Extending Smith's Rule with Task Mandatory Parts and Release Dates

Bonnin Camille *,o Nattaf Margaux* Malapert Arnaud°
Espinouse Marie-Laure*

*Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France
{camille.bonnin, margaux.nattaf, marie-laure.espinouse}@grenoble-inp.fr

°Université Côte d'Azur, CNRS, I3S, France

arnaud.malapert@univ-cotedazur.fr

PMS 2022, April 6-7, Ghent

1. Introduction

Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$

Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$

Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$

Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$

Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$

Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$

Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

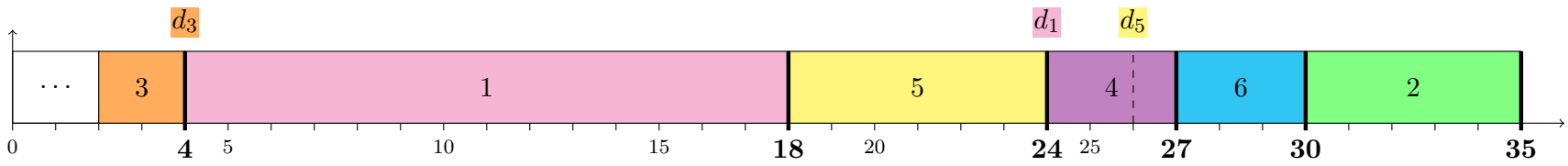
Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$



Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

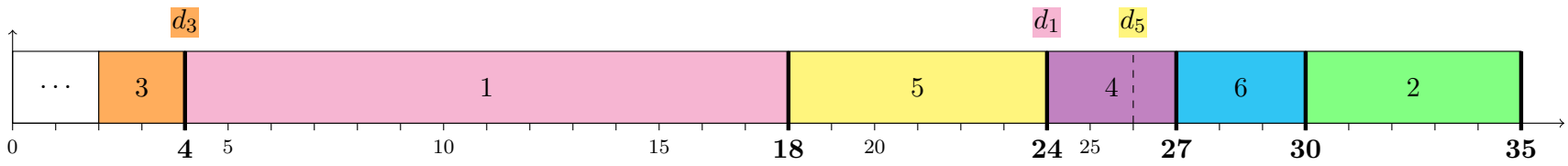
Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$



Problem

Note: here and in the following, we will use d_j instead of $\tilde{d}_j$ for deadlines

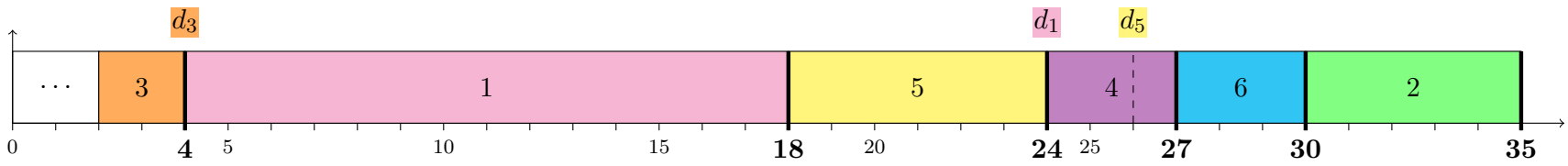
Long term objective: improve CP integration of $\sum C_j$ objective

- General problem (even without *prec*): $1|r_j; d_j|\sum C_j$
- Polynomial relaxation with a list algorithm: $1|r_j; pmtn|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly

id_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 138$$

$$(4 + 18 + 24 + 27 + 30 + 35)$$



Context

Literature

- $1||\sum C_j$: Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn|\sum C_j$: Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006] $\Rightarrow 1^{st}$ Lower Bound (LB)

Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of $1|r_j; d_j|\sum C_j$ respecting MPs
- Long term objective: improve CP solving of $1|r_j; d_j|\sum C_j$

Context

Literature

- $1||\sum C_j$: Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn|\sum C_j$: Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006] $\Rightarrow 1^{st}$ Lower Bound (LB)

Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of $1|r_j; d_j|\sum C_j$ respecting MPs
- Long term objective: improve CP solving of $1|r_j; d_j|\sum C_j$

Context

Literature

- $1||\sum C_j$: Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn|\sum C_j$: Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006] $\Rightarrow 1^{st}$ Lower Bound (LB)

Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of $1|r_j; d_j|\sum C_j$ respecting MPs
- Long term objective: improve CP solving of $1|r_j; d_j|\sum C_j$

Context

Literature

- $1||\sum C_j$: Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn|\sum C_j$: Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006] $\Rightarrow 1^{st}$ Lower Bound (LB)

Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of $1|r_j; d_j|\sum C_j$ respecting MPs
- Long term objective: improve CP solving of $1|r_j; d_j|\sum C_j$

Context

Literature

- $1||\sum C_j$: Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn|\sum C_j$: Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006] $\Rightarrow 1^{st}$ Lower Bound (LB)

Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of $1|r_j; d_j|\sum C_j$ respecting MPs
- Long term objective: improve CP solving of $1|r_j; d_j|\sum C_j$

Context

Literature

- $1||\sum C_j$: Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn|\sum C_j$: Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006] $\Rightarrow 1^{st}$ Lower Bound (LB)

Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of $1|r_j; d_j|\sum C_j$ respecting MPs
- Long term objective: improve CP solving of $1|r_j; d_j|\sum C_j$

Context

Literature

- $1||\sum C_j$: Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn|\sum C_j$: Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006] $\Rightarrow 1^{st}$ Lower Bound (LB)

Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of $1|r_j; d_j|\sum C_j$ respecting MPs
- Long term objective: improve CP solving of $1|r_j; d_j|\sum C_j$

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

eft_j = earliest finishing time of task j
lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

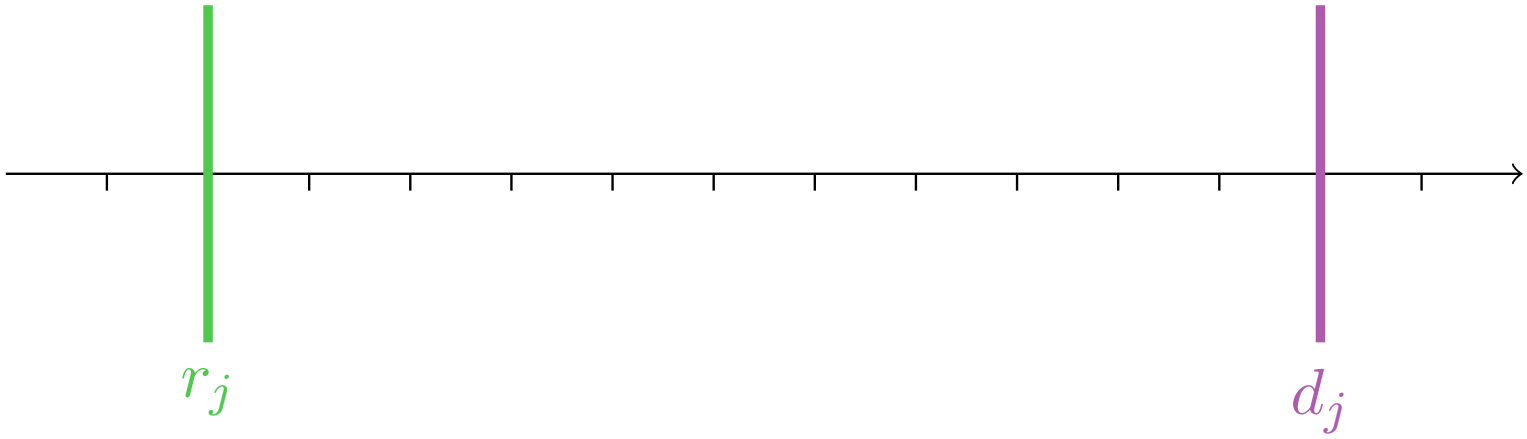
Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

eft_j = earliest finishing time of task j
lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

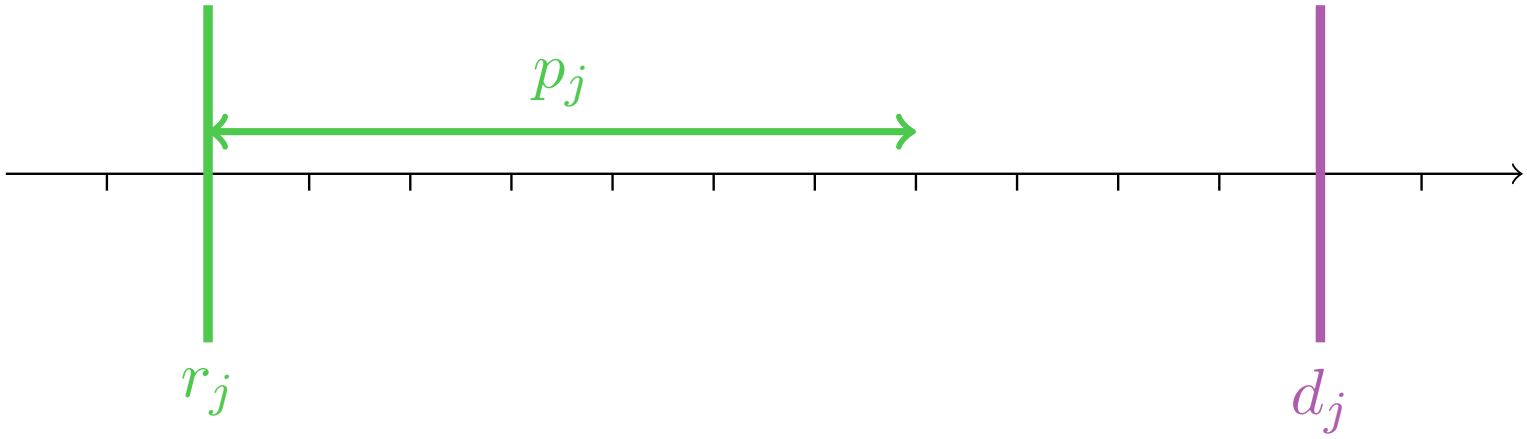


eft_j = earliest finishing time of task j
lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

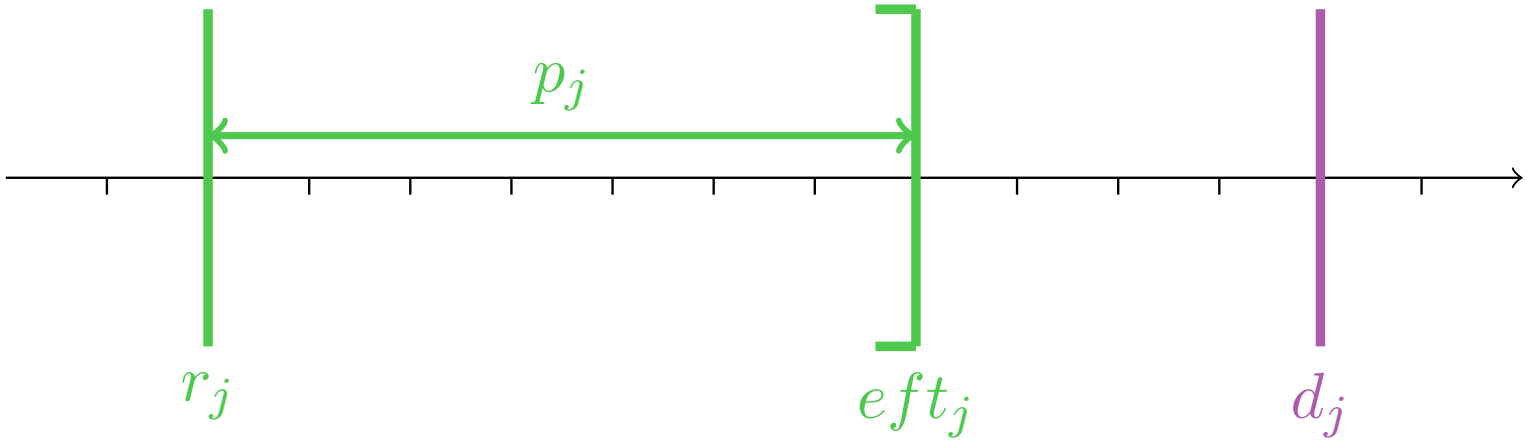


eft_j = earliest finishing time of task j
lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



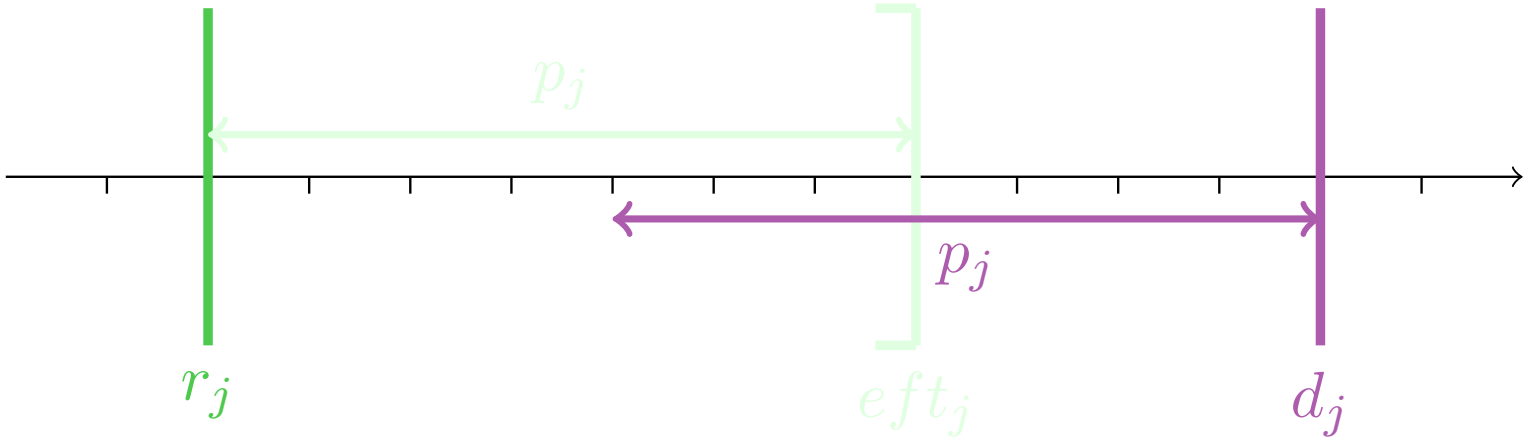
eft_j = earliest finishing time of task j

lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



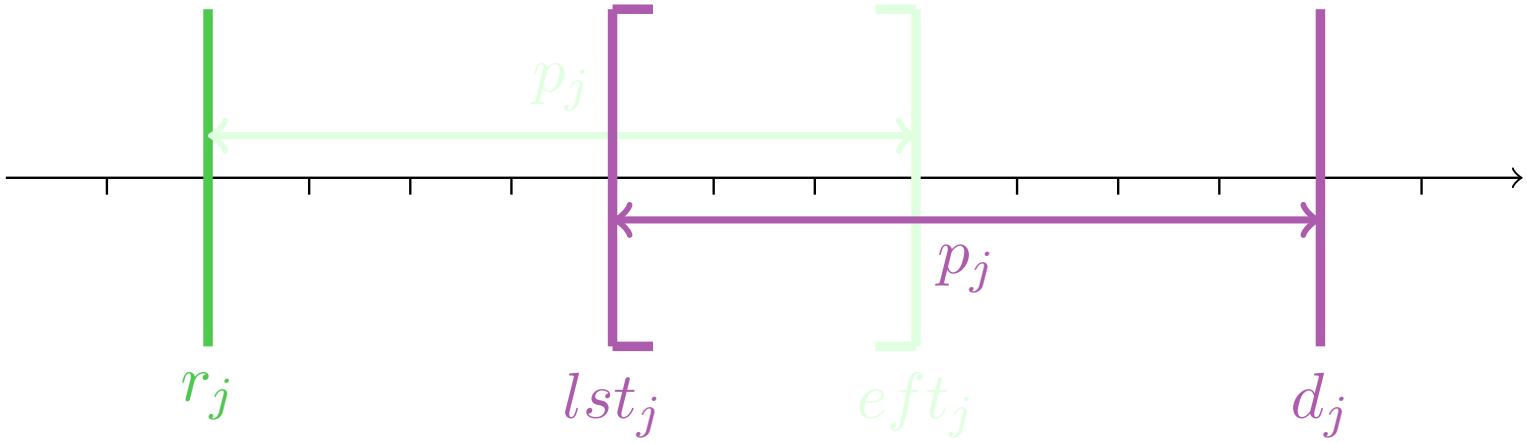
eft_j = earliest finishing time of task j

lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

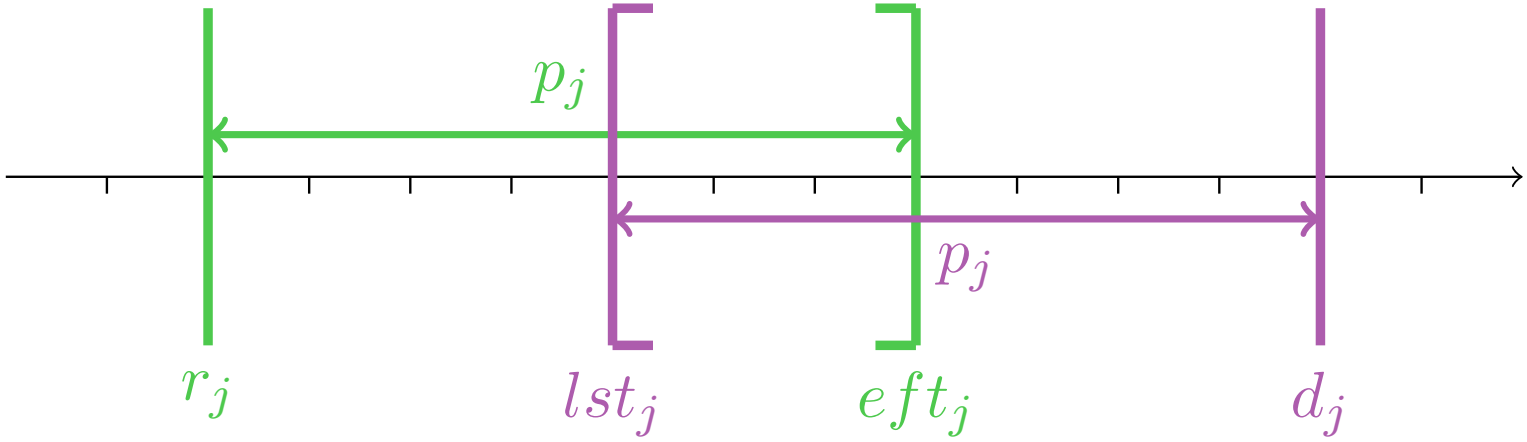


eft_j = earliest finishing time of task j
lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

