離散最適化基礎論

ジョブ・スケジューリングのアルゴリズム

第 2 回

整列による解法

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2024 年 10 月 22 日

最終更新：2024 年 10 月 23 日　09:47

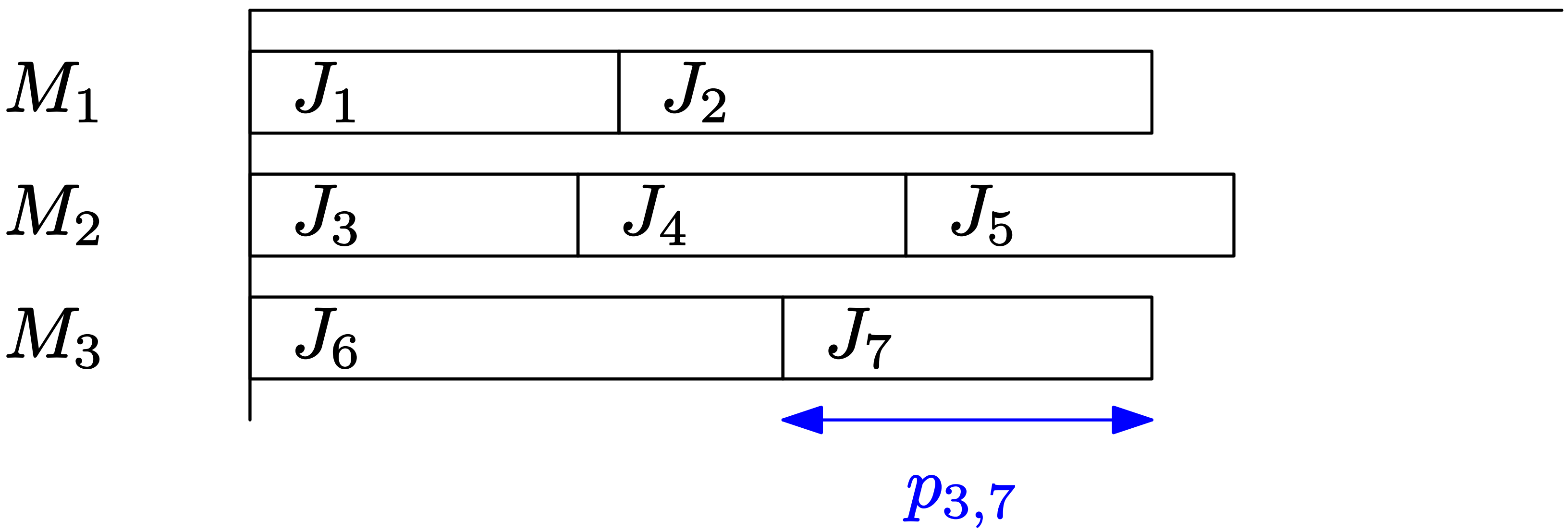
スケジュール (前半)

1. スケジューリング問題の分類	(10/1)
＊ 休み (出張)	(10/8)
＊ 休み (体育祭)	(10/15)
2. **整列による解法**	(10/22)
3. 動的計画法	(10/29)
4. NP 困難性と計算量の分類	(11/5)
5. 計算複雑性による問題の分類	(11/12)
6. リスト・スケジューリング	(11/19)

スケジュール (後半)

7. 先行制約：基礎 (11/26)
＊ 休み (秋ターム試験) (12/3)
8. 先行制約：多機械 (12/10)
9. 先行制約：他の半順序 (12/17)
10. ショップ・スケジューリング：基礎 (12/24)
＊ 休み (冬季休業) (12/31)
11. ショップ・スケジューリング：機械数が定数 (1/7)
12. ショップ・スケジューリング：機械数が可変 (1/14)
13. 近似可能性と近似不可能性 (1/21)
14. 多項式時間近似スキーム (1/28)
＊ なし (2/4)

ジョブ・スケジューリングの構成要素



- 機械の集合　$\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$
  添字 $i \in \{1, 2, \ldots, m\} =: [m]$
- ジョブの集合 (仕事)　$\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$
  添字 $j \in \{1, 2, \ldots, n\} =: [n]$
- 処理時間　$p \in \mathbb{Q}_{\geq 0}^{m \times n}$ (非負有理数の行列)
  $p_{ij} = M_i$ における J_j の処理時間

まとめ：一機械，ジョブの特性なし

$\boxed{\text{まとめ}}$: $1 \,||\, \gamma$

γ		計算量	(文献)
$C_{\max}$	最大完了時刻	$O(n)$	
$\sum C_j$	総完了時刻	$O(n \log n)$	(Smith ’56)
$L_{\max}$	最大納期ずれ	$O(n \log n)$	(Jackson ’55)
$\sum T_j$	総納期遅れ	$O(n^4 \sum p_j)$	(Lawler ’77)
		NP 困難	(Du, Leung ’90)
$\sum U_j$	納期遅れジョブ数	$O(n \log n)$	(Moore ’68)

重要なメッセージ：アルゴリズム全般において

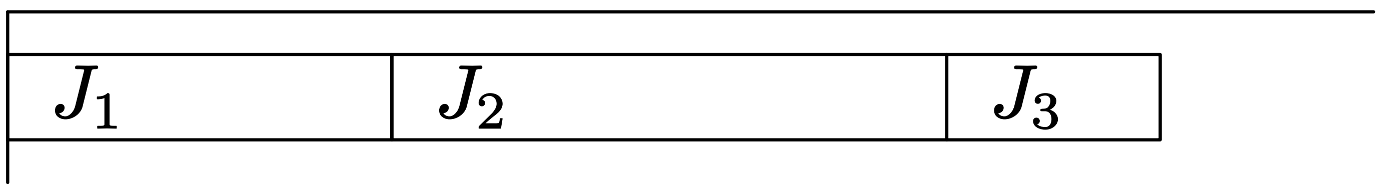
設定における少しの違いが
解きやすさにおける大きな違いを生む

最大完了時刻最小化

観察：$1 \mid\mid C_{\max}$ の最適値

$1 \mid\mid C_{\max}$ の最適値は $\displaystyle\sum_{j \in [n]} p_j$

$p_j =$ ジョブ J_j の処理時間

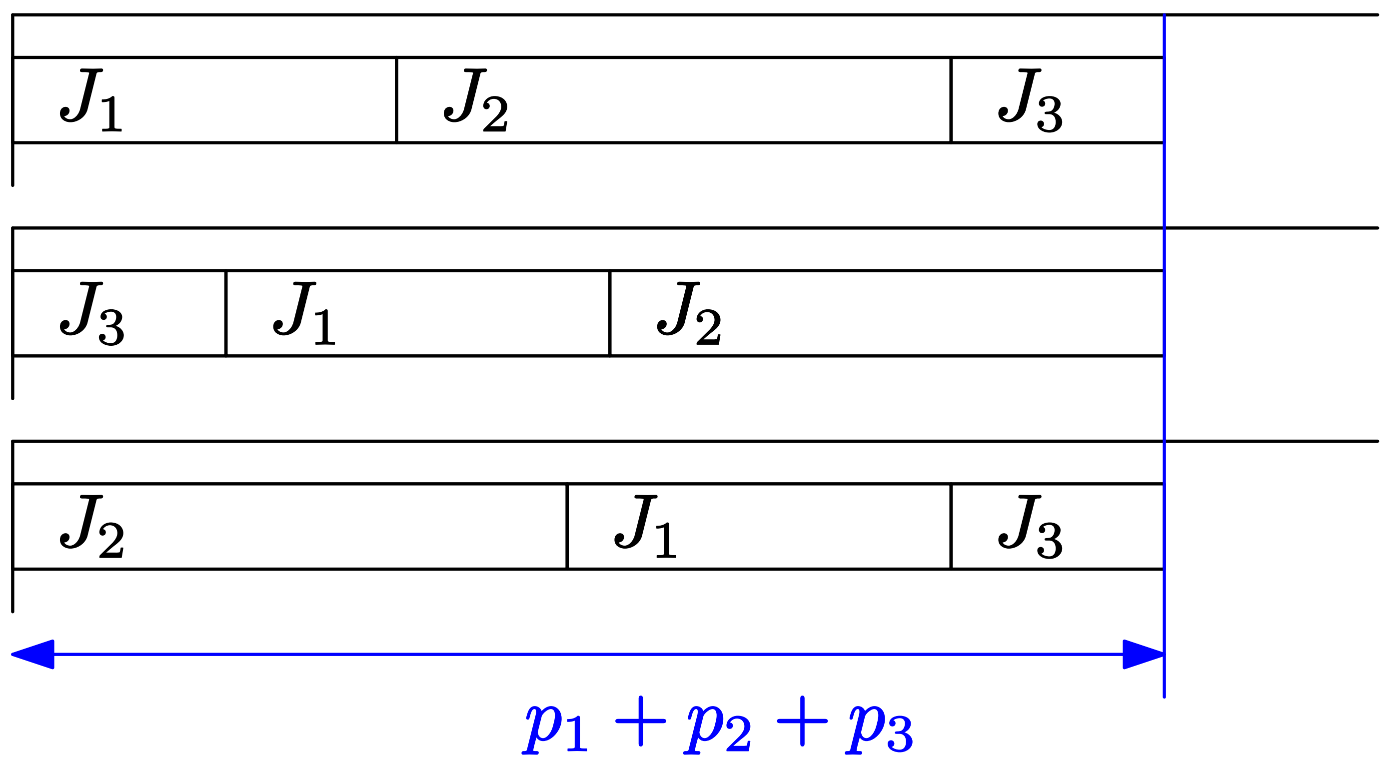


最大完了時刻最小化

観察：$1 \mid\mid C_{\max}$ の最適値

$1 \mid\mid C_{\max}$ の最適値は $\displaystyle\sum_{j \in [n]} p_j$

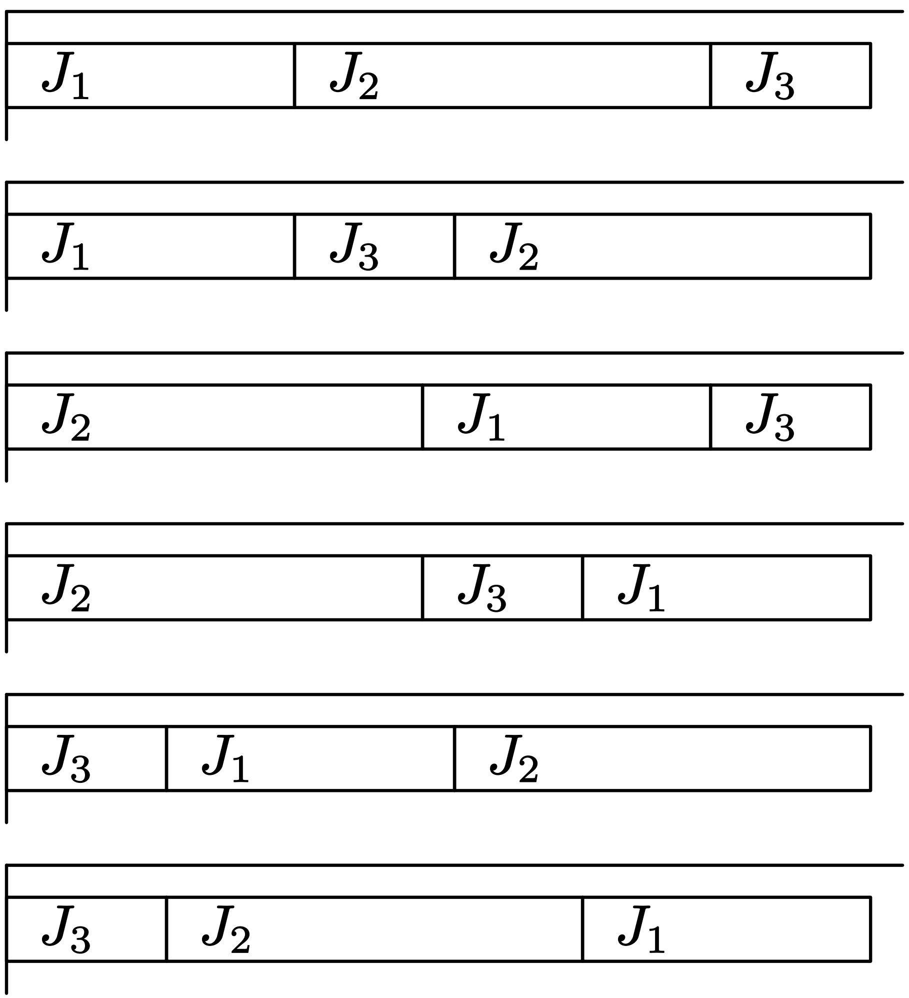
p_j = ジョブ J_j の処理時間



一機械：「自明」な解法

観察：$1 \mid\mid \gamma$ の解法

$1 \mid\mid \gamma$ はすべての並べ方を調べれば解ける



$n!$ 個

$$\sim \sqrt{2\pi n}\left(\frac{n}{\mathrm{e}}\right)^n$$

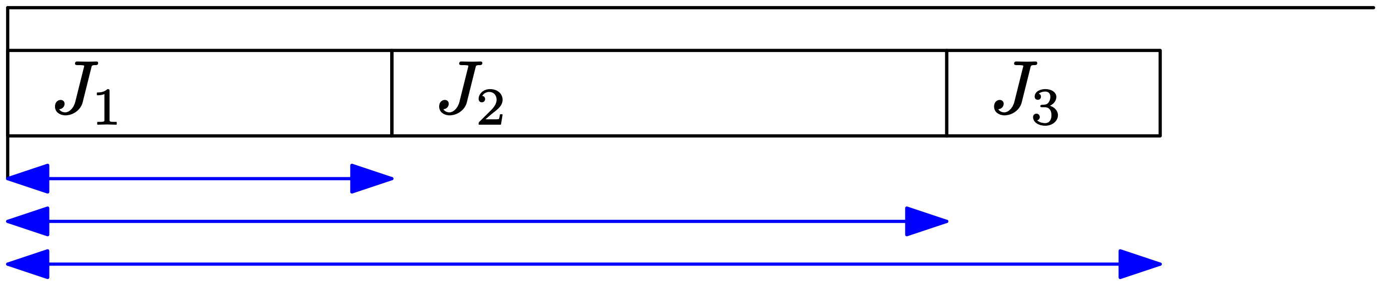
(Stirling の公式)

本日の内容

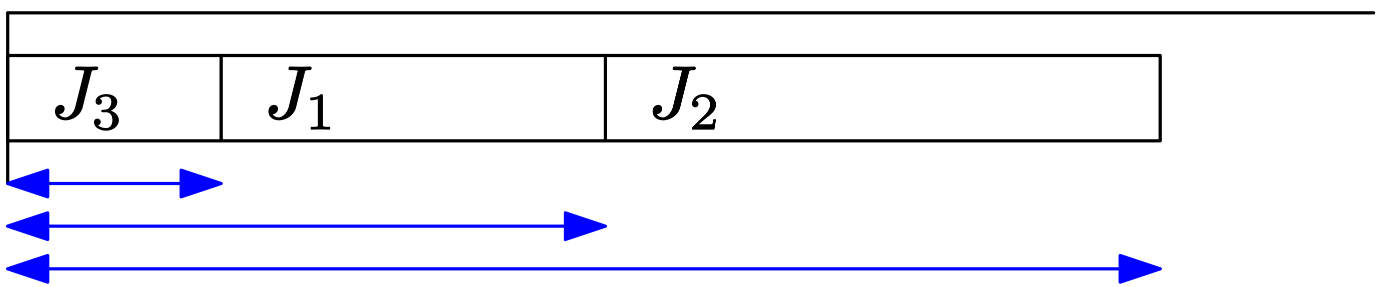
1. **総完了時刻最小化と最短処理時間優先規則**
2. 最大納期ずれ最小化と最早納期優先規則
3. 納期遅れジョブ数最小化と Moore-Hodgson アルゴリズム

- W.E. Smith, Various optimizers for single stage production, *Naval Research Logistics Quarterly* **3** (1956) 59–66.

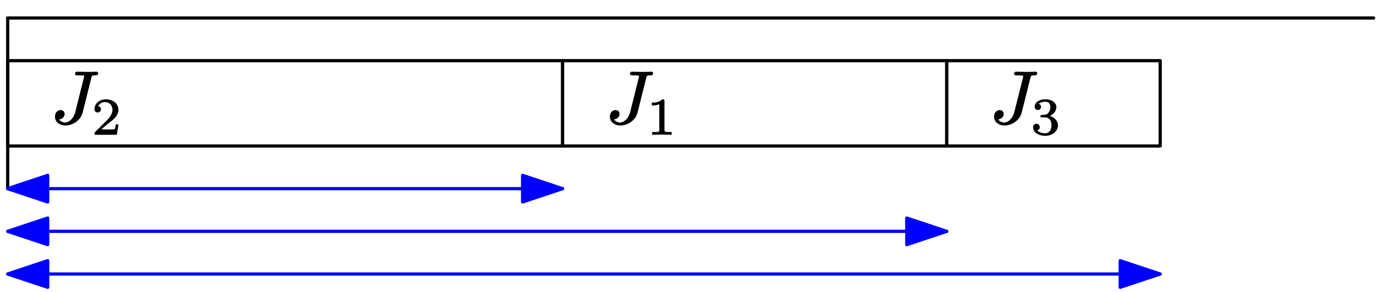
総完了時刻最小化：例



$$9 + (9 + 13) + (9 + 13 + 5) = 58$$

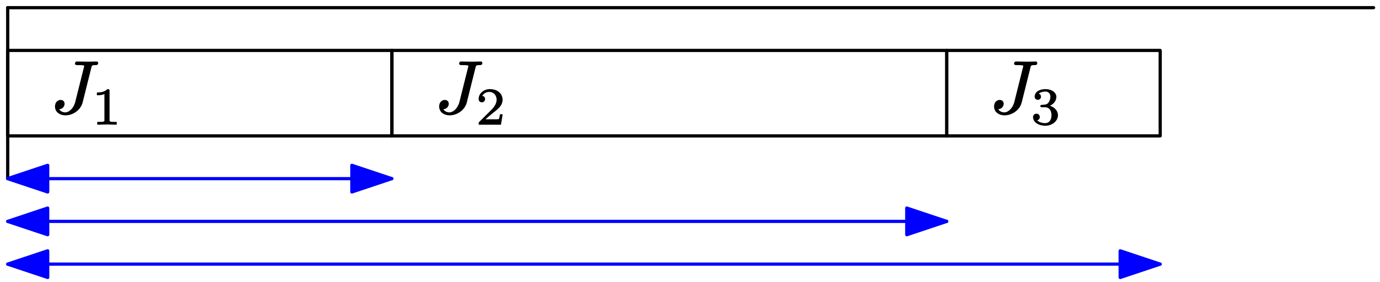


$$5 + (5 + 9) + (5 + 9 + 13) = 46$$

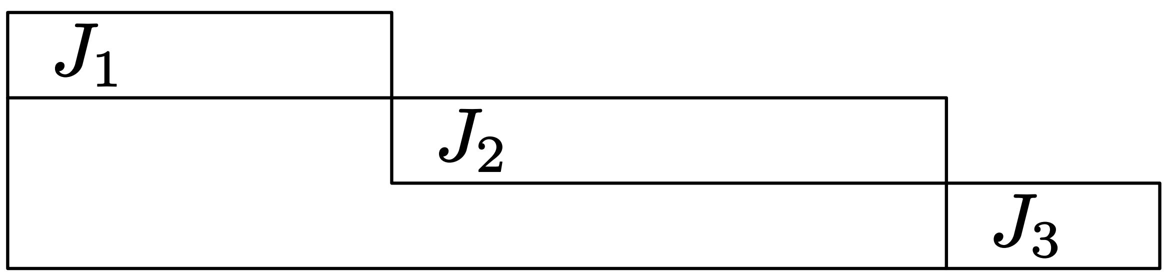


$$13 + (13 + 9) + (13 + 9 + 5) = 62$$

総完了時刻最小化：図 (1/2)



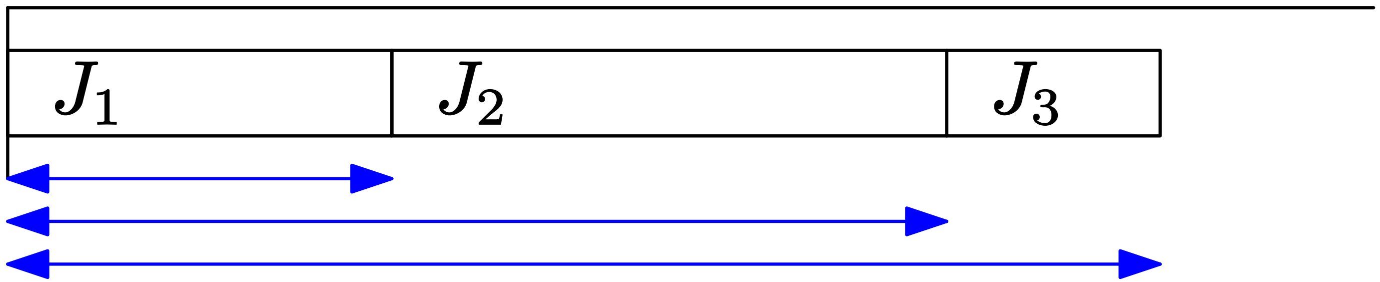
$$9 + (9 + 13) + (9 + 13 + 5) = 58$$



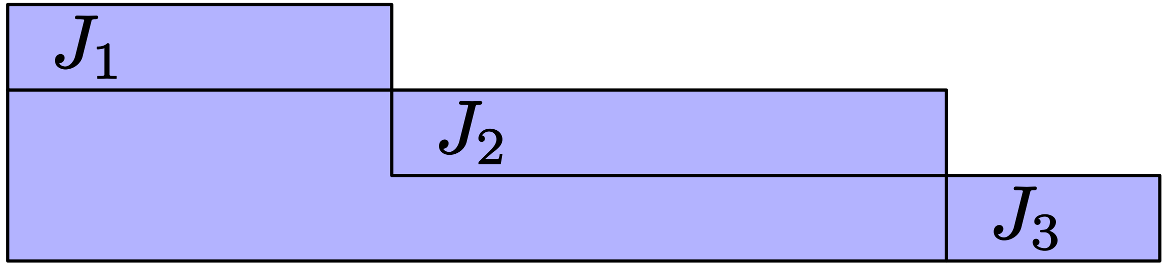
重要なメッセージ：理工学全般において

図を描いて考える，図を描けるように考える

総完了時刻最小化：図 (1/2)



