

# 並列機械モデルにおける 最大待ち時間最小化問題の計算論的分析

15713004 天本 祐希

宋研究室

2018 年 2 月 6 日

## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

## ④ まとめと今後の課題

## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

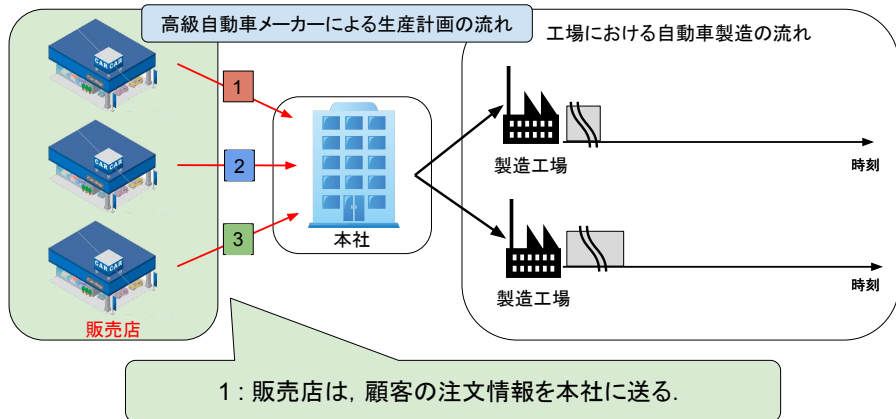
- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

## ④ まとめと今後の課題

# 研究背景と目的：受注生産方式における生産計画の流れ

## 受注生産方式とは

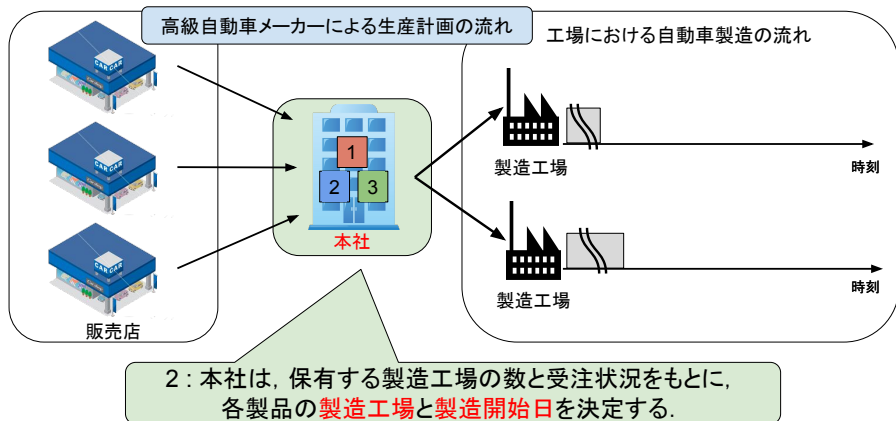
顧客注文を受けてから、その受注製品の生産を全体の生産計画に組み込むため、どの**製造工場**でいつ**製造を開始する**かを決定する。（例えば、高級自動車メーカー、システムインテグレーターなどは受注生産方式。）



# 研究背景と目的：受注生産方式における生産計画の流れ

## 受注生産方式とは

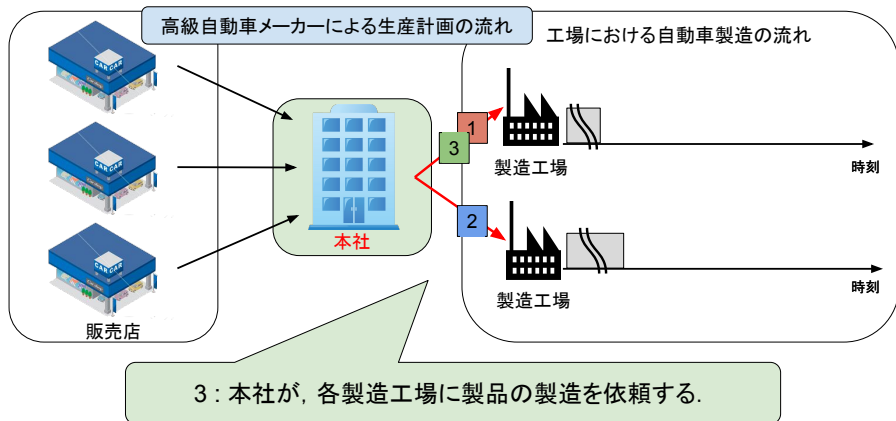
顧客注文を受けてから、その受注製品の生産を全体の生産計画に組み込むため、どの**製造工場**でいつ**製造を開始する**かを決定する。（例えば、高級自動車メーカー、システムインテグレーターなどは受注生産方式。）



# 研究背景と目的：受注生産方式における生産計画の流れ

## 受注生産方式とは

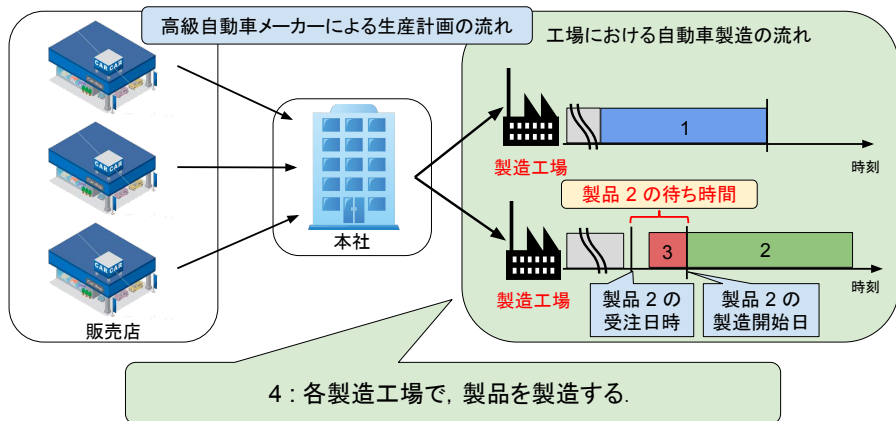
顧客注文を受けてから、その受注製品の生産を全体の生産計画に組み込むため、どの**製造工場**でいつ**製造を開始する**かを決定する。（例えば、高級自動車メーカー、システムインテグレーターなどは受注生産方式。）



# 研究背景と目的：受注生産方式における生産計画の流れ

## 受注生産方式とは

顧客注文を受けてから、その受注製品の生産を全体の生産計画に組み込むため、どの**製造工場**でいつ製造を開始するかを決定する。（例えば、高級自動車メーカー、システムインテグレーターなどは受注生産方式。）



