2004－AL－94　(1)
2004／3／19

# 周期的なタイムスロット付きジャストインタイムスケジューリング問題のヒューリスティックアルゴリズム

千葉英史 and 平石邦彦
北陸先端科学技術大学院大学情報科学研究科
〒 923-1292 石川県能美郡辰口町旭台 1-1
email:{e-chiba, hira}@jaist.ac.jp

**概要:** 与えられた納期ちょうどにジョブの処理を完了する様なスケジュールを求める問題をジャストインタイムスケジューリング問題と呼ぶ．我々は周期的なタイムスロットという現実的な仮定を設けたジャストインタイムスケジューリング問題を取り扱う．本稿ではまず，$P \neq NP$ の仮定の下で近似不可能であることを証明する．次に，機械数が 1 である場合のヒューリスティックアルゴリズムを提案する．アイデアは最小費用流問題への帰着であり，最小費用流を求める以外はフロー値の単純な更新をするだけなので，ヒューリスティックアルゴリズムは高速である．実際に計算機実験も行って確かめた．また，ある制約の下でヒューリスティックアルゴリズムが最適解を返すことを示す．さらに，ある制約の下でヒューリスティックアルゴリズムの近似比を与える．

**キーワード:** スケジューリング，ジャストインタイム，セットアップタイム，ヒューリスティックアルゴリズム，最小費用流問題

# A Heuristic Method for Just-In-Time Scheduling Problem with Periodic Time Slots

Eishi Chiba and Kunihiko Hiraishi
School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi-machi, Nomi-gun, Ishikawa-ken, 923-1292, Japan
email:{e-chiba, hira}@jaist.ac.jp

**Abstract:** Just-in-time scheduling problem is the problem of finding an optimal scheduling such that each job processed by parallel identical machines finishes exactly at its due time. We study the problem under a realistic assumption called periodic time slots. In this paper, we prove that this problem cannot be approximated, assuming $P \neq NP$. Next, we present a heuristic algorithm, assuming that the number of machines is only one. The key idea is to reduce the problem to a network flow problem. The heuristic algorithm is fast because it includes simple updations of some values of a flow over and above the minimum cost flow that dominates the total time. Next, we show some simulation results. Our heuristic algorithm returns an optimal solution under a constraint. For other cases, we give an approximation bound of our heuristic solution to the oprimal one.

**Keywords:** Scheduling, Just-in-time, Set-up times, Heuristic algorithm, Minimum cost flow

# 1　はじめに

これまで多くの研究者がスケジューリング問題に取り組み，その成果として優れたテキストも世に多数出版されている．それにも関わらず，現在でも活発に研究されており，その勢いはますます強まってさえいる印象を受ける．その理由は，近年製造業などからの要請が以前よりも高まってきていることである．また，アルゴリズム設計という立場からも，必ずしも最適な解を見出すことを諦めて，経験的に良い解を求めるというヒューリスティック解法が注目を浴びている．本稿で取り上げるスケジューリング問題は *NP*-困難のクラスに属し，さらに $P \neq NP$ の仮定の下で，近似不可能である．従って，そのような問題に対してヒューリスティックアルゴリズムを設計し，実際的に役に立つかどうかを計算機実験で評価することが大切である．

本稿で扱うスケジューリング問題は，予めジョブごとにプロセス終了時刻が与えられる．その理由は，例えば自動車の販売において，工場での製造終了から出荷までの時間があると，その分だけ余計にストック代がかかるからである．このように予めジョブの終了時刻が与えられているスケジューリング問題を，ジャスインタイムスケジューリング問題と呼ぶ．また，製造業などでは，例えば月単位に商品の発送が決まっている場合がしばしば見られる．このように周期的な時間単位も考慮に入れる場合は，周期的なタイムスロット付きジャストインタイムスケジューリング問題と呼ぶ．自動車製造工程の決定など，他にも多くの現実問題を周期的なタイムスロット付きジャストインタイムスケジューリング問題として考えることが出来る．

あるクラスのジャストインタイムスケジューリング問題の解法が [1] で示された．そこでは，それぞれのジョブごとに価値を付けて，全体としての価値が最大となるように処理するジョブを選択している．選択されなかったジョブについては，どうするかを考慮していない．それに対して，本研究では周期的タイムスロットという現実的な仮定を設け，選択されなかったジョブも処理する方法を提案する．

