High Multiplicity Single Machine Scheduling with Forbidden Start and Completion times

Nadia Brauner
Michaël Gabay
Christophe Rapine

3 July 2013

Sommaire

Chemical experiments planning

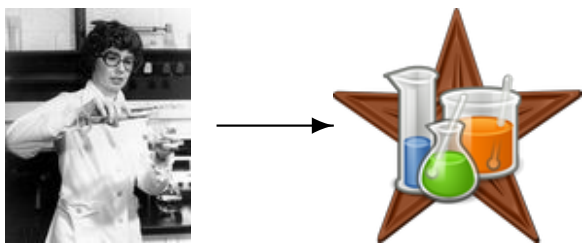
A chemist launches and stops experiments.

Chemical experiments planning

A chemist launches and stops experiments.

Chemical experiments planning

A chemist launches and stops experiments.

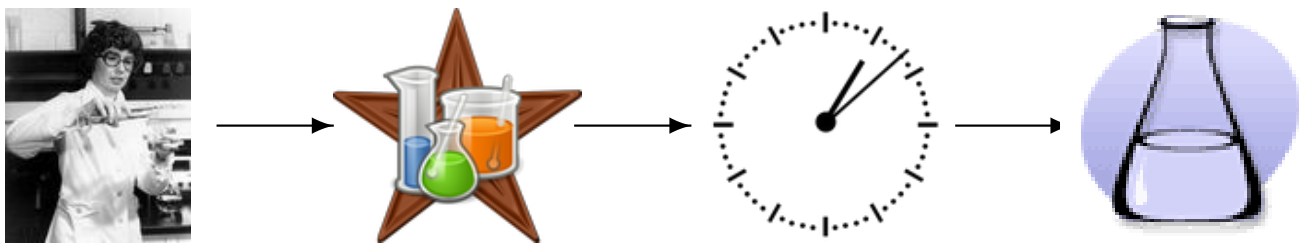


Chemical experiments planning

A chemist launches and stops experiments.

Chemical experiments planning

A chemist launches and stops experiments.



Chemical experiments planning

A chemist launches and stops experiments.

Chemical experiments planning

A chemist launches and stops experiments.

The chemist might be unavailable...

Chemical experiments planning

A chemist launches and stops experiments.

The chemist might be unavailable...

$\Rightarrow$ Resource unavailability

Single machine scheduling problem with an additionnal resource

$$1 \left| \mathrm{FSE} \right| C_{\max}$$

Single machine scheduling problem with an additionnal resource

$$1\left|\mathrm{FSE}\right|C_{\max}$$

n tasks: $(p_i)_{i=1,\dots,n}$

Single machine scheduling problem with an additionnal resource

$$1\left|\mathrm{FSE}\right|C_{\max}$$

n tasks: $(p_i)_{i=1,\dots,n}$

K unavailability dates: $\Gamma = \{\gamma_1, \dots, \gamma_K\}$ $\gamma_1 < \cdots < \gamma_K$

γ_i is an FSE: no task can start or end on time γ_i.

Single machine scheduling problem with an additionnal resource

$$1\left|\mathrm{FSE}\right|C_{\max}$$

n tasks: $(p_i)_{i=1,\dots,n}$

K unavailability dates: $\Gamma = \{\gamma_1, \dots, \gamma_K\}$ $\gamma_1 < \cdots < \gamma_K$

γ_i is an FSE: no task can start or end on time γ_i.

Does an idle-free schedule exist ?

Machines unavailability [Schmidt, 2000]

Operator non-availability (ONA) [Brauner et al., 2008, Rapine et al., 2012]: $\mathcal{NP}$-hard, $\notin \mathcal{APX}$

Forbidden start times [Billaut and Sourd., 2009], $1|FS|C_{\max}$:

- Strongly $\mathcal{NP}$-hard
- $\notin \mathcal{APX}$
- If $N \geq 2K(K+1)$ then there is an idle-free schedule

$1|\text{FSE}|C_{\max}$ [Rapine et Brauner, 2010]:

- Strongly $\mathcal{NP}$-hard
- $\notin \mathcal{APX}$
- If $N > K$ (*large diversity*) then there is an idle-free schedule
- We can compute an idle-free schedule in $\mathcal{O}(K^3 n)$ time

$$1\left|\text{FSE}, HM\right|C_{\max}$$

N kinds of tasks: $(p_i, m_i)_{i=1,\ldots,n}$ $\quad p_1 < \cdots < p_N$

$$n = \sum_{i=1}^{K} m_i$$

K unavailability dates: $\Gamma = \{\gamma_1, \ldots, \gamma_K\}$ $\quad \gamma_1 < \cdots < \gamma_K$

$$1\left|\text{FSE}, HM\right|C_{\max}$$

N kinds of tasks: $(p_i, m_i)_{i=1,\ldots,n}$ $\quad p_1 < \cdots < p_N$

$$n = \sum_{i=1}^{K} m_i$$

K unavailability dates: $\Gamma = \{\gamma_1, \ldots, \gamma_K\}$ $\quad \gamma_1 < \cdots < \gamma_K$

Input:

$$1\left|\text{FSE}, HM\right|C_{\max}$$

N kinds of tasks: $(p_i, m_i)_{i=1,\ldots,n}$ $\quad p_1 < \cdots < p_N$

$$n = \sum_{i=1}^{K} m_i$$

K unavailability dates: $\Gamma = \{\gamma_1, \ldots, \gamma_K\}$ $\quad \gamma_1 < \cdots < \gamma_K$

Input: $2N + K$ integers
Size: $|\mathcal{I}| =$

$$1\left|\text{FSE}, HM\right|C_{\max}$$

N kinds of tasks: $(p_i, m_i)_{i=1,\dots,n}$ $\quad p_1 < \cdots < p_N$

$$n = \sum_{i=1}^{K} m_i$$

K unavailability dates: $\Gamma = \{\gamma_1, \dots, \gamma_K\}$ $\quad \gamma_1 < \cdots < \gamma_K$

Input: $2N + K$ integers
Size: $|\mathcal{I}| = \mathcal{O}((N+K)(\log(n) + \log(p_N)))$

High Multiplicity encoding

$$1\left|\text{FSE}, HM\right|C_{\max}$$

N kinds of tasks: $(p_i, m_i)_{i=1,\dots,n}$ $\quad p_1 < \cdots < p_N$

$$n = \sum_{i=1}^{K} m_i$$

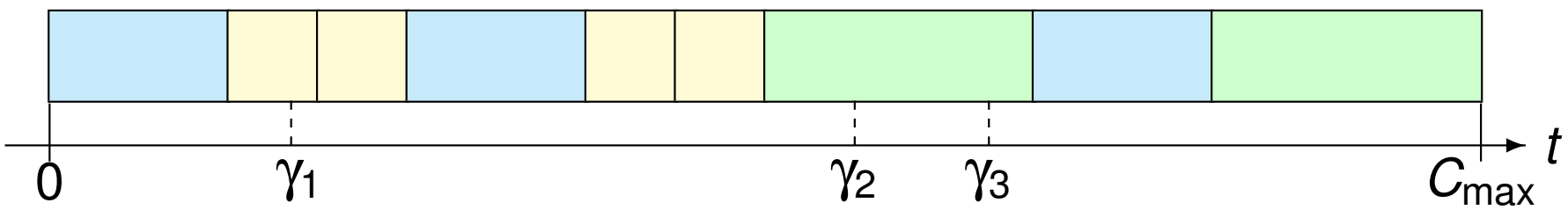
K unavailability dates: $\Gamma = \{\gamma_1, \dots, \gamma_K\}$ $\quad \gamma_1 < \cdots < \gamma_K$

Input: $2N + K$ integers
Size: $|\mathcal{I}| = \mathcal{O}((N+K)(\log(n) + \log(p_N)))$