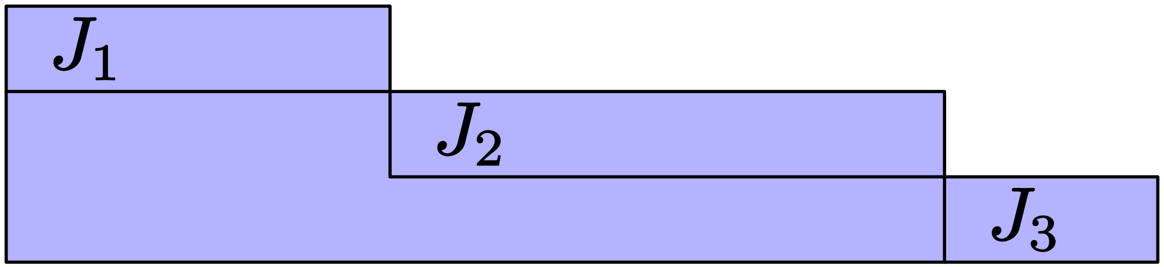
$9 + (9 + 13) + (9 + 13 + 5) = 58$



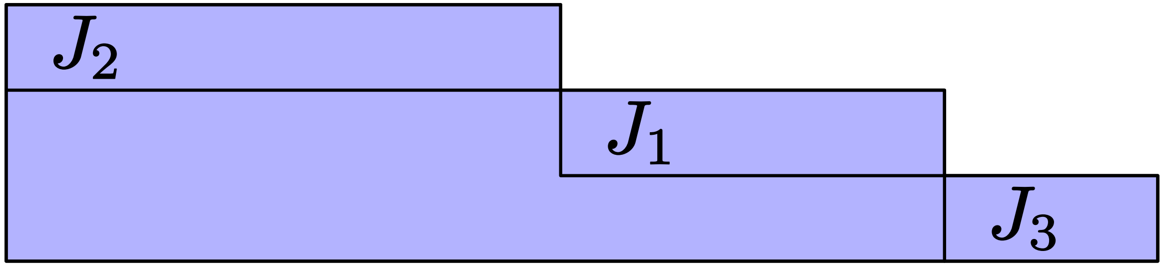
面積 $= 58$

重要なメッセージ：理工学全般において

図を描いて考える，図を描けるように考える

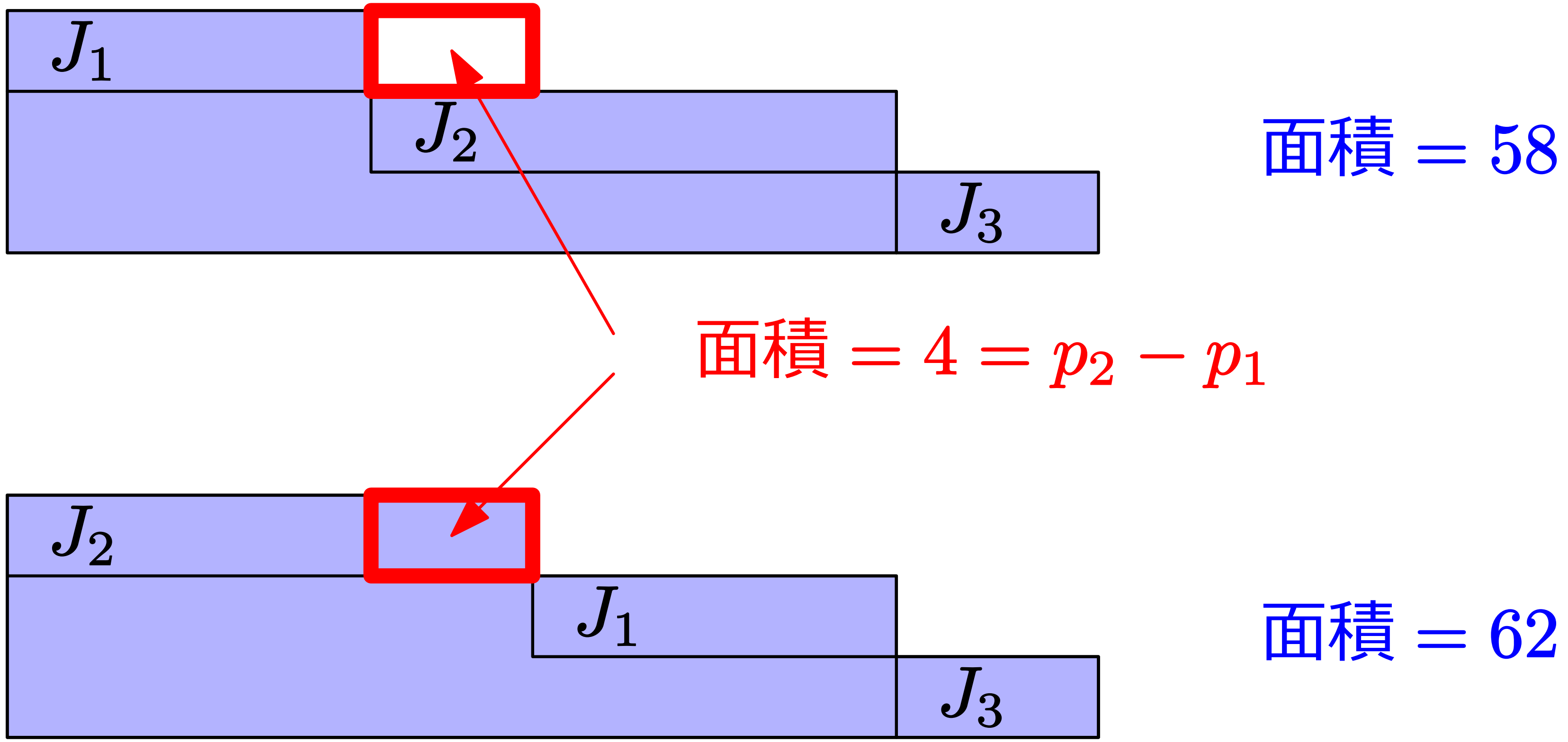


面積 $= 58$

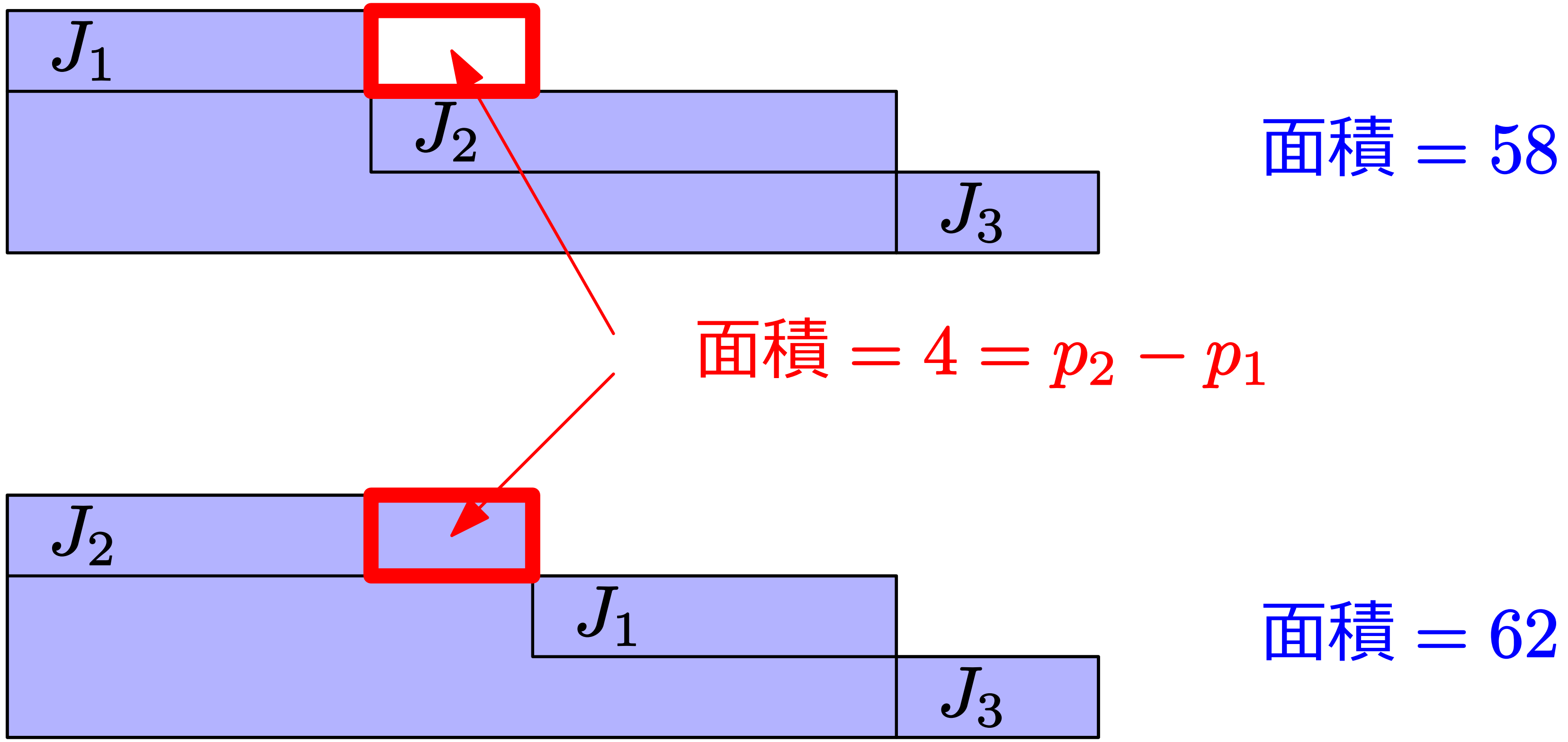


面積 $= 62$

総完了時刻最小化：図 (2/2)



総完了時刻最小化：図 (2/2)

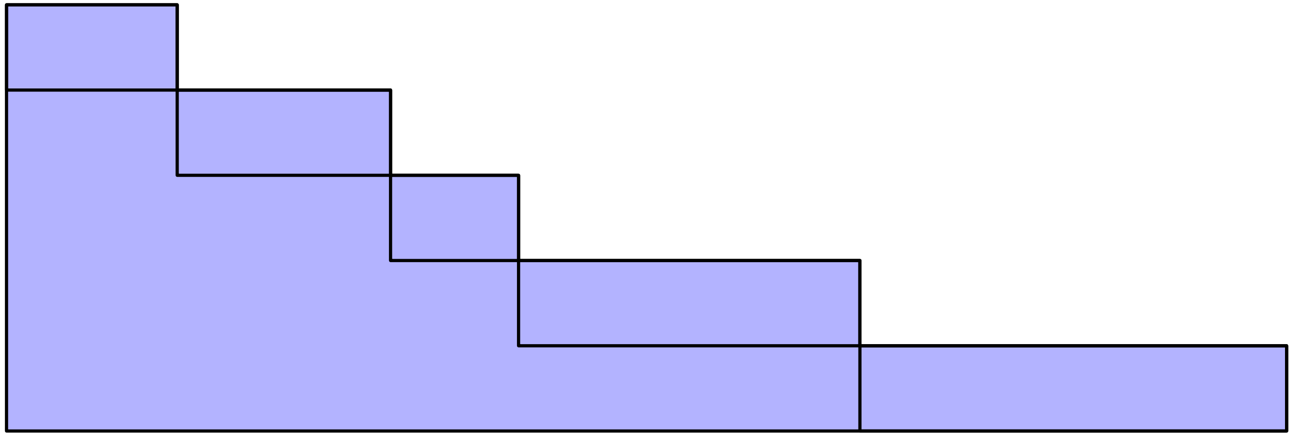


アルゴリズムのアイディア

処理時間の短いジョブから処理するとよさそう

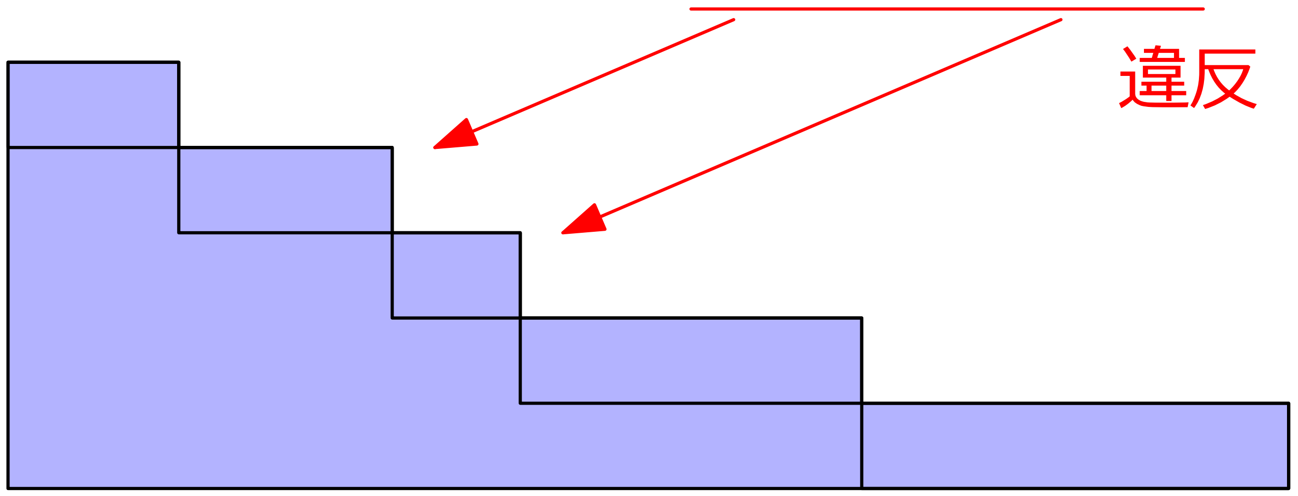
最適性：図で説明

処理時間が短い方から順に処理していないスケジュール



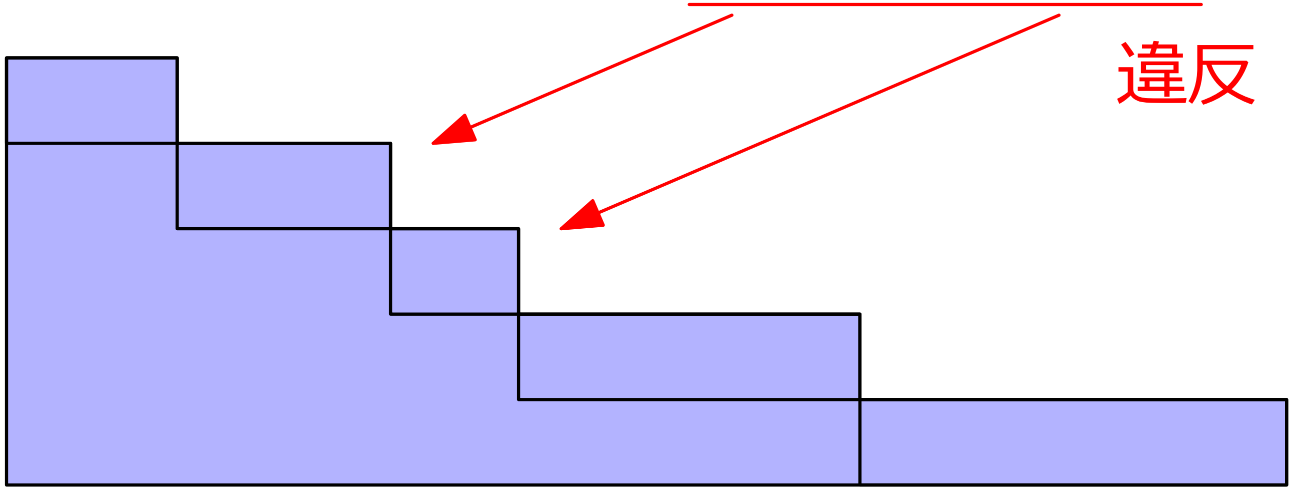
最適性：図で説明

処理時間が短い方から順に処理していないスケジュール



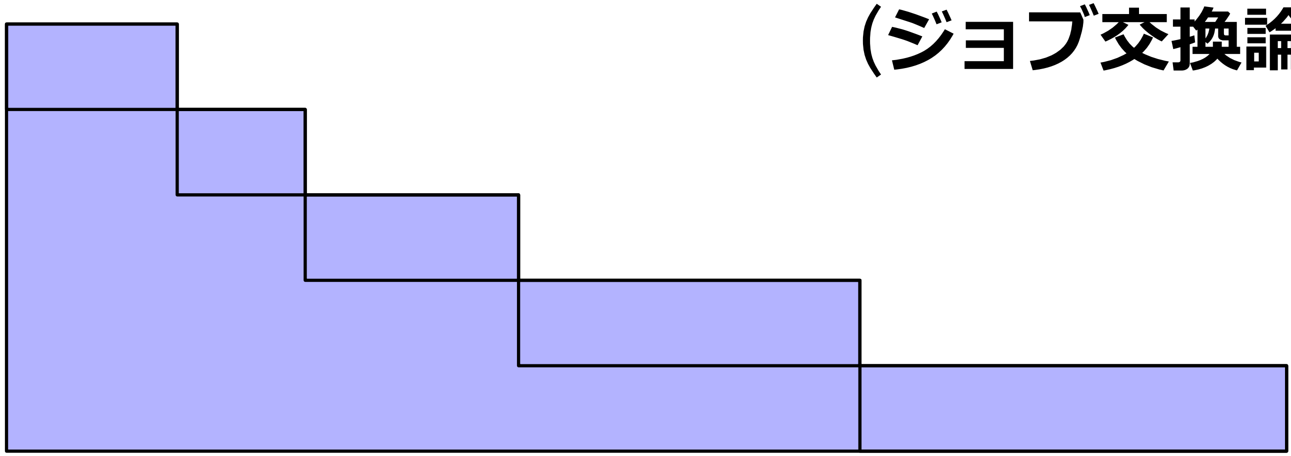
最適性：図で説明

処理時間が短い方から順に処理していないスケジュール



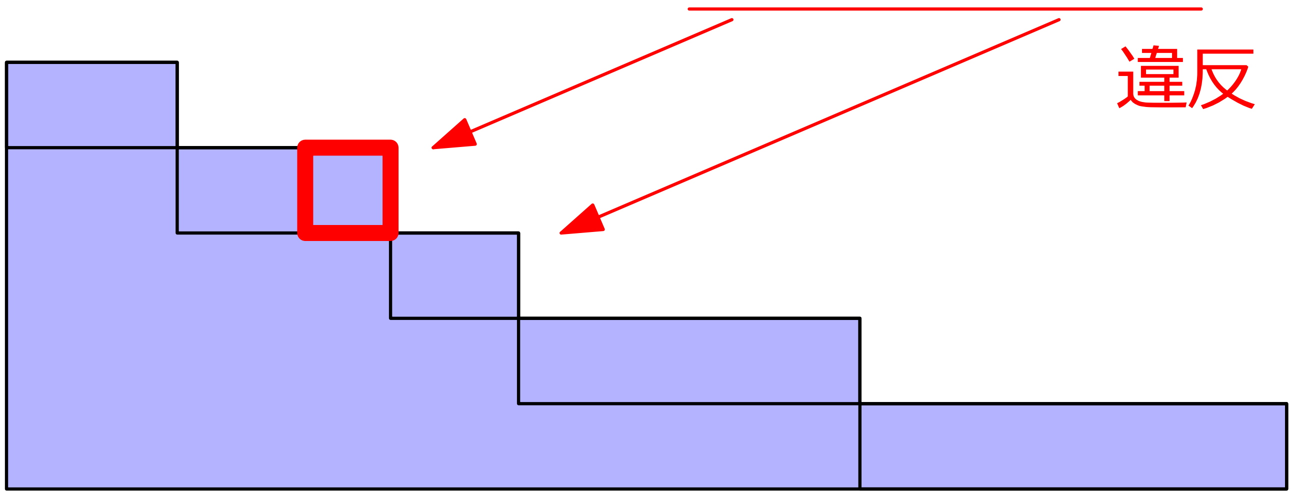
隣り合う違反ジョブを入れ替えたスケジュール

(ジョブ交換論法)



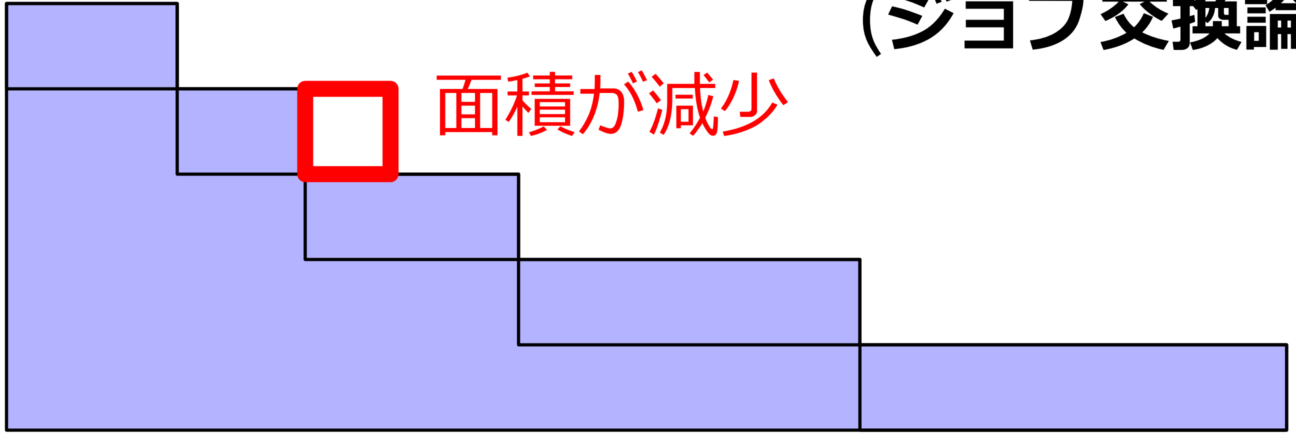
最適性：図で説明

処理時間が短い方から順に処理していないスケジュール

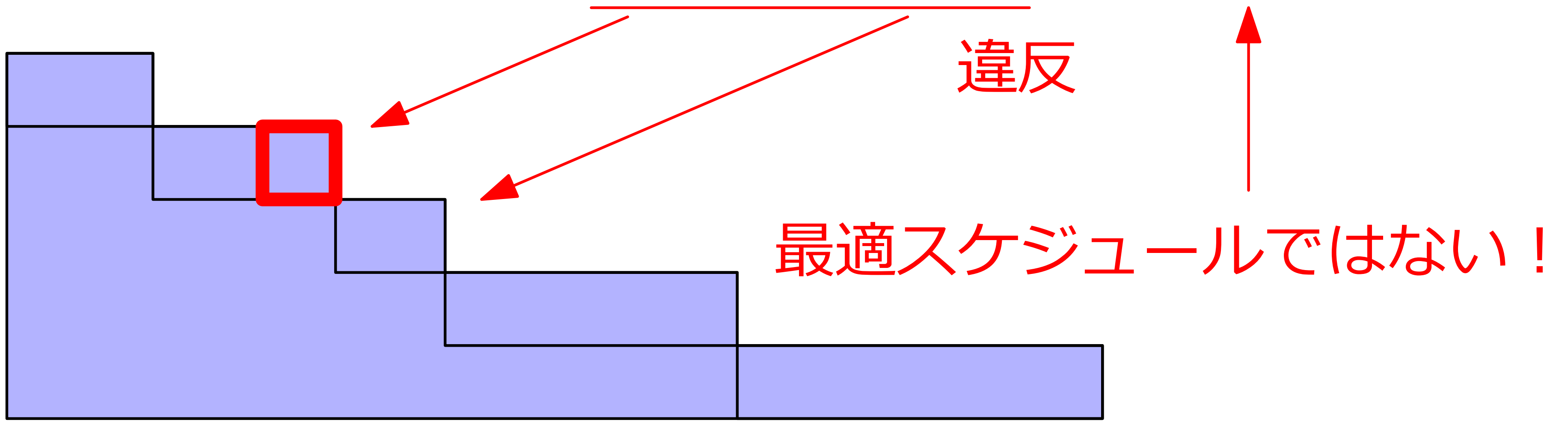


隣り合う違反ジョブを入れ替えたスケジュール

(ジョブ交換論法)