# 研究背景と目的：スケジューリング問題との対応

## 受注生産方式における問題点

生産拠点数と受注状況によっては、受注から製造開始までの待ち時間が長くなり、顧客満足度の低下や注文のキャンセルなどに繋がる。

- 待ち時間を短縮するための生産計画を立てることは重要な課題である。

## スケジューリング問題との対応

各注文をジョブ，受注日時を処理開始可能時刻，製造期間を処理時間と対応させることで，受注生産方式における生産計画を待ち時間を目的関数とするスケジューリング問題として捉えた。

## 研究目的

上記の問題を最大待ち時間最小化問題として定式化し，

- 最大待ち時間最小化問題の計算複雑さを明らかにする。
- 最大待ち時間最小化問題の計算複雑さを踏まえて，解法の提案を行う。



## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

## ④ まとめと今後の課題

## 機械モデル

ジョブを処理する機械の台数について

- 1つの機械で処理を行う **単一機械モデル**,
- 複数の機械で処理を行う **並列機械モデル**,

に分類される。並列機械モデルはさらに,

- すべての機械の性能が等しい **同一並列機械モデル**,
- 機械ごとに処理速度が異なる **一様並列機械モデル**,
- ジョブと機械の組み合わせによって処理時間が異なる **無関連並列機械モデル**,

に分類される。

## 諸定義：最大待ち時間最小化問題

最大待ち時間最小化問題を *Sequencing to minimize maximum Waiting Time* の頭文字をとって、**SWT** と表記する． 無関連並列機械モデルにおける SWT を定式化する．

インスタンス： $\mathcal{I} = (\mathcal{J}, \mathcal{M}, r, p)$

- ジョブの集合  $\mathcal{J} = \{J_1, J_2, \dots, J_n\}$
- 無関連機械の集合  $\mathcal{M} = \{M_1, M_2, \dots, M_m\}$
- ジョブの処理開始可能時刻を返す関数  $r : \mathcal{J} \rightarrow \mathbb{N}$
- ジョブの処理時間を返す関数  $p : \mathcal{J} \times \mathcal{M} \rightarrow \mathbb{N}$

解：スケジュール  $(A, s)$

- ジョブの機械への割り当てを返す関数  $A : \mathcal{J} \rightarrow \mathcal{M}$ .
- ジョブの開始時刻を返す関数  $s : \mathcal{J} \rightarrow \mathbb{N}$ .

# 諸定義：最大待ち時間最小化問題

## 制約

- 各ジョブは処理開始可能時刻以降に処理を開始する.
  - $\forall J \in \mathcal{J} [s(J) \geq r(J)]$
- 各機械は同時に複数のジョブを処理しない.
- 各ジョブの処理を開始すると、完了するまで中断しない.
  - $\forall J, J' \in \mathcal{J} \left[ [J \neq J' \wedge A(J) = A(J')] \Rightarrow [s(J), s(J) + p(J, A(J))] \cap [s(J'), s(J') + p(J', A(J'))] = \emptyset \right]$

## 目的関数：最大待ち時間 $W_{\max}$

- ジョブの処理開始可能時刻と開始時刻の差の最大値.
  - $W_{\max} = \max_{J \in \mathcal{J}} \{s(J) - r(J)\}$
- 最大待ち時間  $W_{\max}$  の最小化を目的とする.

## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

## ④ まとめと今後の課題

## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

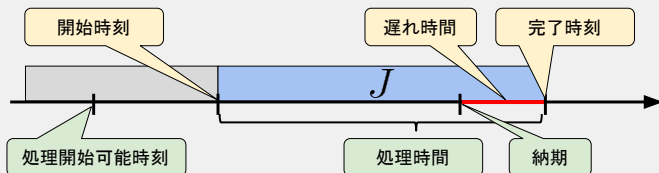
- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

## ④ まとめと今後の課題

## 既存のスケジューリング問題との対応

処理開始可能時刻付き最大遅れ時間最小化問題 (SRTD) は、ジョブの納期から処理が完了するまでの遅れ時間の最大値の最小化を目的とするスケジューリング問題である。

### 処理開始可能時刻付き最大遅れ時間最小化問題 (SRTD) との対応

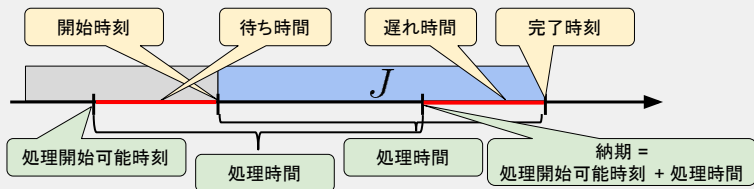


- 納期を処理開始可能時刻と処理時間の和と設定したとき、SRTD における遅れ時間は待ち時間として捉えることができる。
- SWT は SRTD の部分問題であり、SRTD は、単一機械モデルにおいて強 NP 困難であることが示されている [Garey & Johnson(1990)] が、SWT の計算複雑さは明らかでない。

## 既存のスケジューリング問題との対応

処理開始可能時刻付き最大遅れ時間最小化問題 (SRTD) は、ジョブの納期から処理が完了するまでの遅れ時間の最大値の最小化を目的とするスケジューリング問題である。

### 処理開始可能時刻付き最大遅れ時間最小化問題 (SRTD) との対応



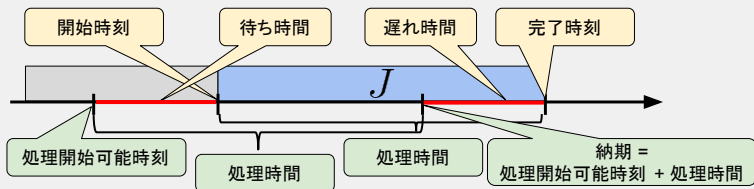
- 納期を処理開始可能時刻と処理時間の和と設定したとき、SRTD における遅れ時間は**待ち時間**として捉えることができる。
- SWT は SRTD の**部分問題**であり、SRTD は、単一機械モデルにおいて**強 NP 困難**であることが示されている [Garey & Johnson(1990)] が、SWT の計算複雑さは明らかでない。



## 既存のスケジューリング問題との対応

処理開始可能時刻付き最大遅れ時間最小化問題 (SRTD) は、ジョブの納期から処理が完了するまでの遅れ時間の最大値の最小化を目的とするスケジューリング問題である。

### 処理開始可能時刻付き最大遅れ時間最小化問題 (SRTD) との対応



- 納期を処理開始可能時刻と処理時間の和と設定したとき、SRTD における遅れ時間は**待ち時間**として捉えることができる。
- SWT は SRTD の**部分問題**であり、SRTD は、単一機械モデルにおいて**強 NP 困難**であることが示されている [Garey & Johnson(1990)] が、SWT の計算複雑さは明らかでない。

