組合せ最適化問題に対する解法

厳密解法 最悪の場合，求解までに時間がかかるかもしれないが厳密に最適解を求める

- 分枝限定法
- 分枝カット法
- 動的計画法
- ...

近似解法 最適解を求めることを諦めて，ある程度最適値に近い値をもつ実行可能解をなるべく早く求める

- 制度保証付き近似解法
- メタヒューリスティック（タブーサーチ，遺伝的アルゴリズム，アニーリング法...）
- 乱択アルゴリズム
- ...

精度保証付き近似解法

最大化問題

- OPT: **最適値**
- Obj： **アルゴリズムで得られる目的関数値**
- OPT/Obj： **このアルゴリズムの** [　　]
- **どんな問題例に対しても，**$\mathrm{OPT}/\mathrm{Obj} \leq \alpha$ **を満たすとき，このアルゴリズムを** [　　　　] **という．**

最小化問題に対しては，$\mathrm{Obj}/\mathrm{OPT} \leq \alpha$ **を満たす** α **で評価する．**

0-1 ナップサック問題に対する 2-近似アルゴリズム

$$
\begin{array}{lllllll}
\textbf{最大化} & c_1x_1 & + & c_2x_2 & + & \cdots + & c_nx_n \\
\underset{\boldsymbol{x}}{\textbf{条件}} & a_1x_1 & + & a_2x_2 & + & \cdots + & a_nx_n \leq b \\
 & x_j \in \{0,1\}, & & & & j = 1,2,\ldots,n &
\end{array}
$$

- **貪欲アルゴリズムの利用**
- **2-近似アルゴリズム**
  $=$ アルゴリズムで得られる解の目的関数値は悪くても最適値の半分以上

貪欲アルゴリズムを用いて解 $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ を求める
if $\sum_{j=1}^{n} c_j\hat{x}_j < \max\{c_j \mid \hat{x}_j = 0\}$ **then**
$c_{\hat{p}} = \max\{c_j \mid \hat{x}_j = 0\}$ **である添字** $\hat{p}$ **に対して，**

$$
\tilde{x}_j \leftarrow \begin{cases} 1 & (j = \hat{p}) \\ 0 & (j \neq \hat{p}) \end{cases}
$$

else
$(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n) \leftarrow (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$
end if

2-近似アルゴリズムの正当性

Theorem

0-1 **ナップサック問題に対する最適解を** $(x_1^*, x_2^*, \ldots, x_n^*)$, **アルゴリズムで得られた解** $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ **は，**

$$\frac{\sum_{j=1}^{n} c_j x_j^*}{\sum_{j=1}^{n} c_j \tilde{x}_j} \leq 2.$$

【証明】
$\sum_{j=1}^{n} c_j \hat{x}_j + c_{\hat{p}}$ **はナップサック問題を連続緩和した最適値よりも小さくはない．よって，** $\sum_{j=1}^{n} c_j \hat{x}_j + c_{\hat{p}} \geq \sum_{j=1}^{n} c_j x_j^*$ **が成り立ち，**

$$\sum_{j=1}^{n} c_j \tilde{x}_j = \max\{\sum_{j=1}^{n} c_j \hat{x}_j, c_{\hat{p}}\} \geq \frac{1}{2} \sum_{j=1}^{n} c_j x_j^*$$

を得る． □

ナップサック問題に対する 2-近似アルゴリズム

4 つのアイテム a,b,c,d が与えられ，$b = 5$ のときの 0-1 ナップサック問題を考える．

アイテム	a	b	c	d
体積 a_j	2	5	2	3
価値 c_j	3	6	2	1
効率 c_j/a_j	1.5	1.2	1.0	0.3

- **貪欲アルゴリズムの解は**

  となり，目的関数値は　である．
- **2-近似アルゴリズムの解は**

0-1 ナップサック問題に対する $(1+\varepsilon)$-近似アルゴリズム

- 任意の $\varepsilon > 0$ に対して，近似比が悪くても $(1+\varepsilon)$ となるアルゴリズム
- ε が小さいと最適解に近づくが，その分，計算時間が必要となり，[　　]と[　　]にはトレードオフの関係