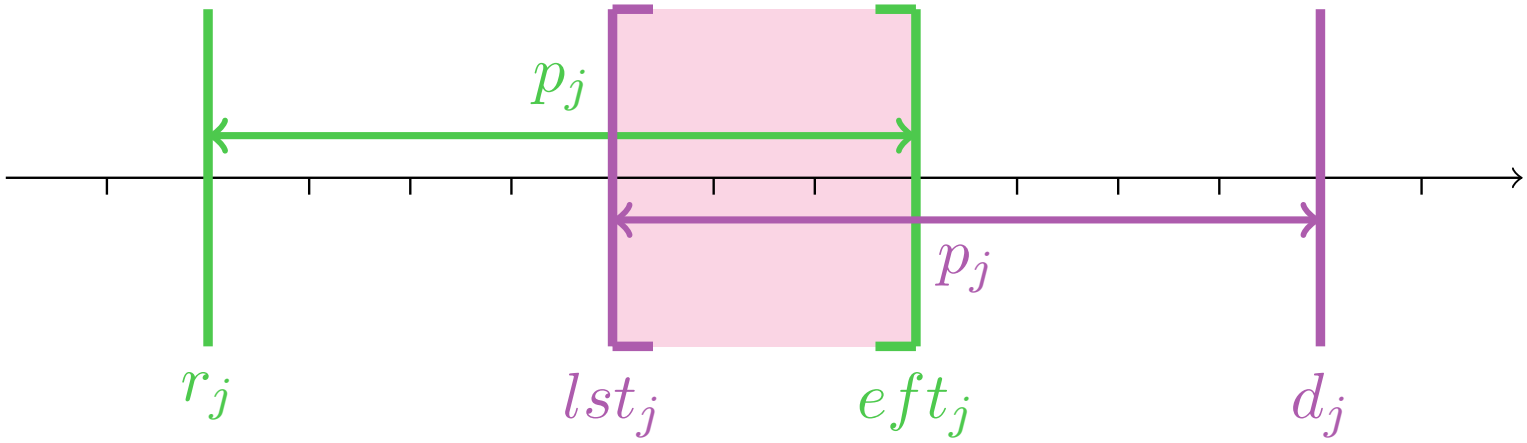


eft_j = earliest finishing time of task j
lst_j = latest starting time of task j

Mandatory Part (MP)

Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



eft_j = earliest finishing time of task j
lst_j = latest starting time of task j

2. Modified Smith Rule with MPs

Modified Smith Rule (MSR)

Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$

Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$

Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

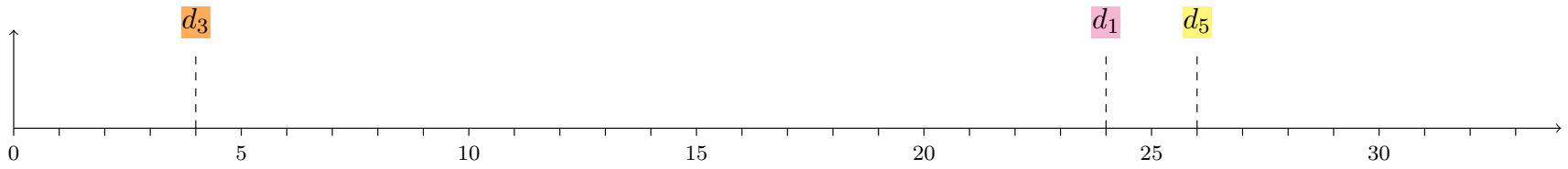
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

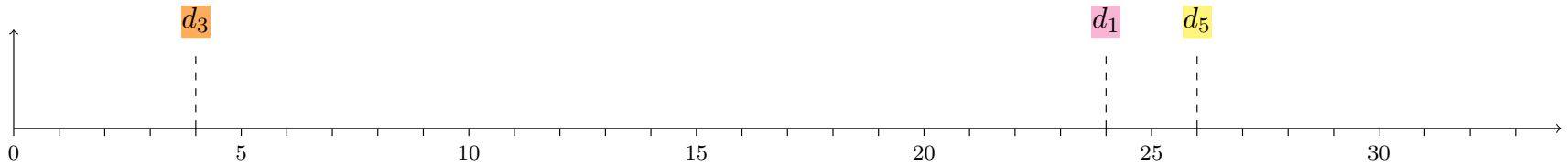
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**14**	**5**	2	3	6	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

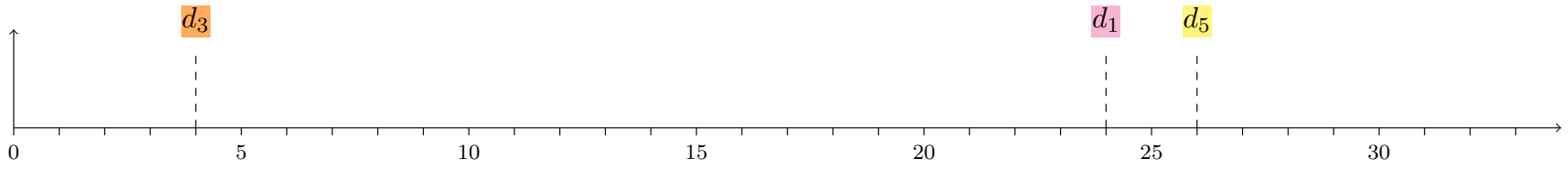
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**14**	**5**	2	3	6	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

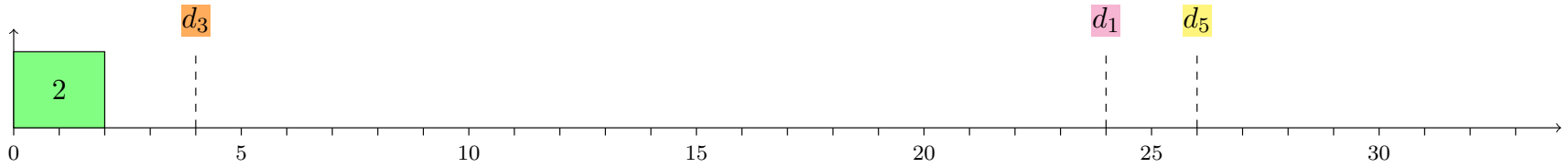
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**14**	**3**	2	3	6	3
r_j	**0**	**0**	**2**	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

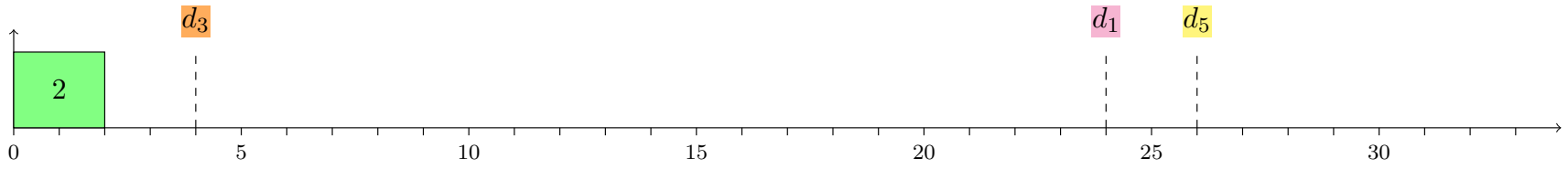
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	14	3	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

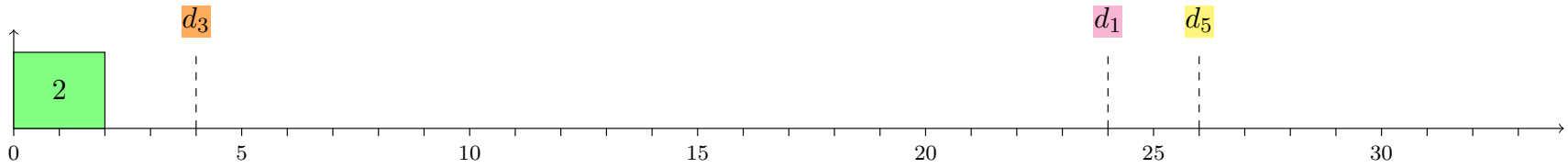
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**14**	**3**	**2**	3	6	3
r_j	**0**	**0**	**2**	12	16	18
d_j	**24**	∞	**4**	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

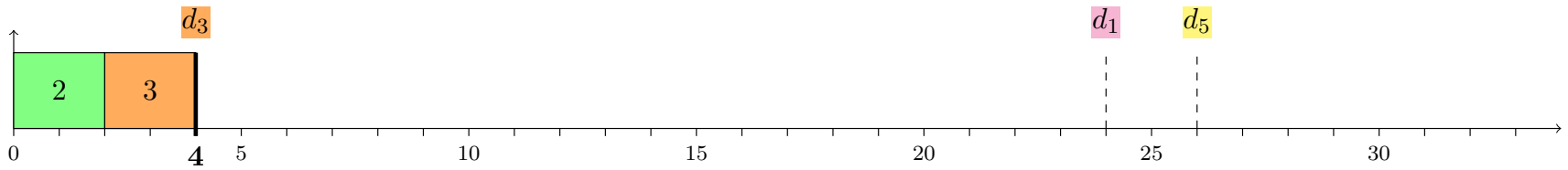
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**14**	**3**	**0**	3	6	3
r_j	**0**	**0**	**2**	12	16	18
d_j	**24**	∞	**4**	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

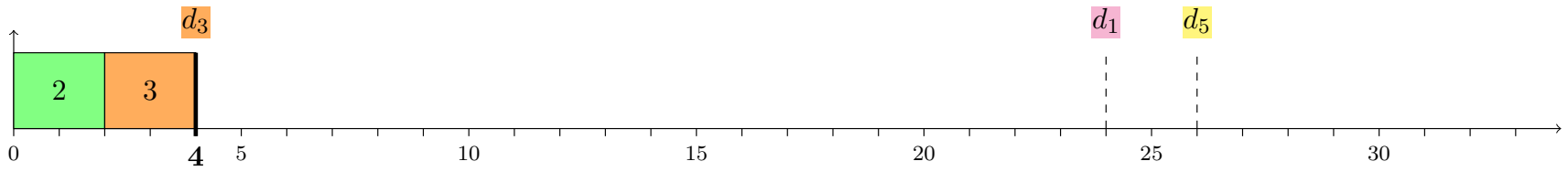
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**14**	**3**	0	3	6	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

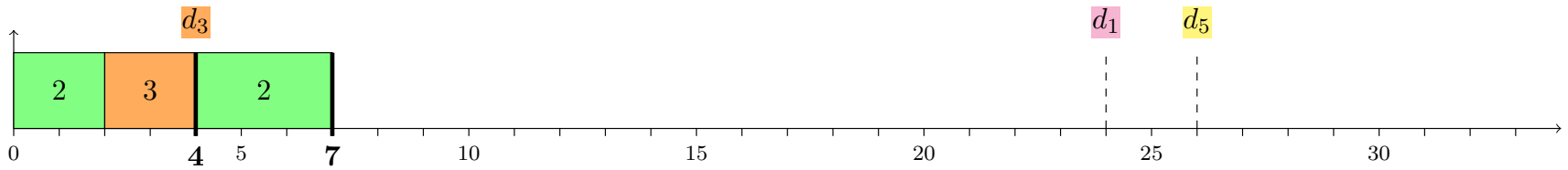
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**14**	**0**	0	3	6	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

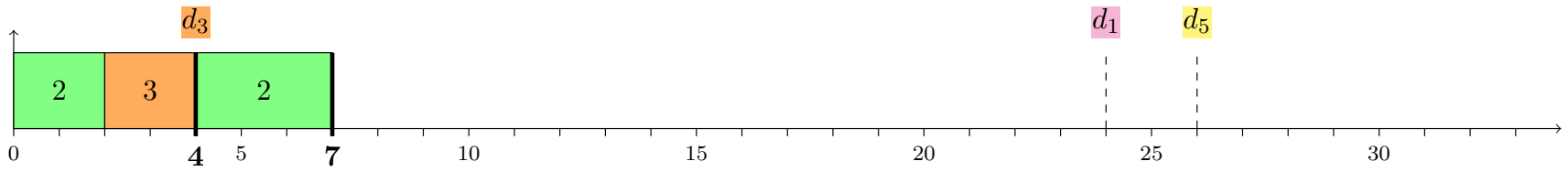
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**14**	0	0	3	6	3
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

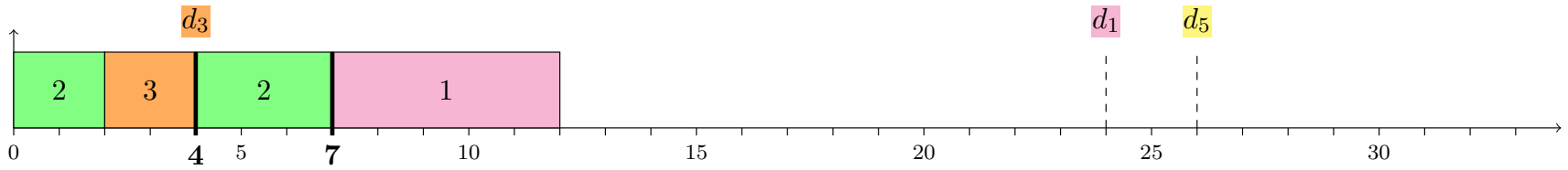
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**9**	0	0	3	6	3
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

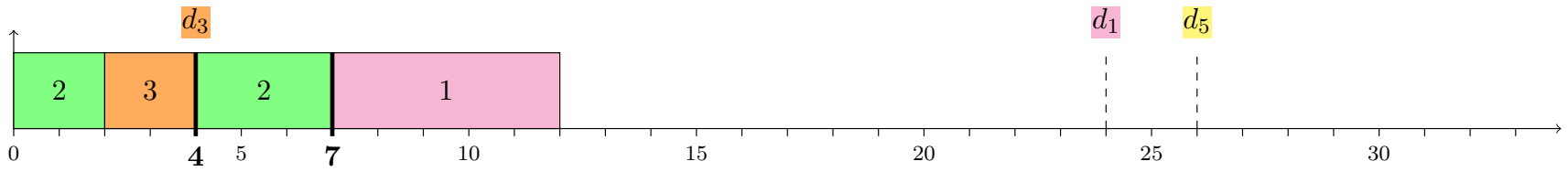
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	**4**	5	6
$\widehat{p}_j$	**9**	0	0	**3**	6	3
r_j	**0**	0	2	**12**	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

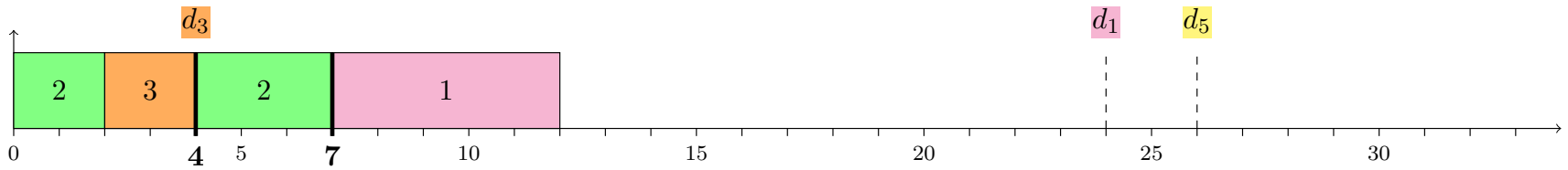
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	**4**	5	6
$\widehat{p}_j$	**9**	0	0	**3**	6	3
r_j	**0**	0	2	**12**	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

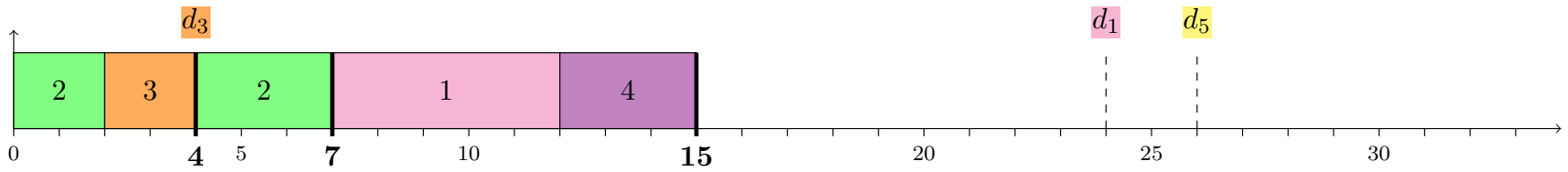
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	**4**	5	6
$\widehat{p}_j$	**9**	0	0	**0**	6	3
r_j	**0**	0	2	**12**	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

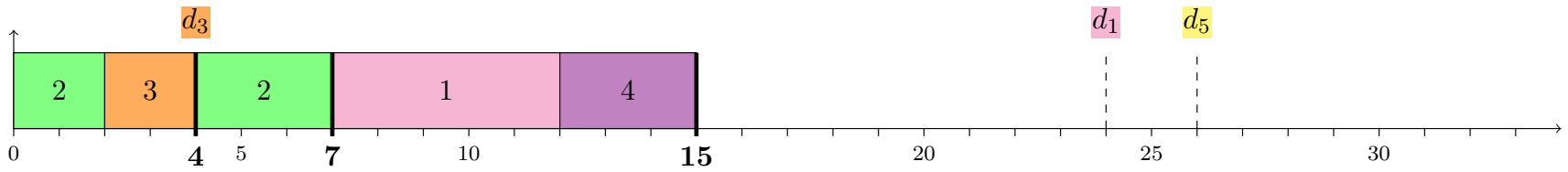
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**9**	0	0	0	6	3
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

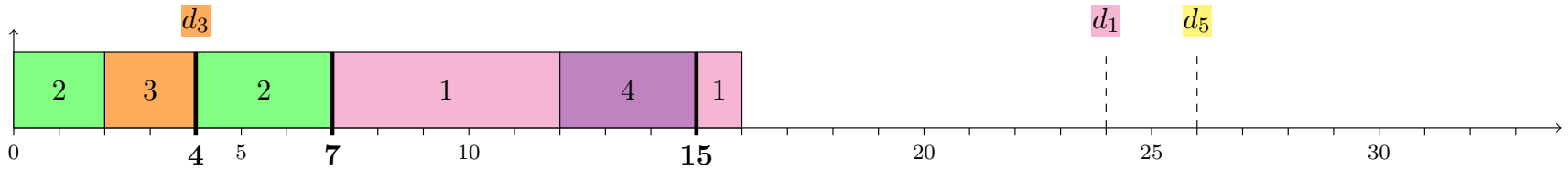
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**8**	0	0	0	6	3
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

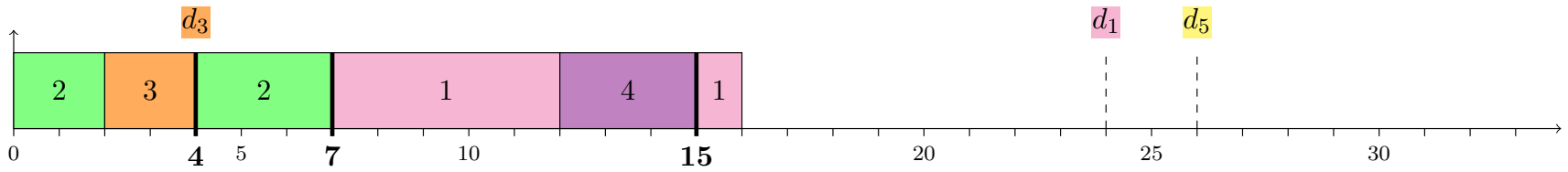
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**8**	0	0	0	**6**	3
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