# 多項式還元の流れ

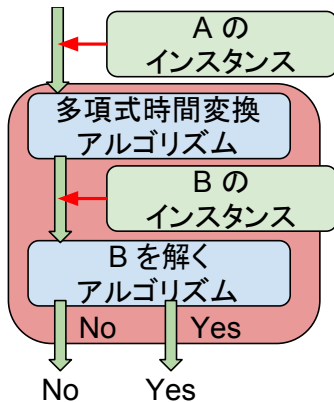
SWT の計算複雑さを明らかにするために、NP 完全であることが知られている 3-SATISFIABILITY (3-SAT) から多項式還元を行う。

## 多項式還元

問題  $A$  の任意のインスタンスを、別の問題  $B$  のあるインスタンスに多項式時間で変換して、判定結果を同じにできるとき、

「 $A$  は  $B$  に多項式還元可能」という。

$B$  を解くアルゴリズムを用いて、 $A$  も解くことができるため、 $B$  は  $A$  と同等か、それ以上難しいと言える。



3-SAT は決定問題の 1 つで、判定として Yes または No のいずれかを持つ。判定結果を同じにするために、問題  $B$  を SWT のインスタンスに対して、待ち時間  $w$  以下となるスケジュールが存在するか、という決定問題として定義し、待ち時間制約付きスケジューリング問題と表す。

## 成果 1

無関連並列機械モデルにおいて、機械数が入力の一部の場合、SWT が NP 困難であることを明らかにした。

## 還元の結果

待ち時間制約付きスケジューリング問題における最大待ち時間が、

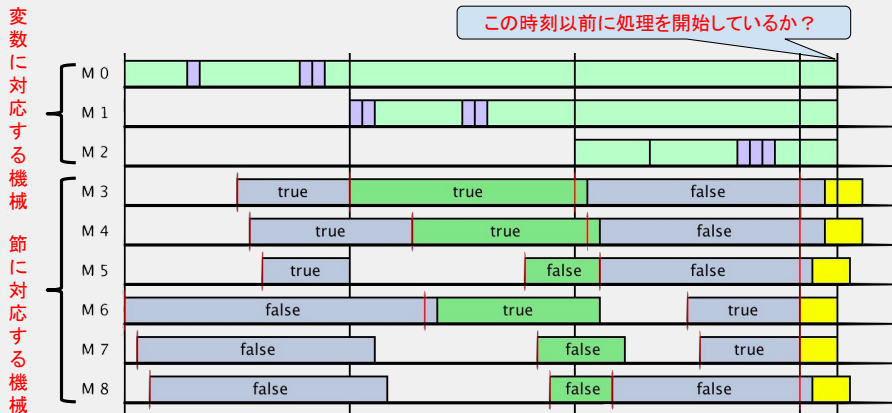
- 3-SAT の判定結果が Yes のとき、2 以下、
- 3-SAT の判定結果が No のとき、3 以上、

となるスケジュールが存在する。

無関連並列機械モデルにおいて機械数が入力の一部の場合、SWT が NP 困難である限り、近似率 1.5 未満の定数近似アルゴリズムは存在しない。

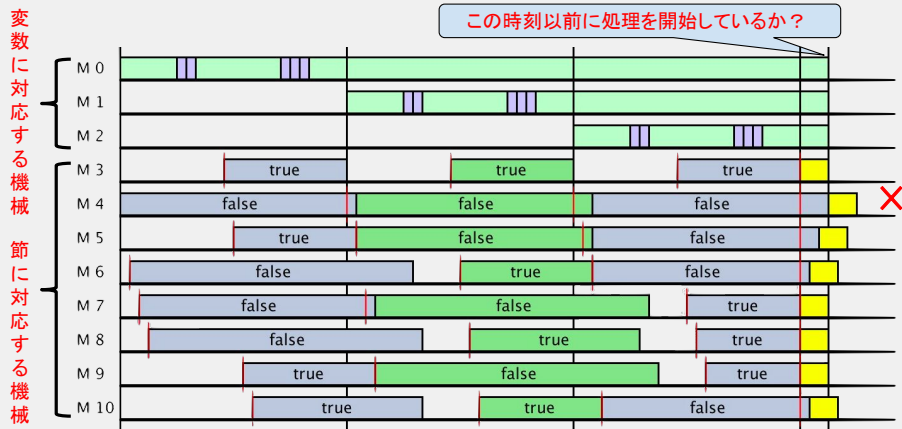
NP 困難な問題に対する効率的な解法は発見されていないため、最適解を求めるために、解の全列挙に基づいた解法が必要である。

## 3-SAT の判定結果が Yes のとき



- 3-SAT の判定結果が **Yes** のとき，すべての機械において，待ち時間制約付きスケジューリング問題における最大待ち時間が **2 以下** となる．

## 3-SAT の判定結果が No のとき



- 3-SAT の判定結果が No のとき，ある機械において，待ち時間制約付きスケジューリング問題における最大待ち時間が 3 以上となる。

## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

## ④ まとめと今後の課題

## 成果 2

同一並列機械モデルにおける SWT に対し，分割生成アルゴリズムおよび分枝限定法に基づいた厳密解法の開発と計算時間の分析を行った。

### 分割生成アルゴリズムの改良

- 分割の要素数 = 機械数 となる分割のみ生成するように改良。
  - 考慮する分割の数を減らすことができる。

### 分枝限定法の改良

- SRTD の部分問題に対する多項式アルゴリズムの概念を導入。
  - 列挙する実行可能解を減らすことができる。

### 上記以外の改良

- 各機械におけるジョブ集合のコストの降順で，探索を始める。
  - 各分割における探索初期段階で，探索の中断を判定できる。

## 研究成果：厳密解法の実験的評価

機械数 5 における計算時間の比較. ただし, 計算時間の単位はミリ秒.

- 分割生成アルゴリズムの改良前と, 改良後の計算時間の比較

ジョブ数	10	11	12	13	14
改良前	553	3,463	30,271	240,917	3,121,089
改良後	146	943	7,576	40,748	249,759
削減率	73%	72%	75%	83%	92%

- 分枝限定法の改良前と, 改良後の計算時間の比較

ジョブ数	10	11	12	13	14	15
改良前	313	1,709	9,758	54,966	295,287	1,668,411
改良後	168	971	6,887	43,903	225,043	1,301,890
削減率	46%	43%	30%	20%	23%	22%

### 成果 2 のまとめ

改良により, 計算時間を大幅に減らすことを可能にしたが, 全探索では, ジョブの増加に伴い計算時間が指数的に増大する. そのため, より大きいインスタンスに対しては, ヒューリスティックの開発が必要である.



## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

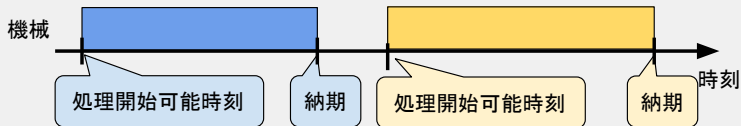
## ④ まとめと今後の課題

## 既存のスケジューリング問題との対応

JIT ジョブ荷重和最大化問題 (SJIT) は、納期以前に処理を完了したジョブの荷重和の最大化を目的とするスケジューリング問題である。

### JIT ジョブ荷重和最大化問題 (SJIT) との対応

SJIT において、各ジョブを JIT で処理可能なとき、各ジョブは納期ちょうどで処理が完了する。つまり、各ジョブの待ち時間は 0 である。



SJIT において、各ジョブを JIT で処理できるとき、処理開始可能時刻の昇順で処理することで最適解が求まる。[Cepek & Sung(2004)]

SWT において、各ジョブの待ち時間が 0 のとき、処理開始可能時刻の昇順で処理することで最適解が求まる。したがって、貪欲アリゴリズムに基づいたヒューリスティックの開発を行う。

## 成果 3

同一並列機械モデルにおける SWT に対して、貪欲アルゴリズムに基づいたヒューリスティックの開発と解法に対する実験的評価を行った。

## ヒューリスティックの評価

本研究では、ヒューリスティックから得られた解における最大待ち時間と最適解における最大待ち時間の比に基づき評価を行った。

$$\text{競合比} = \frac{\text{ヒューリスティックから得られた解における最大待ち時間} \cdots A}{\text{最適解における最大待ち時間} \cdots B}$$

## 成果 3 のまとめ

ヒューリスティックは、競合比が最大 24 となった。また、最適解における最大待ち時間が 0 のとき、ヒューリスティックは常に最適解を返す。

## ① 研究背景と目的

## ② 諸定義

## ③ 研究成果

- 最大待ち時間最小化問題の計算複雑さ
- 厳密解法の提案と実験的評価
- ヒューリスティックの提案とその精度評価

## ④ まとめと今後の課題

## まとめと今後の課題

### まとめ

- 無関連並列機械モデルにおいて機械数が入力の一部の場合，SWT が NP 困難であることを明らかにした。
  - 近似率 1.5 未満の定数近似ができないことを明らかにした。
- SWT に対する厳密解法とヒューリスティックを開発し，実験的評価を行った。

### 今後の課題

- 無関連並列機械モデル以外の機械モデルにおける SWT の計算複雑さを明らかにし，問題の難しさに影響を与える特徴を分析する。
- 3-SAT からの還元手法を工夫して，3-SAT の存在判定が Yes のとき，No のときの待ち時間制約付きスケジューリング問題における最大待ち時間の比を調整する。
  - 上記の比を大きくすることができれば，近似率がその比未満の定数近似アルゴリズムが存在しないことを保証できる。
- 上記の結果に基づき，近似アルゴリズムを開発する。



3-SATISFIABILITY (3-SAT) は決定問題の 1 つで、この問題は NP 完全であることが知られている。

### 3-SATISFIABILITY

インスタンス：  $(X, H)$

- ブール型の変数集合  $X = \{x_1, x_2, \dots, x_n\}$ 
  - リテラルの集合  $L_X = X \cup \{\bar{x} \mid x \in X\}$ 
    - $X$  と  $X$  の各要素の論理否定をとった変数の集合
- 3 つのリテラルからなる節  $h$  の集合  $H$ 
  - $H \subseteq 2^{L_X}$  s.t.  $\forall h \in H [ |h| = 3 ]$

問題： 以下を満たす真理値割り当て  $f: X \rightarrow \{0, 1\}$  が存在するか？

$$\bigwedge_{h \in H} \left( \bigvee_{x \in h} f(x) \vee \bigvee_{\bar{x} \in h} \neg f(x) \right) = 1$$

## 3-SAT の例

インスタンス：  $(X, H)$

- ブール型の変数集合  $X = \{x_1, x_2, x_3\}$ 
  - リテラルの集合  $L_X = \{x_1, x_2, x_3, \bar{x}_1, \bar{x}_2, \bar{x}_3\}$ 
    - $X$  と  $X$  の各要素の論理否定をとった変数の集合
- 3つのリテラルからなる節  $h$  の集合  $H$ 
  - $H = \{\{x_1, x_2, \bar{x}_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}\}$ 
    - $h_1 = \{x_1, x_2, \bar{x}_3\}$ ,  $h_2 = \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}$

問題： 以下を満たすように  $x_1, x_2, x_3$  を 0, 1 で決定できるか？

$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) = 1$$

$f(x_1) = 1, f(x_2) = 0, f(x_3) = 0$  が存在するので、判定は Yes となる.



### 還元のポイント

- 3-SAT の判定結果が **Yes** のとき、待ち時間制約付きスケジューリング問題における最大待ち時間は **2 以下** になる。
- 3-SAT の判定結果が **No** のとき、待ち時間制約付きスケジューリング問題における最大待ち時間は **3 以上** になる。

### 表記の導入

3-SAT のインスタンス を  $X = \{x_1, \dots, x_n\}$  と  $H = \{h_1, \dots, h_\lambda\}$  からなる 2 項組  $(X, H)$  とする.  $(X, H)$  に基づき表記を導入する.

- $\forall i \in \{1, \dots, n\}, \alpha_i = |\{h \in H \mid x_i \in h\}|$ 
  - $H$  において,  $x_i$  が現れる回数を表す自然数を  $\alpha_i$  とする.
- $\forall i \in \{1, \dots, n\}, \beta_i = |\{h \in H \mid \bar{x}_i \in h\}|$ 
  - $H$  において,  $\bar{x}_i$  が現れる回数を表す自然数を  $\beta_i$  とする.
- $\mathcal{A} = \sum_{i \in \{1, \dots, n\}} \alpha_i, \mathcal{B} = \sum_{i \in \{1, \dots, n\}} \beta_i$

$2(\mathcal{A} + \mathcal{B}) + \lambda$  個のジョブ  $\mathcal{J}$  を  $n + \lambda$  個の無関連機械で処理する.