1: 2-近似アルゴリズムを用いて解 $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ を求める
2: $\tilde{z} \leftarrow \sum_{j=1}^{n} c_j \tilde{x}_j$
3: $t \leftarrow (\varepsilon \tilde{z})/n$; $\bar{c}_j \leftarrow \lfloor c_j/t \rfloor$ $(j = 1, \ldots, n)$
4: $\bar{c}_j$ を価値とする 0-1 ナップサック問題の最適解 $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ を求める
5: **if** $\sum_{j=1}^{n} c_j \tilde{x}_j > \sum_{j=1}^{n} c_j \bar{x}_j$ **then**
6: 　$(\check{x}_1, \check{x}_2, \ldots, \check{x}_n) \leftarrow (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$
7: **else**
8: 　$(\check{x}_1, \check{x}_2, \ldots, \check{x}_n) \leftarrow (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$
9: **end if**

$(1+\varepsilon)$-近似アルゴリズムの正当性

Theorem

0-1 ナップサック問題に対する最適解を $(x_1^*, x_2^*, \ldots, x_n^*)$, アルゴリズムで得られた解を $(\check{x}_1, \check{x}_2, \ldots, \check{x}_n)$ とすると，

$$\frac{\sum_{j=1}^{n} c_j x_j^*}{\sum_{j=1}^{n} c_j \check{x}_j} \leq 1 + \varepsilon.$$

【証明】

$\frac{c_j}{t} - 1 < \bar{c}_j \leq \frac{c_j}{t}$ **なので，** $(\because$

$$\begin{aligned}
\sum c_j \bar{x}_j &\geq t \sum \bar{c}_j \bar{x}_j && (\because \\
&\geq t \sum \bar{c}_j x_j^* && (\because \\
&> \sum (c_j - t) x_j^* && (\because \\
&\geq \sum c_j x_j^* - tn && (\because \\
&= \sum c_j x_j^* - \varepsilon \sum c_j \tilde{x}_j && (\because
\end{aligned}$$

よって，

$$\sum c_j \check{x}_j > \sum c_j x_j^* - \varepsilon \sum c_j \check{x}_j$$

□

$\bar{c}$ を価値とする 0-1 ナップサック問題の求解

1: 2-近似アルゴリズムを用いて解 $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$ を求める
2: $\tilde{z} \leftarrow \sum_{j=1}^{n} c_j \tilde{x}_j$
3: $t \leftarrow (\varepsilon \tilde{z})/n$; $\bar{c}_j \leftarrow \lfloor c_j/t \rfloor$ $(j = 1, \ldots, n)$
4: $\bar{c}_j$ を価値とする 0-1 ナップサック問題の最適解 $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ を求める
5: **if** $\sum_{j=1}^{n} c_j \tilde{x}_j > \sum_{j=1}^{n} c_j \bar{x}_j$ **then**
6: $\quad (\check{x}_1, \check{x}_2, \ldots, \check{x}_n) \leftarrow (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)$
7: **else**
8: $\quad (\check{x}_1, \check{x}_2, \ldots, \check{x}_n) \leftarrow (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$
9: **end if**

アルゴリズムの 4 行目では，［　　］により最適解を求める．

- 上界値 Z は［　　］
- 動的計画法の表の大きさは入力データの大きさ（上界値）に無関係になる．（　　）

ε が小さくなると表が大きくなり，計算に時間がかかるが，最適解に近い解が得られる．

ナップサック問題に対する $(1+\varepsilon)$-近似アルゴリズム

5 つのアイテム 1,2,3,4,5 が与えられ，$b = 9$ のときの 0-1 ナップサック問題を考える．

アイテム	1	2	3	4	5
体積 a_j	3	3	3	7	4
価値 c_j	24	25	26	62	36
効率 c_j/a_j	8	8.3	8.7	8.9	9

$\varepsilon = 0.5$ **とする**.

- **貪欲アルゴリズムの解は**
- 2-**近似アルゴリズムの解は**
- $\tilde{z} =$ **を得る**. $t =$

- $\tilde{z} = 62$ **を得る**.$t = 0.5 \times 62/5 = 6.2$

アイテム	1	2	3	4	5
体積 a_j	3	3	3	7	5
価値 $\bar{c}_j$					

これを動的計画法 2 **で解く**

- **上界値** $Z =$

$k\backslash v$	0	1	2	3	4	5	6	7	8	9	10	11
1												
2												
3												
4												
5												

