $1|prec, r_i, \bar{d}_i|C_{max}$ with parameters μ and q.
- Currently studying $P|tree, p_i = 1, r_i, \bar{d}_i|*$ with respect to q
- Building maps for scheduling relevant parameters

Challenging questions for the study of parameterized complexity of scheduling problems :

- Are FPT algorithms efficient in practice?
- What about Kernelization techniques?

Maher Mallem, Claire Hanen, Alix Munier-Kordon:
Parameterized Complexity of Single-machine Scheduling with Precedence, Release Dates and Deadlines, MAPSP2022
A new structural parameter on single machine scheduling with release dates and deadlines, ISCO 2024