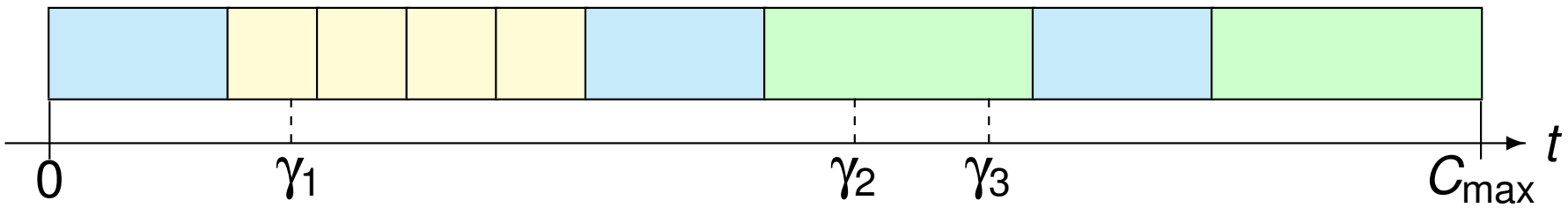
High Multiplicity encoding

n is exponential in $|\mathcal{I}|$

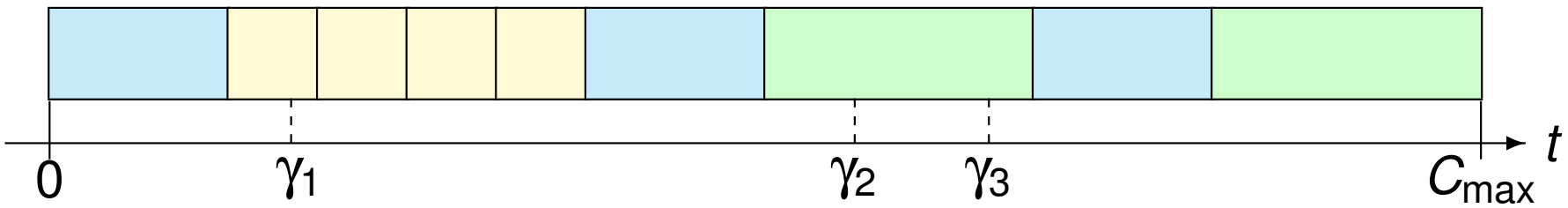
Does a polynomial certificate exist ?



Does a polynomial certificate exist ?



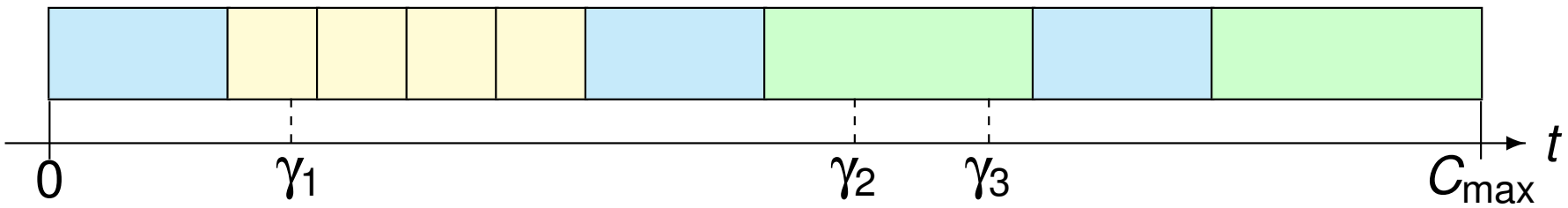
Does a polynomial certificate exist ?



Certificate:

(1,0,0,(yellow,d_1)), (1,3,0,(green,d_2)), (0,0,0,(green,d_2)))

Does a polynomial certificate exist ?



Certificate:

(1,0,0,(yellow,d_1)), (1,3,0,(green,d_2)), (0,0,0,(green,d_2)))

Size: $\mathcal{O}(KN(\log(n)+\log(p_N))) = \mathcal{O}(P(|\mathcal{I}|))$

Large diversity: $K < N$

Theorem (Rapine, Brauner, 2010)

If $K < N$, we can compute an idle-free schedule in time $\mathcal{O}(K^3 n)$

Large diversity: $K < N$

Theorem (Rapine, Brauner, 2010)

If $K < N$, we can compute an idle-free schedule in time $\mathcal{O}(K^3 n)$

Theorem

Under HM encoding, if $K < N$, we can compute an idle-free schedule in time $\mathcal{O}(KN + K^4)$.

Large diversity: $K < N$

Theorem (Rapine, Brauner, 2010)

If $K < N$, we can compute an idle-free schedule in time $\mathcal{O}(K^3 n)$

Theorem

Under HM encoding, if $K < N$, we can compute an idle-free schedule in time $\mathcal{O}(KN + K^4)$.