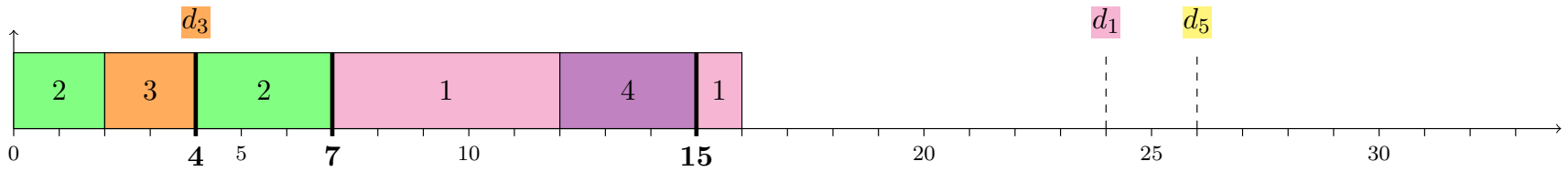
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**8**	0	0	0	**6**	3
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

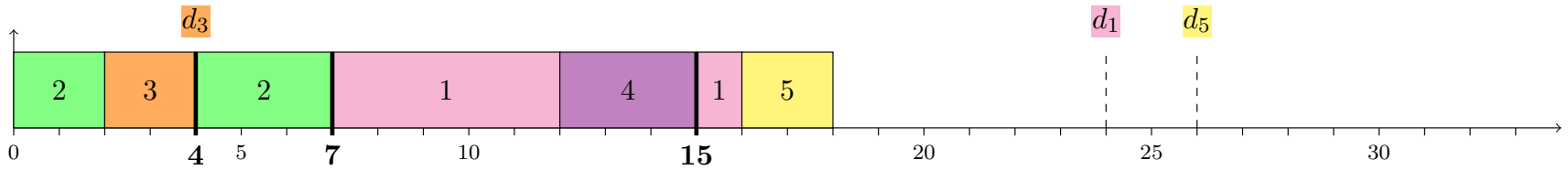
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**8**	0	0	0	**4**	3
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

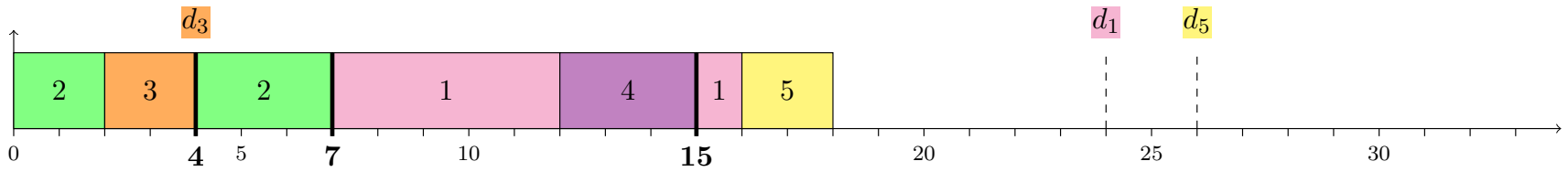
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	**6**
$\widehat{p}_j$	**8**	0	0	0	**4**	**3**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

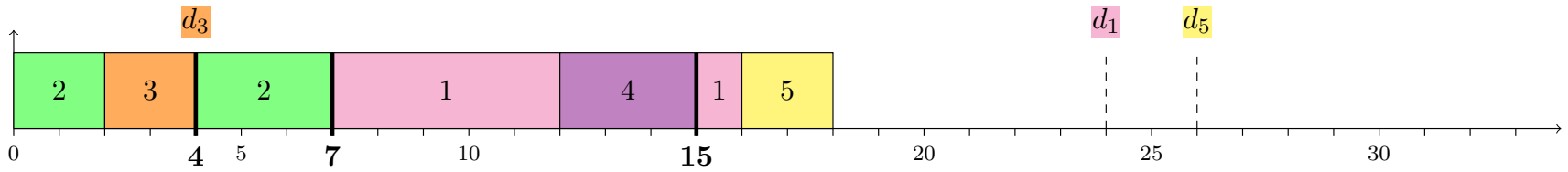
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	**6**
$\widehat{p}_j$	**8**	0	0	0	**4**	**3**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

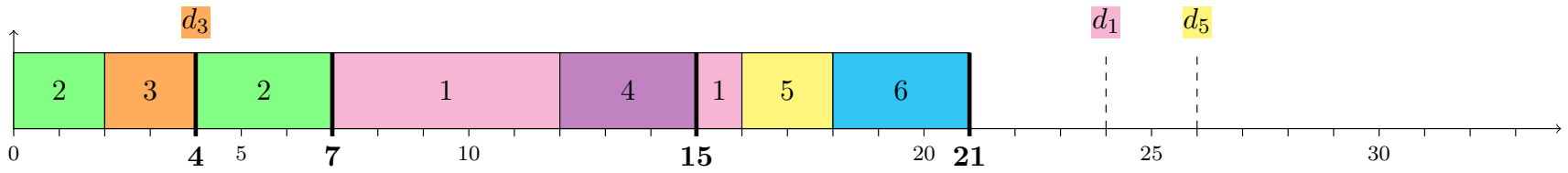
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	**6**
$\widehat{p}_j$	**8**	0	0	0	**4**	**0**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

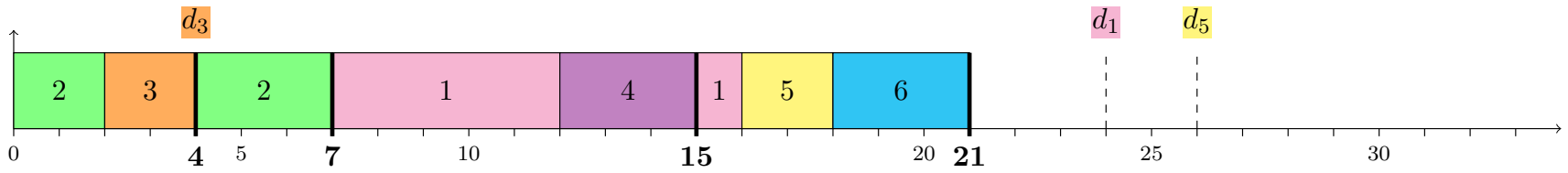
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**8**	0	0	0	**4**	0
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

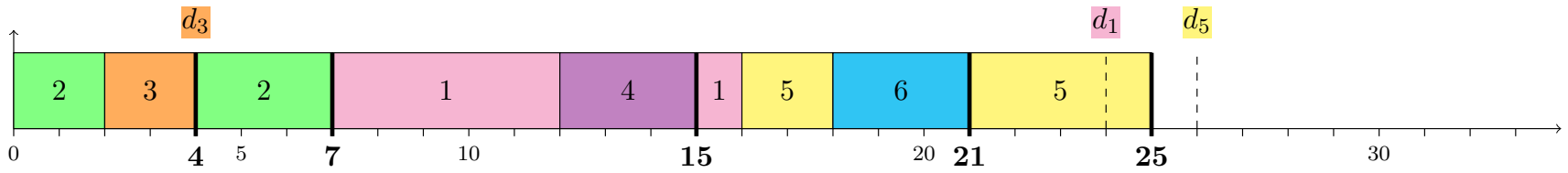
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**8**	0	0	0	**0**	0
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

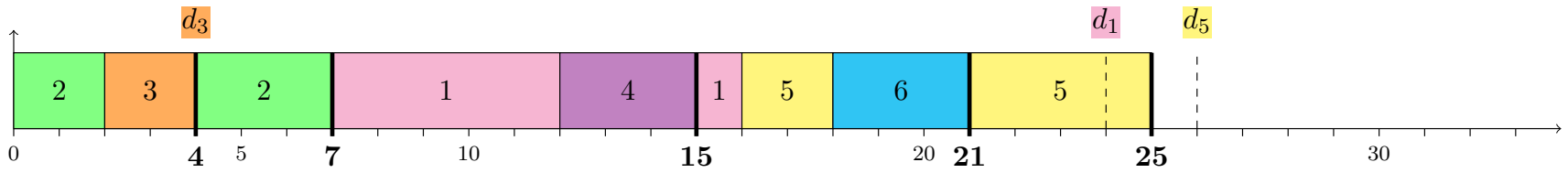
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**8**	0	0	0	0	0
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

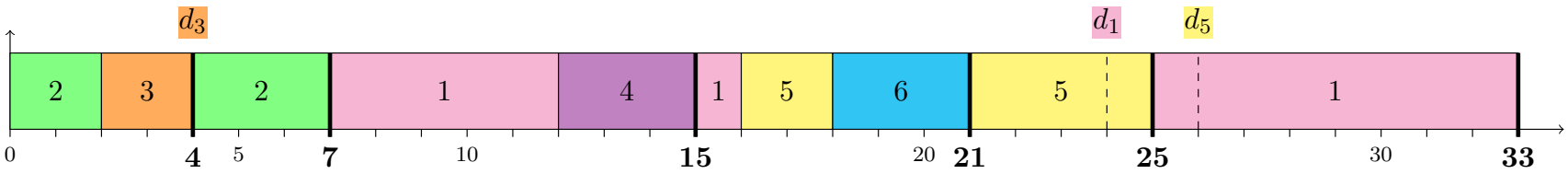
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**0**	0	0	0	0	0
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

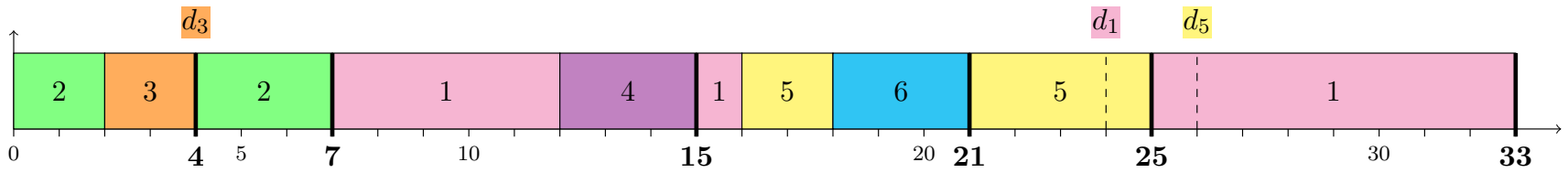
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

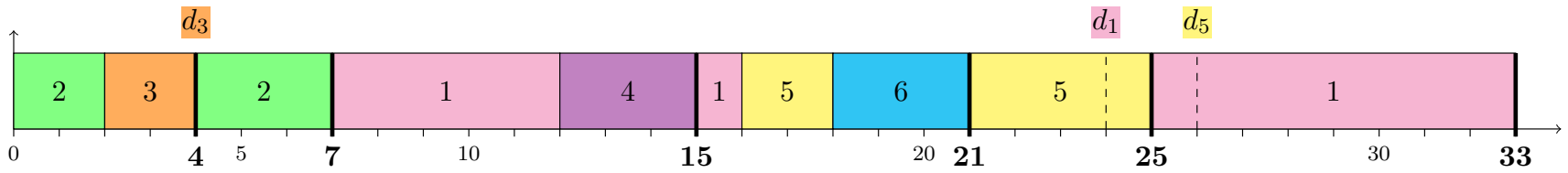
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

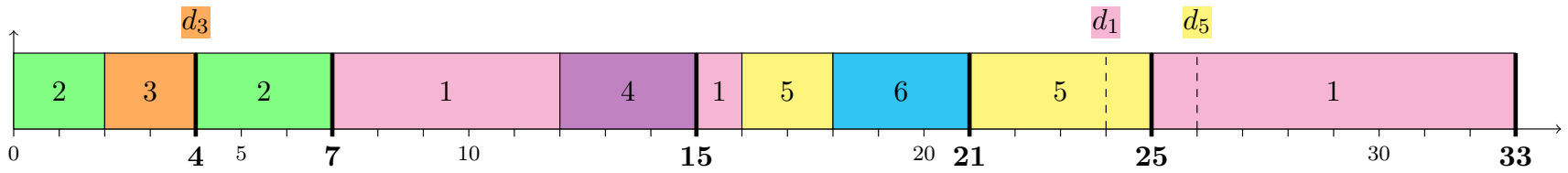
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule (MSR)

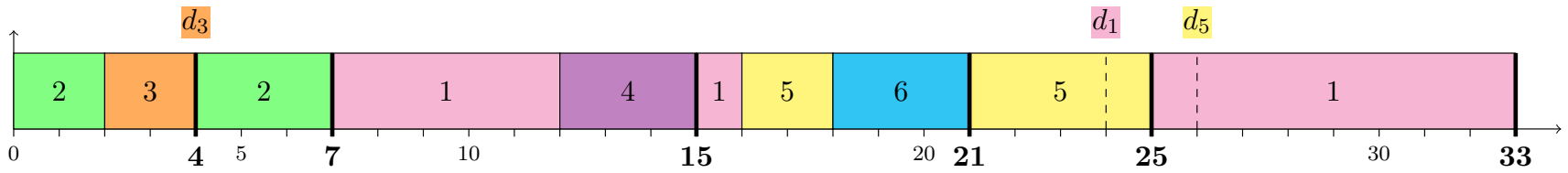
Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time** ($\widehat{p}_j$).

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 105$$

$$(4 + 7 + 15 + 21 + 25 + 33)$$



Complexity: $O(n \log n)$ with a list algorithm

Modified Smith Rule with MPs

Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$

Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

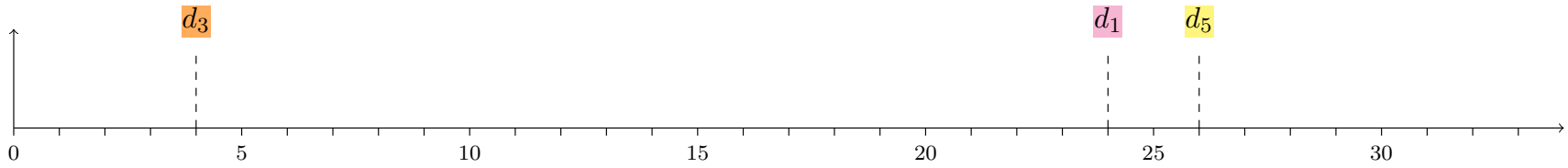
Theorem: Modified Smith Rule with MPs

1. Schedule MPs on their intervals
2. Use MSR on the tasks without their MP
   - In case of equality, schedule the unfinished task without MP or with the earliest MP

id_j	1	2	3	4	5	6
$\widehat{p}_j$	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

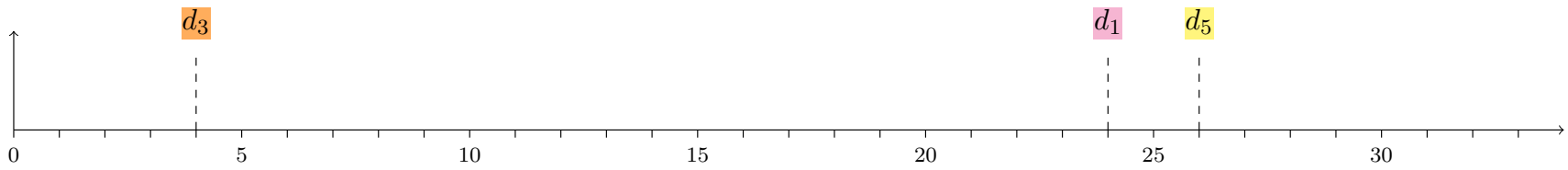
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

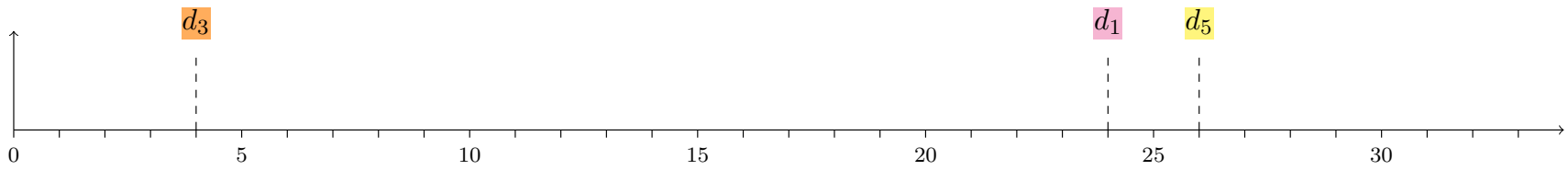
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without** MP or with the **earliest** MP

id_j	**1**	2	**3**	4	**5**	6
$\widehat{p}_j$	**14**	5	**2**	3	**6**	3
r_j	**0**	0	**2**	12	**16**	18
d_j	**24**	∞	**4**	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

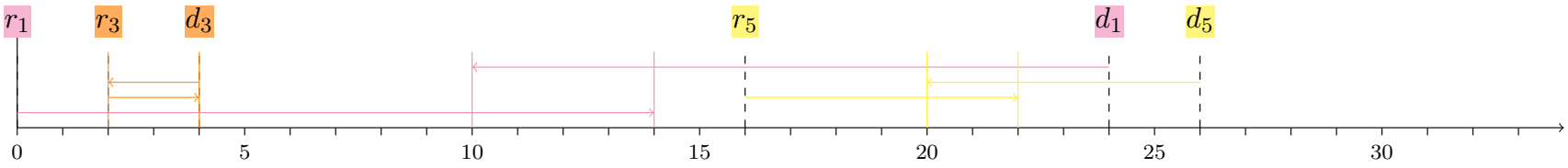
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	**3**	4	**5**	6
$\widehat{p}_j$	**14**	5	**2**	3	**6**	3
r_j	**0**	0	**2**	12	**16**	18
d_j	**24**	∞	**4**	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

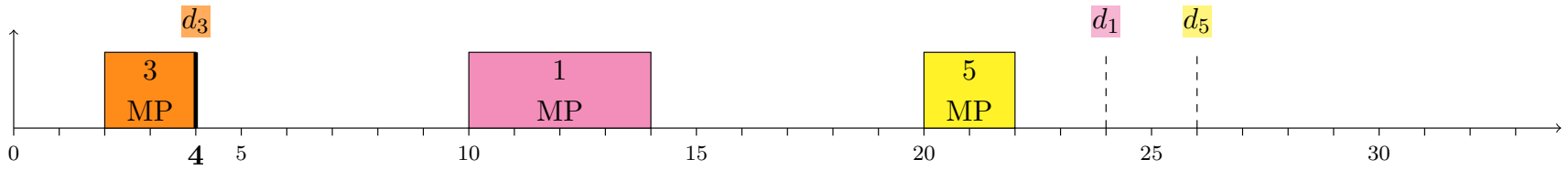
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of equality, schedule the unfinished task without MP or with the earliest MP

id_j	1	2	3	4	5	6
$\widehat{p}_j$	14	5	2	3	6	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