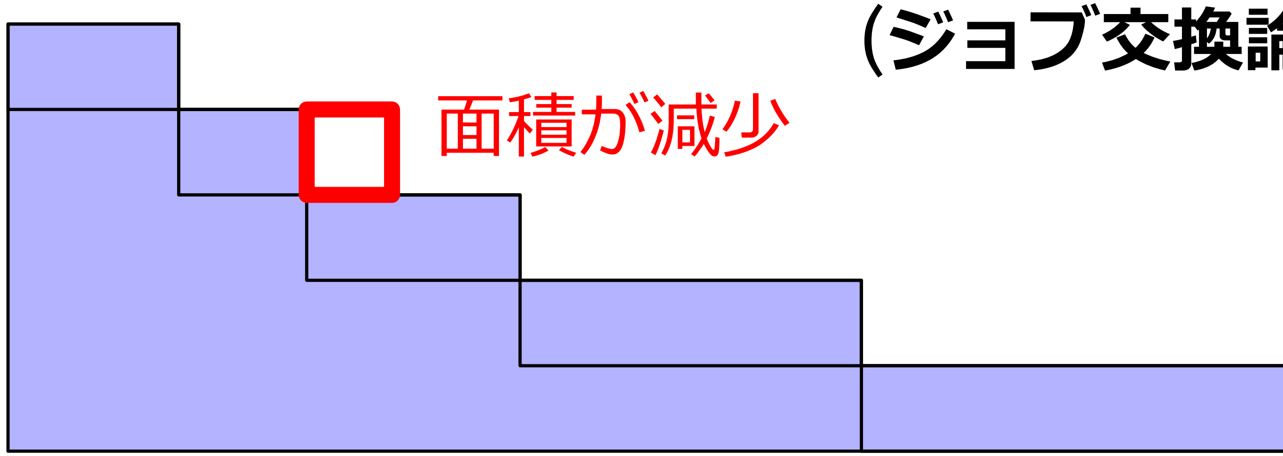


処理時間が短い方から順に処理していないスケジュール



隣り合う違反ジョブを入れ替えたスケジュール

(ジョブ交換論法)



定理：最短処理時間優先規則

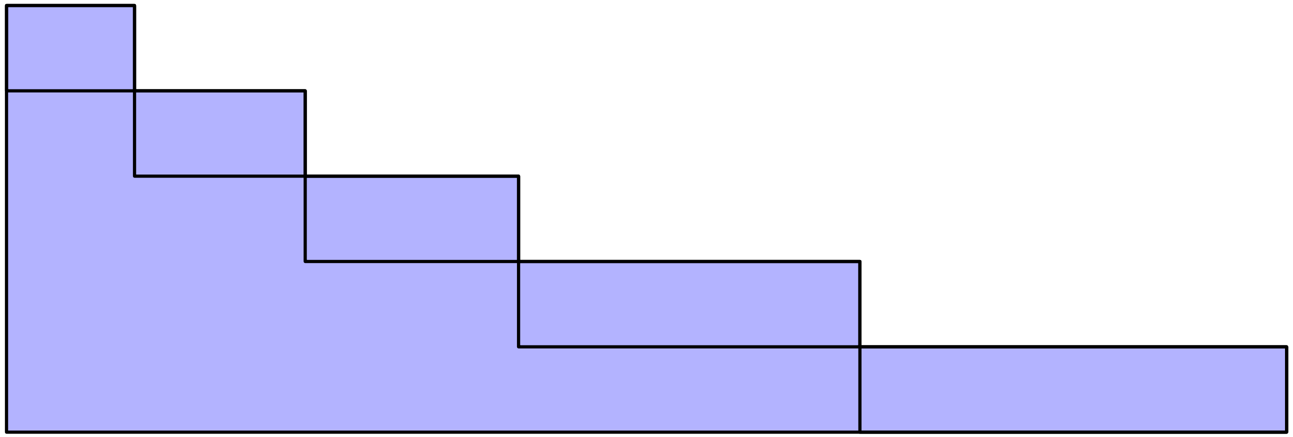
定理：$1 \mid\mid \sum C_j$ のアルゴリズム (Smith ’56)

$1 \mid\mid \sum C_j$ に対して，次のアルゴリズムは最適解を与える

アルゴリズム：最短処理時間優先規則 (SPT)

1. 処理時間が短い順にジョブを並べる
2. その順に従ってジョブを処理する

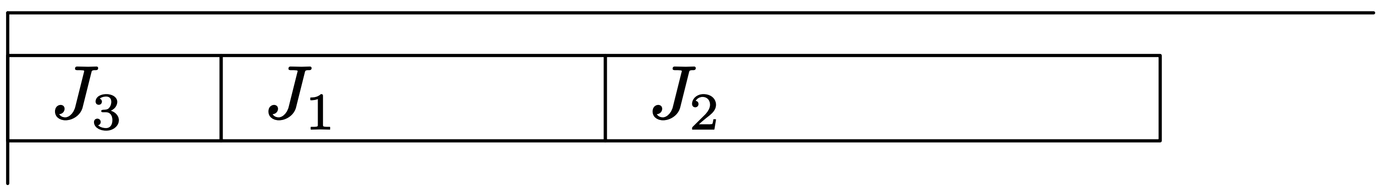
SPT $=$ Shortest Processing Time



ことばで証明するには…

$1 \mid\mid \gamma$ では，ジョブの処理順を決めたい

- 処理順を **置換** $\sigma \colon [n] \to [n]$ で表す
  $\sigma(k) = k$ 番目に処理するジョブの添字

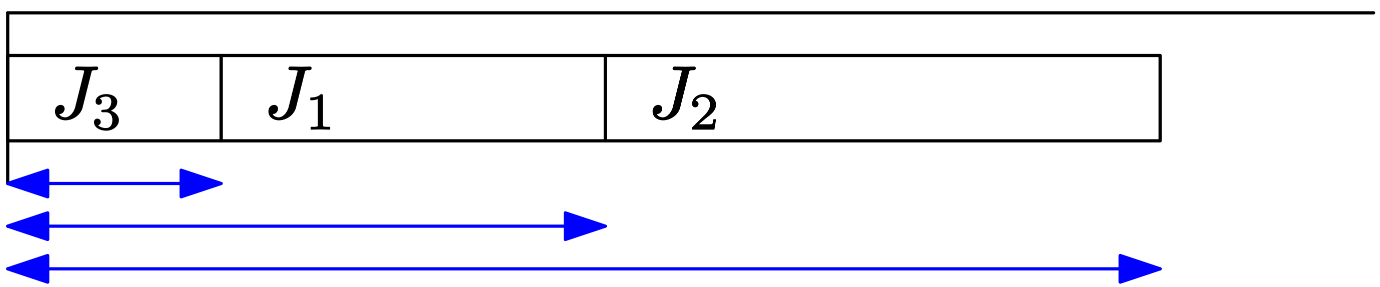


$$\sigma(1) = 3, \sigma(2) = 1, \sigma(3) = 2$$

ことばで証明するには· · ·

$1 \mid\mid \gamma$ では，ジョブの処理順を決めたい

- 処理順を **置換** $\sigma\colon [n] \to [n]$ で表す
  $\sigma(k) = k$ 番目に処理するジョブの添字



$$\sigma(1) = 3, \sigma(2) = 1, \sigma(3) = 2$$

- 処理順 σ における総完了時刻 $\sum C_j(\sigma)$ は次の式で書ける

$$\sum C_j(\sigma) = \sum_{k=1}^{n} (n - k + 1) p_{\sigma(k)}$$

- 置換 $\tau\colon [n] \to [n]$ を考える
- ある ℓ に対して，$p_{\tau(\ell)} > p_{\tau(\ell+1)}$ が成り立つとする

意味：τ では，処理時間の短い順になっていない

$$\tau \to p_{\tau(1)} \lesseqgtr \cdots \lesseqgtr p_{\tau(\ell)} > p_{\tau(\ell+1)} \lesseqgtr \cdots \lesseqgtr p_{\tau(n)}$$

最適性：ことばで説明 (あるいは，証明)

- 置換 $\tau\colon [n] \to [n]$ を考える
- ある ℓ に対して，$p_{\tau(\ell)} > p_{\tau(\ell+1)}$ が成り立つとする
- $\tau(\ell)$ と $\tau(\ell+1)$ を入れ替えた置換を σ とする

$$\sigma(k) = \begin{cases} \tau(k) & (k \neq \ell, \ell+1), \\ \tau(\ell+1) & (k = \ell), \\ \tau(\ell) & (k = \ell+1) \end{cases}$$

$$\tau \to p_{\tau(1)} \leqq \cdots \leqq p_{\tau(\ell)} > p_{\tau(\ell+1)} \leqq \cdots \leqq p_{\tau(n)}$$

$$\sigma \to p_{\tau(1)} \leqq \cdots \leqq p_{\tau(\ell+1)} < p_{\tau(\ell)} \leqq \cdots \leqq p_{\tau(n)}$$

最適性：ことばで説明 (続)

- このとき

$$
\begin{aligned}
&\sum C_j(\tau) \\
= &\sum_{k=1}^{n}(n-k+1)p_{\tau(k)} \\
= &\sum_{k\neq \ell,\ell+1}(n-k+1)p_{\tau(k)} + (n-\ell+1)p_{\tau(\ell)} + (n-\ell)p_{\tau(\ell+1)} \\
> &\sum_{k\neq \ell,\ell+1}(n-k+1)p_{\tau(k)} + (n-\ell)p_{\tau(\ell)} + (n-\ell+1)p_{\tau(\ell+1)} \\
= &\sum_{k=1}^{n}(n-k+1)\sigma(k) = \sum C_j(\sigma)
\end{aligned}
$$

最適性：ことばで説明 (続)

- このとき

$$
\begin{aligned}
&\sum C_j(\tau) \\
= &\sum_{k=1}^{n}(n-k+1)p_{\tau(k)} \\
= &\sum_{k\neq\ell,\ell+1}(n-k+1)p_{\tau(k)} + (n-\ell+1)p_{\tau(\ell)} + (n-\ell)p_{\tau(\ell+1)} \\
> &\sum_{k\neq\ell,\ell+1}(n-k+1)p_{\tau(k)} + (n-\ell)p_{\tau(\ell)} + (n-\ell+1)p_{\tau(\ell+1)} \\
= &\sum_{k=1}^{n}(n-k+1)\sigma(k) = \sum C_j(\sigma)
\end{aligned}
$$

$p_{\tau(\ell)} > p_{\tau(\ell+1)}$

最適性：ことばで説明 (続)

- このとき

- $\tau(\ell)$ と $\tau(\ell+1)$ を入れ替えた置換を σ とする

$$\sigma(k) = \begin{cases} \tau(k) & (k \neq \ell, \ell+1), \\ \tau(\ell+1) & (k = \ell), \\ \tau(\ell) & (k = \ell+1) \end{cases}$$

$$\begin{aligned}
&\sum C_j(\tau) \\
&= \sum_{k=1}^{n} (n-k+1) p_{\tau(k)} \\
&= \sum_{k \neq \ell, \ell+1} (n-k+1) p_{\tau(k)} + (n-\ell+1) p_{\tau(\ell)} + (n-\ell) p_{\tau(\ell+1)} \\
&> \sum_{k \neq \ell, \ell+1} (n-k+1) p_{\tau(k)} + (n-\ell) p_{\tau(\ell)} + (n-\ell+1) p_{\tau(\ell+1)} \\
&= \sum_{k=1}^{n} (n-k+1) \sigma(k) = \sum C_j(\sigma)
\end{aligned}$$

$p_{\tau(\ell)} > p_{\tau(\ell+1)}$

最適性：ことばで説明 (続)

- このとき

$$
\begin{aligned}
&\sum C_j(\tau)\\
&= \sum_{k=1}^{n} (n-k+1) p_{\tau(k)}\\
&= \sum_{k \neq \ell, \ell+1} (n-k+1) p_{\tau(k)} + (n-\ell+1) p_{\tau(\ell)} + (n-\ell) p_{\tau(\ell+1)}\\
&> \sum_{k \neq \ell, \ell+1} (n-k+1) p_{\tau(k)} + (n-\ell) p_{\tau(\ell)} + (n-\ell+1) p_{\tau(\ell+1)}\\
&= \sum_{k=1}^{n} (n-k+1) \sigma(k) = \sum C_j(\sigma)
\end{aligned}
$$

- したがって，τ は最適解ではない $\square$

最短処理時間優先規則：計算量

アルゴリズム：最短処理時間優先規則 (SPT)

1. 処理時間が短い順にジョブを並べる
2. その順に従ってジョブを処理する

$\boxed{\text{計算量}}$：$n =$ ジョブの総数

1. n 個の数値 $p_1, p_2, \ldots, p_n$ の整列
   $\rightsquigarrow O(n \log n)$ 時間 (マージソートなど)
2. 順序に従ってジョブを置くだけ
   $\rightsquigarrow O(n)$ 時間

$\rightsquigarrow$ 合計 $= O(n \log n)$ 時間

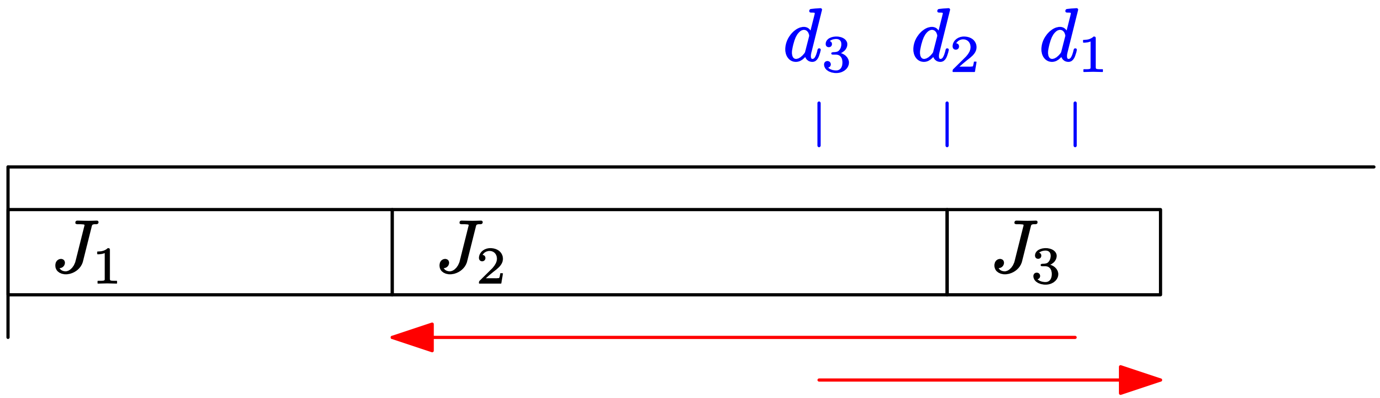
注：計算モデルは第 4 回で扱う予定

本日の内容

1. 総完了時刻最小化と最短処理時間優先規則
2. **最大納期ずれ最小化と最早納期優先規則**
3. 納期遅れジョブ数最小化と Moore-Hodgson アルゴリズム

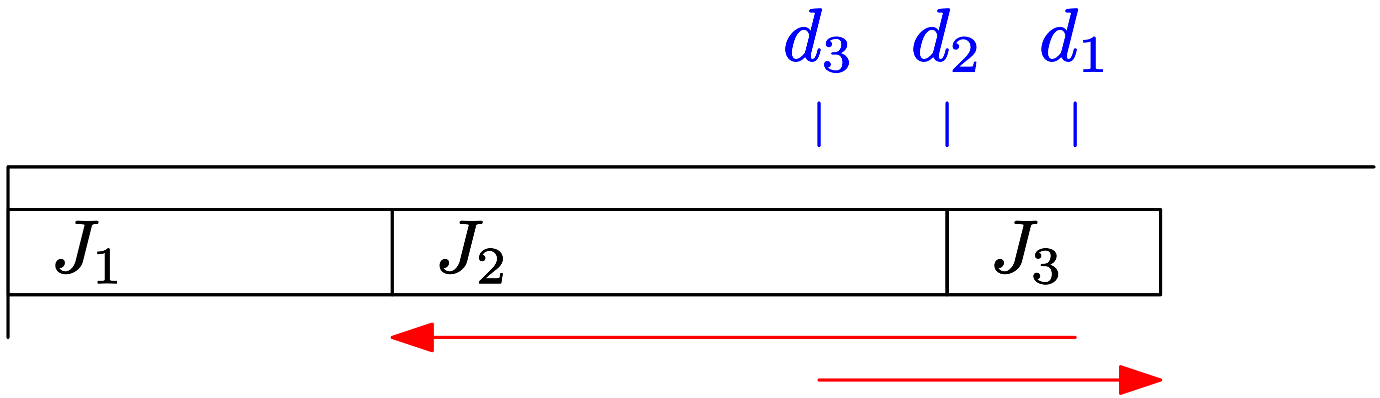
- J.R. Jackson, Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project, University of California, Los Angeles, 1955.