以下，第 2 章では対象となる問題の定義を与えると共に，具体的な例題を示す．第 3 章では多項式時間近似スキーム（PTAS）の非存在性に関する定理を示す．第 4 章では我々が提案するヒューリスティックアルゴリズムを記述する．第 5 章ではそのアルゴリズムの実際的性能を計算機実験により評価する．第 6 章では本問題のいくつかの性質を近似比の観点から述べてみる．第 7 章では結論と今後の課題を述べる．

# 2　問題の定義

本稿で扱う問題を定義するために以下の変数を準備する．ただし，正整数 $k$ に対して，集合 $I_k = \{1, 2, \cdots, k\}$ とする．

- $M_i$ $(i \in I_m)$: $m$ 個の同種機械
- $J_j$ $(j \in I_n)$: $n$ 個のジョブ
- $p_j > 0$ $(j \in I_n)$: 各ジョブの処理時間
- $d_j \geqq p_j$ $(j \in I_n)$: 各ジョブの納期
- $s_{jk} \geqq 0$ $(j \neq k \wedge j, k \in I_n)$: $J_j$ と $J_k$ 間のセットアップタイム
- $L \geqq \max_{j \in I_n}\{d_j\}$: タイムスロット長

タイムスロットは周期的であり，それぞれのタイムスロット中にジョブの納期が与えられる．すなわち，ジョブ $J_j$ の納期は $d_j, L + d_j, 2L + d_j, \cdots$ であり，これらの納期中のどれか一つの時点で $J_j$ の処理が完了する．また，ジョブ間には先行関係が無く，並列に処理される．そして，各機械で要するタイムスロット数の最大値を最小にするスケジュールを求める問題を考えよう．

ジョブ $J_j$ のスケジュールとは写像 $S : J_j \mapsto (M^S_{[j]}, C^S_j)$ である．ただし，$M^S_{[j]}$ はジョブ $J_j$ が処理される機械であり，$C^S_j$ はジョブ $J_j$ の処理が終了する時刻である．そして，以下の 2 つの条件を満たす時，スケジュール $S$ は実行可能であると言う．

(i) 各ジョブ $J_j$ に対して，制約式 $C^S_j = r^S_j \cdot L + d_j$ を満たす非負の整数 $r^S_j$ が存在する．

(ii) $M^S_{[j]} = M^S_{[k]} (j \neq k \wedge j, k \in I_n)$ であり，$J_j$ と $J_k$ が連続して処理されるならば，次の 2 つの制約式 $C^S_k \geqq C^S_j + s_{jk} + p_k, C^S_j \geqq C^S_k + s_{kj} + p_j$ のどちらか一方が成り立つ．

また，$r(S) := \max_{j \in I_n}\{r^S_j\}$ とすると，周期的なタイムスロット付きジャストインタイムスケジューリング問題を次のように問題を記述できる．

**入力:** ジョブ数 $n$，機械数 $m$，処理時間 $p_j$，納期 $d_j$，セットアップタイム $s_{jk}$，タイムスロット長 $L$

**目的関数:** $r(S)$ →最小

**制約条件:** スケジュール $S$ は実行可能である

以下，前後関係から明らかな時は，単に問題と省略して呼ぶことにする．全ての実行可能なスケジュールの中で，$r(S)$ が最小になるスケジュールを最適解と呼び，その時の $r(S)$ を最適コストと呼ぶ．必要なタイムスロットの個数は $r(S)+1$ である．$m \geqq n$ の時は，明らかにどの機械も二つ以上のジョブを処理しないスケジュールが存在するので，簡単に解ける．そのため，以下では $m < n$ と仮定する．また，上で各納期 $d_j \geqq p_j$ と仮定したのは，全てのジョブは一つのタイムスロット内で処理されることに対応する．

ここで，異なる 2 つのジョブ $J_j, J_k$ に対して，制約式 $(g_{jk}-1)\cdot L + d_k < d_j + s_{jk} + p_k \leqq g_{jk}\cdot L + d_k$ を満たす非負整数 $g_{jk}$ を導入する．この制約式は，$J_j$ を処理した後，$g_{jk}$ タイムスロット先で $J_k$ の処理を完了することを意味する．問題の入力が与えられた時，任意の $j,k(j \neq k \wedge j,k \in I_n)$ に対して $g_{jk}$ が一意に定まる．ここで，$g := \max_{j\neq k \wedge j,k\in I_n}\{g_{jk}\}$ とする．先行研究 [2] から，以下の結果が知られている．