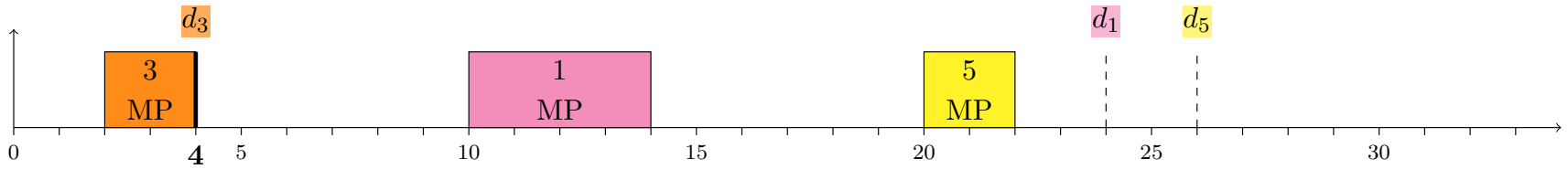
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of equality, schedule the unfinished task without MP or with the earliest MP

id_j	**1**	2	**3**	4	**5**	6
$\widehat{p}_j$	**10**	5	**0**	3	**4**	3
r_j	**0**	0	**2**	12	**16**	18
d_j	**24**	∞	**4**	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

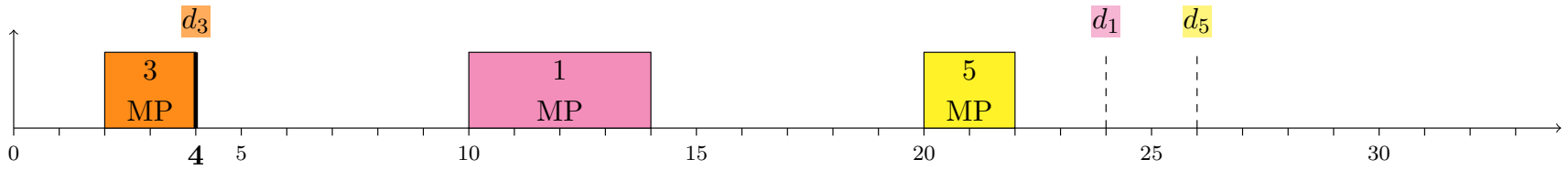
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	10	5	0	3	4	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

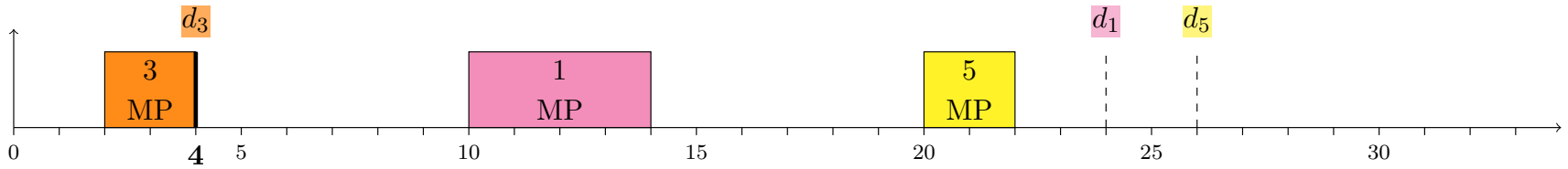
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**10**	**5**	0	3	4	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

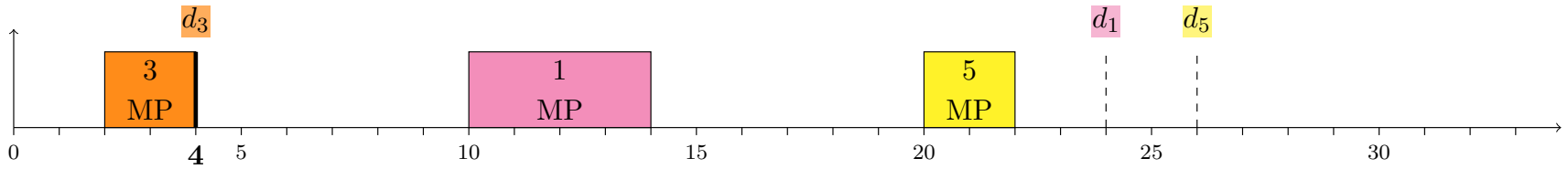
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**10**	**5**	0	3	4	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

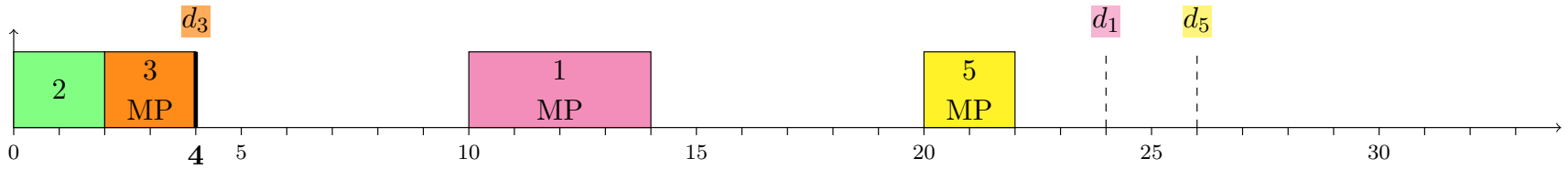
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**10**	**3**	0	3	4	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

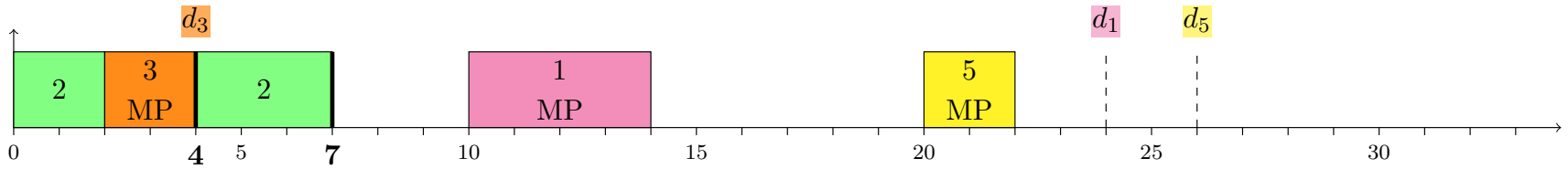
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**10**	**0**	0	3	4	3
r_j	**0**	**0**	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

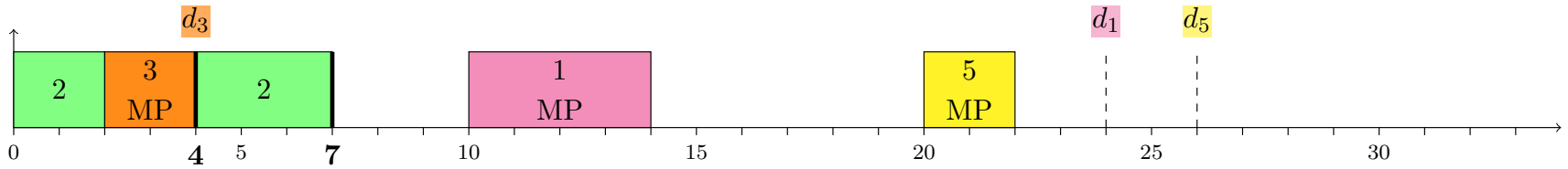
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**10**	0	0	3	4	3
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

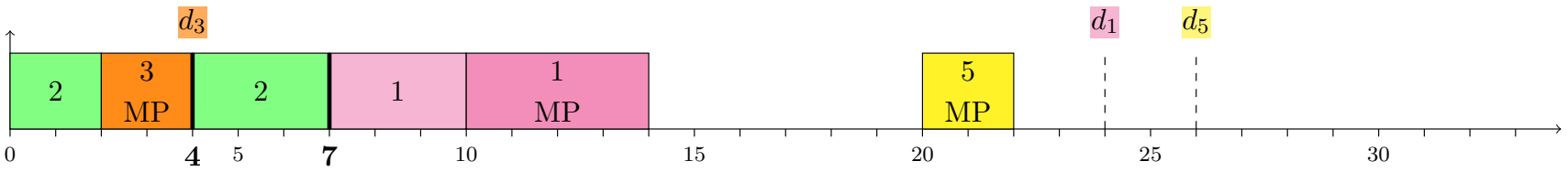
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**7**	0	0	3	4	3
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

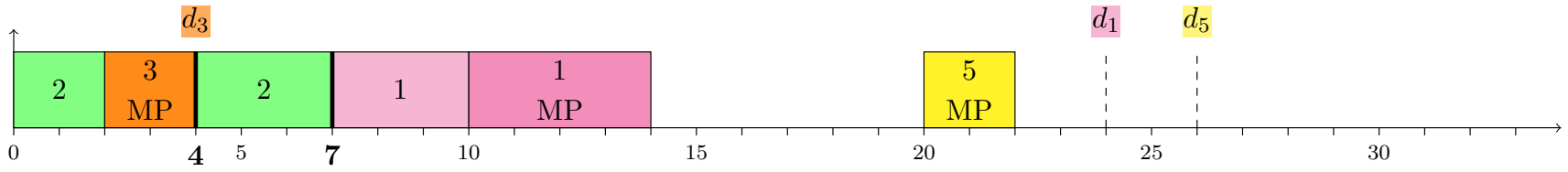
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	**4**	5	6
$\widehat{p}_j$	**7**	0	0	**3**	4	3
r_j	**0**	0	2	**12**	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

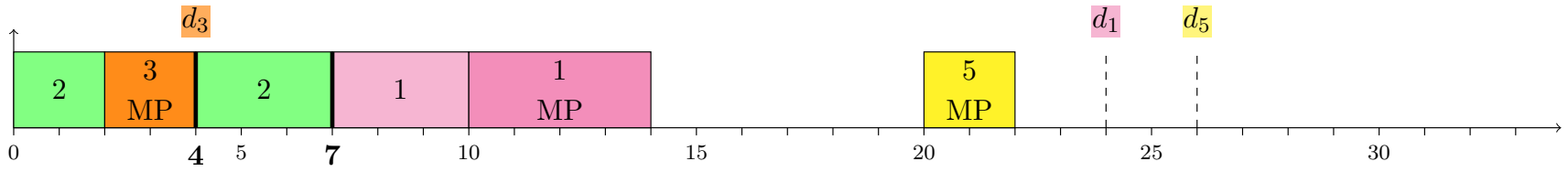
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	**4**	5	6
$\widehat{p}_j$	**7**	0	0	**3**	4	3
r_j	**0**	0	2	**12**	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

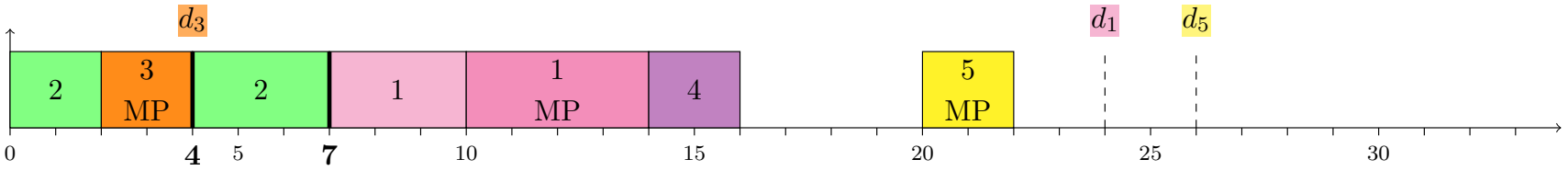
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	**4**	5	6
$\widehat{p}_j$	**7**	0	0	**1**	4	3
r_j	**0**	0	2	**12**	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

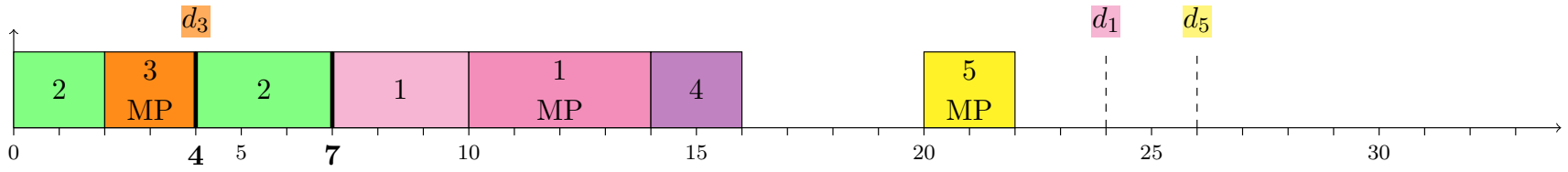
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	**4**	**5**	6
$\widehat{p}_j$	**7**	0	0	**1**	**4**	3
r_j	**0**	0	2	**12**	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

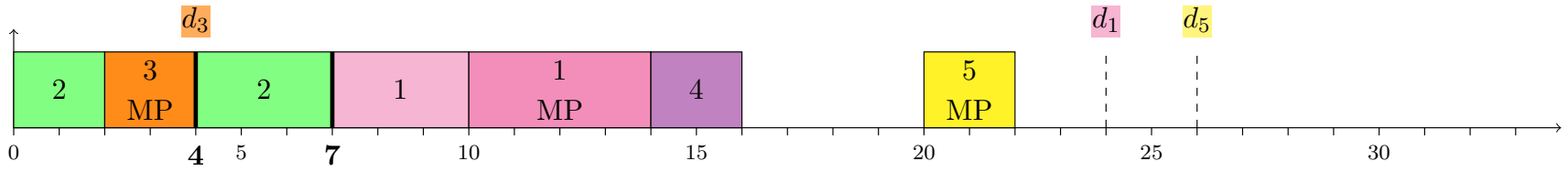
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	**4**	**5**	6
$\widehat{p}_j$	**7**	0	0	**1**	**4**	3
r_j	**0**	0	2	**12**	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

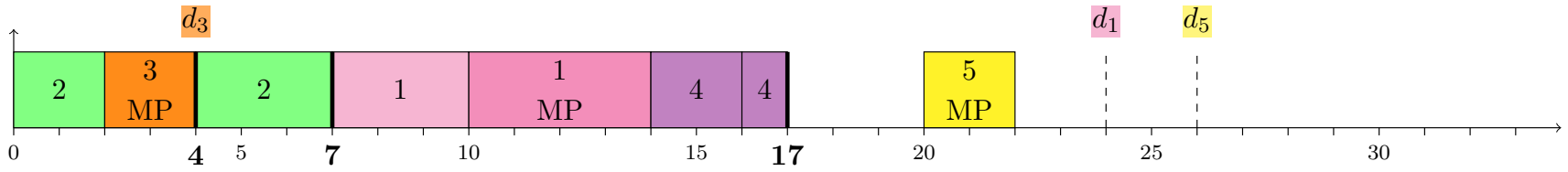
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	**4**	**5**	6
$\widehat{p}_j$	**7**	0	0	**0**	**4**	3
r_j	**0**	0	2	**12**	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

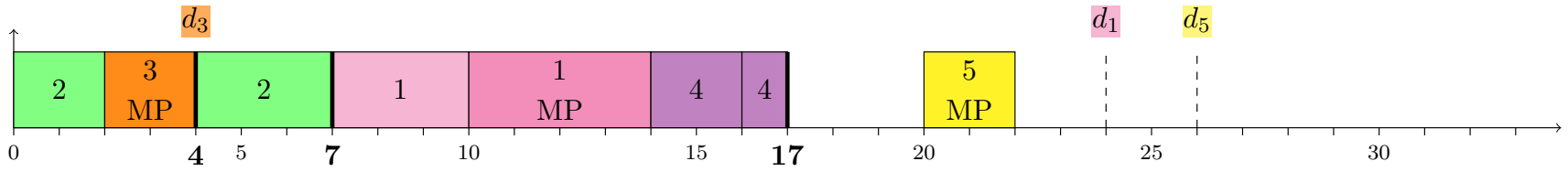
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**7**	0	0	0	**4**	3
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

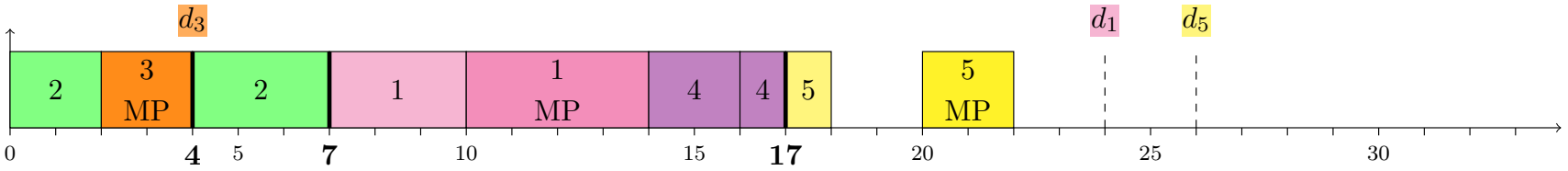
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**7**	0	0	0	**3**	3
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

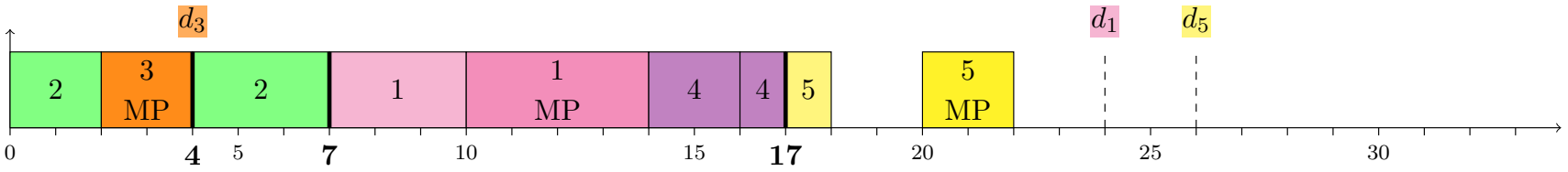
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	7	0	0	0	3	3
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

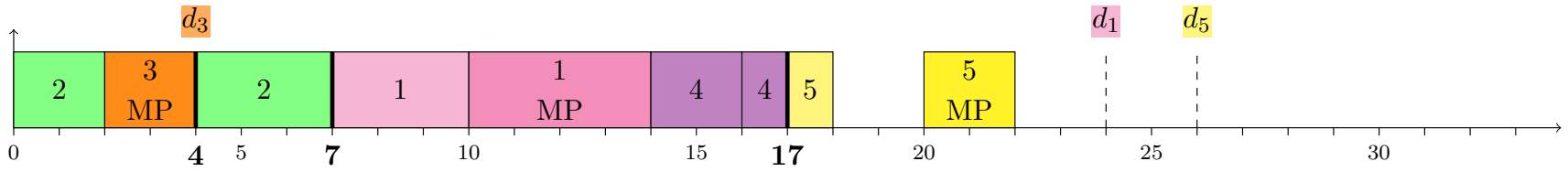
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	**7**	0	0	0	**3**	**3**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