Idea

Schedule additionnal tasks first and spare $K + 1$ different tasks

Large diversity: $K < N$

Algorithm and proof ideas

- F contains 1 task from the $K+1$ largest types
- R contains the remaining tasks

Large diversity: $K < N$

Algorithm and proof ideas

- F contains 1 task from the $K+1$ largest types
- R contains the remaining tasks

Schedule tasks from R:

- By decreasing order of processing times
- Until γ_1 is reached

$i = N$
while $t + m_i p_i < \gamma_1$ (and $i > 0$) **do**
  $t \leftarrow t + m_i p_i$
  $i = i - 1$
end while
$t \leftarrow t + p_i\left(\left\lceil \frac{\gamma_1 - t}{p_i} \right\rceil - 1\right)$

Large diversity: $K < N$

By the end of this first step

- If R is empty ($i = 0$), schedule remaining tasks from F: large diversity, $\mathcal{O}(K^4)$.
- If R is not empty, use some tasks from F to go beyond γ_1. Keep a large diversity instance on the remaining part.

Large diversity: $K < N$

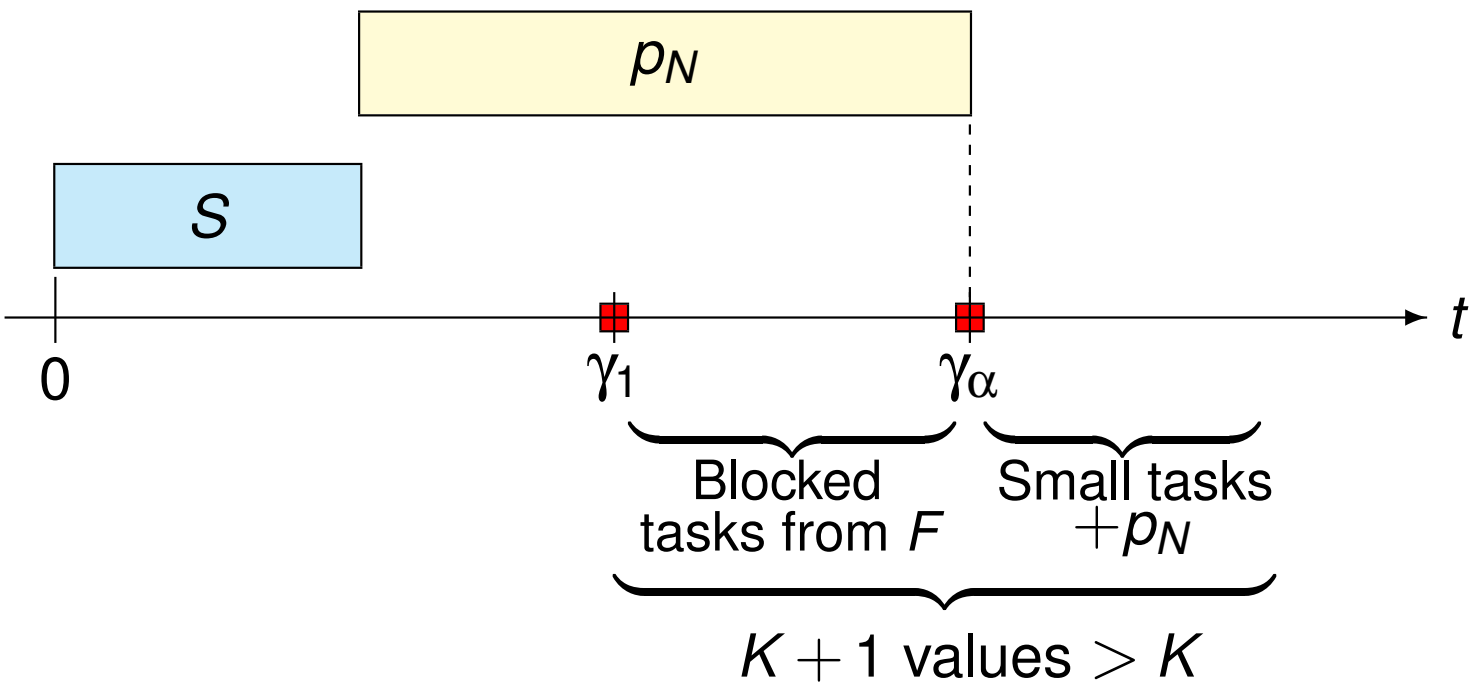
$R \neq \emptyset$

- A task from F goes beyond γ_1 and is not blocked $\Rightarrow$ OK
  $\rightsquigarrow t + p_j > \gamma_1$ and $t + p_j \notin \Gamma \quad \Rightarrow \quad t \leftarrow t + p_j$