### 変換した待ち時間制約付きスケジューリング問題のインスタンス

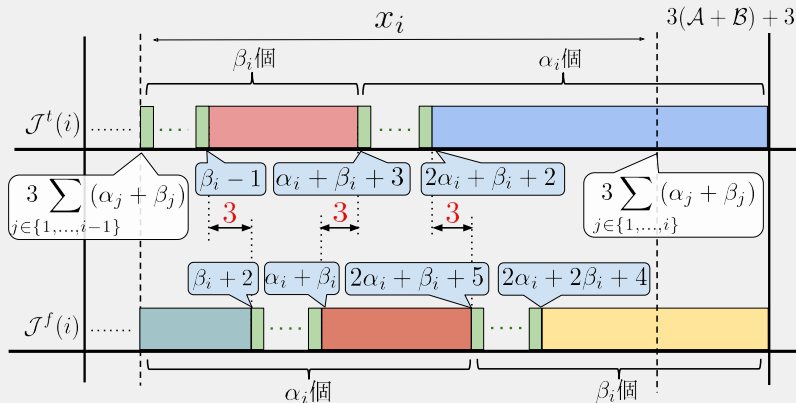
$2(\mathcal{A} + \mathcal{B}) + \lambda$  個のジョブ

- 各  $i \in \{1, \dots, n\}$  における変数  $x_i$  からなるリテラル  $x_i, \bar{x}_i$  に対応するジョブの集合を  $\mathcal{J}^t(i), \mathcal{J}^f(i)$  とする. それぞれの要素数は,  $x_i$  と  $\bar{x}_i$  が  $H$  に現れた回数, つまり,  $|\mathcal{J}^t(i)| = |\mathcal{J}^f(i)| = \alpha_i + \beta_i$ .
- $\lambda$  は,  $H$  の要素数. 節に対応する機械に割り当てる用のダミージョブ.

$n + \lambda$  個の機械

- $n$  は  $X$  の要素数. 各変数に対応する機械が 1 台ある.
  - $\lambda$  は  $H$  の要素数. 各節に対応する機械が 1 台ずつある.
- 
- $f(x_i) = 1$  のとき,  $\mathcal{J}^f(i)$  を機械  $i$  に割り当てる.
  - $f(x_i) = 0$  のとき,  $\mathcal{J}^t(i)$  を機械  $i$  に割り当てる.

## 各機械 $i \in \{1, \dots, n\}$ に対応するジョブのスケジュール



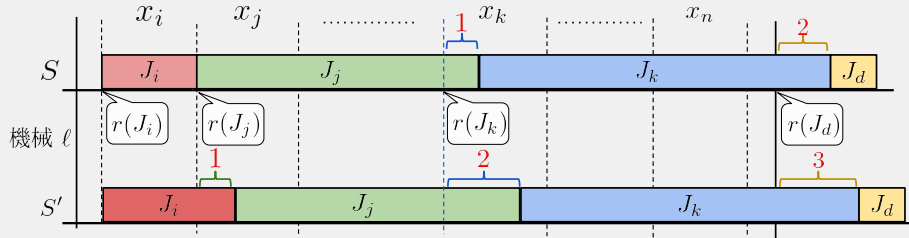
- $x_i$  と  $\bar{x}_i$  に対応するジョブの集合  $J^t(i)$  と  $J^f(i)$  におけるジョブ間の処理開始可能時刻の差を  $3$  に設定する。
- $J^t(i)$  の要素  $J^t(i)$  と  $J^f(i)$  の要素  $J^f(i)$  が機械  $i$  に同時に割り当てられないように、上記の差を設けている。

各機械  $\ell \in \{n+1, \dots, n+\lambda\}$  に対応するジョブのスケジュール

$h_{\ell-n} = \{x_i, x_j, x_k\}$  のとき,

$S$  :  $f(x_i) = 1, f(x_j) = 0, f(x_k) = 0$  のとき.

$S'$  :  $f(x_i) = 0, f(x_j) = 0, f(x_k) = 0$  のとき.



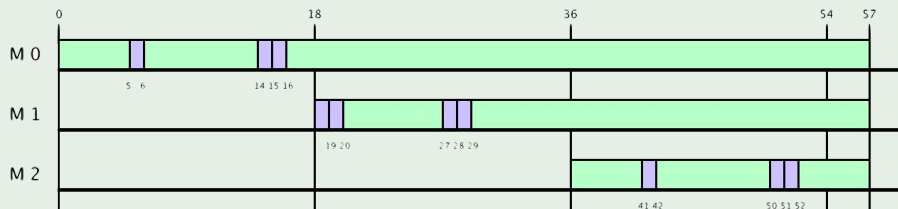
- 1 に割り当てられたリテラルに対応するジョブは,
  - 次のジョブの処理開始可能時刻 - 自身の処理開始可能時刻
- 0 に割り当てられたリテラルに対応するジョブは,
  - 次のジョブの処理開始可能時刻 - 自身の処理開始可能時刻 + 1

## 例：各変数に対応する機械におけるスケジュール

$X = \{x_1, x_2, x_3\}$ ,  $H = \{\{x_1, x_2, \bar{x}_3\}, \{x_1, x_2, x_3\}, \{x_1, \bar{x}_2, \bar{x}_3\}, \{\bar{x}_1, x_2, x_3\}, \{\bar{x}_1, \bar{x}_2, x_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}\}$  の 2 項組  $(X, H)$  が 3-SAT のインスタンスとして与えられたとき，前のスライドで説明した方法により，SWT のインスタンス  $I_{(X, H)}$  を作る．

- $(X, H)$  において， $H$  を充足する真理値割り当ての 1 つ  $f(x_1) = 1$ ,  $f(x_2) = 0$ ,  $f(x_3) = 1$  が存在する．