最大納期ずれ最小化：例

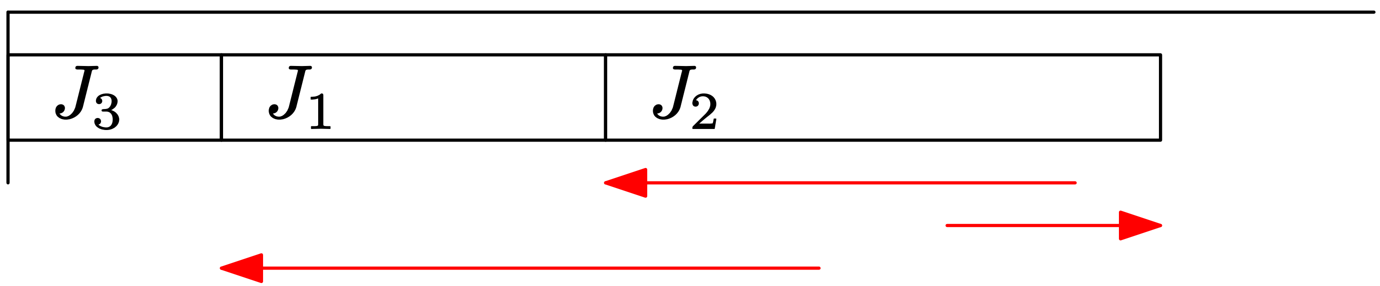


$$L_1 = -16, L_2 = 0, L_3 = 8, \qquad L_{\max} = 8$$

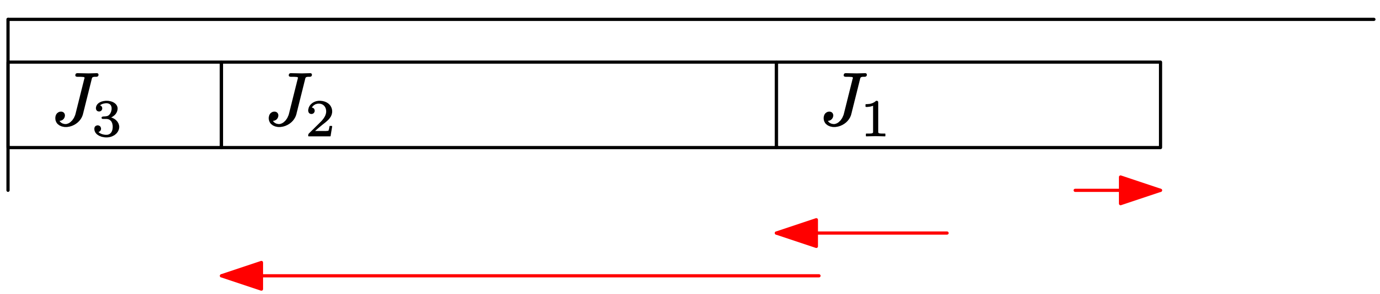
最大納期ずれ最小化：例



$L_1 = -16, L_2 = 0, L_3 = 8,$ $\quad$ $L_{\max} = 8$

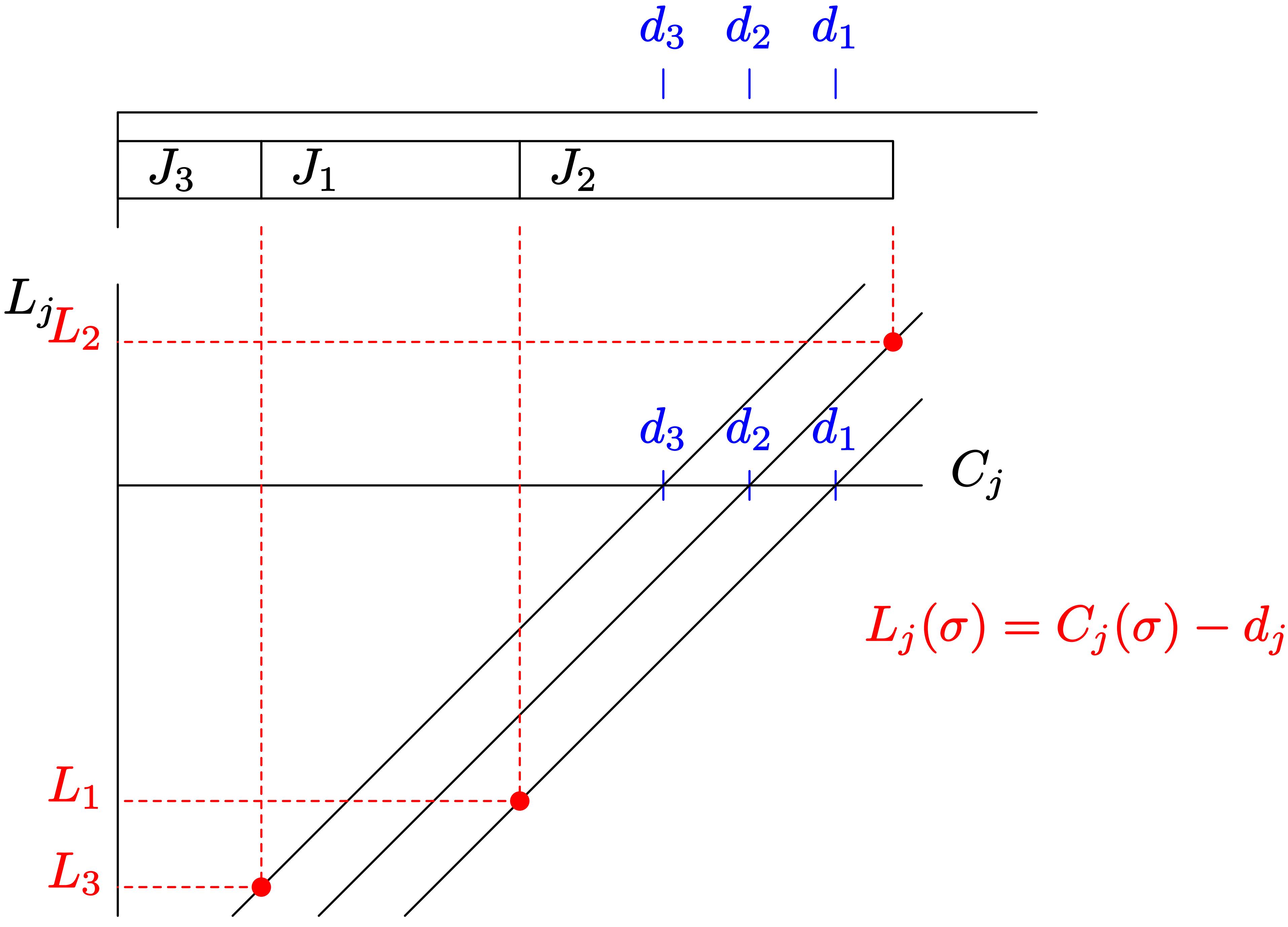


$L_1 = -11, L_2 = 5, L_3 = -14,$ $\quad$ $L_{\max} = 5$

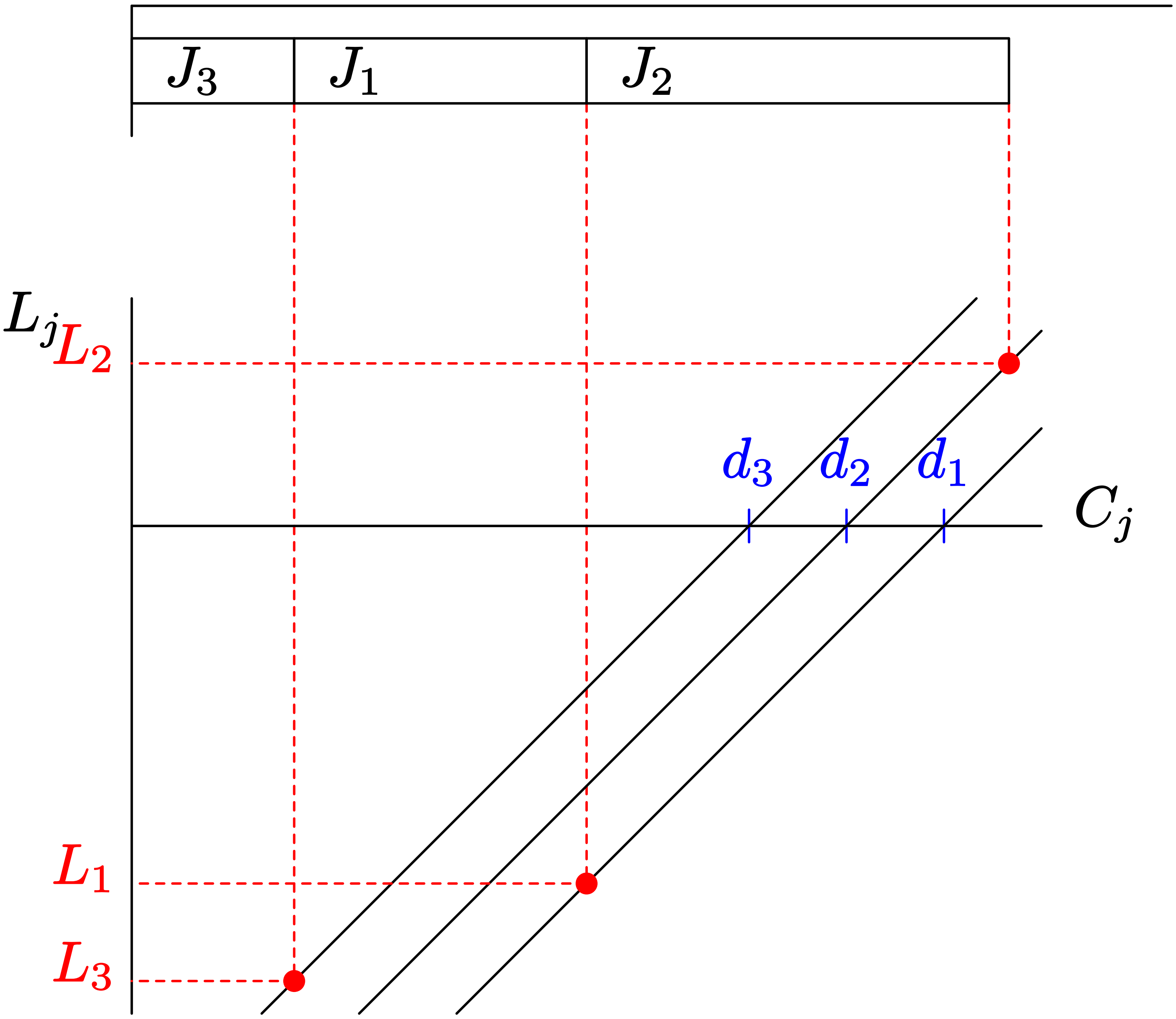


$L_1 = 2, L_2 = -4, L_3 = -14,$ $\quad$ $L_{\max} = 2$

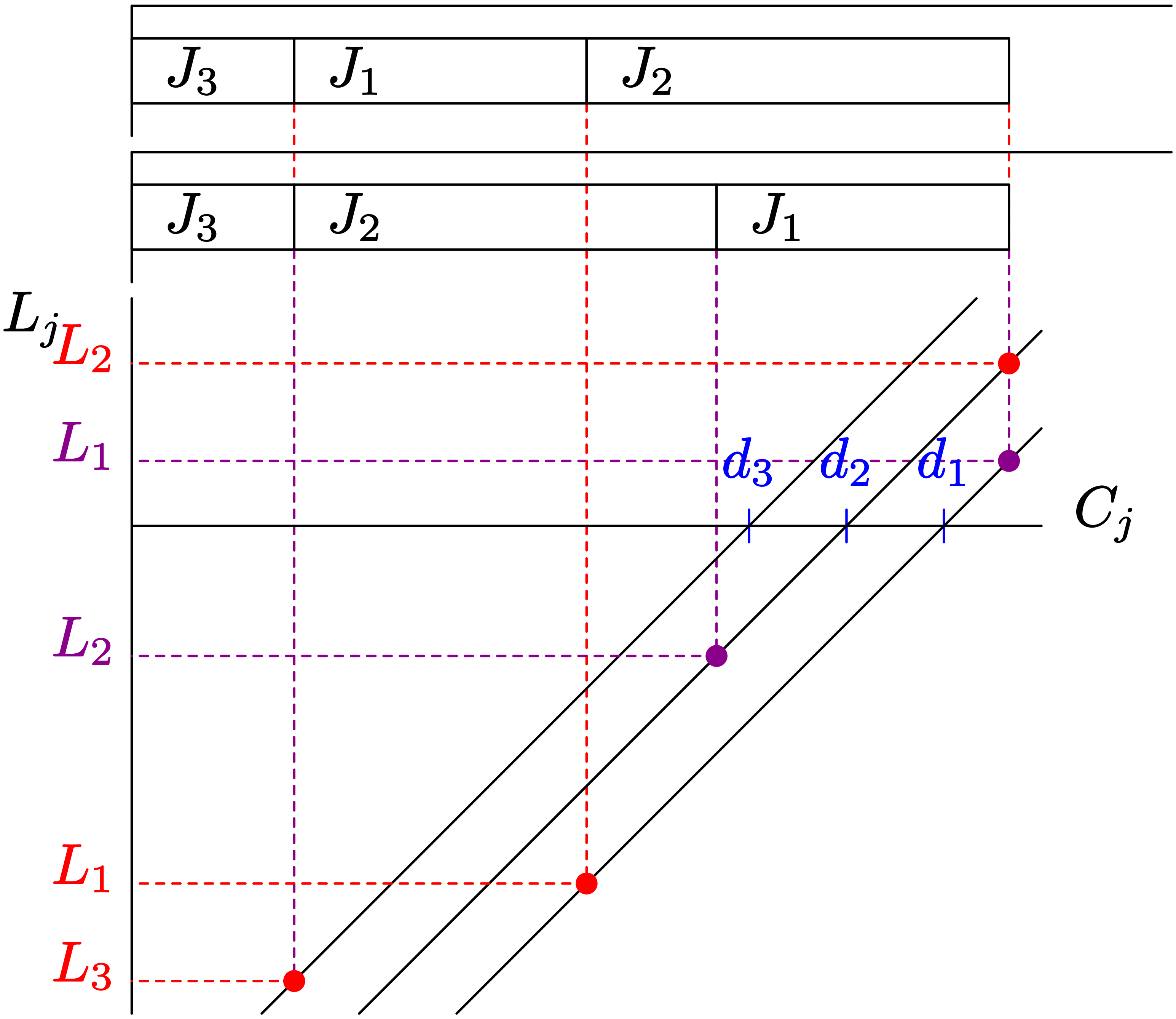
最大納期ずれ最小化：図 (1/2)



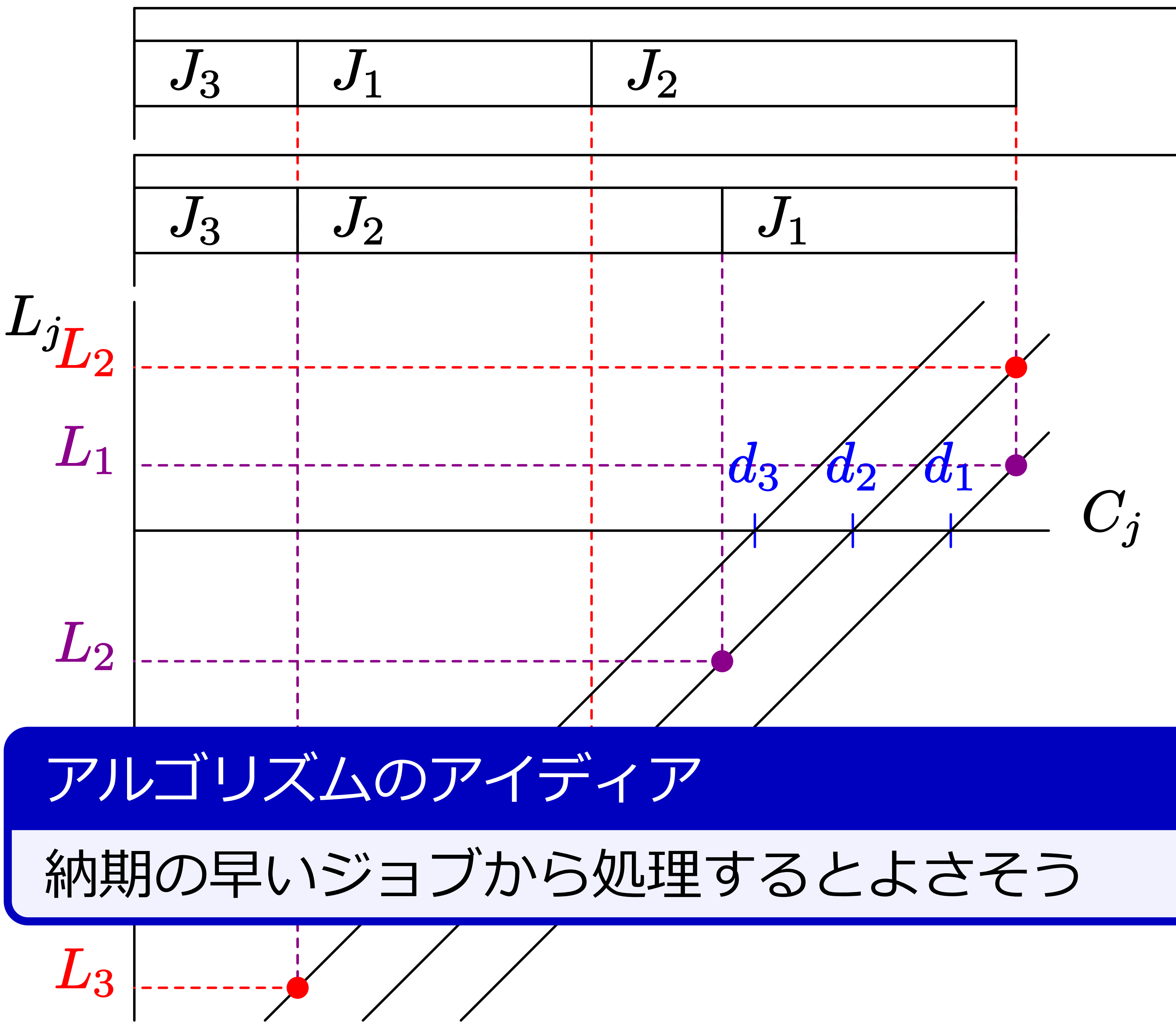
最大納期ずれ最小化：図 (2/2)



最大納期ずれ最小化：図 (2/2)



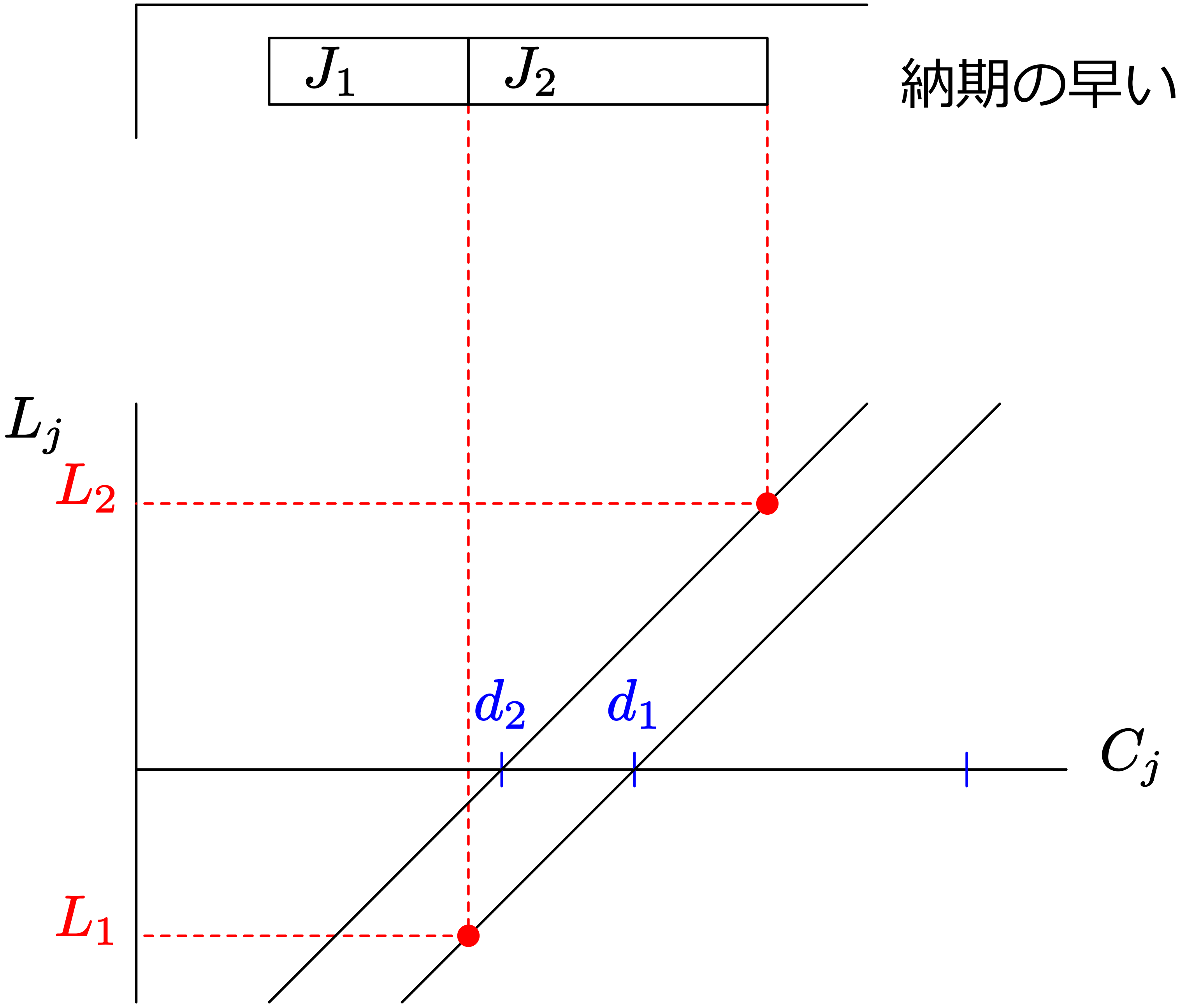
最大納期ずれ最小化：図 (2/2)