12	13	14	15	16	17	18	19	20

$\max\{v \mid Y_n(v) \leq 9\} = 11$

スケジューリング問題に対する近似解法

Definition

n 個のジョブを m 台の機械で処理するとき，ジョブの機械への割当と処理順序 (処理開始時間）を決定する．

- ジョブ j の処理時間 p_j と重み w_j が分かっている
- ジョブ j の処理開始時間 s_j は，同じ機械での処理が重ならないように決定
- $\max_{j=i}^{n} w_j(s_j + p_j)$ $\rightarrow$ 最小化: 最終完了時刻最小化スケジューリング問題

最小化問題なので，近似比の取り方が前項までと異なることに注意．

- 貪欲アルゴリズムで近似解が求められる (リストスケジューリング)
- 処理待ちのジョブをリストで保持
- ある機械のジョブの処理が終了したらすぐにリストからジョブを取り出し，取り出したジョブの処理を開始する．

リストスケジューリング

$C(i)$:機械 i に割り当てられたジョブの処理終了時刻

ジョブをリスト L に格納
$C(i) \leftarrow 0\ (i = 1, \ldots, m)$
while $L \neq \emptyset$ **do**
  $t \leftarrow \min\{C(i) \mid i = 1, \ldots, m\}$
  $\hat{i} \leftarrow C(\hat{i}) = t$ を満たす機械の番号
  リスト L からジョブ j を取り出し，機械 $\hat{i}$ で時刻 t に処理開始する
  $C(\hat{i}) \leftarrow C(\hat{i}) + p_j$
end while

リストスケジューリングの近似比

Theorem

m 台の機械で n 個のジョブを処理するとき

- $C_{\min}$: 最小の最終完了時刻
- C_{LIST}: リストスケジューリングで処理したときの最小完了時刻を

$$\frac{C_{\mathrm{LIST}}}{C_{\min}} \leq 2 - \frac{1}{m}$$

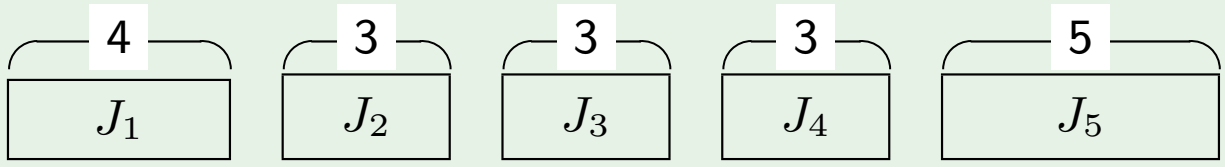
- l: リストスケジューリングで最後に処理が完了するジョブ
- ジョブ l の開始時刻 $C_{\mathrm{LIST}} - p_l$ までの間，すべての機械でジョブが処理されている $\Rightarrow \quad m \cdot (C_{\mathrm{LIST}} - p_l) \leq \sum_j p_j - p_l$
- m 台の機械を $C_{\min}$ まで稼働すれば，すべてのジョブの処理が可能
  $\Rightarrow \quad \sum_j p_j \leq m \cdot C_{\min}$
- $p_l \leq C_{\min}$ が成立

$$C_{\mathrm{LIST}} \leq \frac{\sum_j p_j}{m} + (1 - \frac{1}{m})p_l \leq C_{\min} + (1 - \frac{1}{m})p_l \leq (2 - \frac{1}{m})C_{\min}$$