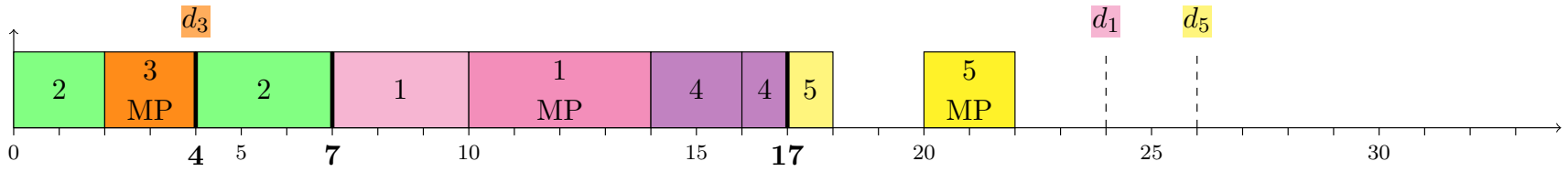
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	**6**
$\widehat{p}_j$	**7**	0	0	0	**3**	**3**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

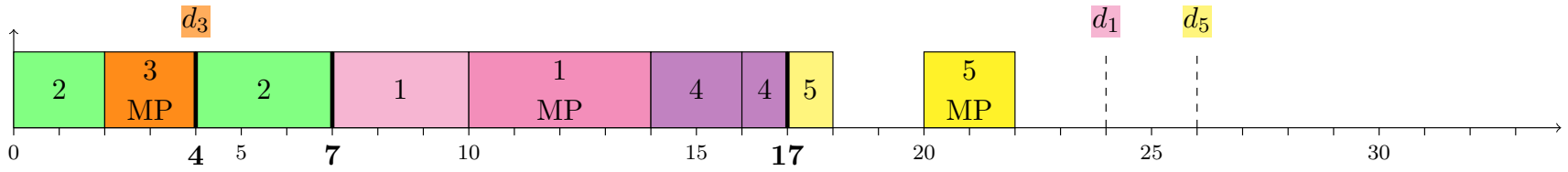
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	**6**
$\widehat{p}_j$	**7**	0	0	0	**3**	**3**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

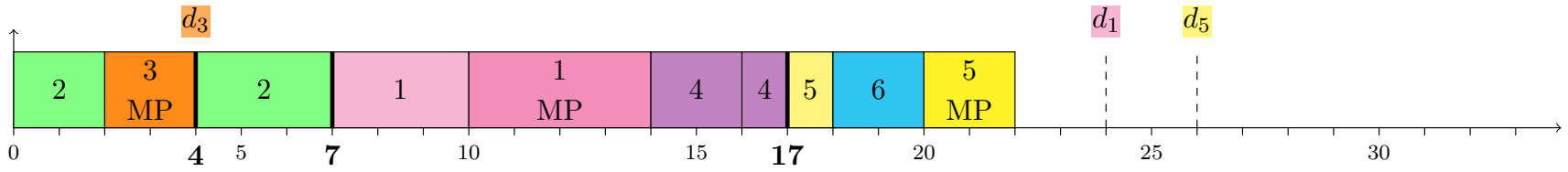
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	**6**
$\widehat{p}_j$	**7**	0	0	0	**3**	**1**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

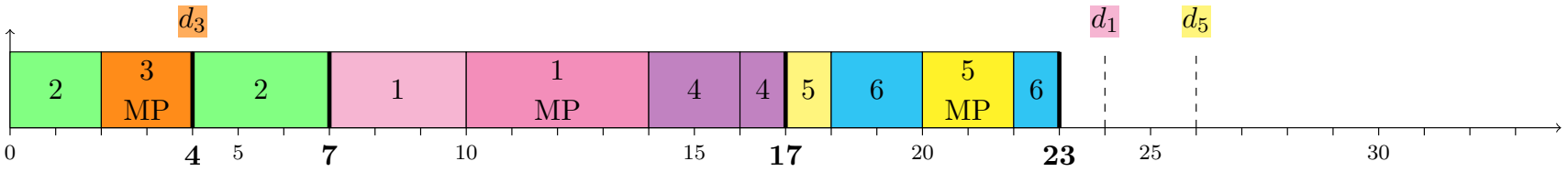
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	**6**
$\widehat{p}_j$	**7**	0	0	0	**3**	**0**
r_j	**0**	0	2	12	**16**	**18**
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

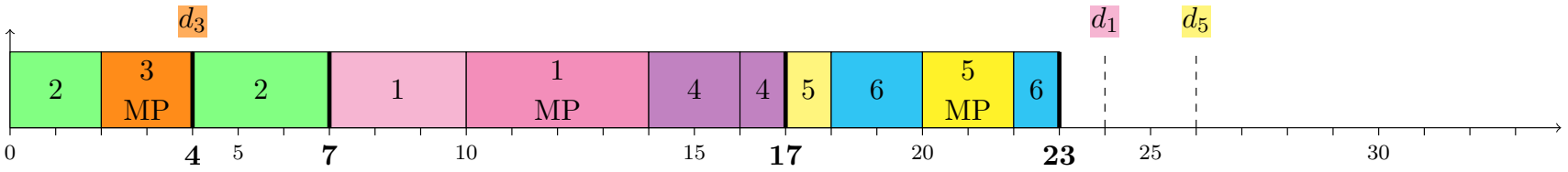
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**7**	0	0	0	**3**	0
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

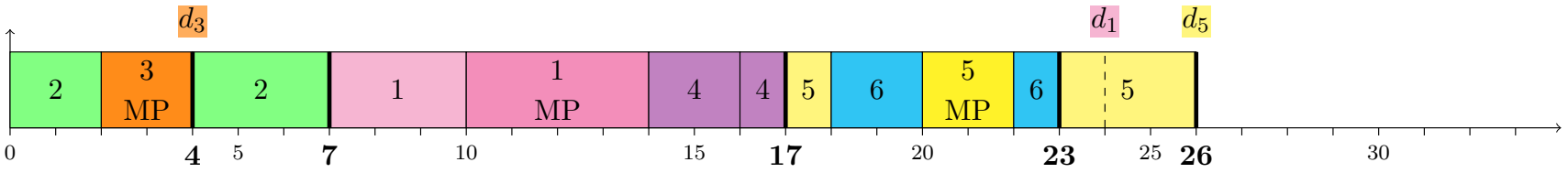
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	**5**	6
$\widehat{p}_j$	**7**	0	0	0	**0**	0
r_j	**0**	0	2	12	**16**	18
d_j	**24**	∞	4	∞	**26**	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

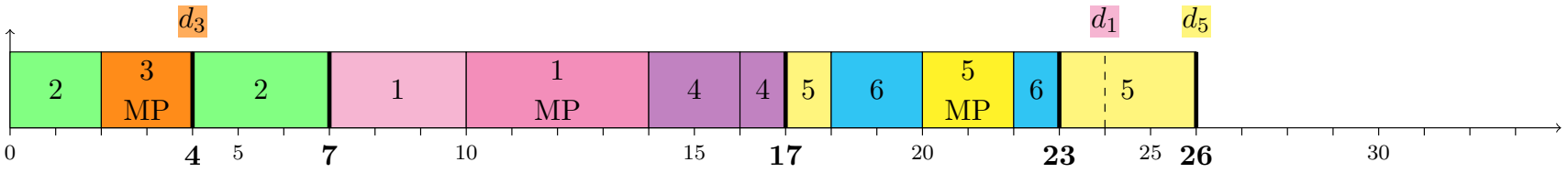
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**7**	0	0	0	0	0
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

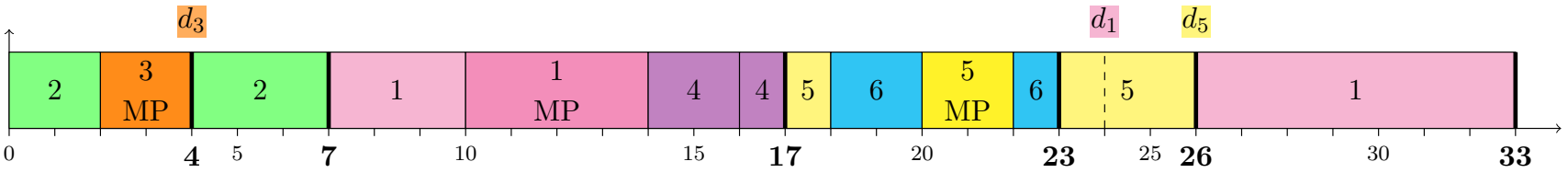
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	**1**	2	3	4	5	6
$\widehat{p}_j$	**0**	0	0	0	0	0
r_j	**0**	0	2	12	16	18
d_j	**24**	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

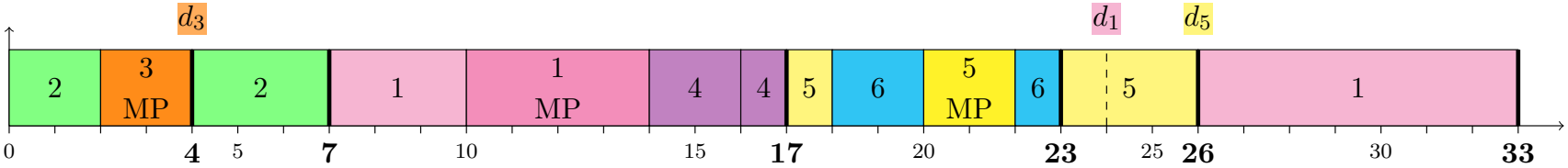
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

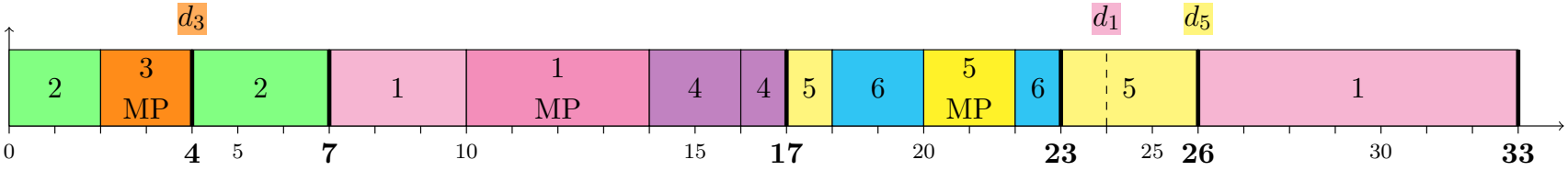
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

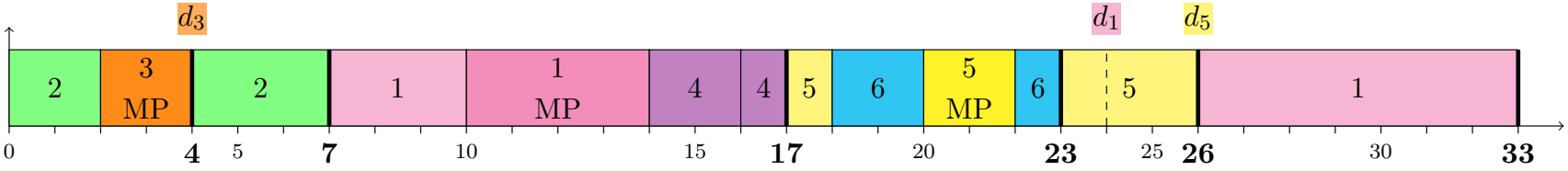
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Modified Smith Rule with MPs

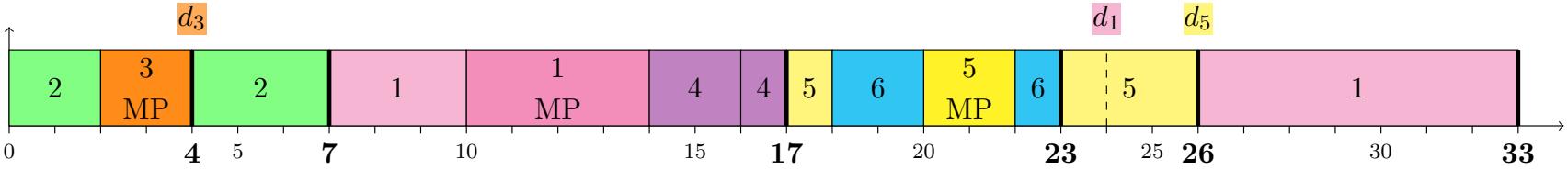
Theorem: Modified Smith Rule with MPs

1. Schedule **MPs** on their intervals
2. Use **MSR** on the tasks without their MP
   - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

id_j	1	2	3	4	5	6
$\widehat{p}_j$	0	0	0	0	0	0
r_j	0	0	2	12	16	18
d_j	24	∞	4	∞	26	∞

$$\sum C_j = 110$$

$$(4 + 7 + 17 + 23 + 26 + 33)$$



Complexity: $O(n \log n)$ with a sweep line algorithm

Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S^* = solution of MSR with MPs ;
- i has a higher priority than j
- Starting at t_i in S^*, take out the (non MP) parts of i and j
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$

Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- $S^* =$ solution of MSR with MPs ;
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$

Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$

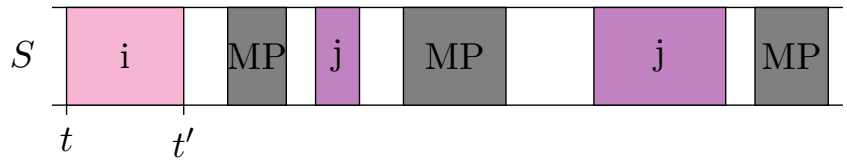
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



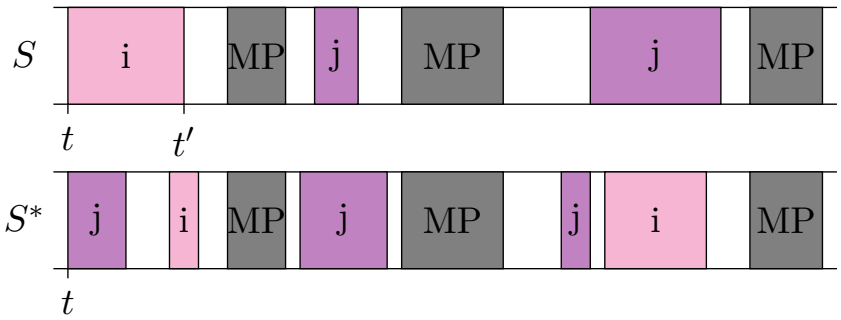
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



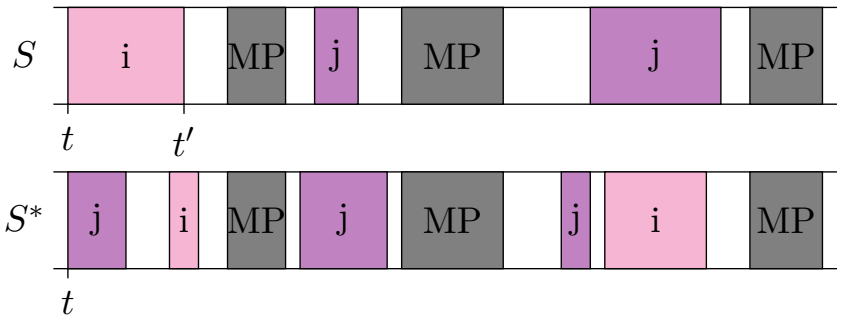
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



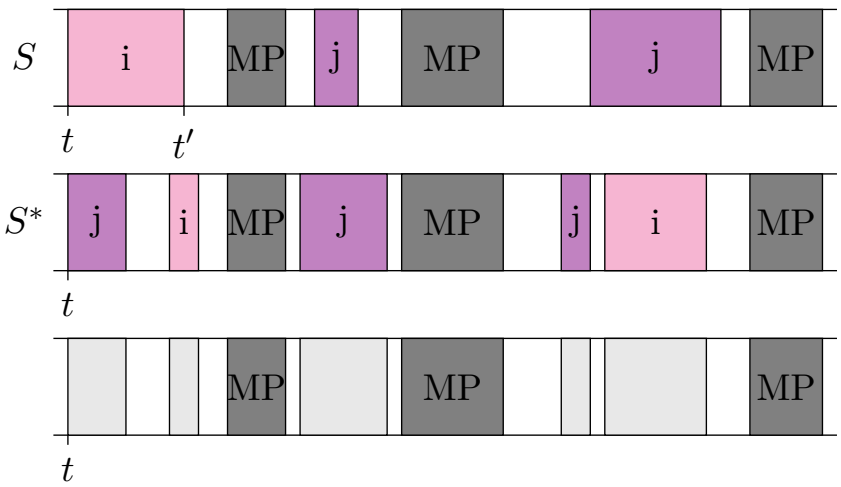
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



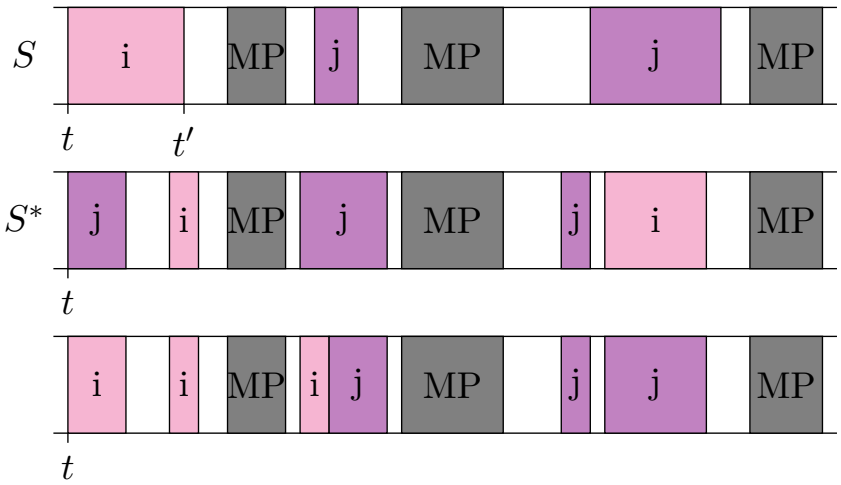
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



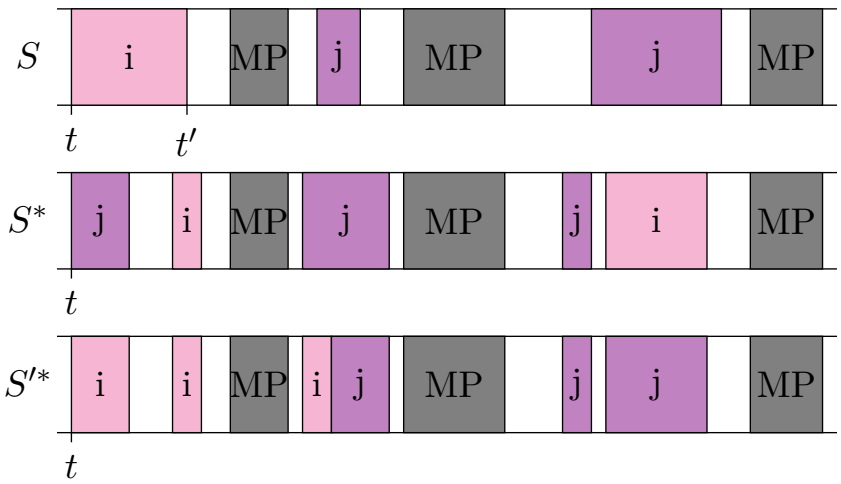
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