アルゴリズムのアイディア

納期の早いジョブから処理するとよさそう

最適性：図で説明



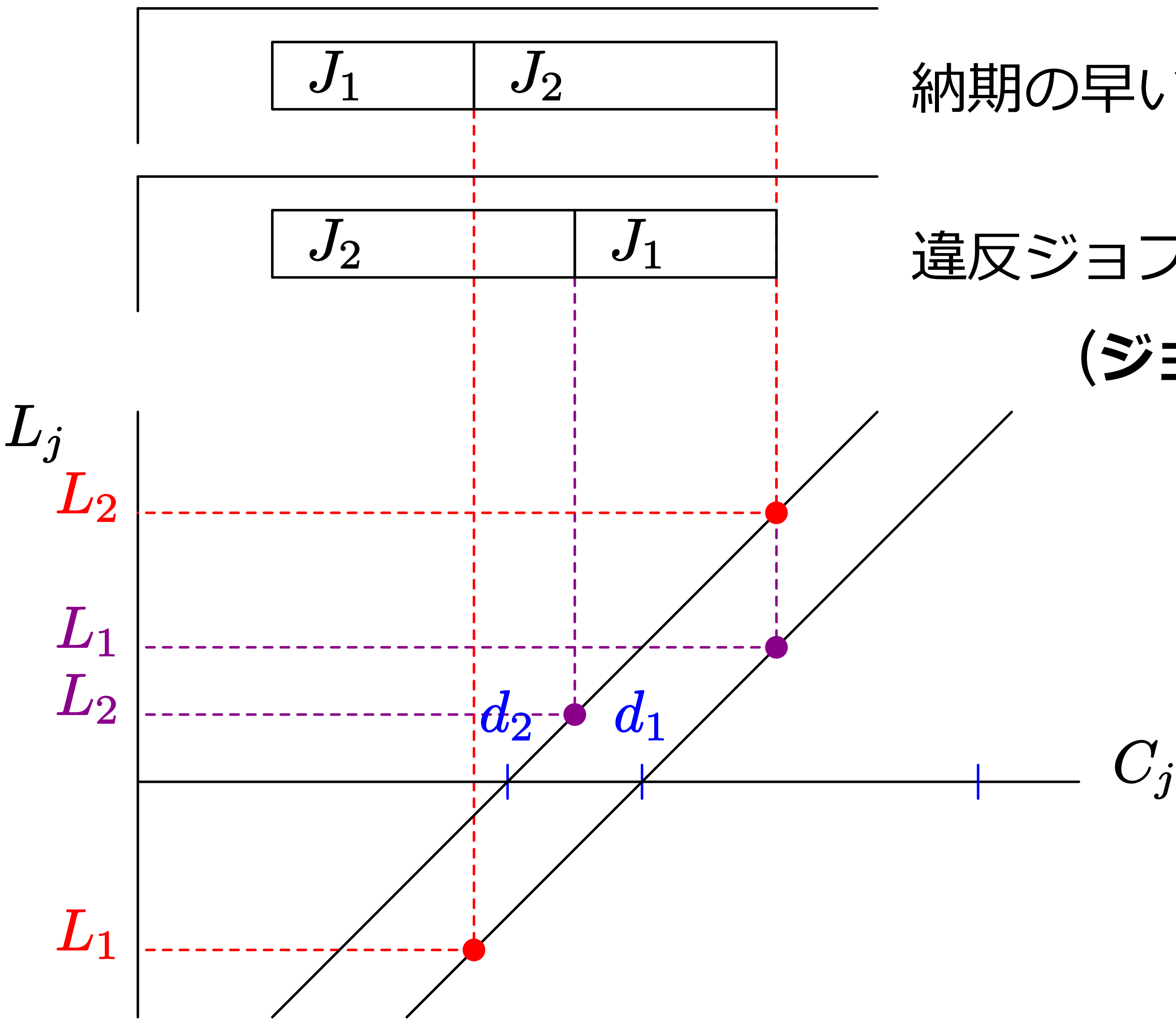
納期の早い順ではない

最適性：図で説明

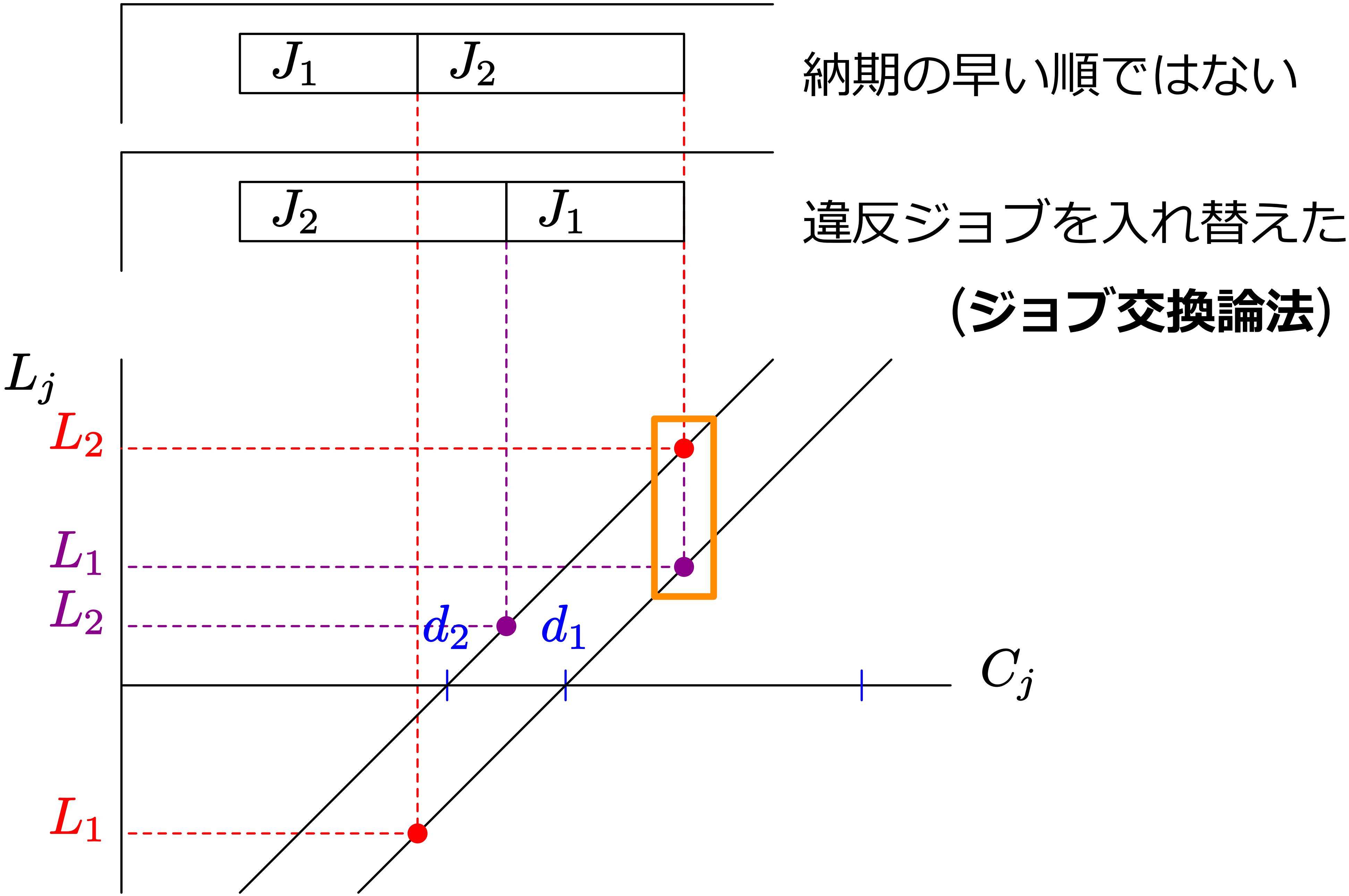
納期の早い順ではない

違反ジョブを入れ替えた

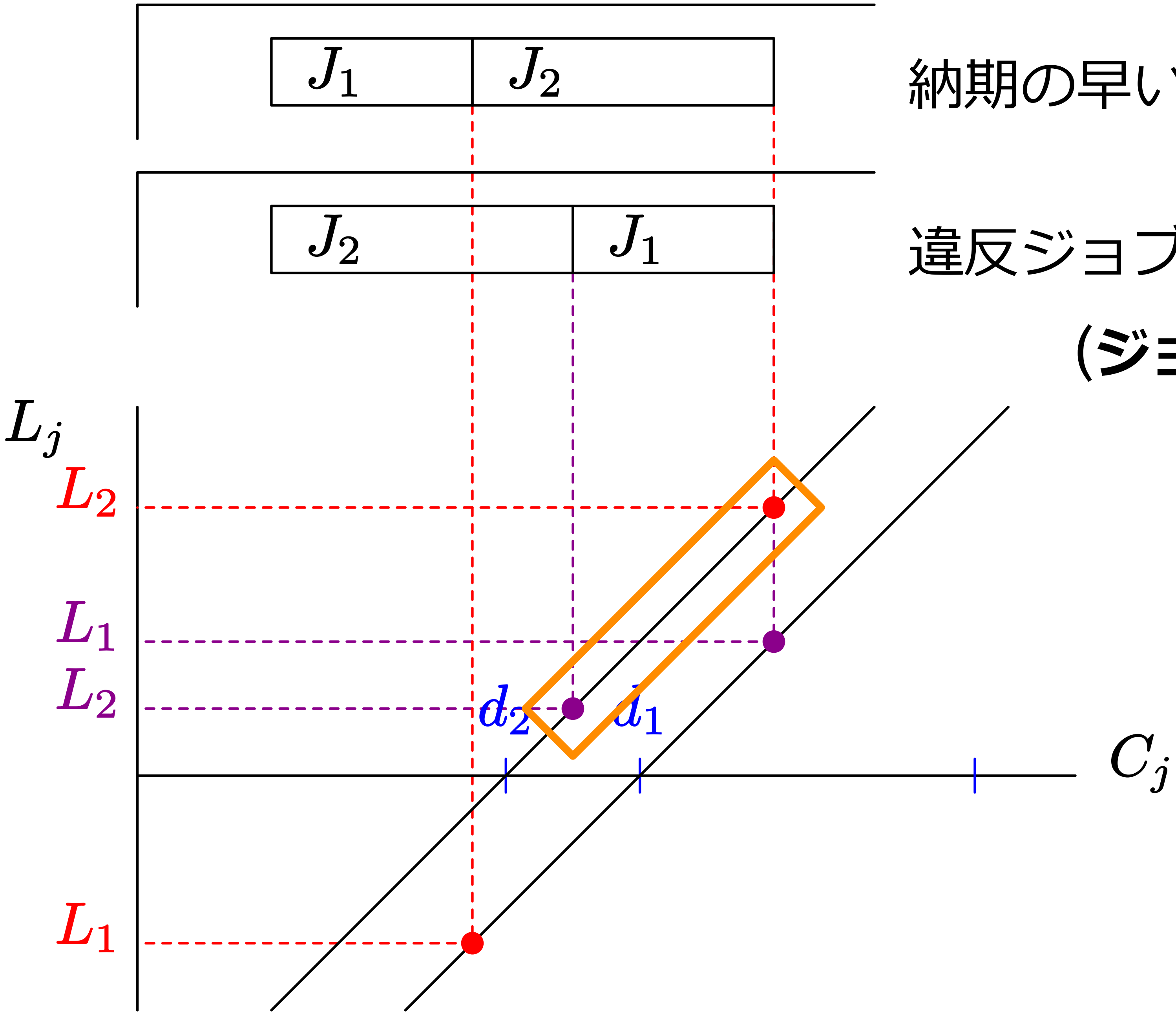
(ジョブ交換論法)



最適性：図で説明


J_1
J_2
納期の早い順ではない
J_2
J_1
違反ジョブを入れ替えた
(ジョブ交換論法)
L_j
L_2
L_1
L_2
d_2
d_1
C_j
L_1

最適性：図で説明



納期の早い順ではない

違反ジョブを入れ替えた

(ジョブ交換論法)

定理：最早納期優先規則

定理：$1 \mid\mid L_{\max}$ のアルゴリズム (Jackson ’55)

$1 \mid\mid L_{\max}$ に対して，次のアルゴリズムは最適解を与える

さいそう

アルゴリズム：最早納期優先規則 (EDD)

1. 納期が早い順にジョブを並べる
2. その順に従ってジョブを処理する

EDD $=$ Earliest Due Date

最早納期優先規則：計算量

アルゴリズム：最早納期優先規則 (EDD)

1. 納期が早い順にジョブを並べる
2. その順に従ってジョブを処理する

$\boxed{\text{計算量}}$：$n =$ ジョブの総数

1. n 個の数値 $d_1, d_2, \ldots, d_n$ の整列
   $\rightsquigarrow O(n \log n)$ 時間 (マージソートなど)
2. 順序に従ってジョブを置くだけ
   $\rightsquigarrow O(n)$ 時間

$\rightsquigarrow$ 合計 $= O(n \log n)$ 時間

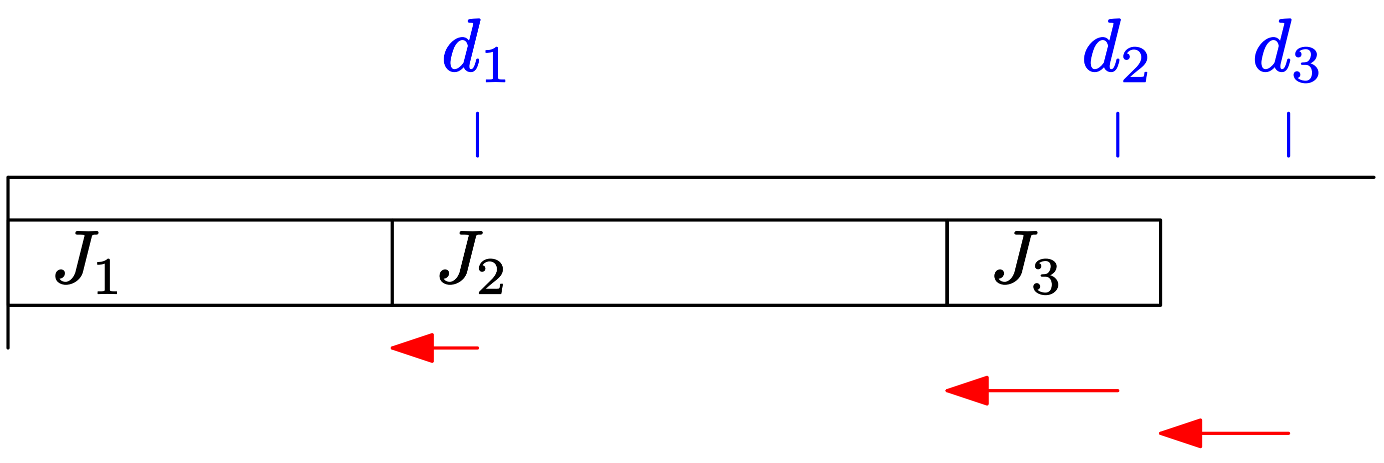
注：計算モデルは第 4 回で扱う予定

補足

性質：EDD と遅れジョブ数

遅れジョブ数 $= 0$ のスケジュールが存在 $\Leftrightarrow$ EDD の最大納期ずれ ≤ 0

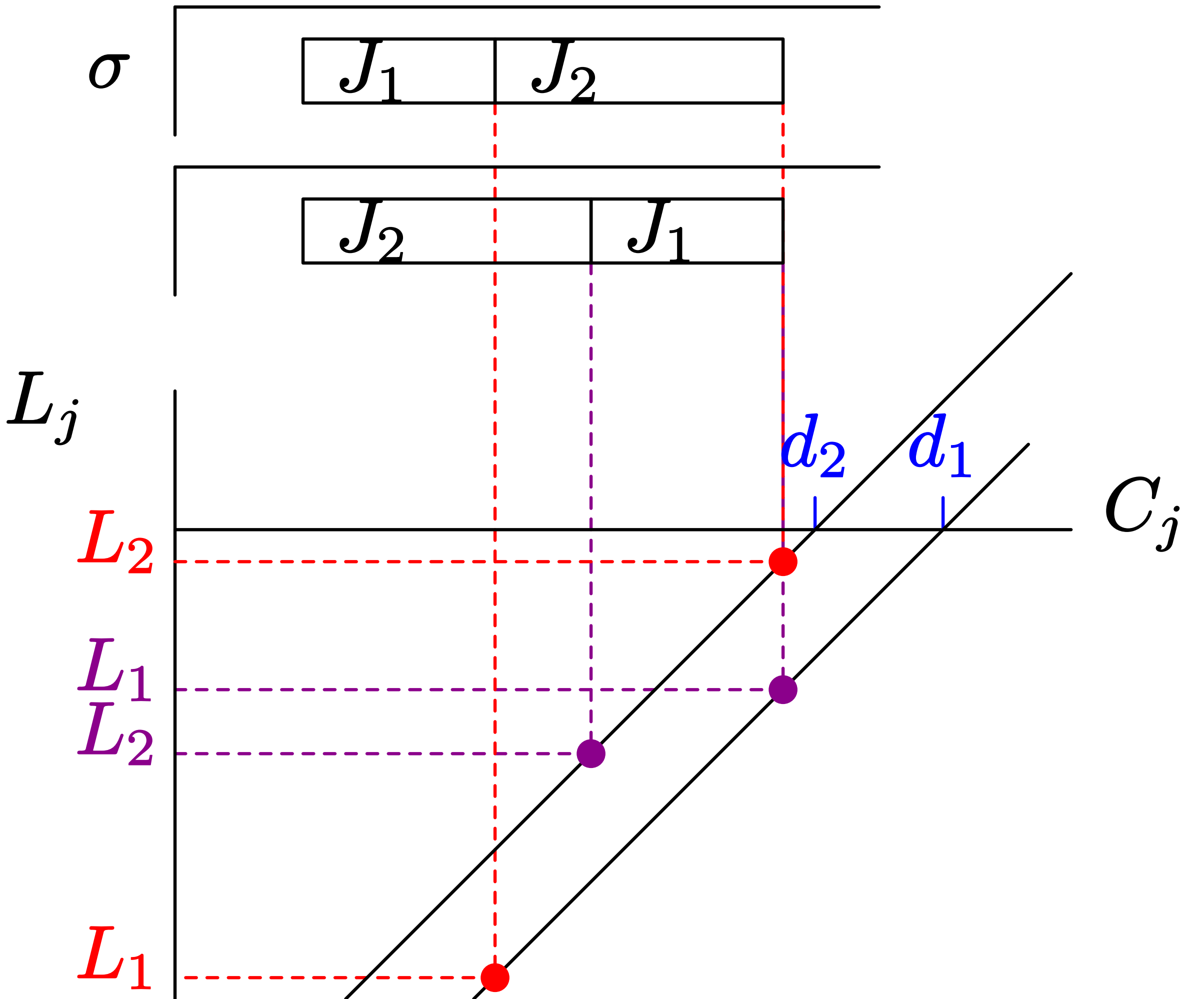
<u>証明 ($\Leftarrow$)</u>：最大納期ずれ ≤ 0 ならば，どのジョブも遅れてない



証明 (⇒) : 遅れジョブ数 $= 0$ のスケジュール σ を考える

納期順になってないジョブを入れ替えても，遅れない

$\therefore$ EDD の最大納期ずれ ≤ 0 □

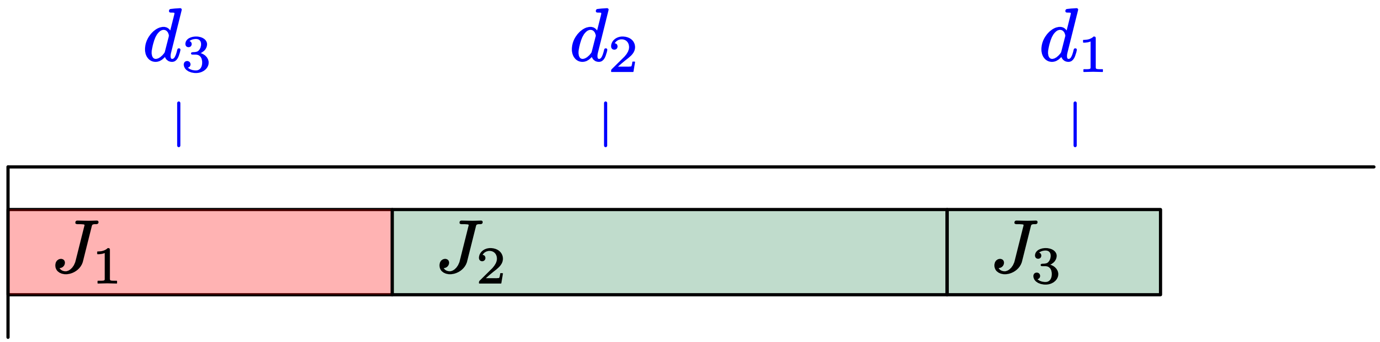


本日の内容

1. 総完了時刻最小化と最短処理時間優先規則
2. 最大納期ずれ最小化と最早納期優先規則
3. **納期遅れジョブ数最小化と Moore-Hodgson アルゴリズム**

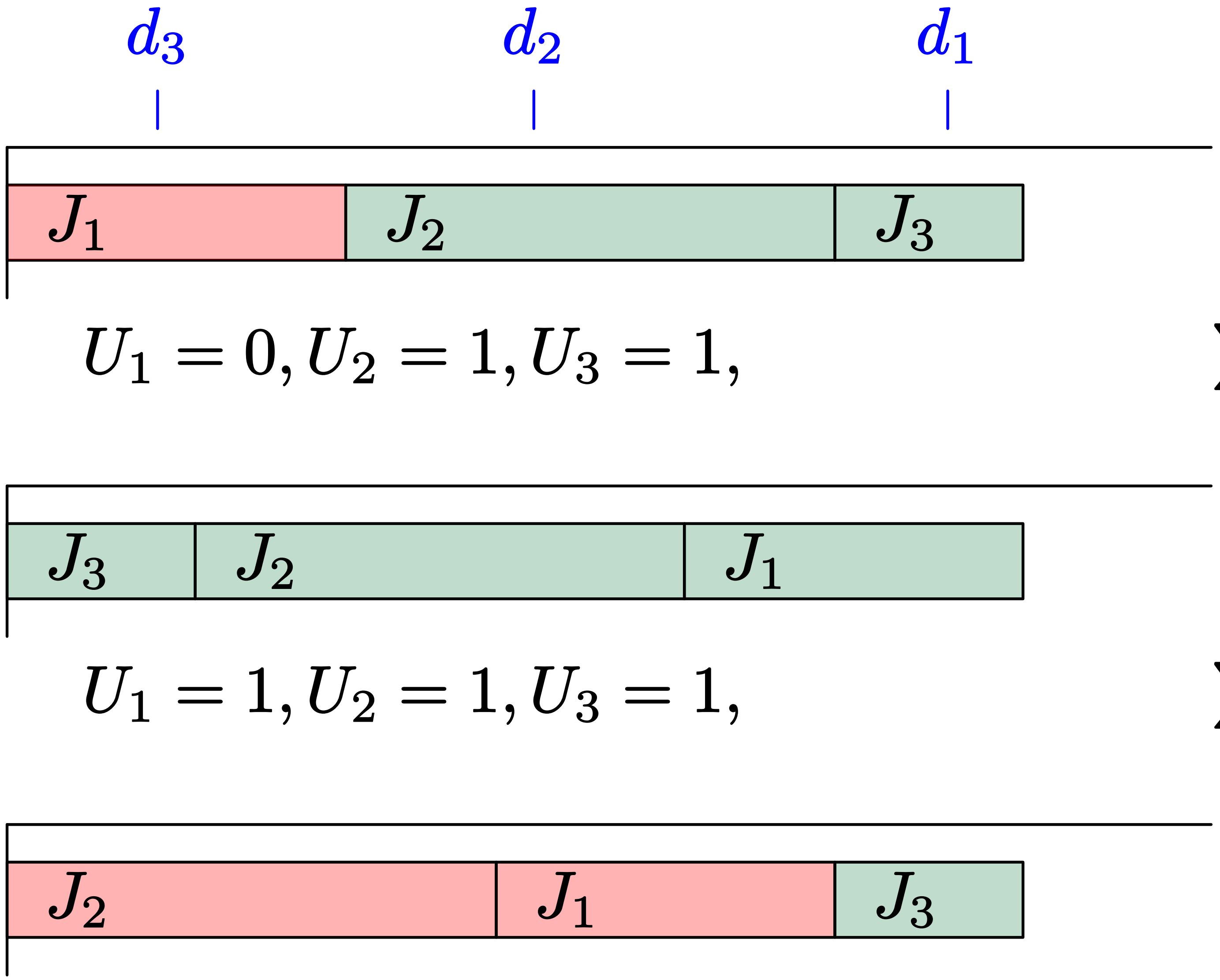
- J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science* 15 (1968) 102–109.
- J. Cheriyan, R. Ravi, M. Skutella, A simple proof of the Moore-Hodgson algorithm for minimizing the number of late jobs. *Operations Research Letters* 49 (2021) 842–843.