Large diversity: $K < N$

$R \neq \emptyset$

- A task from F goes beyond γ_1 and is not blocked $\Rightarrow$ OK
  $\rightsquigarrow t + p_j > \gamma_1$ and $t + p_j \notin \Gamma \quad \Rightarrow \quad t \leftarrow t + p_j$
- All tasks in F are blocked or finish before γ_1



Large diversity: $K < N$

Let $N - K \leq I < N$ s.t. $t + p_N + p_I \notin \Gamma$

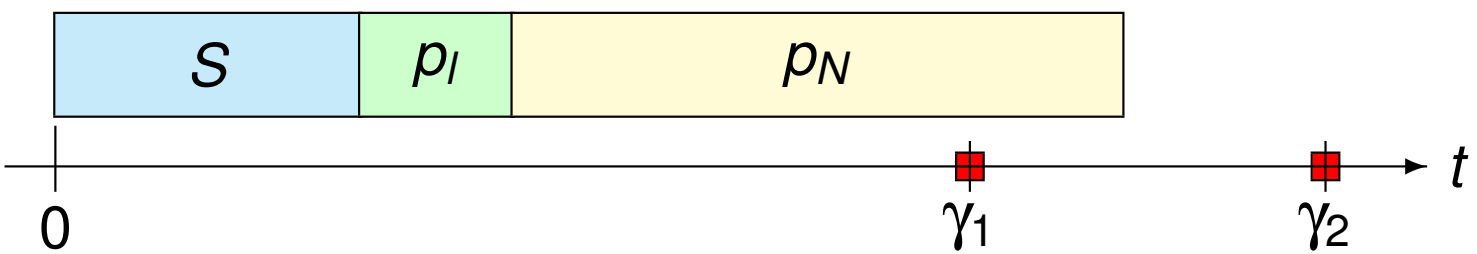
- If $t + p_N + p_I > \gamma_2 \Rightarrow$ OK (we still have large diversity)

Large diversity: $K < N$

Let $N - K \leq l < N$ s.t. $t + p_N + p_l \notin \Gamma$

- If $t + p_N + p_l > \gamma_2 \Rightarrow$ OK (we still have large diversity)
- Otherwise, $i = N$ $\qquad (i \neq N \Rightarrow t + p_N \geq \gamma_2)$
  $\rightsquigarrow$ R contains a task p_N

Use a small task from F with p_N from R to go beyond γ_1.



Large diversity: $K < N$

Theorem

Under HM encoding, if $K < N$, we can compute an idle-free schedule in time $\mathcal{O}(KN + K^4)$.

Improves $\mathcal{O}(K^3 n)$ complexity.

$K \geq N$?

$1|\text{FSE}, HM|C_{\max}$ is $\mathcal{NP} - complete$
$\Rightarrow$ No polynomial algorithm for the general case (unless $\mathcal{P} = \mathcal{NP}$)

What is the complexity for fixed K ?

Input size: $\mathcal{O}((N+K)(\log(n)+\log(p_N)))$

$K \geq N$?

$1|\text{FSE}, HM|C_{\max}$ is $\mathcal{NP} - complete$
$\Rightarrow$ No polynomial algorithm for the general case (unless $\mathcal{P} = \mathcal{NP}$)

What is the complexity for fixed K ?

Input size: $\mathcal{O}(\log(n) + \log(p_N))$

$K \geq N$?

$1|\text{FSE}, HM|C_{\max}$ is $\mathcal{NP} - complete$
$\Rightarrow$ No polynomial algorithm for the general case (unless $\mathcal{P} = \mathcal{NP}$)

What is the complexity for fixed K ?

Input size: $\mathcal{O}(\log(n) + \log(p_N))$

For $K < N$: $\mathcal{O}(KN + K^4) \rightsquigarrow \mathcal{O}(N)$

$K \geq N$?

K fixed ?

Theorem (Lenstra, 1983)

Let $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, n fixed, we can decide in polynomial time whether: $\{x \in \mathbb{Z}^n | Ax \leq b\}$ is empty.

K fixed ?

Theorem (Lenstra, 1983)

Let $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$, n fixed, we can decide in polynomial time whether: $\{x \in \mathbb{Z}^n | Ax \leq b\}$ is empty.