各  $i \in \{0, 1, 2\}$  における機械  $i$  上のスケジュールは以下の通り．

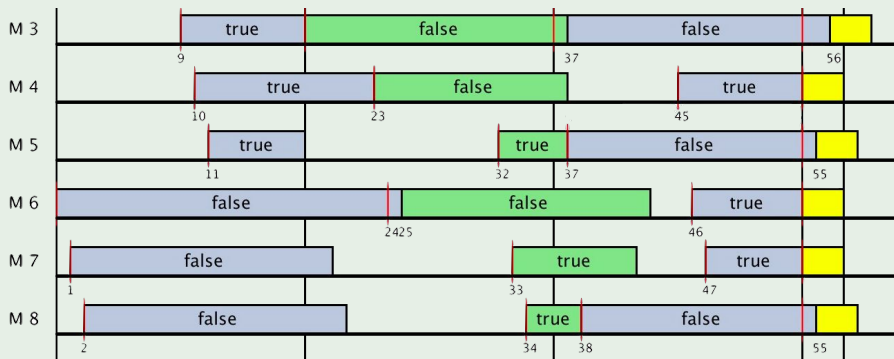


## 付録：多項式還元の例（3-SAT の判定結果が Yes のとき）

### 例：各節に対応する機械におけるスケジュール

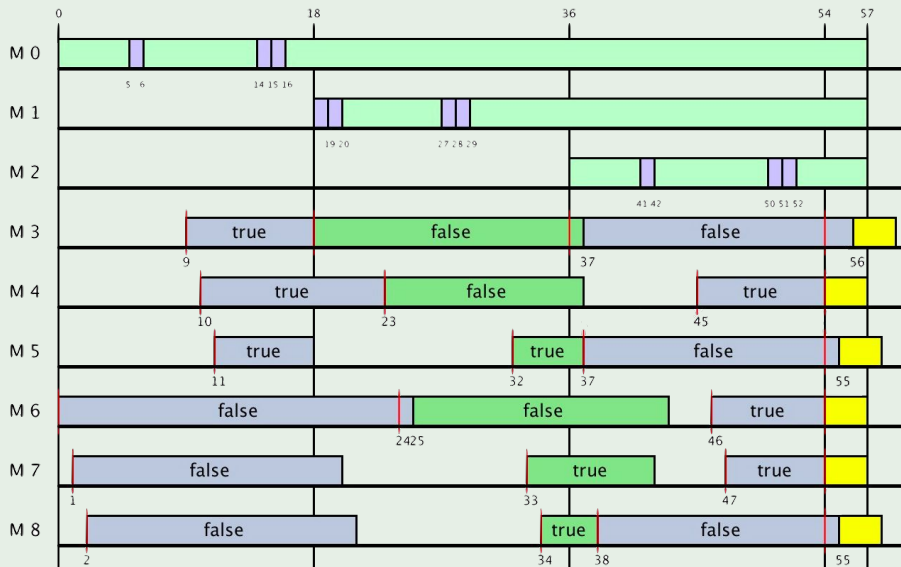
$H$  を充足する真理値割り当てが存在することから、各節におけるリテラルのなかで、少なくとも 1 つ 1 (true) に割り当てられたジョブが存在する。

したがって、各  $k \in \{3, \dots, 8\}$  における機械  $k$  に割り当てられた最後のジョブの待ち時間は高々 2 である。以下の通り。



# 付録：多項式還元の例（3-SAT の判定結果が Yes のとき）

## 例：すべての機械におけるスケジュール

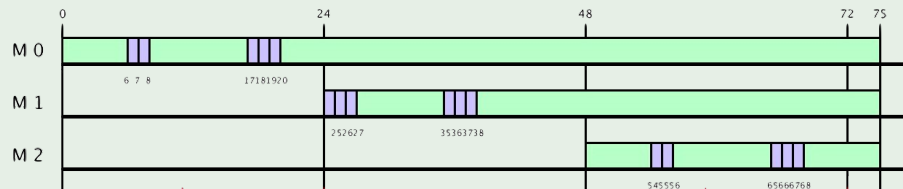


## 付録：多項式還元の例（3-SAT の判定結果が **No** のとき）

### 例：各変数に対応する機械におけるスケジュール

$X = \{x_1, x_2, x_3\}$ ,  $H = \{\{x_1, x_2, x_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}, \{x_1, \bar{x}_2, \bar{x}_3\}, \{\bar{x}_1, x_2, \bar{x}_3\}, \{\bar{x}_1, \bar{x}_2, x_3\}, \{\bar{x}_1, x_2, x_3\}, \{x_1, \bar{x}_2, x_3\}, \{x_1, x_2, \bar{x}_3\}\}$  の 2 項組  $(X, H)$  が 3-SAT のインスタンスとして与えられたとき，前のスライドで説明した方法により，SWT のインスタンス  $I_{(X,H)}$  を作る．

- $(X, H)$  において， $H$  を充足する真理値割り当ては存在しない．
- ここで， $f(x_1) = 1$ ,  $f(x_2) = 1$ ,  $f(x_3) = 1$  としたとき，各  $i \in \{0, 1, 2\}$  における機械  $i$  上のスケジュールは以下の通り．



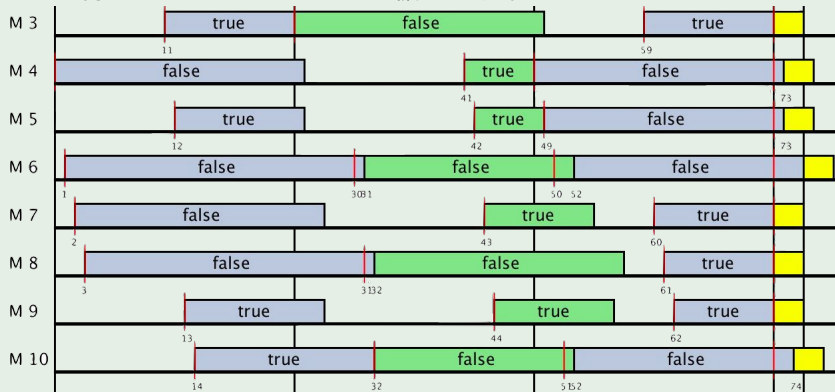


## 付録：多項式還元の例（3-SAT の判定結果が **No** のとき）

### 例：各節に対応する機械におけるスケジュール

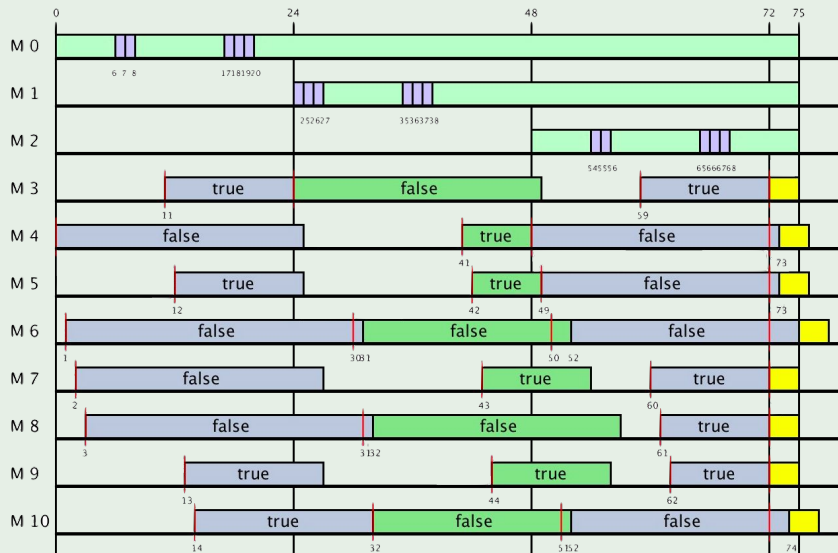
$H$  を充足する真理値割り当てが存在しないことから、各節におけるリテラルのなかで、すべて 0 (false) に割り当てられたジョブが割り当てられた機械が少なくとも 1 つ存在する。