納期遅れジョブ数最小化：例



$$U_1 = 0, U_2 = 1, U_3 = 1, \qquad \sum U_j = 2$$

納期遅れジョブ数最小化：例



$U_1 = 0, U_2 = 1, U_3 = 1,$ $\sum U_j = 2$

$U_1 = 1, U_2 = 1, U_3 = 1,$ $\sum U_j = 3$

$U_1 = 0, U_2 = 0, U_3 = 1,$ $\sum U_j = 1$

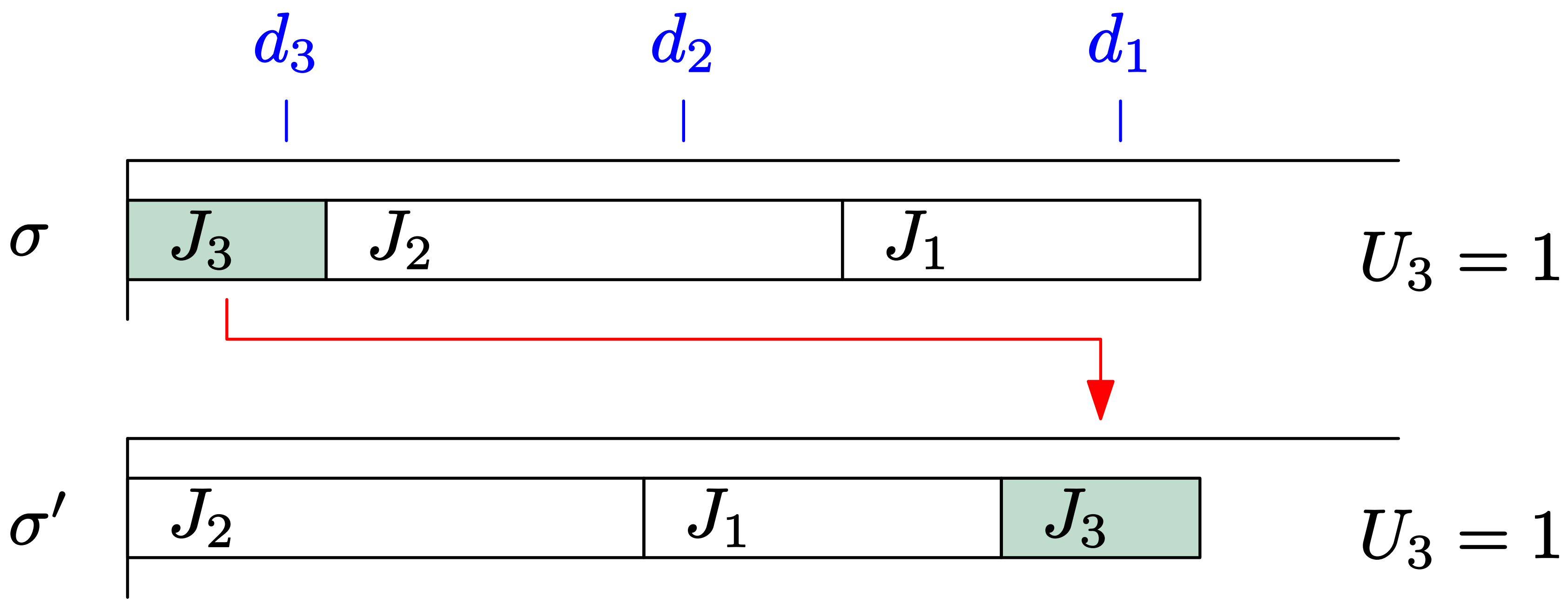
遅れジョブ数最小化：観察

観察：遅れジョブはさらに遅らせても遅れジョブ

スケジュール σ でジョブ J_j が遅れジョブ $\Rightarrow$

ジョブ J_j の完了を σ より遅くしたスケジュール σ' でも
ジョブ J_j は遅れジョブ

$\therefore$ 遅れジョブをどれだけ遅らせても，罰は変わらない



ジョブの分類

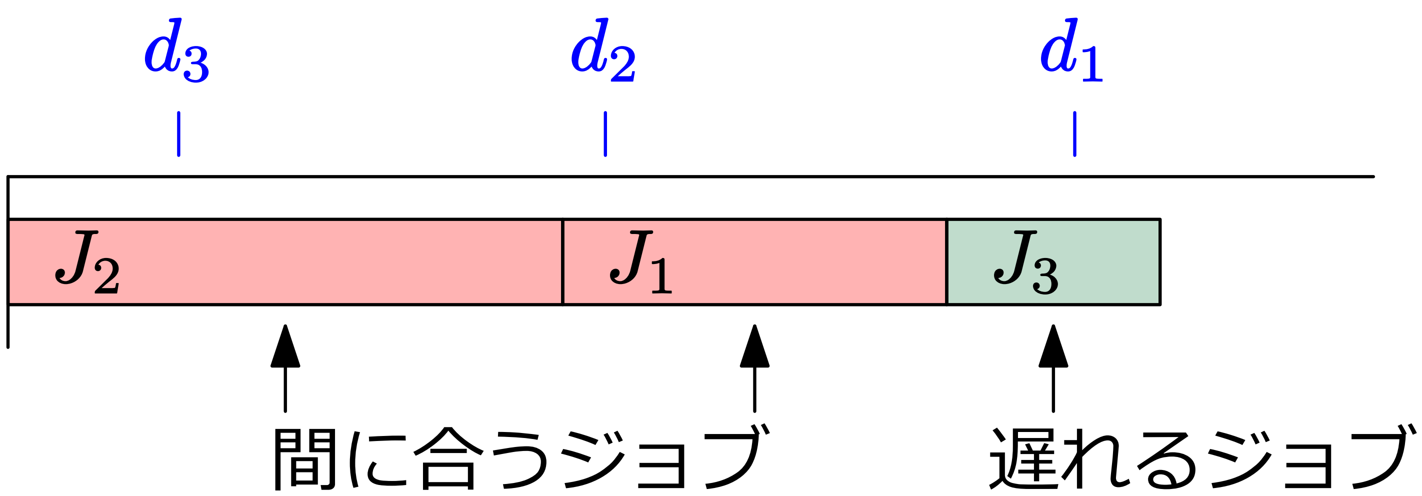
遅れジョブ数最小化：解法の考え方

ジョブを次の 2 つに分類

- 間に合うジョブ
- 遅れるジョブ

分類に従って，次のようにスケジューリング

- 間に合うジョブは優先して，EDD で処理
- 遅れるジョブはその後に，任意の順で処理



ジョブの分類

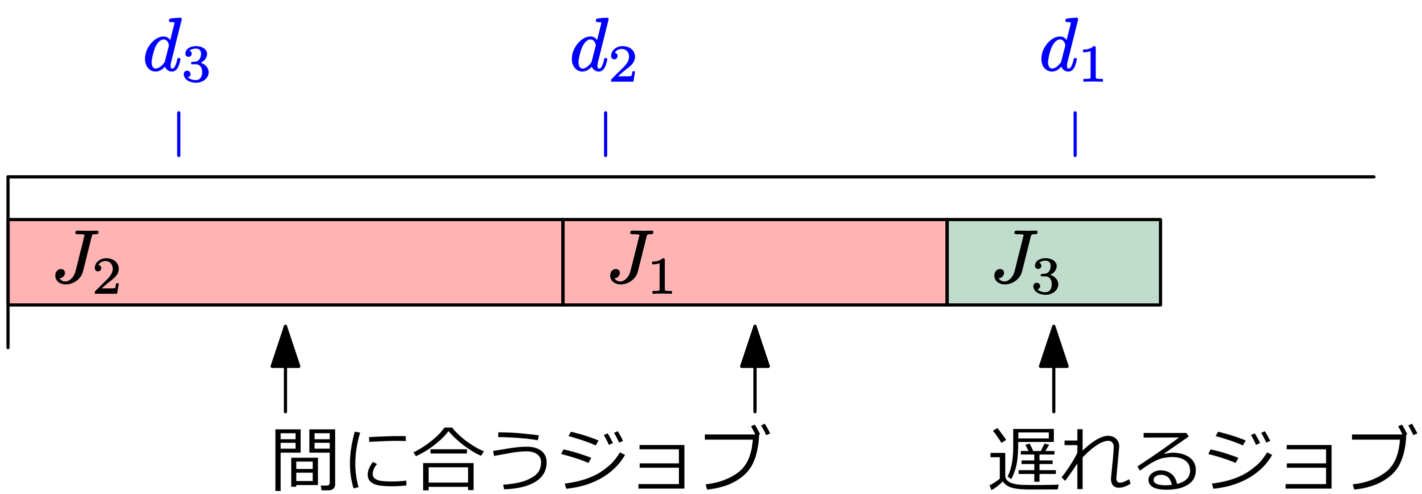
遅れジョブ数最小化：解法の考え方

ジョブを次の 2 つに分類

- 間に合うジョブ
- 遅れるジョブ

分類に従って，次のようにスケジューリング

- 間に合うジョブは優先して，EDD で処理
- 遅れるジョブはその後に，任意の順で処理



このとき，$\sum U_j =$ 遅れるジョブの数

Moore-Hodgson アルゴリズム

遅れジョブ数最小化：解法の考え方

ジョブを次の 2 つに分類

- 間に合うジョブ
- 遅れるジョブ

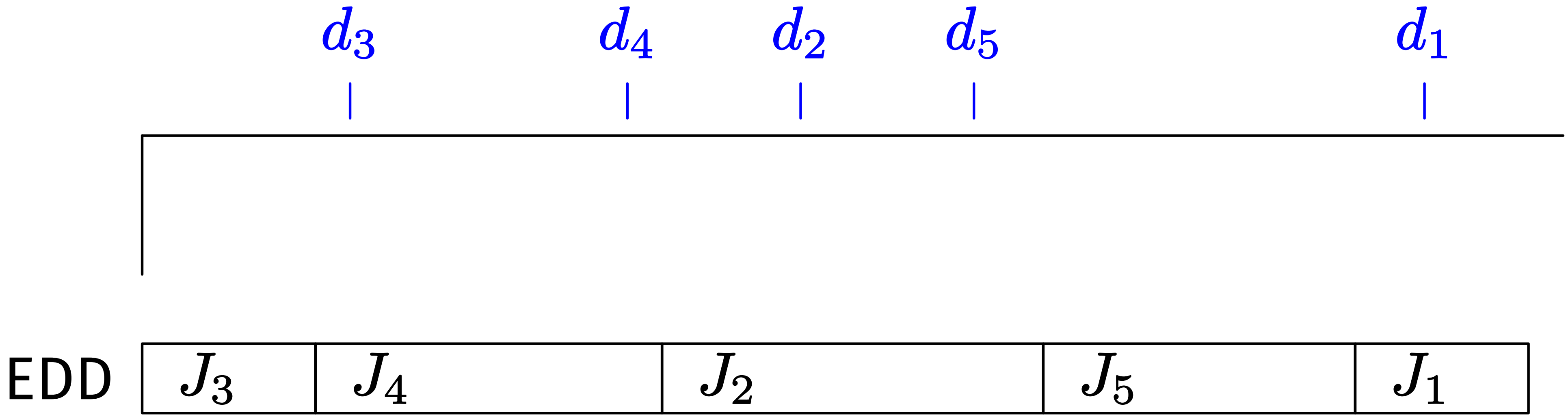
分類に従って，次のようにスケジューリング

- 間に合うジョブは優先して，EDD で処理
- 遅れるジョブはその後に，任意の順で処理

アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

Moore-Hodgson アルゴリズム：例

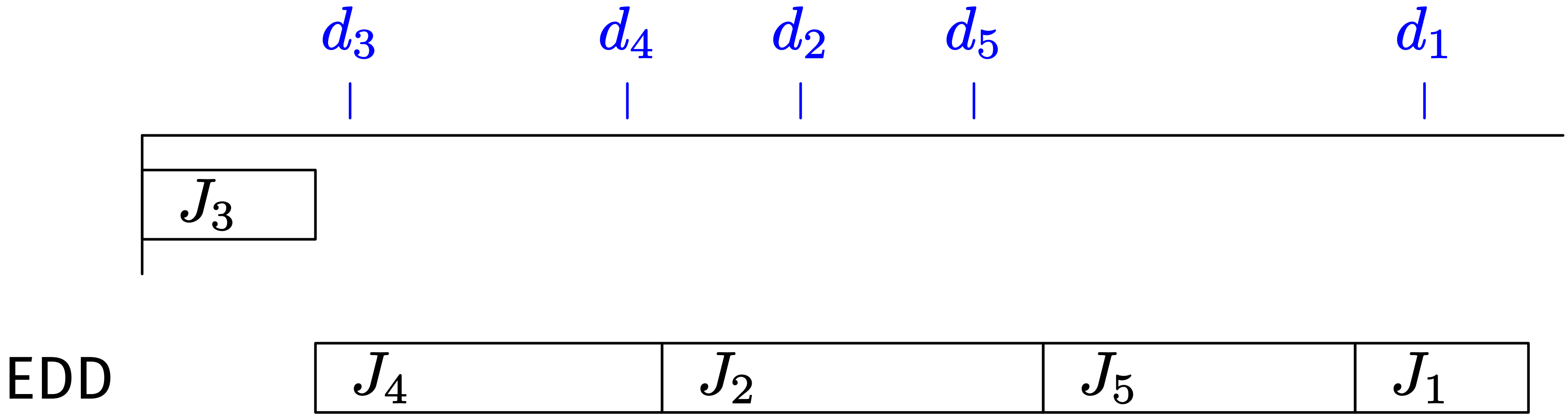


遅れるジョブ

アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   - (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   - (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

Moore-Hodgson アルゴリズム：例

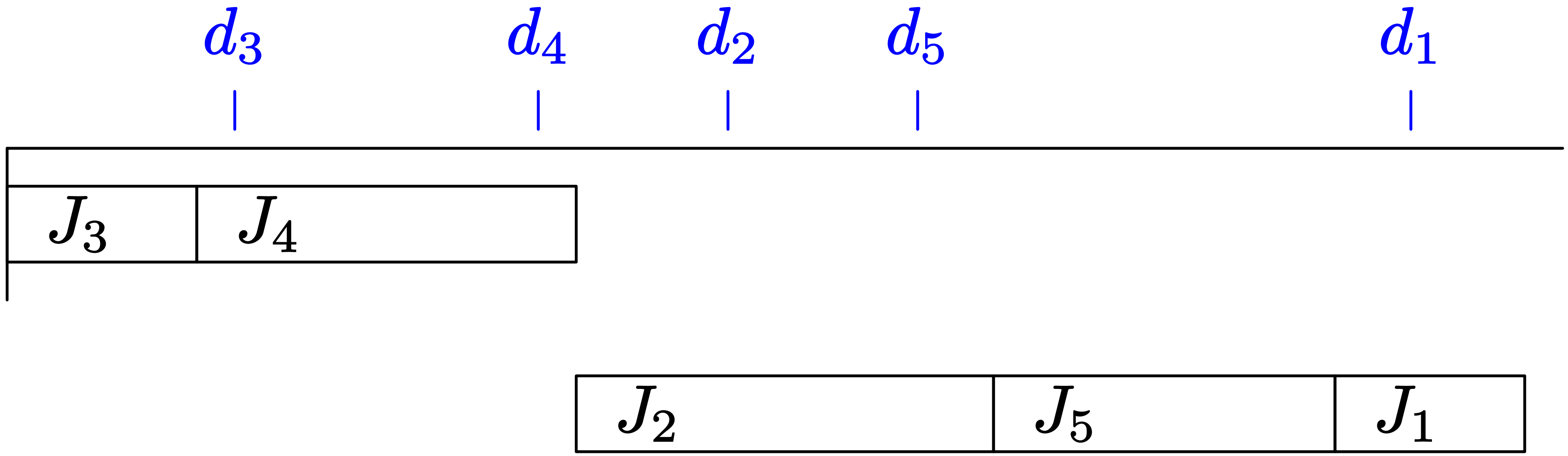


遅れるジョブ

アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   - (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   - (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