- 問題は $m = 1$ の時でさえ，*NP*-困難である．
- $g \leqq 1$ ならば，問題は多項式時間で解ける．

**例題:** 次の入力を考えよう．ジョブ数 $n = 6$，機械数 $m = 1$，セットアップタイム $s_{jk} = 0(j \neq q \wedge j,k \in I_6)$，タイムスロット長 $L = 17$.

| $j$ | $p_j$ | $d_j$ |
|---|---|---|
| 1 | 5 | 7 |
| 2 | 2 | 10 |
| 3 | 5 | 16 |
| 4 | 2 | 3 |
| 5 | 1 | 5 |
| 6 | 7 | 13 |

最初のタイムスロットから順次，[1] の方法を用いて可能な限り多くのジョブを処理していくというグリーディな手法を考えよう．この場合，図 1 に示す様に，全体のスケジュールはタイムスロットを 3 つ必要とする．しかし，図 2 に示す様に，明らかにより少ないタイムスロット数（= 2 個）で全体の処理を完了するスケジュールが存在する．

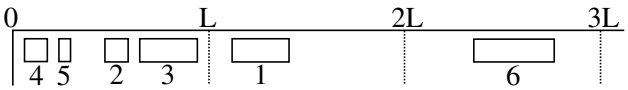


図 1: グリーディな手法によるスケジュール

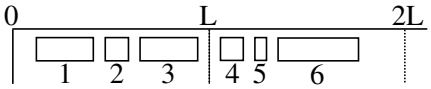


図 2: 最適なスケジュール

## 3 PTAS の非存在性

**定理 1.** 多項式時間で計算可能な任意の関数 $\alpha(n)$ に対して，P = NP でない限り，タイムスロット付きジャストインタイムスケジューリング問題を $\alpha(n)$ 以内の近似率で近似することは不可能である．

**証明**: 背理法を用いる．タイムスロット付きジャストインタイムスケジューリング問題に対して，近似率 $\alpha(n)$ の多項式時間近似アルゴリズム $\mathcal{A}$ が存在したとする．すると，$\mathcal{A}$ が *NP*-困難なハミルトンパスの存在判定問題を多項式時間で解くのに使えてしまい，*P=NP* となってしまうことを以下に示す．

まず，ハミルトンパスの存在判定問題からタイムスロット付きジャストインタイムスケジューリング問題への還元を説明する．ハミルトンパスの存在判定問題は次の様に定義される．

**入力:** 有向グラフ $G = (V, E)$．ただし，$V = \{v_1, \cdots, v_n\}$ とする．

**出力:** ハミルトンパス (各頂点をちょうど一度通る有向パス) が存在するか？

これに対して，タイムスロット付きジャストインタイムスケジューリング問題の入力を以下の様に構成する．

**入力:** ジョブ数 $n$，機械数 $= 1$，タイムスロット長 $= 1$，各納期 $d_j = 1$，各処理時間 $p_j = 1$，
$(v_i, v_j) \in E$ ならば，$s_{ij} = 0$，そうでないなら，$s_{ij} = \alpha(n)\cdot n$

すると，以下の性質を満たす．

- $G$ がハミルトンパスを持つならば，タイムスロット付きジャストインタイムスケジューリング問題の最適解のタイムスロット数は $n$ であり，

- $G$ がハミルトンパスを持たないならば，タイムスロット付きジャストインタイムスケジューリング問題の最適解のタイムスロット数は $\alpha(n)\cdot n$ より大きい．

タイムスロット付きジャストインタイムスケジューリング問題に対して，近似率 $\alpha(n)$ のアルゴリズム $\mathcal{A}$ を走らせると，最初のケースに当てはまるときはタイムスロット数が $\alpha(n)\cdot n$ 以下の解を返し，第 2 のケースに当てはまる時はスロット数が $\alpha(n)\cdot n$ より大きい解を返す．したがって，$\mathcal{A}$ は $G$ がハミルトンパスを持つかどうか判定するのに使える． □

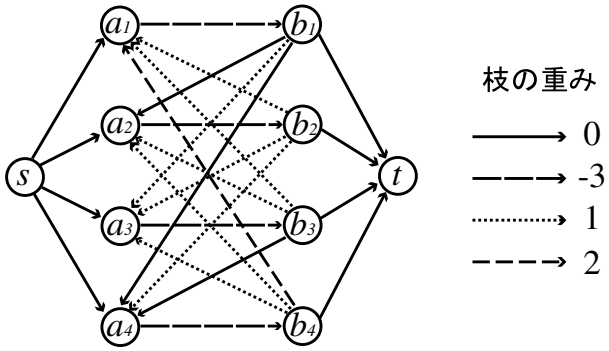


図 3: グラフ $G$ の例

## 4 ヒューリスティックアルゴリズム

本章では，ネットワークフローを利用したヒューリスティックアルゴリズムを記述する．問題の入力が与えられた時，以下に示すような単純な重み付き連結有向グラフ $G=(V,A)$ を構成する．ここで，キャパシティ関数を $c:A\to\mathbb{N}$，重み関数を $w:A\to\mathbb{Z}$ とする．