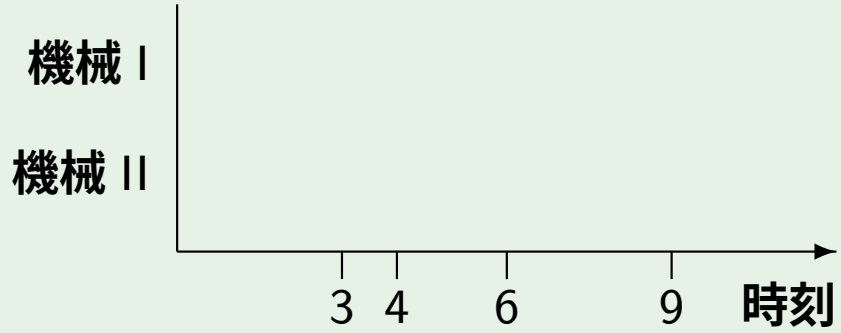
□

最終完了時刻最小化のリストスケジューリング

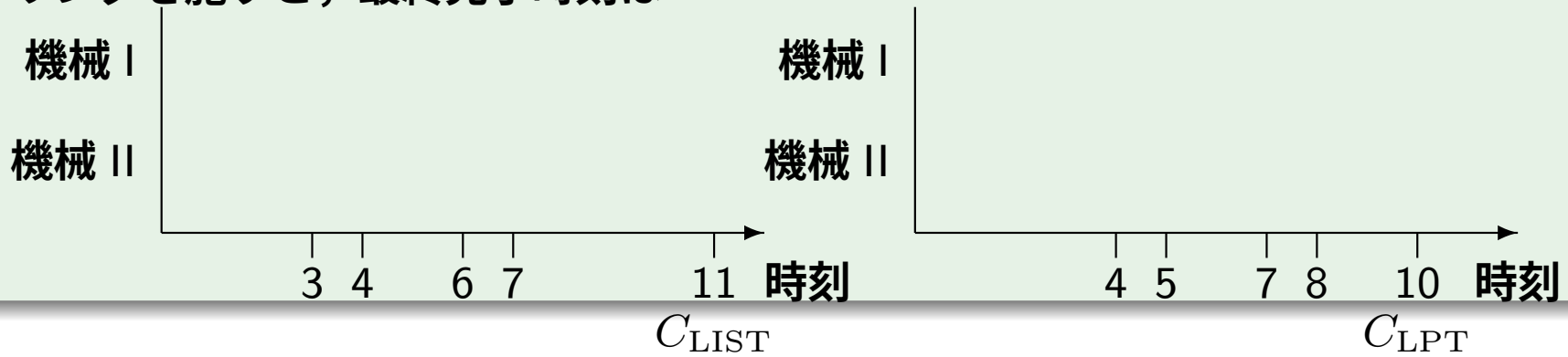
図に示す５つのジョブ J_1, J_2, J_3, J_4, J_5 をこの順番にリストに格納し，2台の機械にリストスケジューリングで割り当てる．



最適なスケジュールでは
最終完了時刻 $C_{\min}$ は



リストスケジューリングでは，最終完了時刻 C_{LIST} は　．
リストに格納するジョブの順番を処理時間の降順に J_5, J_1, J_2, J_3, J_4 としてリストスケジューリングを施すと，最終完了時刻は



最長処理時間 (LPT) ルール

リストスケジューリング+処理時間の長いものから順に割り当てる

Theorem

$m(\geq 2)$ 台の機械で n 個のジョブを処理するときの最小の最終完了時刻を $C_{\min}$，処理時間の長いものから順に割り当てるルールを適用してリストスケジューリングで処理したときの最終完了時刻を C_{LPT} とすると，

$$\frac{C_{\mathrm{LPT}}}{C_{\min}} \leq \frac{4}{3} - \frac{1}{3m}.$$

【証明】
背理法で証明しよう．LPT ルールで処理したときの最終完了時刻 C_{LPT} が $\frac{C_{\mathrm{LPT}}}{C_{\min}} > \frac{4}{3} - \frac{1}{3m}$ となる問題例の中でジョブ数が最小の場合を考える．最後に処理が完了するジョブを l とする．リストスケジューリングの証明と同様にして $C_{\mathrm{LPT}} \leq C_{\min} + (1 - \frac{1}{m})p_l$ が成り立つので，
$\frac{4}{3} - \frac{1}{3m} < \frac{C_{\mathrm{LPT}}}{C_{\min}} \leq 1 + (1 - \frac{1}{m})\frac{p_l}{C_{\min}}$ より，$C_{\min} < 3p_l$ を得る．

ジョブ l より後に開始して l より前に終了するジョブがある場合，このジョブを除いても仮定を満たすので，ジョブ l は最後に処理開始される．よって，LPT ルールより p_l が最も処理時間は短い．ここで，$C_{\min} < 3p_l$ なので，最適なスケジュールでは各機械ではたかだか２つのジョブしか処理しない．この場合，LPT ルールを用いても最適なスケジュールとなり $C_{\min} = C_{\mathrm{LPT}}$ であるので矛盾．

演習問題

1. ナップサック問題に対する 2-近似アルゴリズムで近似比が 2 に近づく問題例を示せ．
2. 最終完了時刻最小化スケジューリング問題に対するリストスケジューリングで近似比が $2 - \frac{1}{m}$ に近づく問題例を示せ．
3. 以下のジョブからなる最終完了時刻最小化スケジューリング問題で，与えられたジョブを 3 台の機械にスケジュールせよ．最小最終完了時刻 $C_{\min}$，リストスケジューリングでの最終完了時刻 C_{LIST}，LPT ルールを用いたときの最終完了時刻 C_{LPT} を求めよ．