したがって、各  $k \in \{3, \dots, 10\}$  における機械  $k$  に割り当てられた最後のジョブの待ち時間が 3 以上となる機械が存在する。以下の通り。



# 付録：多項式還元の例（3-SAT の判定結果が **No** のとき）

## 例：すべての機械におけるスケジュール



### 特殊なケースにおけるインスタンスの作り方

例えば、 $X = \{x_1, x_2, x_3\}$ ,  $H = \{\{x_1, x_2, x_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}\}$  のとき、

- 各  $i \in \{1, 2, 3\}$  に対して、 $\alpha_i, \beta_i$  は 3 以下である。
- このとき、各  $i \in \{1, 2, 3\}$  に対して、 $\alpha_i, \beta_i$  は 3 以上となるように、ダミーの節を作る。以下の通り。

$$H = \{\{x_1, x_2, x_3\}, \{x_1, x_2, x_3\}, \{x_1, x_2, x_3\}, \\ \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}\}$$

- 作成したダミー節は、本来  $H$  の要素である節と同じ節を複数個用意したに過ぎない。そのため、真理値割り当てに関係なく、判定結果にも影響しない。

上記のように、ダミー節を含んだ節の集合を作り、その後、待ち時間制約付きスケジューリング問題のインスタンスへの変換する。インスタンスの変換方法は、前のスライドで記述した通り。

### 3-SAT のインスタンスにおいて除くケース

$H$  に現れるリテラルが各変数、もしくは変数の論理否定をとったもののいずれかしか現れないかつ、そのリテラルがすべての節に現れる場合、 $H$  を充足する真理値割り当ては常に存在するためこの場合を除く。

### 例：3-SAT のインスタンスにおいて除くケース

$X = \{x_1, x_2, x_3\}$ ,  $H = \{\{x_1, x_2, \bar{x}_3\}, \{\bar{x}_1, \bar{x}_2, \bar{x}_3\}\}$  のとき、変数  $x_3$  に対応するリテラルは、 $\bar{x}_3$  しか  $H$  に現れない。さらに、すべての節に  $\bar{x}_3$  が現れる。このとき、 $f(x_3) = 0$  とすることで、 $H$  を充足する真理値割り当てが常に存在する。

## 貪欲アルゴリズムに基づいた解法

入力： $I = (\mathcal{J}, \mathcal{M}, r, p, C)$

出力：スケジュールの集合  $\mathcal{S}$ .

Step 1.  $\mathcal{J}$  を処理開始可能時刻の昇順でソートする.

Step 2. 各  $1 \leq i \leq n$  における  $J_i$  について以下の処理を繰り返す.

Step 2.1. 最小完了時刻を持つスケジュール集合の要素

$S_{M_a} \in \left\{ \arg \min_{M \in \mathcal{M}} C(S_M) \right\}$  を 1 つ求める. ここで,

$S_{M_a} := S_{M_a} \cup J_i$  とする.

Step 3.  $\mathcal{S} = \{S_M \mid M \in \mathcal{M}\}$  として,  $\mathcal{S}$  を出力する.

## 付録：分割生成アルゴリズムの改良

同一並列機械モデルにおいて、ジョブ  $J$  を機械 1 に割り当てるのと、機械 2 に割り当てることは同じである。したがって、ジョブをどの機械に割り当てるかではなく、**どのジョブと同じ機械に割り当てるか**を考える必要がある。つまり、同一並列機械モデルにおけるジョブの機械への割り当ては、**ジョブの分割**として捉えることができる。

例：ジョブ数 3，同一機械数 2 における分割

$(0, 0, 0)$



$(0, 0, 1)$



$(0, 1, 0)$



$(0, 1, 1)$



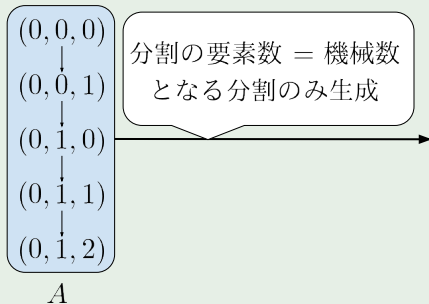
$(0, 1, 2)$

$A$

## 付録：分割生成アルゴリズムの改良

同一並列機械モデルにおいて、ジョブ  $J$  を機械 1 に割り当てるのと、機械 2 に割り当てることは同じである。したがって、ジョブをどの機械に割り当てるかではなく、**どのジョブと同じ機械に割り当てるか**を考える必要がある。つまり、同一並列機械モデルにおけるジョブの機械への割り当ては、**ジョブの分割**として捉えることができる。

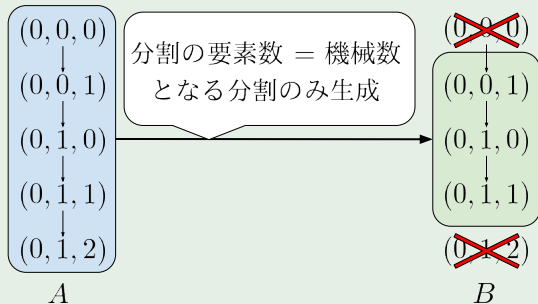
### 例：ジョブ数 3，同一機械数 2 における分割



## 付録：分割生成アルゴリズムの改良

同一並列機械モデルにおいて、ジョブ  $J$  を機械 1 に割り当てるのと、機械 2 に割り当てることは同じである。したがって、ジョブをどの機械に割り当てるかではなく、**どのジョブと同じ機械に割り当てるか**を考える必要がある。つまり、同一並列機械モデルにおけるジョブの機械への割り当ては、**ジョブの分割**として捉えることができる。