- $V=\{s,t,a_i,b_i \mid i\in I_n\}$ であり，$2n+2$ 個の節点から成る．
- $A$ は以下に示す $n^2+2n$ 個の枝から成る．
  - $w(s,a_j)=0$ を付した $(s,a_j)$ $(j\in I_n)$
  - $w(a_j,b_j)=-w_{\mathrm{job}}$ を付した $(a_j,b_j)$ $(j\in I_n)$
  - $w(b_j,t)=0$ を付した $(b_j,t)$ $(j\in I_n)$
  - $w(b_j,a_k)=g_{jk}$ を付した $(b_j,a_k)$ $(j\neq k\wedge j,k\in I_n)$

ただし，節点 $s$ の入次数は 0 で，ソースと呼ばれる．節点 $t$ の出次数は 0 で，シンクと呼ばれる．また，$w_{\mathrm{job}}$ は，制約式 $g<w_{\mathrm{job}}$ を満たす任意の正整数である．点 $a_j,b_j$ $(j\in I_n)$ はジョブ $J_j$ に対応する．

[$G$ **の構成例**] ジョブ数 $n=4$，機械数 $m=1$，セットアップタイム $s_{jk}=1$ $(j\neq k\wedge j,k\in I_4)$，タイムスロット長 $L=8$.

| $j$ | $p_j$ | $d_j$ |
|---|---|---|
| 1 | 2 | 2 |
| 2 | 2 | 6 |
| 3 | 3 | 4 |
| 4 | 2 | 8 |

上に示した様な入力が与えられた時，10 個の節点集合と 24 個の枝集合の対から成るグラフ $G$ を構成する．また，それぞれ $g_{12}=0,g_{13}=1,g_{14}=0,g_{21}=1,g_{23}=1,g_{24}=1,g_{31}=1,g_{32}=1,g_{34}=0,g_{41}=2,g_{42}=1,g_{43}=1$ であることを考慮して，構成される $G$ を図 3 に示す．

さて，各枝 $(u,v)\in A$ の容量 $c(u,v)$ を 1 とし，単位流量当りの費用 $w(u,v)$ が与えられているときに，1 点 $s$ から他の 1 点 $t$ への流量 $m$ のフロー $f:A\to[0,1]$ の中で総費用を最小にするもの（$s$ から $t$ への流量 $m$ の最小費用流）を求める問題を考える．数式で表せば，次の様になる．

『条件

$$\sum_w f(v,w)-\sum_u f(u,v)=\begin{cases} m & (v=s),\\ 0 & (v\in V\setminus\{s,t\}),\\ -m & (v=t),\end{cases}$$

$$f(u,v)\in[0,1]\text{ for any }(u,v)\in A$$

の下で

$$w_f=\sum_{(u,v)\in A} w(u,v)f(u,v)$$

を最小にせよ.』

構成したグラフ $G$ に対して，最小費用流を求める．この最小費用流は，各枝に対して，0 か 1 のどちらか一方の値を決めている．全ての枝 $(a_j,b_j)$ にはフロー値 1 が対応するが，その理由は，枝の重み $-w_{\mathrm{job}}$ が制約式 $g<w_{\mathrm{job}}$ を満たしているからである．ここで，フロー値が 1 の枝をグラフに残し，フロー値が 0 の枝をグラフから削除したグラフを $G'$ とおく．$G'$ には $m$ 個のソースからシンクへのパスが存在することになる．もし，$G$ の全ての枝コストが非負であるならば，$G'$ には $m$ 個のソースからシンクへのパス以外に，パスは存在しない．しかし，$G$ には枝コストに負のものがあるので，$G'$ には $m$ 個のソー