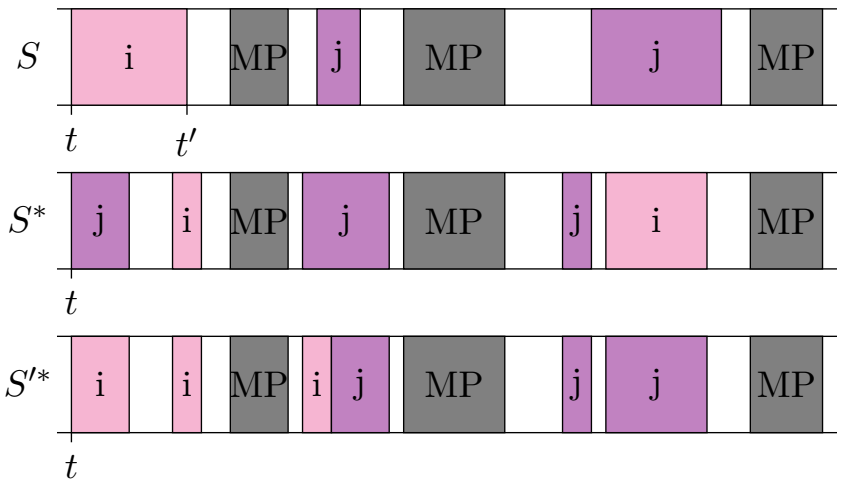
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



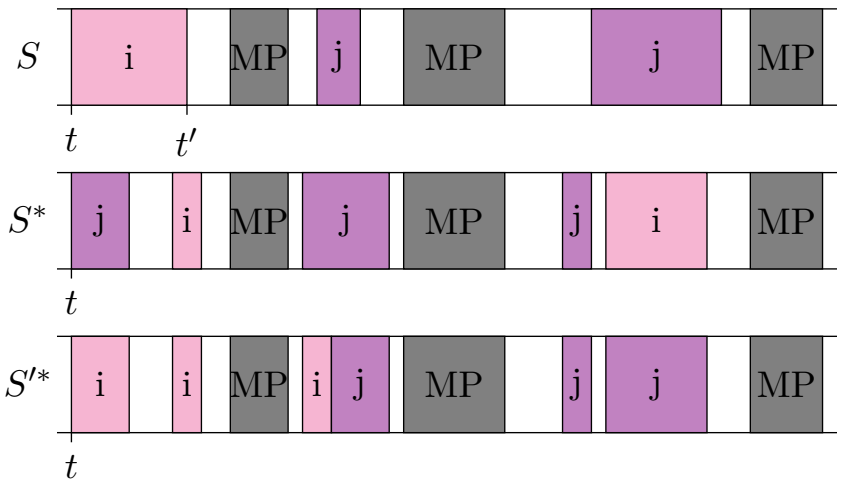
Optimality of MSR with MPs

Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for $1|r_j; pmtn; MP|\sum C_j$.

Sketch of proof

- S = solution of MSR with MPs ; S^* = optimal solution
- i has a higher priority than j
- Starting at t, in S^*, take out the (non MP) parts of i and j and re-schedule i before $j \Rightarrow S'^*$
- Lemma: the objective value of S'^* is no greater than those of S^*
- Repeat until $S = S^*$



Lemma

Lemma: Optimality of MSR with MP

The objective value of S'^* is no greater than those of S^*.

Notations

- $\widehat{p}_k$ = remaining time of k
- C_k (resp. C'_k) = completion time of k in S^* (resp. in S'^*)
- eft_k = earliest finishing time of k (start of the MP)
- lst_k = latest starting time of k (end of the MP)

Lemma

Lemma: Optimality of MSR with MP

The objective value of S'^* is no greater than those of S^*.

Notations

- $\widehat{p}_k$ = remaining time of k
- C_k (resp. C'_k) = completion time of k in S^* (resp. in S'^*)
- eft_k = earliest finishing time of k (start of the MP)
- lst_k = latest starting time of k (end of the MP)

Lemma

Lemma: Optimality of MSR with MP

The objective value of S'^* is no greater than those of S^*.

Notations

- $\widehat{p}_k$ = remaining time of k
- C_k (resp. C'_k) = completion time of k in S^* (resp. in S'^*)
- eft_k = earliest finishing time of k (start of the MP)
- lst_k = latest starting time of k (end of the MP)

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma : proof

3 cases

- Case 1: No MP after $t \Rightarrow$ MSR
- Case 2: MP of i or j in the middle of its execution in S'^*
- Case 3: i or j ends by its MP in S'^*

Goal

- Show that the objective value of S'^* is no greater than those of S^*
- ie. show that $C'_i + C'_j \leq C_i + C_j$

Lemma: proof of case 2

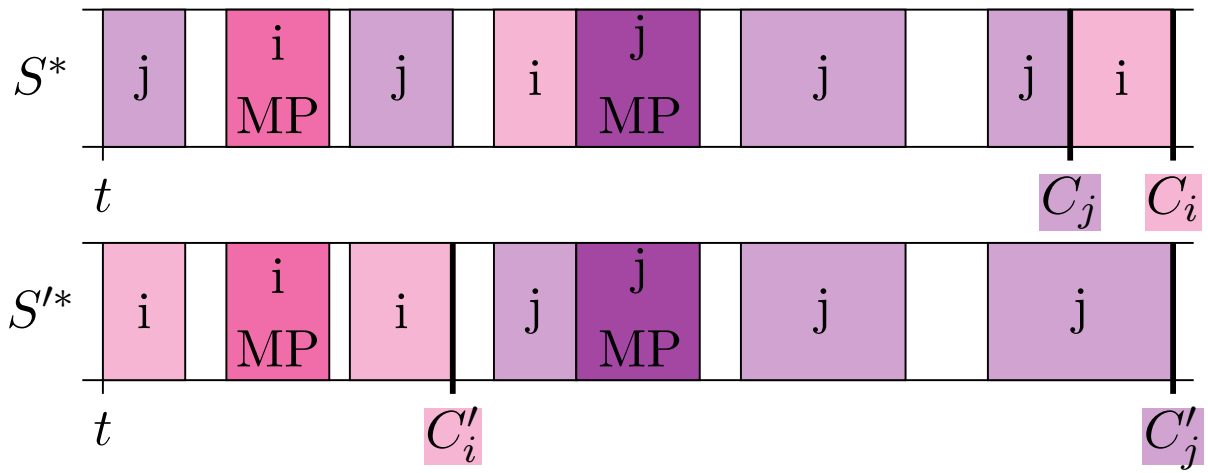
Case 2: MP of i or j in the middle of its execution in S'^*

- Definition: MPs are at the same place in S^* and S'^*
- MPs in the middle $\Rightarrow$ tasks scheduled around the MPs
- These MPs have no influence on C'_i and C'_j
- We can ignore these MPs $\Rightarrow$ go to case 1 or 3

Lemma: proof of case 2

Case 2: MP of i or j in the middle of its execution in S'^*

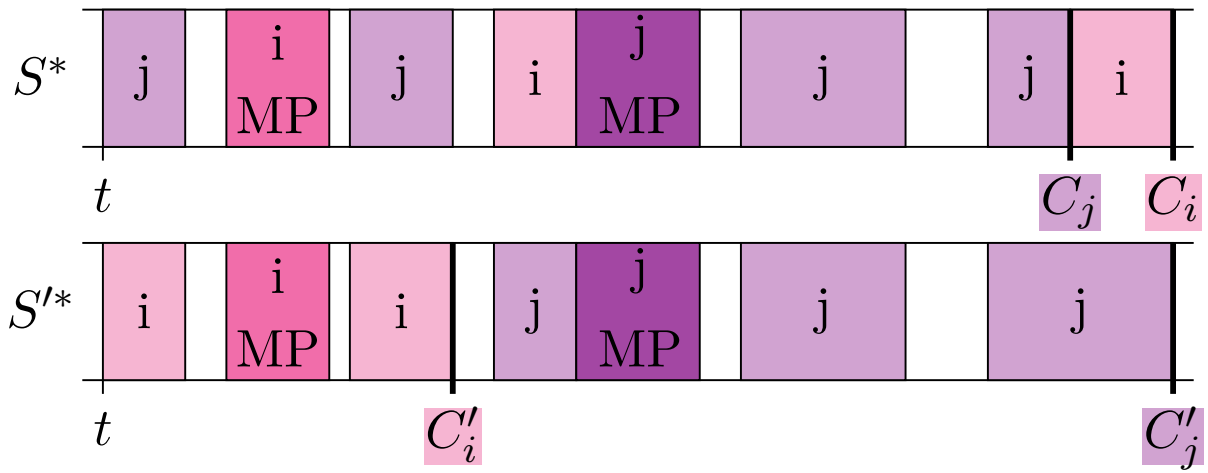
- Definition: MPs are at the same place in S^* and S'^*
- MPs in the middle $\Rightarrow$ tasks scheduled around the MPs
- These MPs have no influence on C'_i and C'_j
- We can ignore these MPs $\Rightarrow$ go to case 1 or 3



Lemma: proof of case 2

Case 2: MP of i or j in the middle of its execution in S'^*

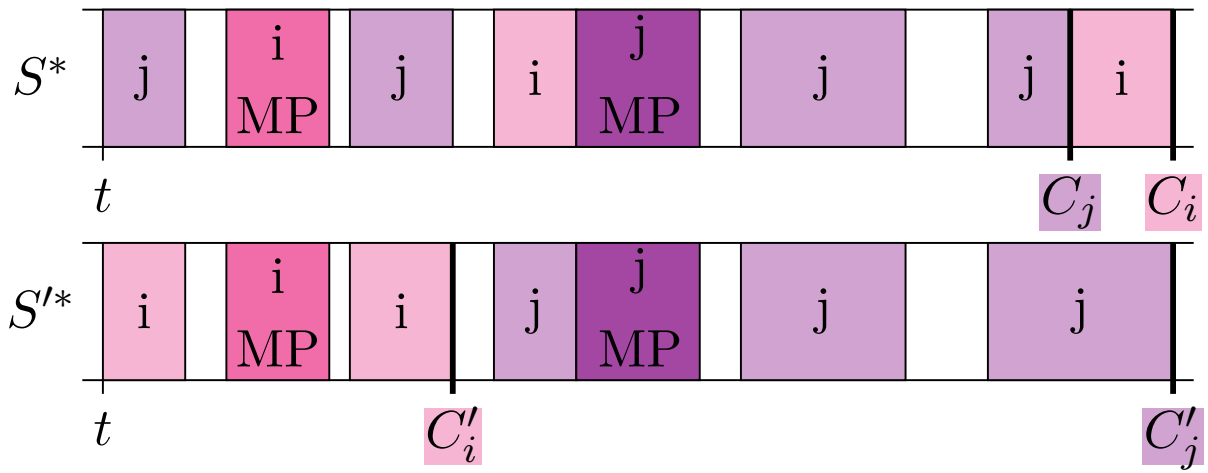
- Definition: MPs are at the same place in S^* and S'^*
- MPs in the middle $\Rightarrow$ tasks scheduled around the MPs
- These MPs have no influence on C'_i and C'_j
- We can ignore these MPs $\Rightarrow$ go to case 1 or 3



Lemma: proof of case 2

Case 2: MP of i or j in the middle of its execution in S'^*

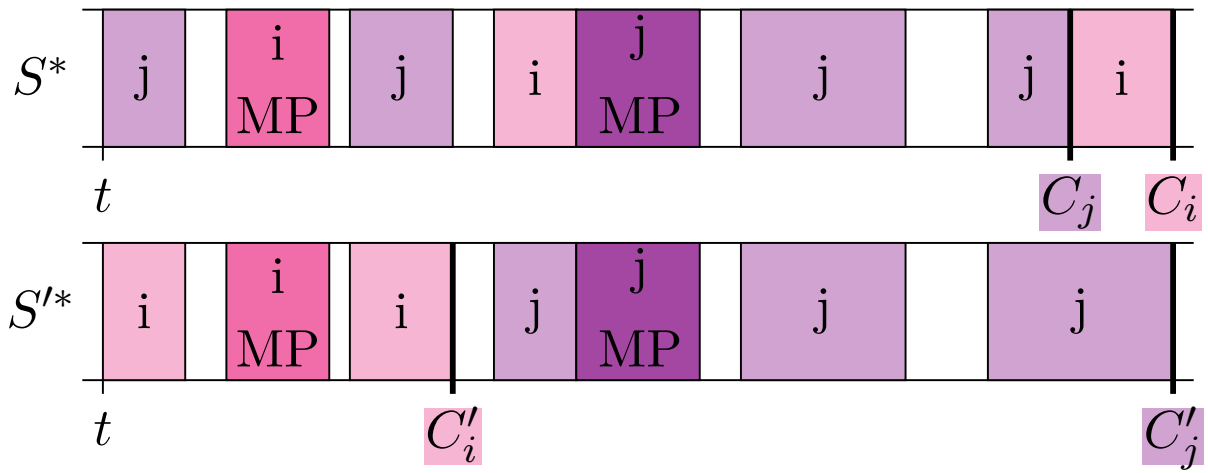
- Definition: MPs are at the same place in S^* and S'^*
- MPs in the middle $\Rightarrow$ tasks scheduled around the MPs
- These MPs have no influence on C'_i and C'_j
- We can ignore these MPs $\Rightarrow$ go to case 1 or 3



Lemma: proof of case 2

Case 2: MP of i or j in the middle of its execution in S'^*

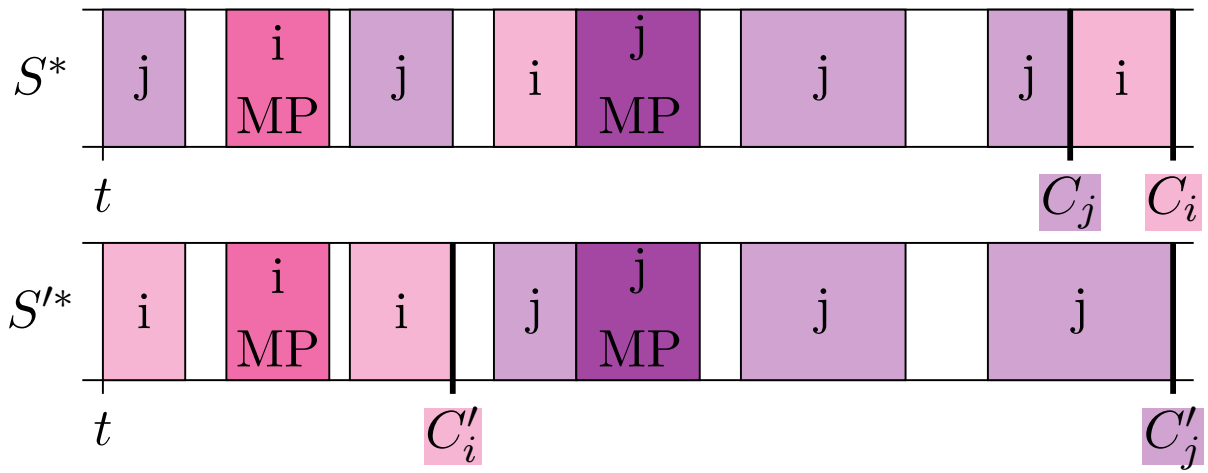
- Definition: MPs are at the same place in S^* and S'^*
- MPs in the middle $\Rightarrow$ tasks scheduled around the MPs
- These MPs have no influence on C'_i and C'_j
- We can ignore these MPs $\Rightarrow$ go to case 1 or 3



Lemma: proof of case 2

Case 2: MP of i or j in the middle of its execution in S'^*

- Definition: MPs are at the same place in S^* and S'^*
- MPs in the middle $\Rightarrow$ tasks scheduled around the MPs
- These MPs have no influence on C'_i and C'_j
- We can ignore these MPs $\Rightarrow$ go to case 1 or 3



Lemma: proof of case 3 (sub-cases)

Case 3: i or j ends by its MP in S'^*

3 sub-cases:

- Sub-case a): j ends with its MP
- Sub-case b): i ends with its MP
- Sub-case c): j and i end with their MP

Lemma: proof of case 3 (sub-cases)

Case 3: i or j ends by its MP in S'^*

3 sub-cases:

- Sub-case a): j ends with its MP
- Sub-case b): i ends with its MP
- Sub-case c): j and i end with their MP

Lemma: proof of case 3 (sub-cases)

Case 3: i or j ends by its MP in S'^*

3 sub-cases:

- Sub-case a): j ends with its MP
- Sub-case b): i ends with its MP
- Sub-case c): j and i end with their MP

Lemma: proof of case 3 (sub-cases)

Case 3: i or j ends by its MP in S'^*

3 sub-cases:

- Sub-case a): j ends with its MP
- Sub-case b): i ends with its MP
- Sub-case c): j and i end with their MP

Lemma: proof of case 3 (sub-cases)

Case 3: i or j ends by its MP in S'^*

3 sub-cases:

- Sub-case a): j ends with its MP
- Sub-case b): i ends with its MP
- Sub-case c): j and i end with their MP

Lemma: proof of case 3 a)

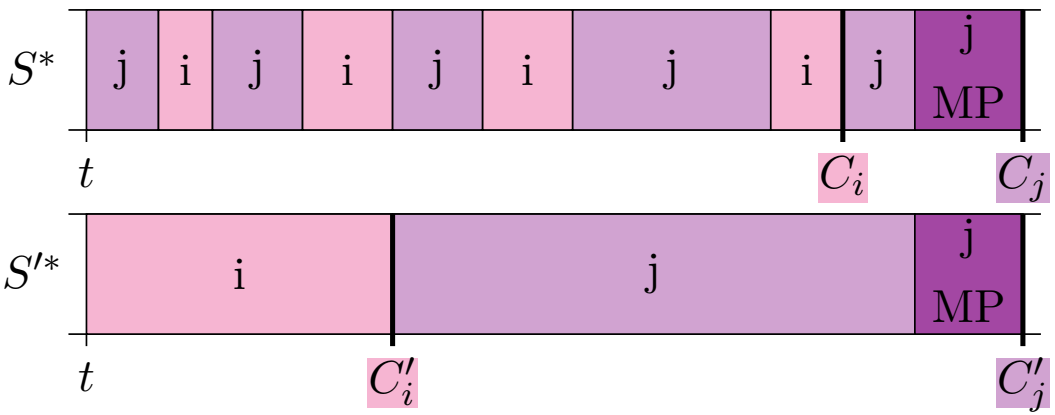
Sub-case a): j ends with its MP

- $C_j = C'_j = eft_j$
- Construction: i processed at t in S'^* (1)
- Construction: j processed in $[t, eft_j[$ in S'^* (2)
- (1) + (2) $\Rightarrow$ CONTRADICTION !