### 例：ジョブ数 3，同一機械数 2 における分割

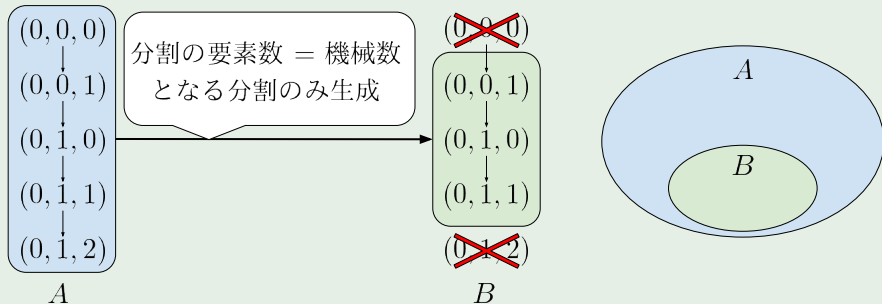




## 付録：分割生成アルゴリズムの改良

同一並列機械モデルにおいて、ジョブ  $J$  を機械 1 に割り当てるのと、機械 2 に割り当てることは同じである。したがって、ジョブをどの機械に割り当てるかではなく、**どのジョブと同じ機械に割り当てるか**を考える必要がある。つまり、同一並列機械モデルにおけるジョブの機械への割り当ては、**ジョブの分割**として捉えることができる。

### 例：ジョブ数 3，同一機械数 2 における分割



## 付録：分割生成アルゴリズムの改良部分

### 表記の導入

- ジョブ数  $n$ , 機械数  $m$
- 要素数  $n$  の配列  $b$  は分割を表す.
- 要素数  $n$  の配列  $b_{\max}$  は 各  $1 \leq i \leq n$  に対して,  $i$  番目の要素は, それまでのジョブが割り当てられた機械の最大数を格納している.  
つまり,  $b_{\max}[i] = \max_{1 \leq j \leq i-1} b[j]$

### 分割生成アルゴリズムの改良部分

入力 :  $I = (b, b_{\max}, m)$

出力 : 分割  $b$

Step 1. 各  $1 \leq i \leq n$  に対して, 以下の処理を繰り返す.

Step 1.1.  $b_{\max}[n-i] \geq m-i$  のとき, 処理を終了する.

Step 1.2.  $b[n-i] := m-i$ ,  $b_{\max}[n-i] := m-i$  とする.

Step 2.  $b$  を出力する.

例：  $b = (0, 1, 0, 1)$ ,  $b_{\max} = (0, 1, 1, 1)$ ,  $m = 3$  のとき

Step 1.  $b_{\max}(0, 1, 1, \mathbf{1})$  より,  $1 < 3 - 1$ .

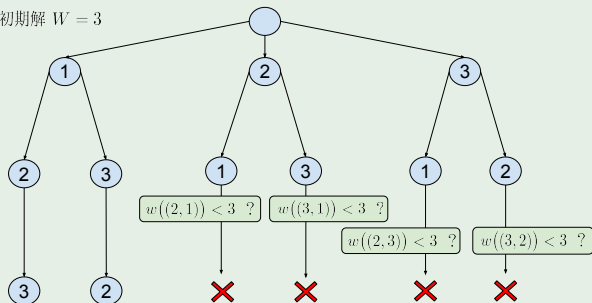
Step 2. したがって,  $b = (0, 1, 0, 2)$ ,  $b_{\max} = (0, 1, 1, 2)$  とする.

Step 3.  $b_{\max}(0, 1, \mathbf{1}, 2)$  より,  $1 \geq 3 - 2$ .

Step 4.  $b$  を出力する.

## 例：以下のインスタンスにおけるスケジュール生成

初期解  $W = 3$

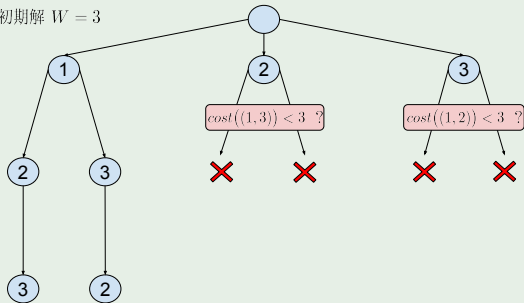


$J$	$r(J)$	$p(J)$
$J_1$	2	4
$J_2$	5	7
$J_3$	10	15

- それまでに割り当てたジョブのスケジュールにおける最大待ち時間がそれまでの最良の解  $W$  より良い値か？
- 割り当てていない残りのジョブを最適にスケジュールしたときの最大待ち時間がそれまでの最良の解  $W$  より良い値か？
  - SWT は、処理時間が一定のとき、処理開始可能時刻の昇順で処理することで最適解が求まる。

例：以下のインスタンスにおけるスケジュール生成

初期解  $W = 3$



$\mathcal{J}$	$r(J)$	$p(J)$
$J_1$	2	4
$J_2$	5	7
$J_3$	10	15

- それまでに割り当てたジョブのスケジュールにおける最大待ち時間がそれまでの最良の解  $W$  より良い値か？
- 割り当てていない残りのジョブを最適にスケジュールしたときの最大待ち時間がそれまでの最良の解  $W$  より良い値か？
  - SWT は、処理時間が一定のとき、処理開始可能時刻の昇順で処理することで最適解が求まる。

### 分枝限定法の改良部分

入力：  $I = (\mathcal{J}', S)$

出力： スケジュール  $S$ .

Step 1. 各  $1 \leq i \leq |\mathcal{J}'|$  における  $J'_i$  について、以下の処理を繰り返す.

Step 1.1.  $p(J'_i) = \min_{j \in \{1, \dots, |\mathcal{J}'|\}} p(J'_j)$  とする.

Step 1.2.  $S := S \cup J'_i$  とする.

Step 2.  $S$  を出力する.

- Step 1. で  $\mathcal{J}'$  におけるすべてのジョブの処理時間を  $\mathcal{J}'$  中の最小の処理時間に設定している.
- 上記の操作により、 $\mathcal{J}'$  におけるジョブの本来の処理時間は、設定した処理時間以上である.
- このとき、処理開始可能時刻順で処理して得られるスケジュールにおける最大待ち時間は、 $\mathcal{J}'$  におけるコストの下限である.

分割生成アルゴリズムと分枝限定法の改良下部分を用いて、さらに計算効率を向上させるために、以下の改良を加えた。

**例：以下の分割のとき、順列を生成する機械順の違い**

ジョブ数 5，同一機械数 3 において，分割が  $(0, 1, 2, 1, 1)$  のとき，通常版と改良版における順列を生成する機械順は以下の通り。

**通常版** 機械 0 から順に処理を行う。  $0 \rightarrow 1 \rightarrow 2$ 。

**改良版** 各機械に割り当てられたジョブ集合におけるコストの下限の降順で処理を行う。  $1 \rightarrow 0 \rightarrow 2$ 。

上記改良により，コストの下限が悪いスケジュールを生成した機械から順に，その機械におけるジョブ集合の順列を生成することで，目的関数に与える影響の大きいスケジュールを早い段階で生成できる。