Moore-Hodgson アルゴリズム：例



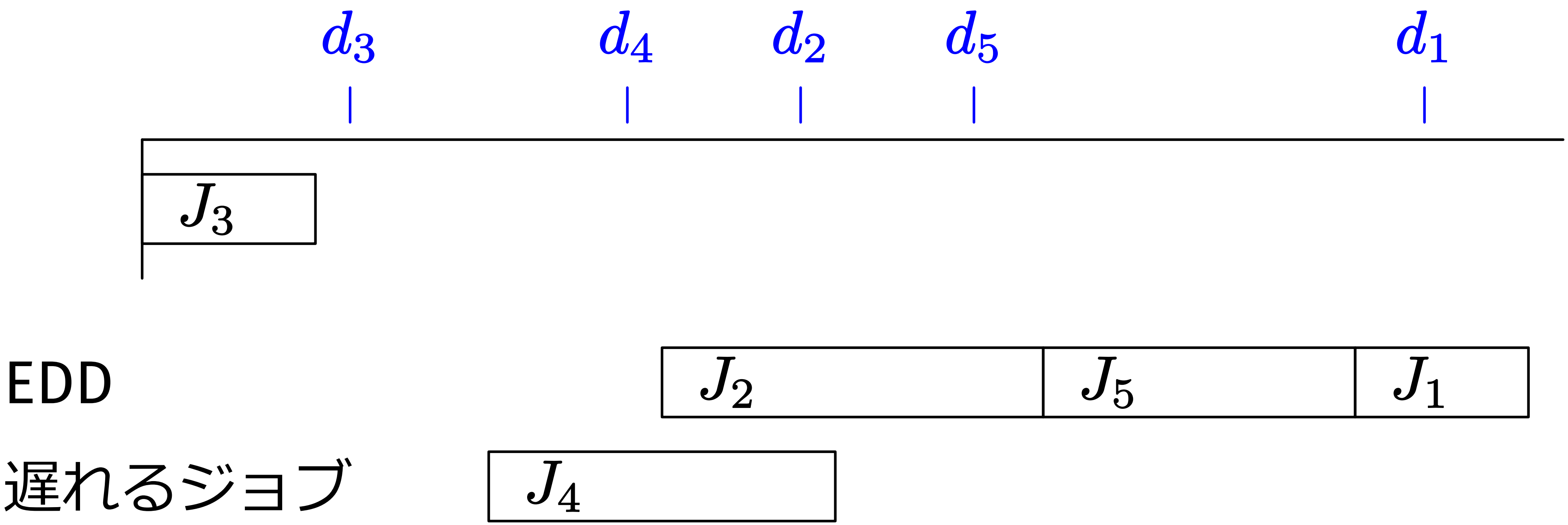
EDD

遅れるジョブ

アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

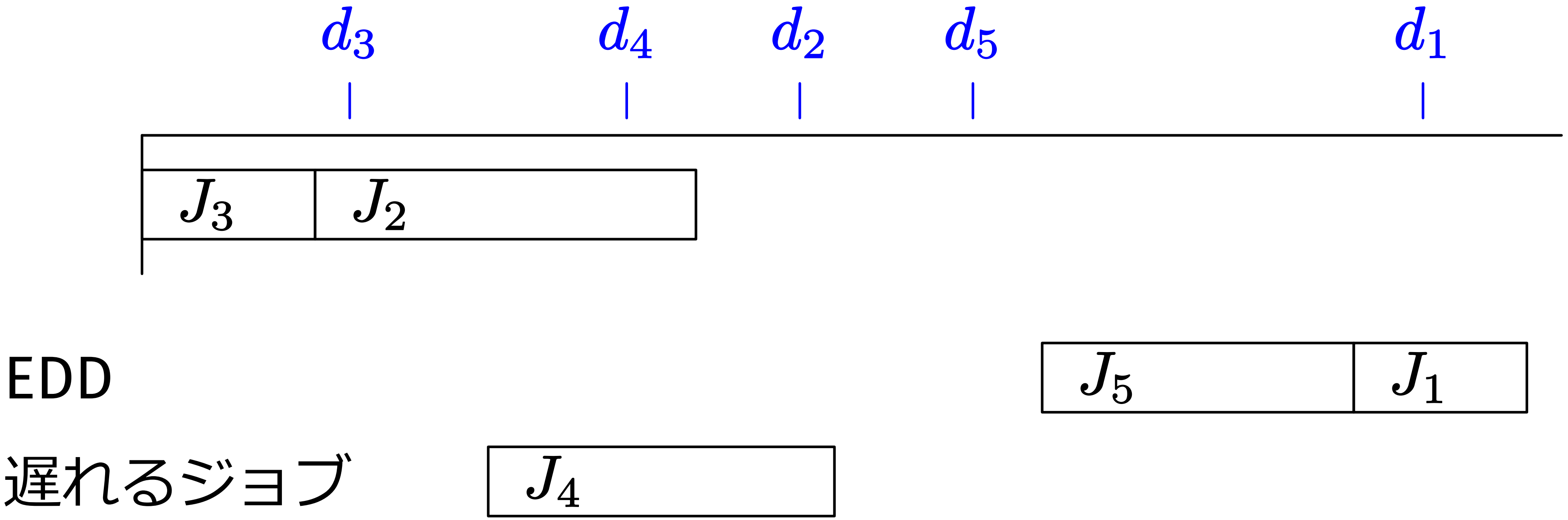
Moore-Hodgson アルゴリズム：例



アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

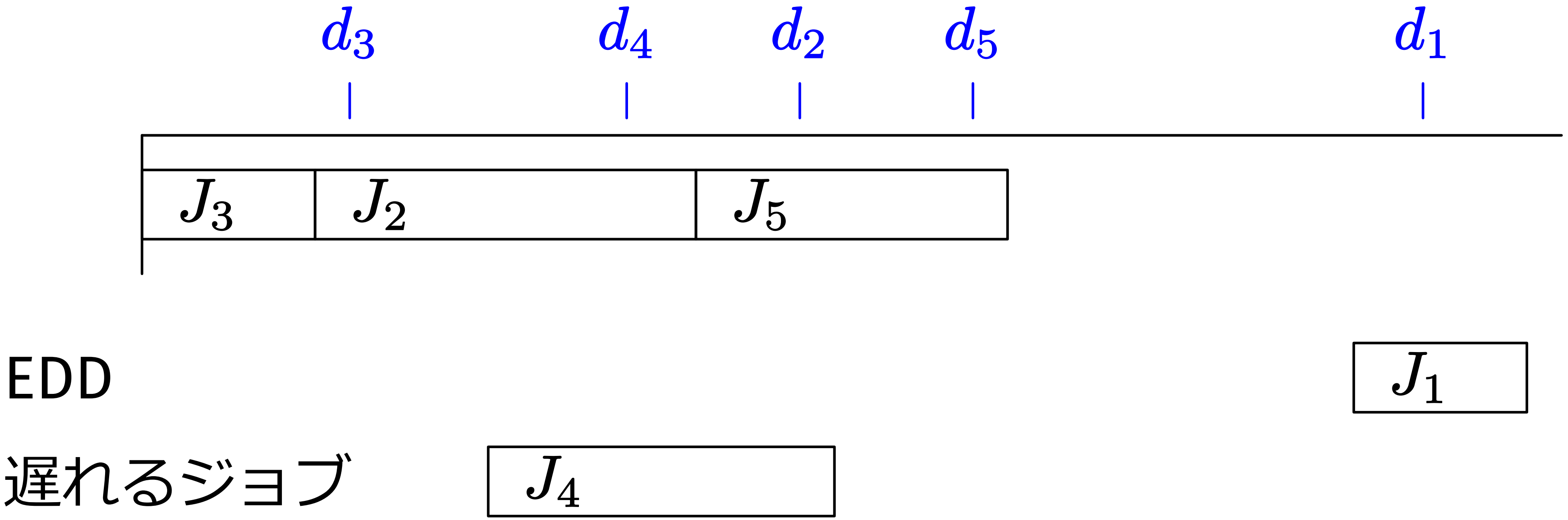
Moore-Hodgson アルゴリズム：例



アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

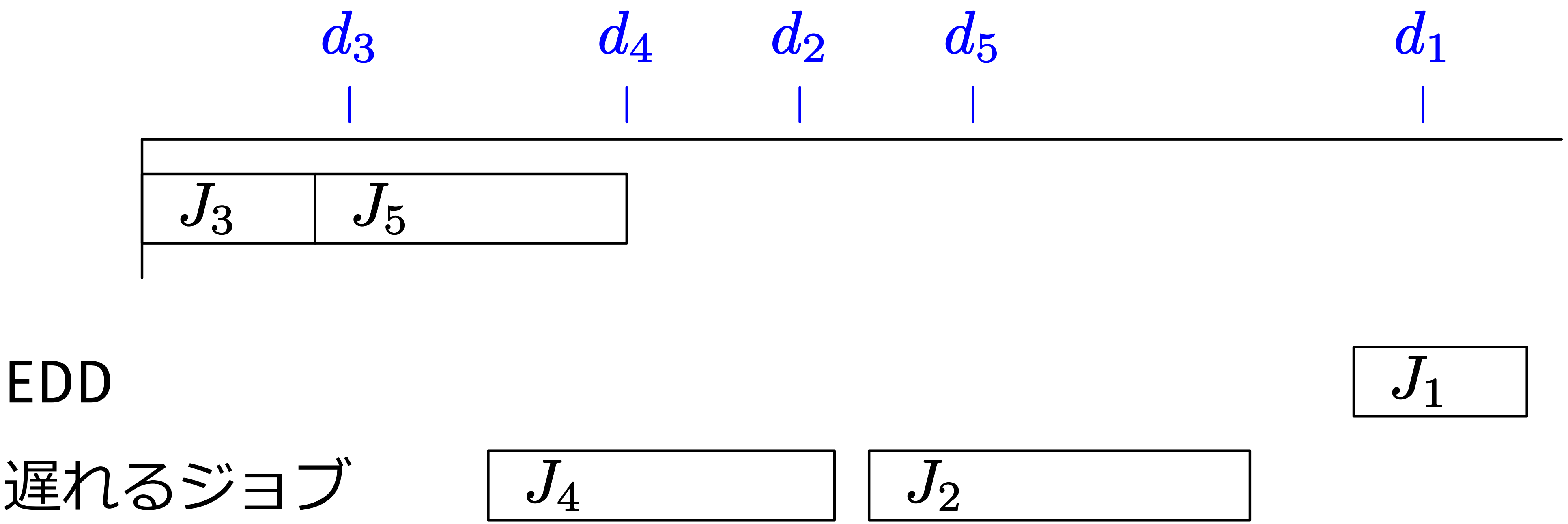
Moore-Hodgson アルゴリズム：例



アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

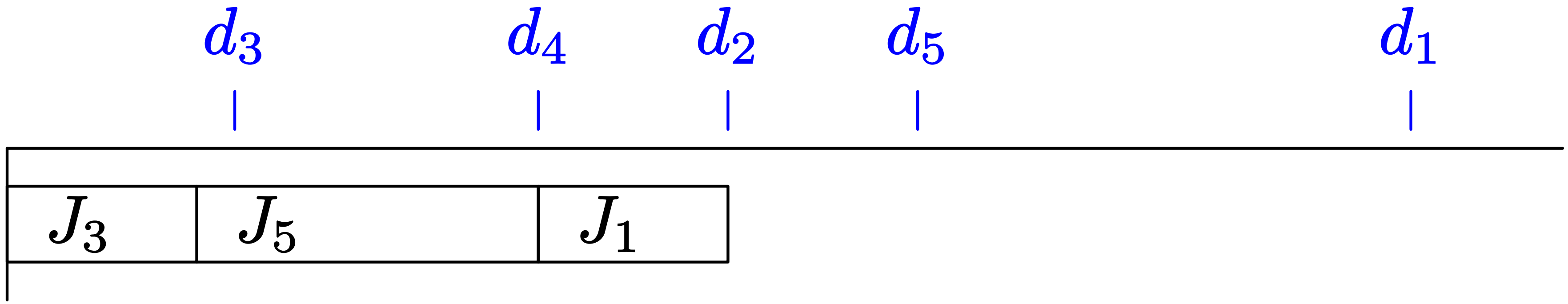
Moore-Hodgson アルゴリズム：例



アルゴリズム：Moore-Hodgson アルゴリズム

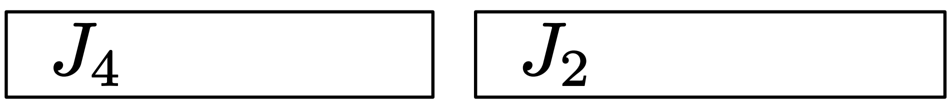
1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

Moore-Hodgson アルゴリズム：例



EDD

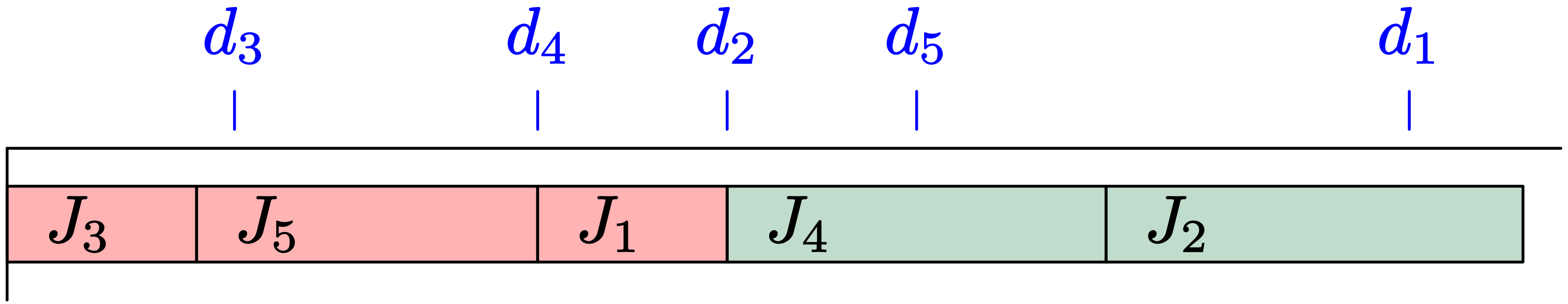
遅れるジョブ



アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

Moore-Hodgson アルゴリズム：例



EDD

遅れるジョブ

アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

定理：Moore-Hodgson アルゴリズム

定理：$1 \,||\, \sum U_j$ のアルゴリズム (Moore ’68)

$1 \,||\, \sum U_j$ に対して，次のアルゴリズムは最適解を与える

アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う ⇒「仮に間に合うジョブ」とする
   (b) 納期に間に合わない ⇒ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

証明：Moore-Hodgson の正当性

証明 (Cheriyan, Ravi, Skutella '21)：

最適値に関する帰納法

- 最適値 $= 0 \Rightarrow$ Moore-Hodgson $=$ EDD なので，OK

性質：EDD と遅れジョブ数 (再掲)

遅れジョブ数 $= 0$ のスケジュールが存在 $\Leftrightarrow$ EDD の最大納期ずれ ≤ 0

証明：Moore-Hodgson の正当性

証明 (Cheriyan, Ravi, Skutella '21)：

最適値に関する帰納法

- 最適値 $= 0 \Rightarrow$ Moore-Hodgson = EDD なので，OK
- 最適値 $= t \geq 0$ のときに正しいと仮定して，
  最適値 $= t + 1$ のときを考える

証明：Moore-Hodgson の正当性

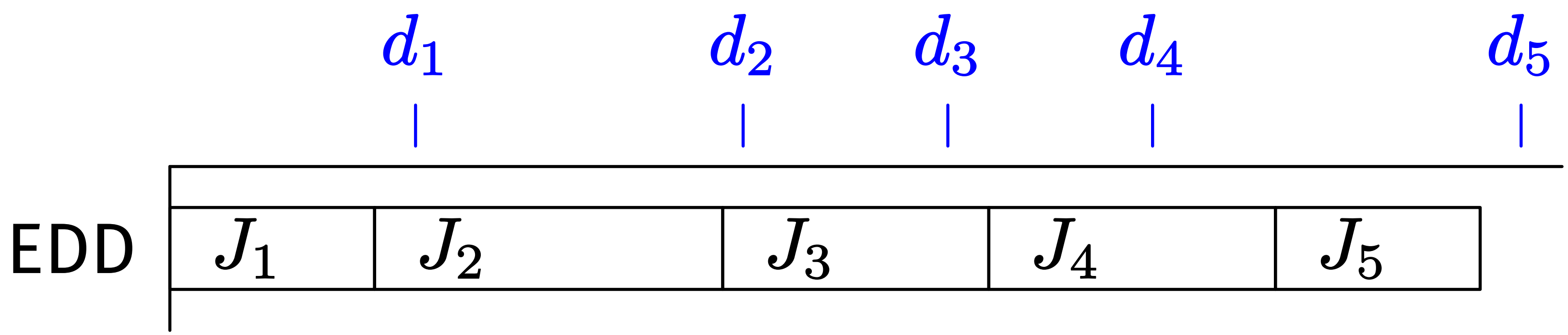
証明 (Cheriyan, Ravi, Skutella ’21)：

最適値に関する帰納法

- 最適値 $= 0 \Rightarrow$ Moore-Hodgson = EDD なので，OK
- 最適値 $= t \geq 0$ のときに正しいと仮定して，
  最適値 $= t + 1$ のときを考える

仮定：WLOG EDD 順が $J_1, J_2, \ldots, J_n$ である

Without loss of generality (一般性を失わずに)



証明 (Cheriyan, Ravi, Skutella '21)：

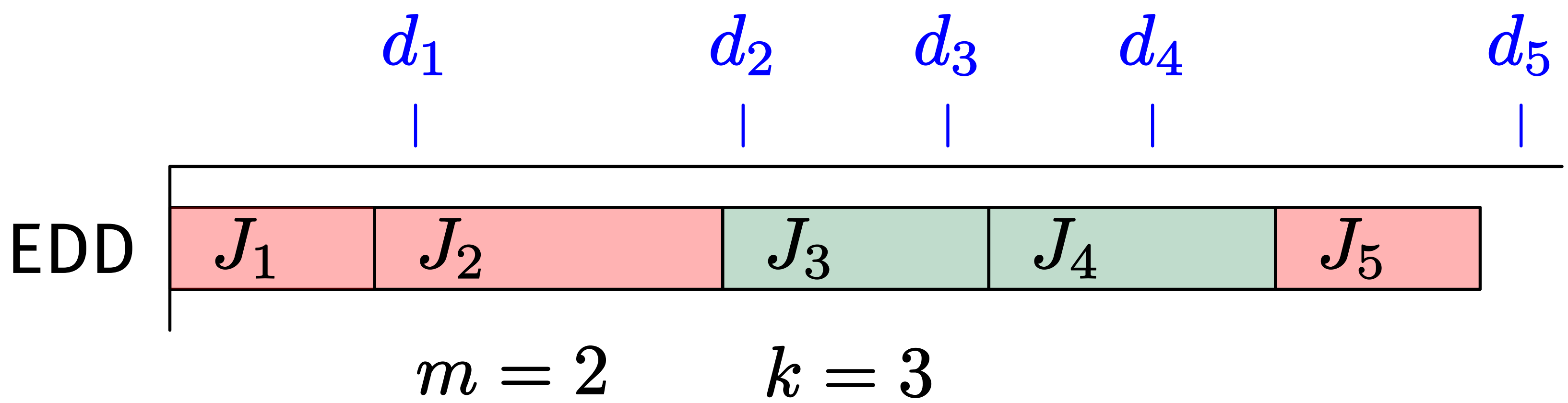
最適値に関する帰納法

- 最適値 $= 0 \Rightarrow$ Moore-Hodgson $=$ EDD なので，OK
- 最適値 $= t \geq 0$ のときに正しいと仮定して，
  最適値 $= t + 1$ のときを考える

仮定：WLOG EDD 順が $J_1, J_2, \ldots, J_n$ である

$J_k =$ EDD 順ではじめに遅れるジョブ

$J_m =$ アルゴリズムがはじめに除去するジョブ



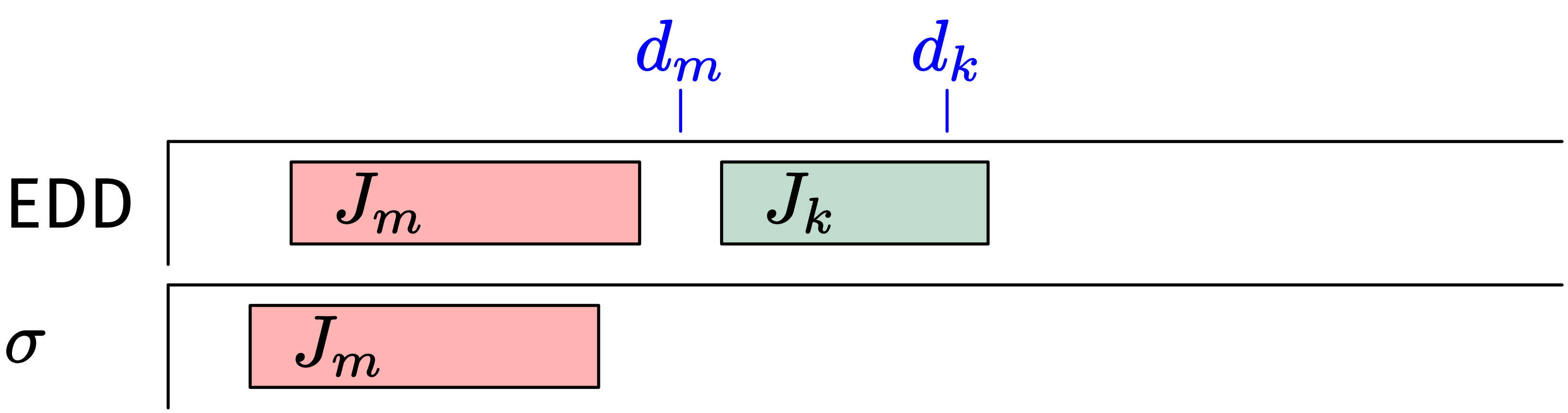
補題とその証明

補題

J_m が遅れるジョブとなる最適スケジュールが存在

<u>補題の証明</u>：任意の最適スケジュール σ を考える

- σ で J_m が遅れる $\Rightarrow$ 証明終わり
- σ で J_m が間に合う $\Rightarrow$



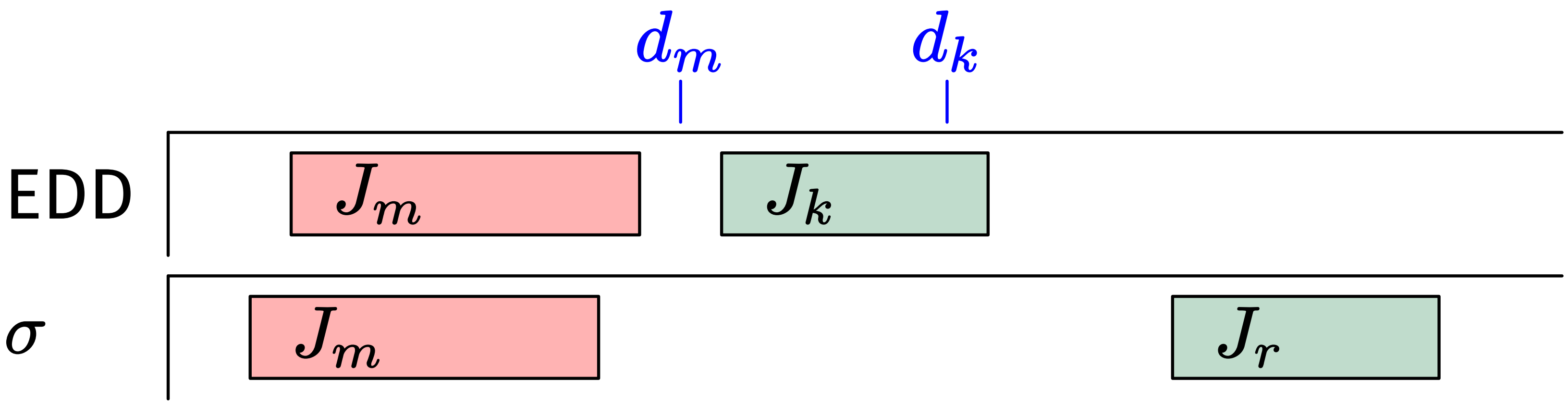
補題とその証明

補題

J_m が遅れるジョブとなる最適スケジュールが存在

<u>補題の証明</u>：任意の最適スケジュール σ を考える

- σ で J_m が遅れる $\Rightarrow$ 証明終わり
- σ で J_m が間に合う $\Rightarrow$
  - ある $r \in \{1, \ldots, k\}$ に対して，σ で J_r が遅れる



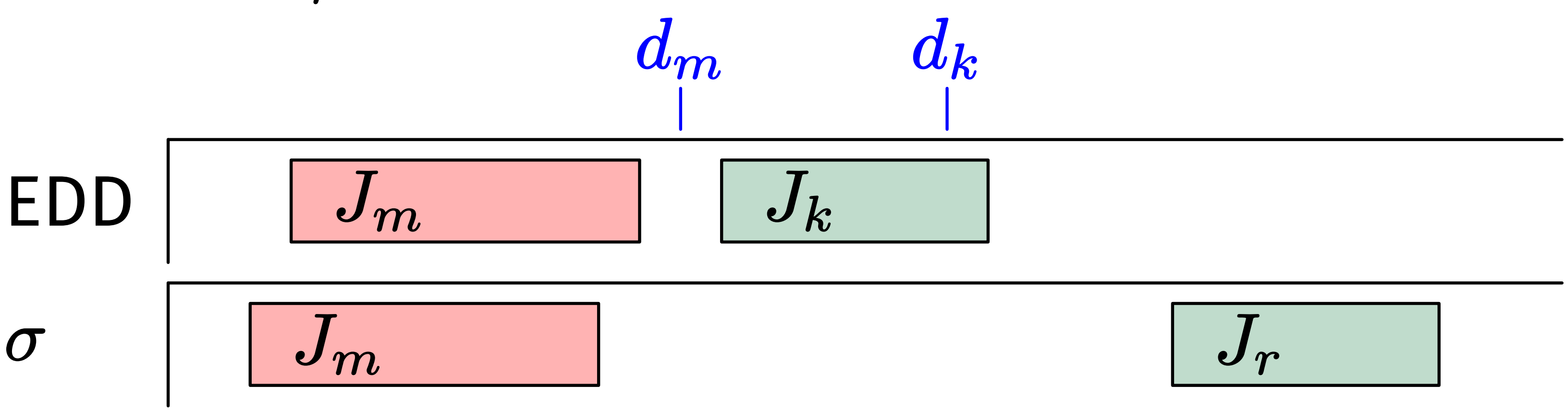
補題とその証明

補題

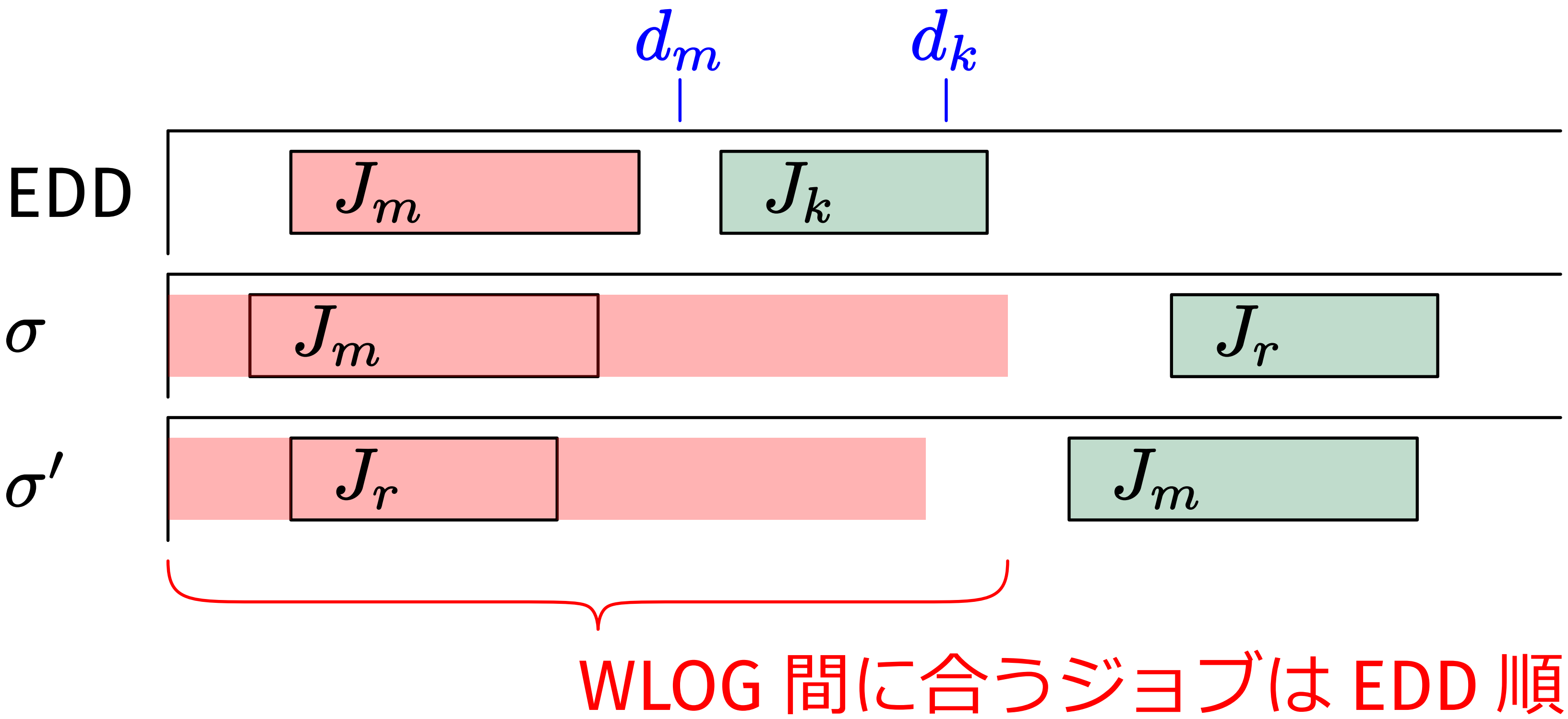
J_m が遅れるジョブとなる最適スケジュールが存在

補題の証明：任意の最適スケジュール σ を考える

- σ で J_m が遅れる $\Rightarrow$ 証明終わり
- σ で J_m が間に合う $\Rightarrow$
  - ある $r \in \{1, \ldots, k\}$ に対して，σ で J_r が遅れる
  - σ を次のように変更して，σ' を作る
    - J_m を遅れるジョブとする
    - J_r を間に合うジョブとする