Lemma: proof of case 3 a)

Sub-case a): j ends with its MP

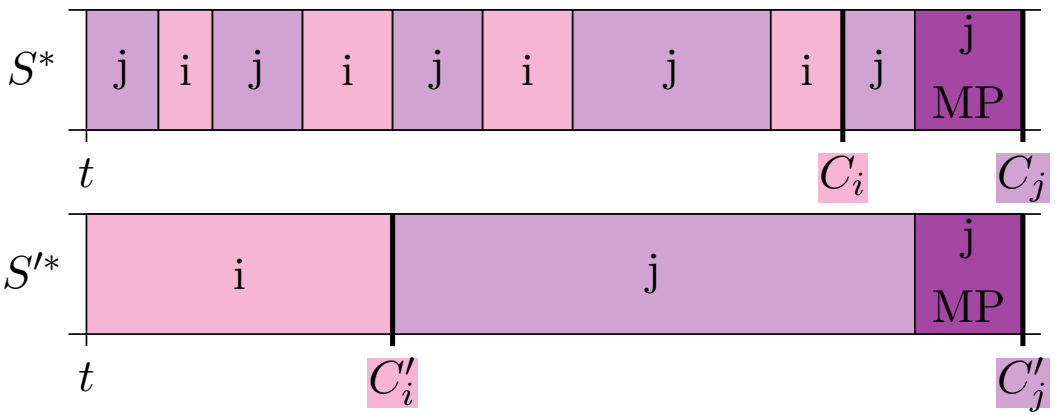
- $C_j = C'_j = eft_j$
- Construction: i processed at t in S'^* (1)
- Construction: j processed in $[t, eft_j[$ in S'^* (2)
- (1) + (2) $\Rightarrow$ CONTRADICTION !



Lemma: proof of case 3 a)

Sub-case a): j ends with its MP

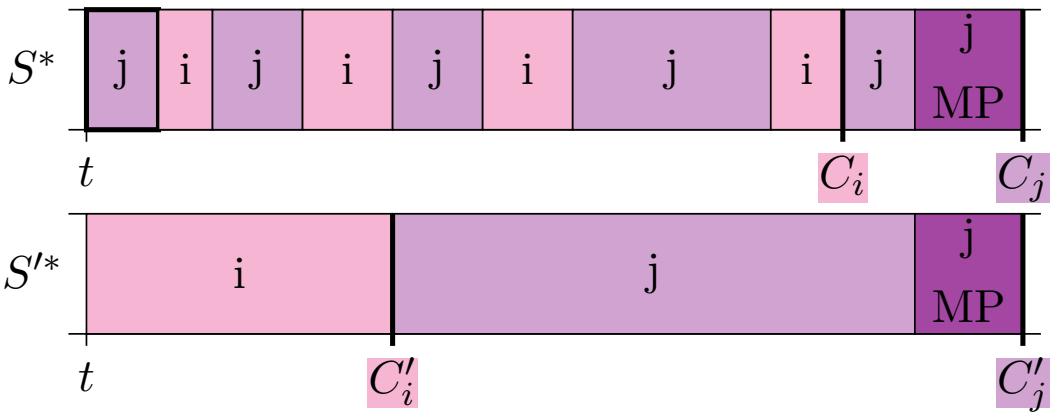
- $C_j = C'_j = eft_j$
- Construction: i processed at t in S'^* (1)
- Construction: j processed in $[t, eft_j[$ in S'^* (2)
- (1) + (2) $\Rightarrow$ CONTRADICTION !

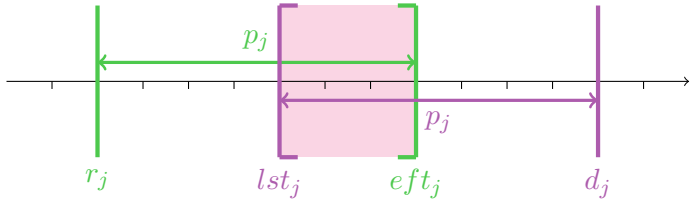


Lemma: proof of case 3 a)

Sub-case a): j ends with its MP

- $C_j = C'_j = eft_j$
- Construction: i processed at t in S'^* (1)
- Construction: j processed in $[t, eft_j[$ in S'^* (2)
- (1) + (2) $\Rightarrow$ CONTRADICTION !

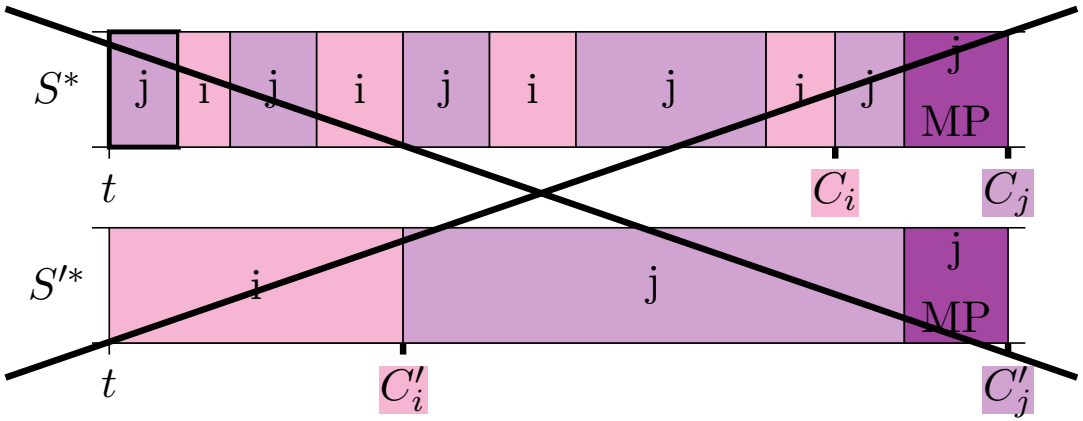


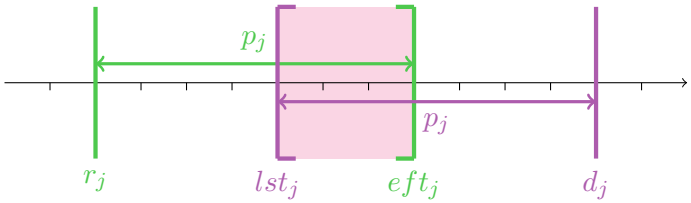


Lemma: proof of case 3 a)

Sub-case a): j ends with its MP

- $C_j = C'_j = eft_j$
- Construction: i processed at t in S'^* (1)
- Construction: j processed in $[t, eft_j[$ in S'^* (2)
- (1) + (2) $\Rightarrow$ CONTRADICTION !





Lemma: proof of case 3 c)

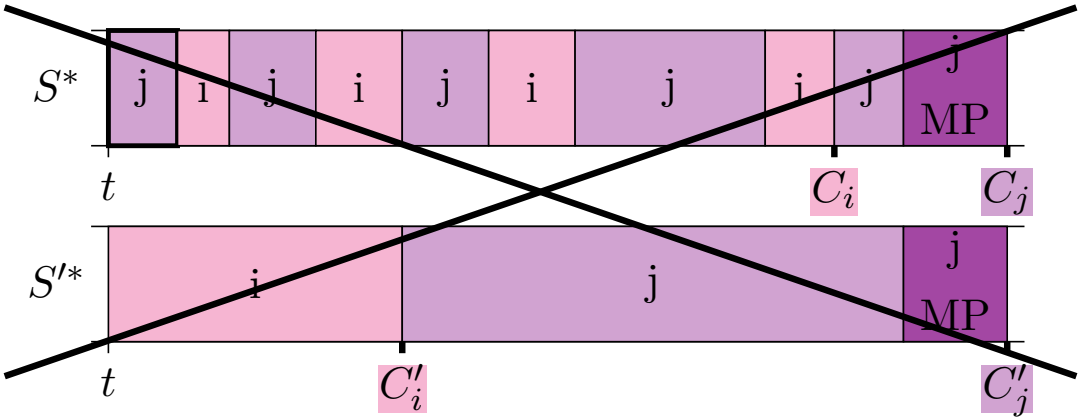
Sub-case c): i and j ends with their MP

- Sub-case a): j ends with its MP IMPOSSIBLE !
- Sub-case c): i and j ends with their MP $\Rightarrow$ IMPOSSIBLE !

Lemma: proof of case 3 c)

Sub-case c): i and j ends with their MP

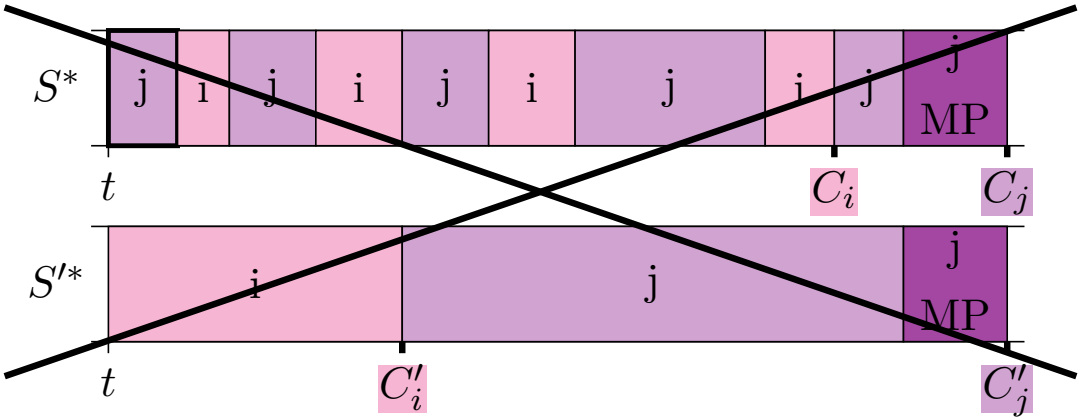
- Sub-case a): j ends with its MP IMPOSSIBLE !
- Sub-case c): i and j ends with their MP $\Rightarrow$ IMPOSSIBLE !



Lemma: proof of case 3 c)

Sub-case c): i and j ends with their MP

- Sub-case a): j ends with its MP IMPOSSIBLE !
- Sub-case c): i and j ends with their MP $\Rightarrow$ IMPOSSIBLE !



Lemma: proof of case 3 b)

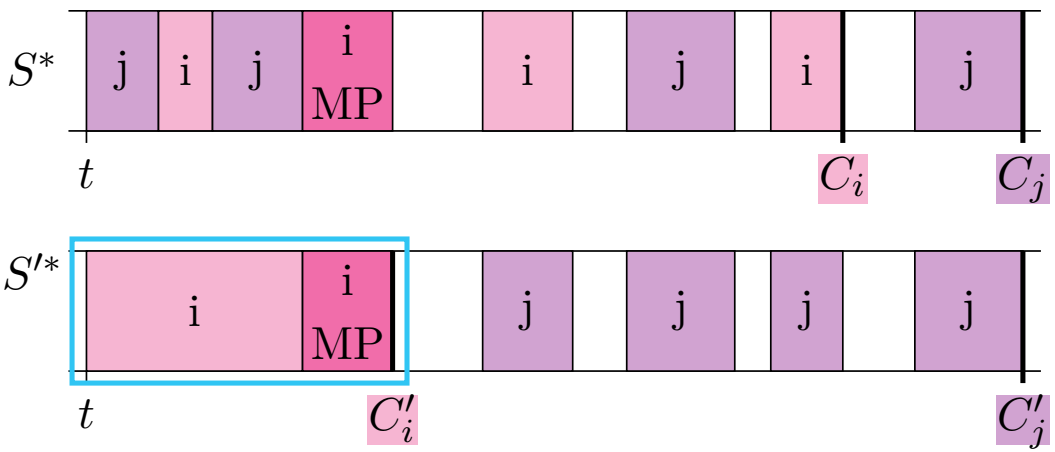
Sub-case b): i ends with its MP

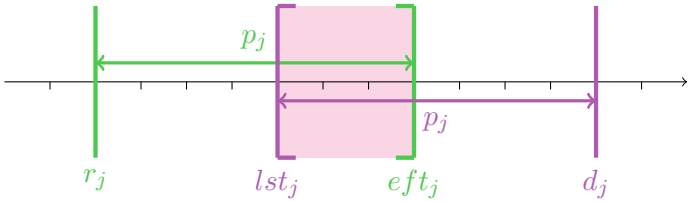
- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$
  - $C'_j = C_i$

Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$
  - $C'_j = C_i$

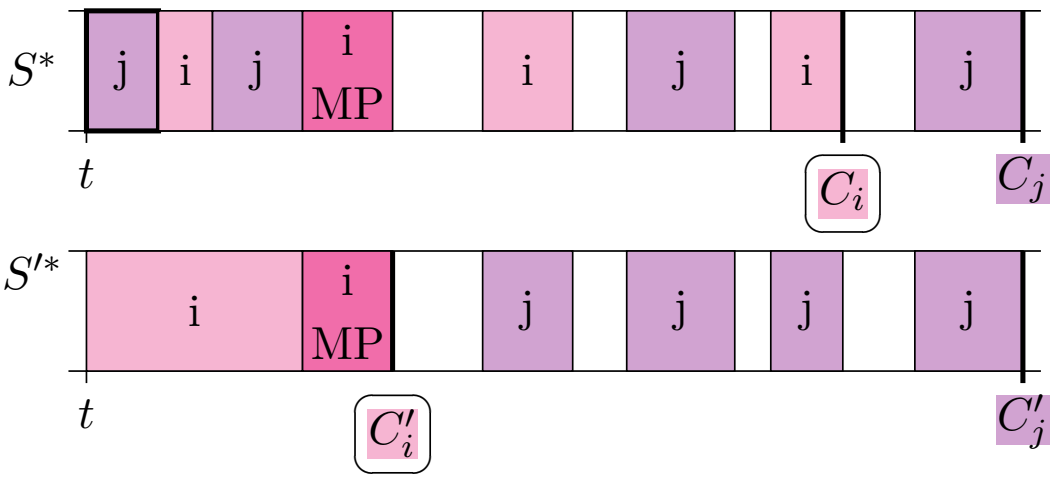


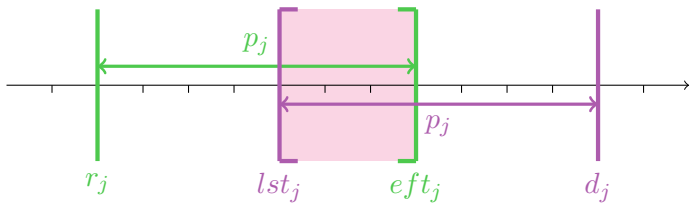


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$
  - $C'_j = C_i$

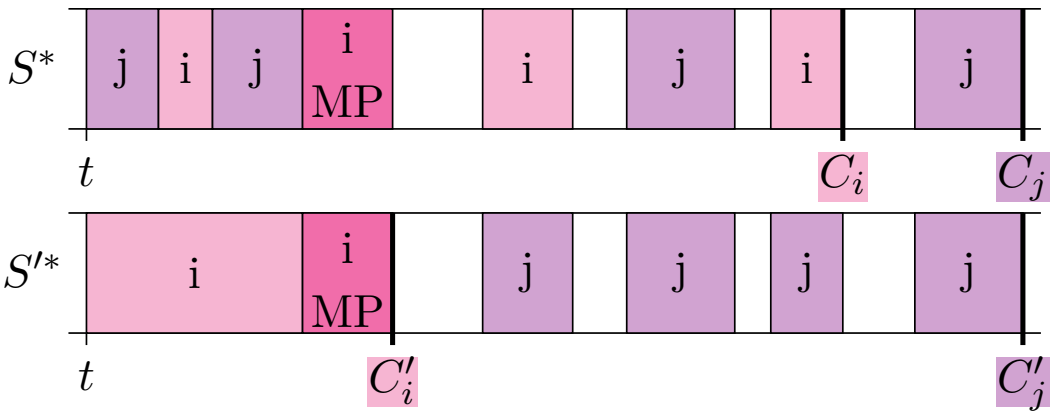


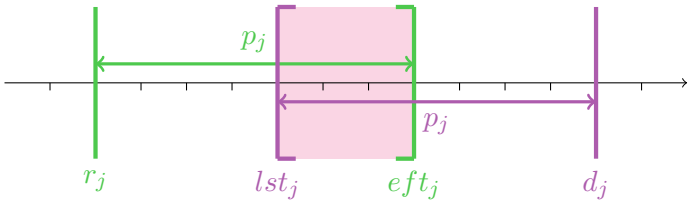


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_j + C'_j < C_j + C_j$
  - $C'_j = C_i$: $p_i < p_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_j + C'_j < C_j + C_j$

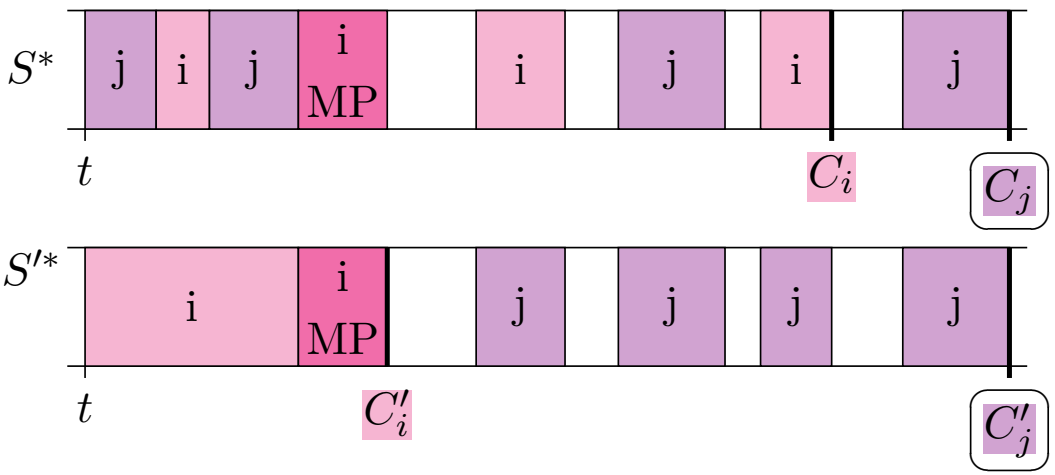


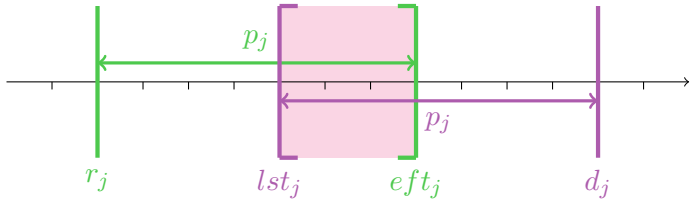


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_i + C'_j < C_i + C_j$
  - $C'_j = C_i$: $p_i < p_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