Theorem (Eisenbrand, 2003)

An integer program of binary encoding length s in fixed dimension, which is defined by a fixed number of constraints, can be solved in $\mathcal{O}(s)$ arithmetic operations on rational numbers of binary encoding $\mathcal{O}(s)$.

K fixed ?

Idea

If $K < N$ use the prefix algorithm, otherwise, solve an IP with a fixed number of variables and constraints.

K fixed ?

Idea

If $K < N$ use the prefix algorithm, otherwise, solve an IP with a fixed number of variables and constraints.

Problems:

- Model idle times
- Linearize lots of equations
- Be careful with big M !

Theorem (Rapine, Brauner, 2010)

Any list scheduling algorithm delivers a schedule with makespan at most $\sum_{i=1}^{N} m_i p_i + 2K$.

Theorem (Rapine, Brauner, 2010)

Any list scheduling algorithm delivers a schedule with makespan at most $\sum_{i=1}^{N} m_i p_i + 2K$.

Model idle times as additionnal tasks with processing times $1, \ldots, 2K$.

Ensure we can skip all FSE : add an additionnal task with processing time $\gamma_K + 1$.

Find an idle-free schedule finishing the last *real* task asap.

Variables

m_{ij}	:	number of tasks of type i completed by time γ_j
W_j	:	total work completed by time γ_j. Short-hand for $\sum_i p_i m_{ij}$
S_{jk}	$= 1$	if a task covers exactly the instants γ_j till γ_{k-1}
x_{ij}	$= 1$	if a task of type i covers the instant γ_j
		and this task does not cover the previous FSE instant.
y_j	$= 1$	if all the (real) tasks are completed by time γ_j
$C_{\max}$	:	makespan of the schedule

Variables

m_{ij}	:	number of tasks of type i completed by time γ_j
W_j	:	total work completed by time γ_j. Short-hand for $\sum_i p_i m_{ij}$
S_{jk}	$= 1$	if a task covers exactly the instants γ_j till γ_{k-1}
x_{ij}	$= 1$	if a task of type i covers the instant γ_j
		and this task does not cover the previous FSE instant.
y_j	$= 1$	if all the (real) tasks are completed by time γ_j
$C_{\max}$	:	makespan of the schedule

Number of variables $= \mathcal{O}(K^2)$

Constraints

- S_{jk} defines a $1-(K+1)$ path
- Work W_j must be completed by time γ_j
- If some consecutive FSE are covered by the same task, no task can be scheduled during their interval (big M)
- Covering tasks finishing dates depend on covered FSE
- Decides whichs tasks covers FSE dates
- If a task covers some FSE , count it.
- Assign all real tasks
- Set $y_j = 1$ iff all real tasks are completed before γ_j
- $C_{\max}$ is equal the first W_j such that $y_j = 1$ (big M)

Constraints

- S_{jk} defines a $1-(K+1)$ path
- Work W_j must be completed by time γ_j
- If some consecutive FSE are covered by the same task, no task can be scheduled during their interval (big M)
- Covering tasks finishing dates depend on covered FSE
- Decides whichs tasks covers FSE dates
- If a task covers some FSE , count it.
- Assign all real tasks
- Set $y_j = 1$ iff all real tasks are completed before γ_j
- $C_{\max}$ is equal the first W_j such that $y_j = 1$ (big M)

$\mathcal{O}(K^2)$ constraints and M bounded by $\sum_{i=1}^{N} m_i p_i + 2K$

Ideas

Use previous results to make the model:

- Ensure there is an idle-free schedule ($N > K$)
- Ensure we have enough tasks to build all possible combinations of idle times (at most $2K$)

And number of constraints and variables are bounded by $P(K)$

Ideas

Use previous results to make the model:

- Ensure there is an idle-free schedule ($N > K$)
- Ensure we have enough tasks to build all possible combinations of idle times (at most $2K$)

And number of constraints and variables are bounded by $P(K)$

Theorem

$1|\text{FSE}, HM|C_{\max}$ can be solved in linear time for fixed K.

What's next ?...

- Reduce dimension (e.g. only $K+1$ different artificial tasks are needed)
- Polynomial algorithm for $N = K$?
- Non parametrized complexity for some subcases
- Arithmetical approach ?

Questions ?