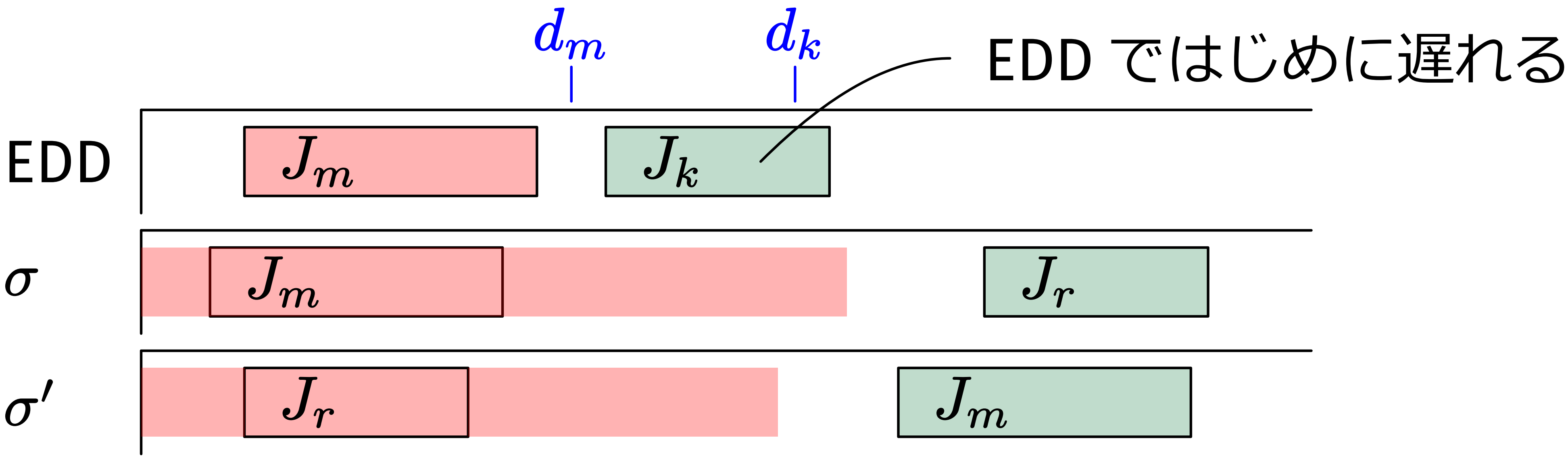
補題とその証明 (続)



性質：EDD と遅れジョブ数 (再掲)

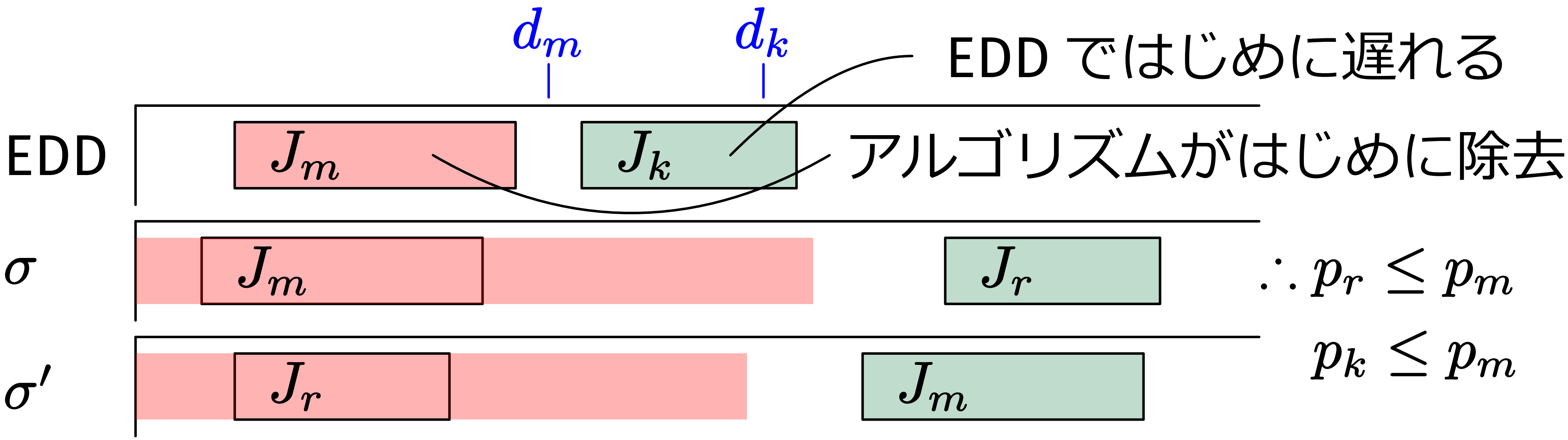
遅れジョブ数 $= 0$ のスケジュールが存在 $\Leftrightarrow$ EDD の最大納期ずれ ≤ 0

補題とその証明 (続)



- このようなスケジュール σ' があれば，σ' も最適
- 実際，スケジュール σ' は存在する
  - 任意の $\ell \in [k-1] \setminus \{m\}$ に対して，
    J_ℓ が σ で間に合う $\Rightarrow$ J_ℓ は σ' で間に合う

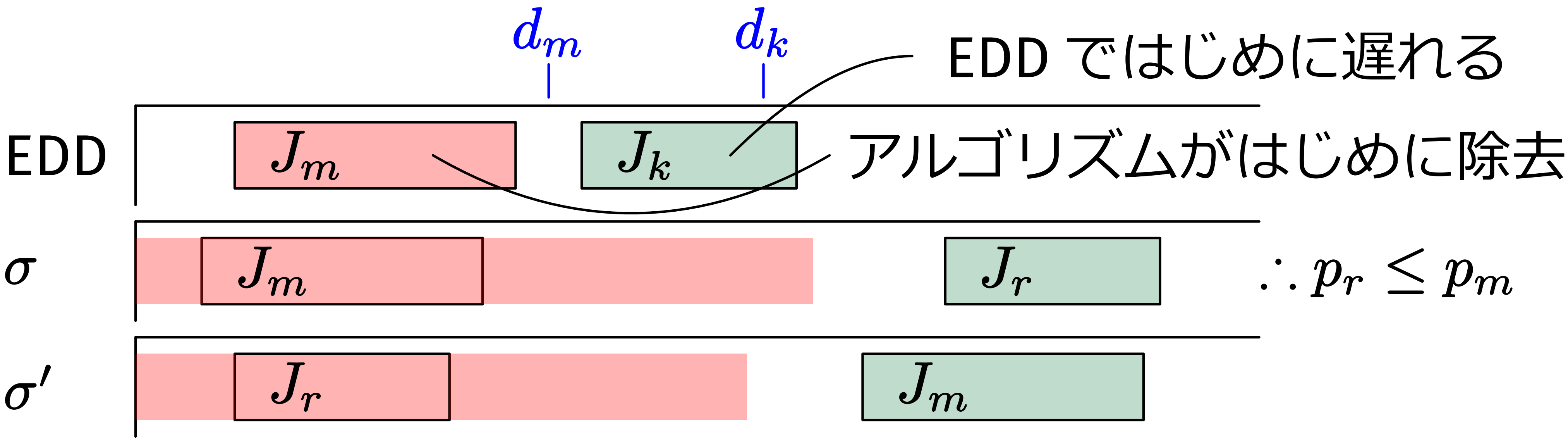
補題とその証明 (続)



- このようなスケジュール σ' があれば，σ' も最適
- 実際，スケジュール σ' は存在する
  - 任意の $\ell \in [k-1] \setminus \{m\}$ に対して，
    J_ℓ が σ で間に合う $\Rightarrow$ J_ℓ は σ' で間に合う
  - J_r は σ' で間に合う ($r = k$ のときに注意)

$$r = k \quad \Rightarrow \quad C_k(\sigma') \leq \sum_{j \in [k] \setminus \{m\}} p_j \leq \sum_{j \in [k-1]} p_j \leq d_{k-1} \leq d_k$$

補題とその証明 (続)



- このようなスケジュール σ' があれば，σ' も最適
- 実際，スケジュール σ' は存在する
  - 任意の $\ell \in [k-1] \setminus \{m\}$ に対して，
    J_ℓ が σ で間に合う $\Rightarrow$ J_ℓ は σ' で間に合う
  - J_r は σ' で間に合う ($r = k$ のときに注意)
  - 任意の $\ell \in \{k, \ldots, n\}$ に対して，
    J_ℓ が σ で間に合う $\Rightarrow$ J_ℓ は σ' で間に合う

$\square$

証明：Moore-Hodgson の正当性 (続)

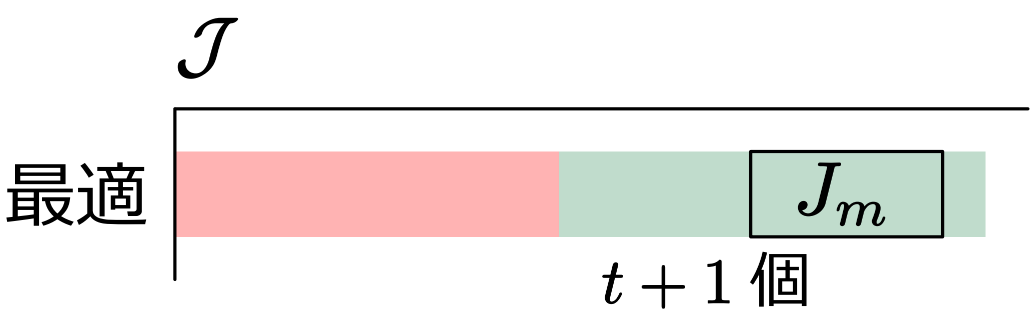
証明 (Cheriyan, Ravi, Skutella '21)：

最適値に関する帰納法

- 最適値 $= t \geq 0$ のときに正しいと仮定して，
  最適値 $= t+1$ のときを考える

  仮定：WLOG EDD 順が $J_1, J_2, \ldots, J_n$ である

  $J_m =$ アルゴリズムがはじめに除去するジョブ



(補題が存在を保証)

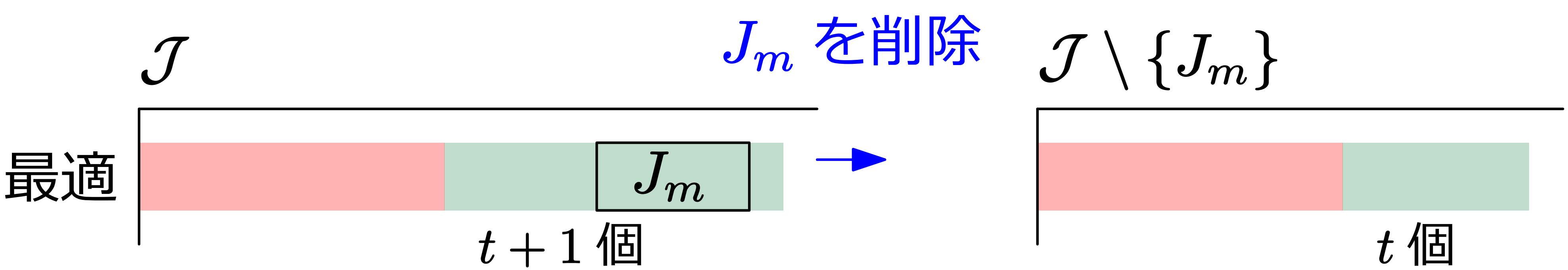
証明 (Cheriyan, Ravi, Skutella '21)：

最適値に関する帰納法

- 最適値 $= t \geq 0$ のときに正しいと仮定して，
  最適値 $= t + 1$ のときを考える

  仮定：WLOG EDD 順が $J_1, J_2, \ldots, J_n$ である

  $J_m =$ アルゴリズムがはじめに除去するジョブ



証明：Moore-Hodgson の正当性 (続)

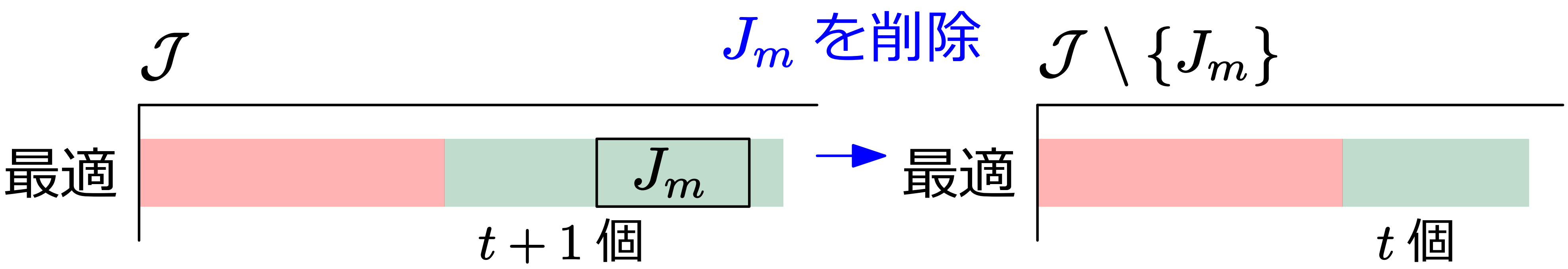
証明 (Cheriyan, Ravi, Skutella '21)：

最適値に関する帰納法

- 最適値 $= t \geq 0$ のときに正しいと仮定して，
  最適値 $= t + 1$ のときを考える

  仮定：WLOG EDD 順が $J_1, J_2, \ldots, J_n$ である

  $J_m =$ アルゴリズムがはじめに除去するジョブ



証明：Moore-Hodgson の正当性 (続)

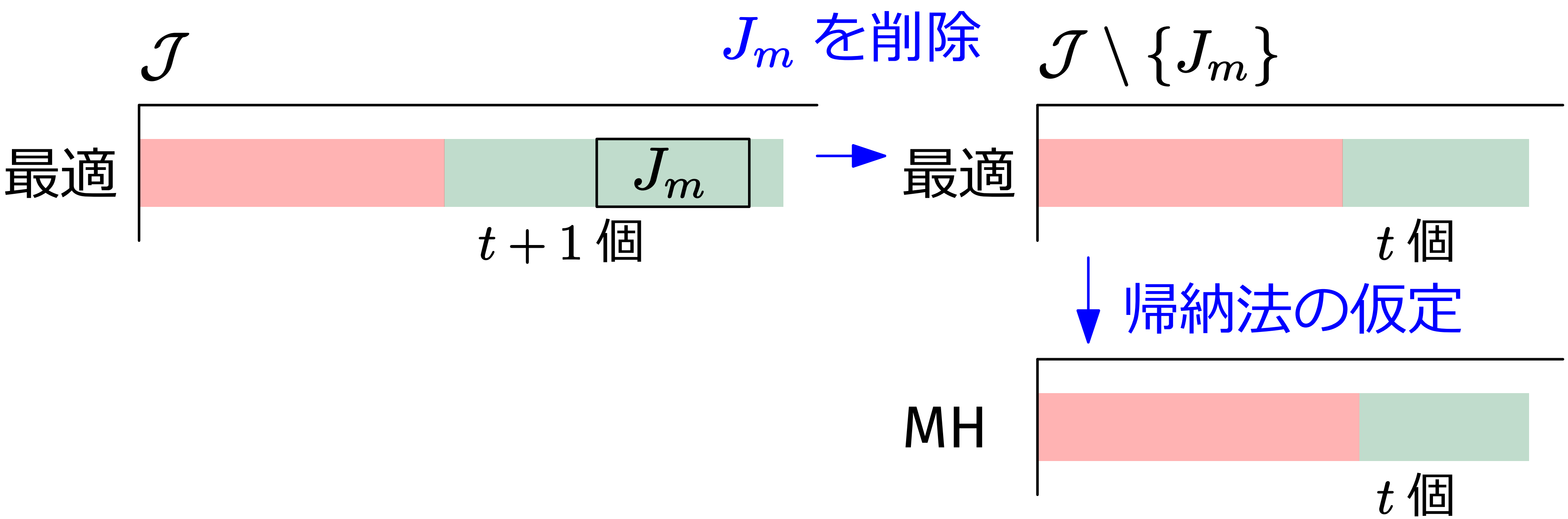
証明 (Cheriyan, Ravi, Skutella ’21)：

最適値に関する帰納法

- 最適値 $= t \geq 0$ のときに正しいと仮定して,
  最適値 $= t + 1$ のときを考える

仮定：WLOG EDD 順が $J_1, J_2, \ldots, J_n$ である

$J_m =$ アルゴリズムがはじめに除去するジョブ



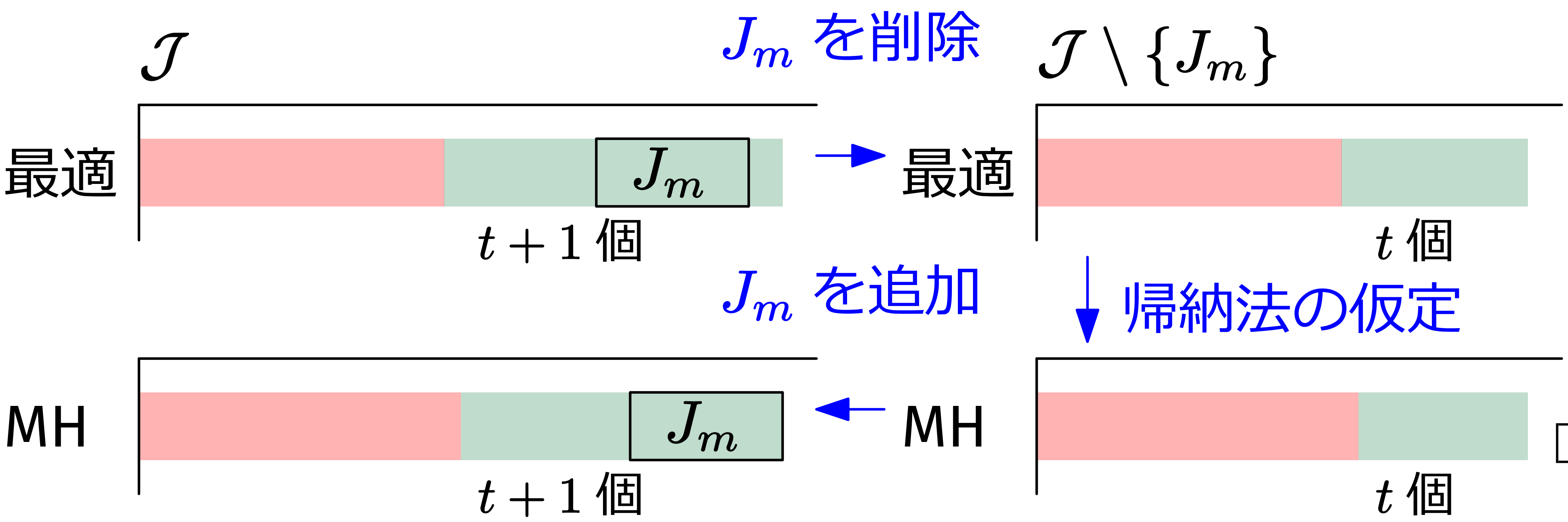
証明：Moore-Hodgson の正当性 (続)

証明 (Cheriyan, Ravi, Skutella '21)：
最適値に関する帰納法

- 最適値 $= t \geq 0$ のときに正しいと仮定して，
  最適値 $= t + 1$ のときを考える

  仮定：WLOG EDD 順が $J_1, J_2, \ldots, J_n$ である

  $J_m =$ アルゴリズムがはじめに除去するジョブ



$\square$

アルゴリズム：Moore-Hodgson アルゴリズム

1. EDD 順でジョブを処理しようとする
   (a) 納期に間に合う $\Rightarrow$「仮に間に合うジョブ」とする
   (b) 納期に間に合わない $\Rightarrow$ 仮に間に合うジョブの中で処理時間最大のジョブを遅らせる
2. 遅らせるジョブは任意の順に処理する

$\boxed{\text{計算量}}$: $n =$ ジョブの総数

0. EDD 順の計算 $\rightsquigarrow$ $O(n \log n)$ 時間
1. ヒープの利用 $\rightsquigarrow$ ジョブ 1 つあたり $O(\log n)$ 時間
2. 並べるだけ $\rightsquigarrow$ ジョブ 1 つあたり $O(1)$ 時間

$\rightsquigarrow$ 合計 $= O(n \log n)$ 時間

まとめ：一機械，ジョブの特性なし

まとめ : $1 \,||\, \gamma$

γ		計算量	(文献)
$C_{\max}$	最大完了時刻	$O(n)$	
$\sum C_j$	総完了時刻	$O(n \log n)$	(Smith ’56)
$L_{\max}$	最大納期ずれ	$O(n \log n)$	(Jackson ’55)
$\sum T_j$	総納期遅れ	$O(n^4 \sum p_j)$	(Lawler ’77)
		NP 困難	(Du, Leung ’90)
$\sum U_j$	納期遅れジョブ数	$O(n \log n)$	(Moore ’68)

重要なメッセージ：アルゴリズム全般において

設定における少しの違いが
解きやすさにおける大きな違いを生む

次回の予告

次回の予告

アルゴリズム設計技法：動的計画法

- $P2 \mid\mid C_{\max}$ (Bellman ’57)
- $1 \mid\mid \sum T_j$ (Lawler ’77)