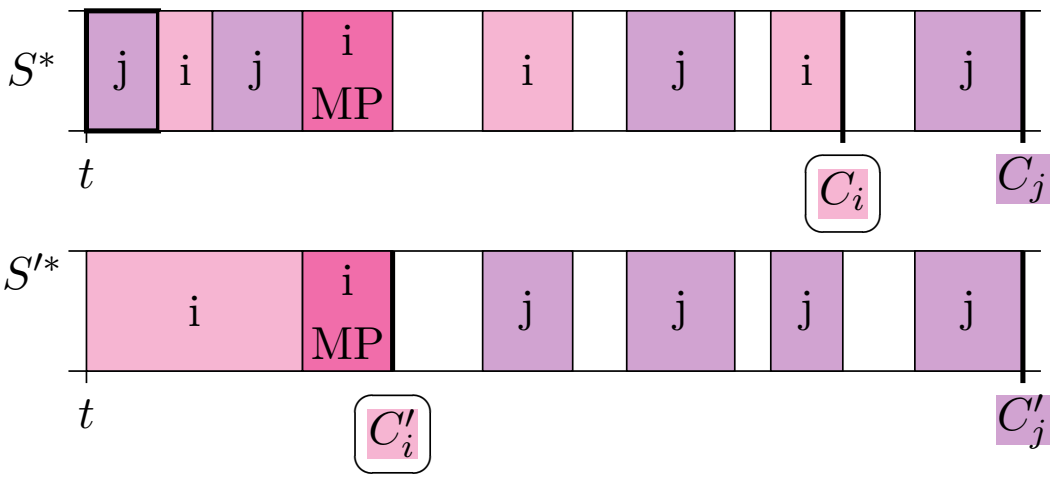


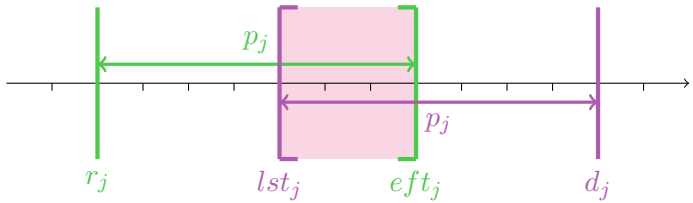


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_i + C'_j < C_i + C_j$
  - $C'_j = C_i$: $p_i < p_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

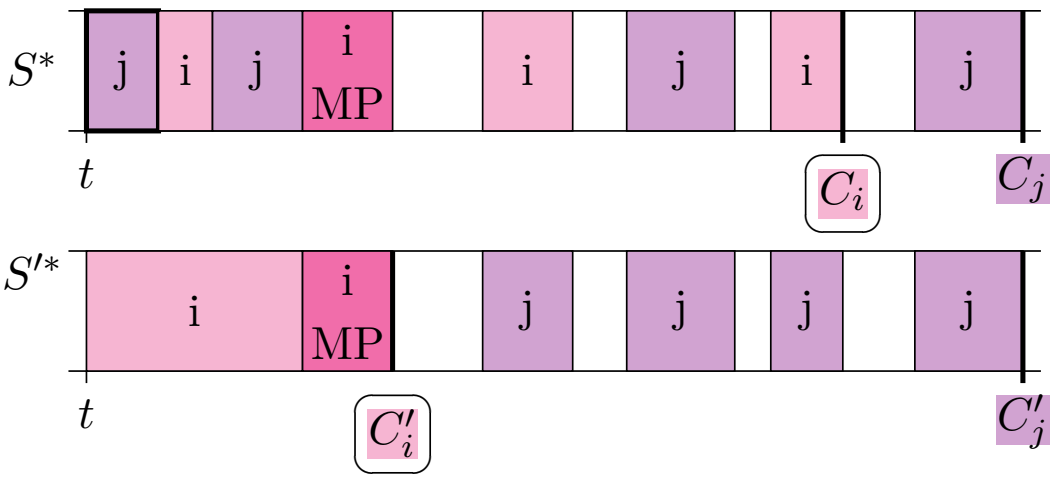


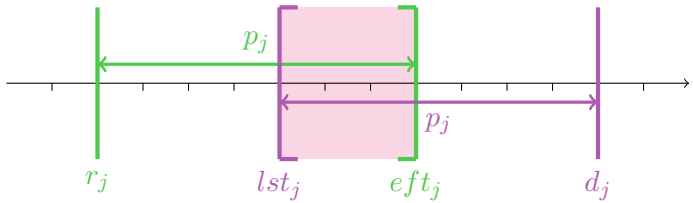


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_i + C'_j < C_i + C_j$
  - $C'_j = C_i$: $p_i < p_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

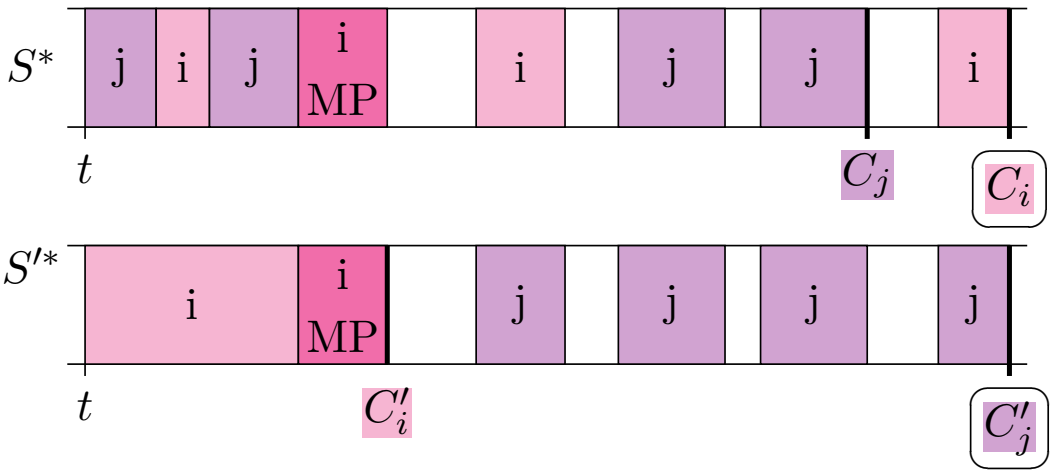


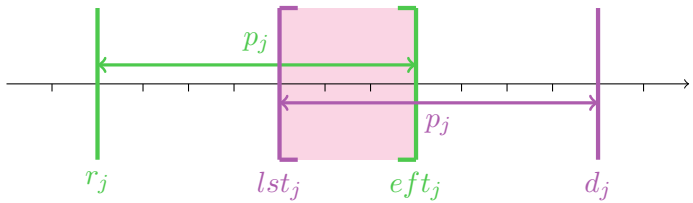


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_i + C'_j < C_i + C_j$
  - $C'_j = C_i$: $\widehat{p}_i < \widehat{p}_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

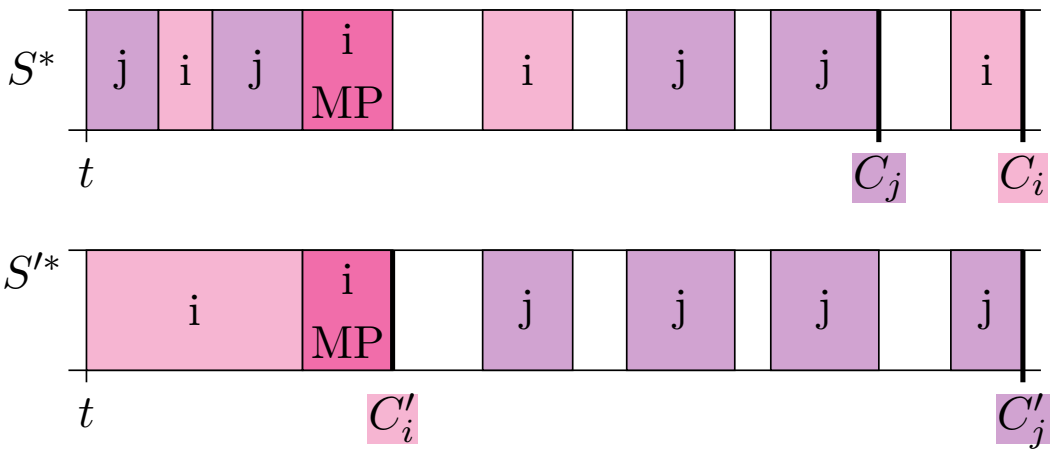


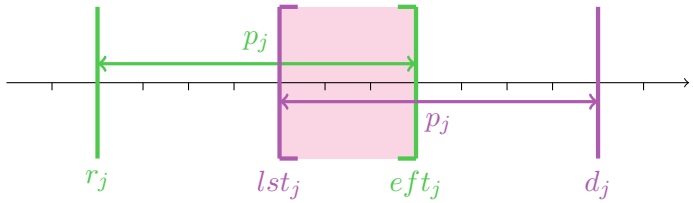


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_i + C'_j < C_i + C_j$
  - $C'_j = C_i$: $\widehat{p}_i < \widehat{p}_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

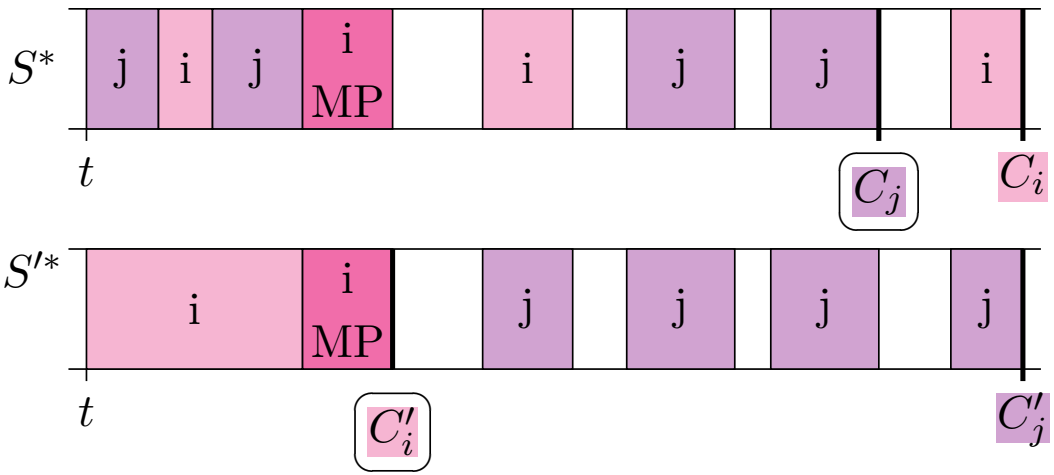


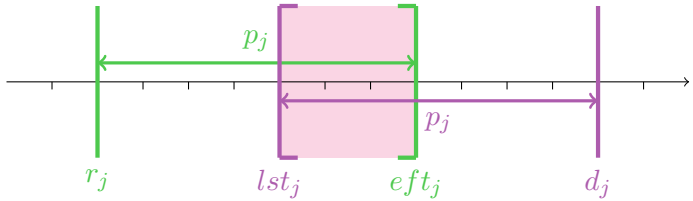


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_i + C'_j < C_i + C_j$
  - $C'_j = C_i$: $\widehat{p}_i < \widehat{p}_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

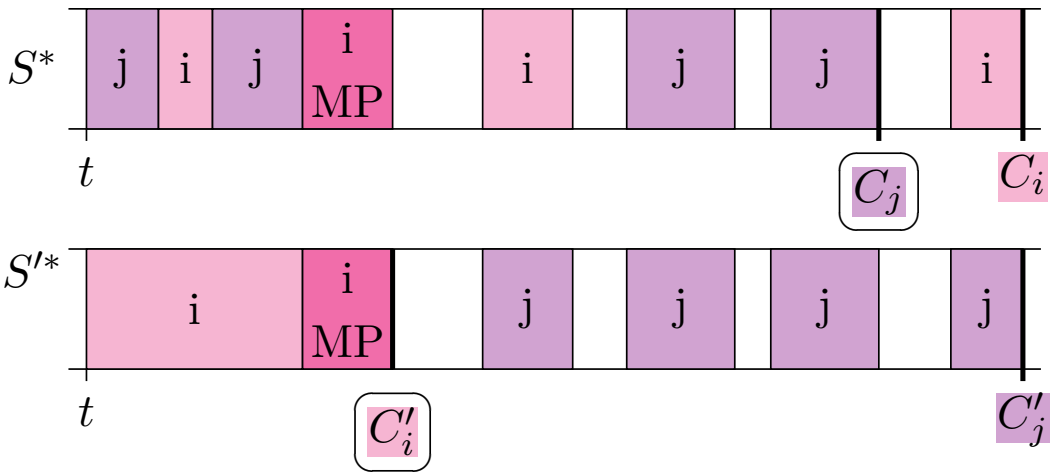


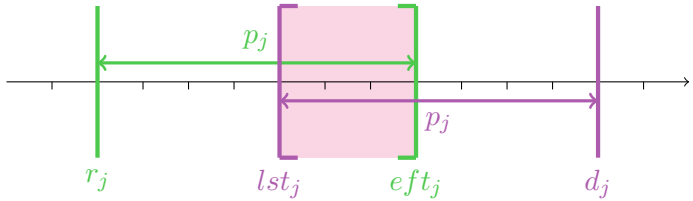


Lemma: proof of case 3 b)

Sub-case b): i ends with its MP

- Construction: i processed in $[t, eft_i[$ in S'^*
- $S^* \neq S'^*$ at time t : $C_i > C'_i = eft_i$ (1)
- $C'_j = \max(C_j, C_i)$
  - $C'_j = C_j$ and (1) $\Rightarrow C'_i + C'_j < C_i + C_j$
  - $C'_j = C_i$: $\widehat{p}_i < \widehat{p}_j$ (rule) $\Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$





3. Experimentations

Experimental protocol

570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- with **durations** in $[1, 10]$ or $[1, 100]$
- with **time windows** $[r_j, d_j]$ more or less close of those of a **possible schedule**

Problems studied

RD-CP	$1\|r_j; d_j\|\sum C_j$	OPT
RDP-MP-IP	$1\|r_j; d_j; pmtn; MP\|\sum C_j$	LB
RDP-IP	$1\|r_j; d_j; pmtn\|\sum C_j$	LB
RP-MP-S	$1\|r_j; pmtn; MP\|\sum C_j$	LB
RP-S	$1\|r_j; pmtn\|\sum C_j$	LB

Experimental protocol

570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- with **durations** in $[1, 10]$ or $[1, 100]$
- with **time windows** $[r_j, d_j]$ more or less close of those of a possible schedule

Problems studied

RD-CP	$1\|r_j; d_j\|\sum C_j$	OPT
RDP-MP-IP	$1\|r_j; d_j; pmtn; MP\|\sum C_j$	LB
RDP-IP	$1\|r_j; d_j; pmtn\|\sum C_j$	LB
RP-MP-S	$1\|r_j; pmtn; MP\|\sum C_j$	LB
RP-S	$1\|r_j; pmtn\|\sum C_j$	LB

Experimental protocol

570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- with **durations** in $[1, 10]$ or $[1, 100]$
- with **time windows** $[r_j, d_j]$ more or less close of those of a possible schedule

Problems studied

RD-CP	$1\|r_j; d_j\|\sum C_j$	OPT
RDP-MP-IP	$1\|r_j; d_j; pmtn; MP\|\sum C_j$	LB
RDP-IP	$1\|r_j; d_j; pmtn\|\sum C_j$	LB
RP-MP-S	$1\|r_j; pmtn; MP\|\sum C_j$	LB
RP-S	$1\|r_j; pmtn\|\sum C_j$	LB

Experimental protocol

570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- with **durations** in $[1, 10]$ or $[1, 100]$
- with **time windows** $[r_j, d_j]$ more or less close of those of a **possible schedule**

Problems studied

RD-CP	$1\|r_j; d_j\|\sum C_j$	OPT
RDP-MP-IP	$1\|r_j; d_j; pmtn; MP\|\sum C_j$	LB
RDP-IP	$1\|r_j; d_j; pmtn\|\sum C_j$	LB
RP-MP-S	$1\|r_j; pmtn; MP\|\sum C_j$	LB
RP-S	$1\|r_j; pmtn\|\sum C_j$	LB

Experimental protocol

570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- with **durations** in $[1, 10]$ or $[1, 100]$
- with **time windows** $[r_j, d_j]$ more or less close of those of a **possible schedule**

Problems studied

RD-CP	$1\|r_j; d_j\|\sum C_j$	OPT
RDP-MP-IP	$1\|r_j; d_j; pmtn; MP\|\sum C_j$	LB
RDP-IP	$1\|r_j; d_j; pmtn\|\sum C_j$	LB
RP-MP-S	$1\|r_j; pmtn; MP\|\sum C_j$	LB
RP-S	$1\|r_j; pmtn\|\sum C_j$	LB

Experimental protocol

570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- with **durations** in $[1, 10]$ or $[1, 100]$
- with **time windows** $[r_j, d_j]$ more or less close of those of a **possible schedule**

Problems studied

RD-CP	$1\|r_j; d_j\|\sum C_j$	OPT
RDP-MP-IP	$1\|r_j; d_j; pmtn; MP\|\sum C_j$	LB
RDP-IP	$1\|r_j; d_j; pmtn\|\sum C_j$	LB
RP-MP-S	$1\|r_j; pmtn; MP\|\sum C_j$	LB
RP-S	$1\|r_j; pmtn\|\sum C_j$	LB

Experimentations

Gaps to the optimum analysis: LB ÷ OPT (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

Remarks

- RP-S $\leq$ RP-MP-S $\leq$ RDP-IP $\leq$ RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ($\mathcal{NP}$-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

Experimentations

Gaps to the optimum analysis: LB ÷ OPT (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

Remarks

- RP-S $\leq$ RP-MP-S $\leq$ RDP-IP $\leq$ RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ($\mathcal{NP}$-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

Experimentations

Gaps to the optimum analysis: LB ÷ OPT (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

Remarks

- RP-S $\leq$ RP-MP-S $\leq$ RDP-IP $\leq$ RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ($\mathcal{NP}$-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

Experimentations

Gaps to the optimum analysis: LB ÷ OPT (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

Remarks

- RP-S $\leq$ RP-MP-S $\leq$ RDP-IP $\leq$ RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ($\mathcal{NP}$-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

Experimentations

Gaps to the optimum analysis: LB ÷ OPT (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

Remarks

- RP-S $\leq$ RP-MP-S $\leq$ RDP-IP $\leq$ RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ($\mathcal{NP}$-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

Experimentations

Gaps to the optimum analysis: LB ÷ OPT (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

Remarks

- RP-S $\leq$ RP-MP-S $\leq$ RDP-IP $\leq$ RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ($\mathcal{NP}$-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

Experimentations

Gaps to the optimum analysis: LB ÷ OPT (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

Remarks

- RP-S $\leq$ RP-MP-S $\leq$ RDP-IP $\leq$ RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ($\mathcal{NP}$-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

4. Conclusion

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Conclusion

Conclusion

- MSR with MP compute LB for $1|r_j; d_j|\sum C_j$ in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of $1|r_j; d_j; pmtn|\sum C_j$
- No direct integration of d_j

Prospects

Prospects

- Partial integration of d_j
- CP applications
- ...

Prospects

Prospects

- Partial integration of d_j
- CP applications
- ...

Prospects

Prospects

- Partial integration of d_j
- CP applications
- ...

Prospects

Prospects

- Partial integration of d_j
- CP applications
- ...

Thank you for your attention !

5. Appendix

Sweep line algo VS list algo

Differences

- List: sorts elements with the **same level**
- Sweep line: sorts elements with the **different levels**

- List: all elements have **same actions**
- Sweep line: elements can have **different actions**