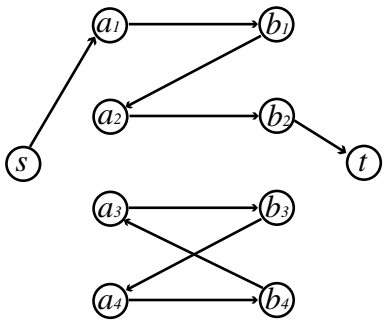


図 4: グラフ $G'$ の例

スからシンクへのパス以外に，いくつかのサイクルが存在することもあり得る．このようなサイクルが生じる例を図 4 に示す．この例は，図 3 のグラフ $G$ に対して最小費用流を求め，グラフ $G'$ を表したものである．もし，$G'$ においてサイクルが生じなければ，次の様にして $G'$ を実行可能解へ対応させる．$G'$ においてソースからシンクへのパスの個数は流量 $m$ に等しいが，これらの $m$ 個のパスをフロー $f_1, f_2, \cdots, f_m$ で表す．すなわち，フロー $f_i$ は，対応するパス上の枝に対して値 1 を与え，そうでない枝に対して値 0 を与える．すると，各フロー $f_i$ を 1 台の機械へ対応させることが出来る．$\{f_1, f_2, \cdots, f_m\}$ から $\{M_1, M_2, \cdots, M_m\}$ への写像は全部で $m!$ 個あるが，機械は同種なので，どの写像を選んでもよい．また，フロー $f_i$ に対応するパス $\pi$ は，正の重みを持つ全ての枝を取り除くことによって，複数のパス $\pi_1, \pi_2, \cdots, \pi_l$ に分解できる．各パス $\pi_i$ $(i = 1, 2, \cdots, l)$ に以下の様にタイムスロットを割り当てる．

(i) $\pi_1$ に最初のタイムスロットを割り当てる．

(ii) $\pi_j$ に $k$ 番目のタイムスロットを割り当てるとする．この時，$\pi_j$ と $\pi_{j+1}$ を接続する枝の重みが $k'$ であるならば，$\pi_{j+1}$ に $(k + k')$ 番目のタイムスロットを割り当てる．

以上が $G'$ においてサイクルが生じない場合の実行可能解への対応の取り方である．また，その時の「延べ使用タイムスロット数」は

$$m + \sum_{(u,v) \in A \ \wedge \ w(u,v) > 0} w(u,v) f(u,v)$$

である．最小費用流問題の最適解は上の「延べ使用タイムスロット数」を最小化するが,「各機械で要するタイムスロット数の最大値」を最小化するわけではない．ただし，機械数 $m = 1$ であれば両者は等しくなるので，最小費用流問題の最適解が対応するスケジュールは本問題の最適解でもある．ここで問題となるのは $G'$ にサイクルが存在するときである．この時，$G'$ は実行可能なスケジュールに対応していない．

以下，機械数 $m = 1$ の場合に限定して議論する．$G'$ にサイクルが存在する時でも，次の様にして実行可能解を得る．

**Heuristic Algorithm:**

1. グラフ $G$ を構成する．
2. $G$ に対して，最小費用流を求め，$G'$ にサイクルが存在しなければ，終了．
3. $G'$ のサイクル上の全ての枝 $(b_k, a_j)$ に対して，$\min\{w(b_k, a_i) - w(b_k, a_j), w(b_l, a_j) - w(b_k, a_j)\}$ が最小である枝 $(b_k, a_j)$ を求める．ただし，ソースからシンクへのパス上で，$a_i$ はソースの次の節点であり，$b_l$ はシンクの直前の節点である．
4. ステップ 3 で求めた枝 $(b_k, a_j)$ に対して，
   IF $w(b_k, a_i) - w(b_k, a_j) < w(b_l, a_j) - w(b_k, a_j)$
   $G'$ において，枝 $(b_k, a_j)$，$(s, a_i)$ を削除，枝 $(b_k, a_i)$，$(s, a_j)$ を付け加える．
   ELSE
   $G'$ において，枝 $(b_l, t)$，$(b_k, a_j)$ を削除，枝 $(b_l, a_j)$，$(b_k, t)$ を付け加える．
5. $G'$ にサイクルが存在しなければ，終了．あれば，ステップ 3 へ．

ステップ 4 は枝の付け替え処理であり，その様子を図 5 に示す．図 5(a) は枝の付け替え前に対応している．図 5(b) は，ステップ 4 の IF 文で条件式が成り立つ時の枝の付け替えに対応している．図 5(c) は，ステップ 4 の IF 文で条件式が成り立たない時の枝の付け替えに対応している．このヒューリスティックアルゴリズムにより，最終的に $G'$ にはサイクルが存在しないので，実行可能解へ対応させることが出来る．また，その時のタイムスロット数は

$$1 + \sum_{(u,v) \in A \ \wedge \ w(u,v) > 0} w(u,v) f(u,v)$$

である．

**補題 1.** 入力のジョブ数が 2 の時，ヒューリスティックアルゴリズムは最適解を返す．

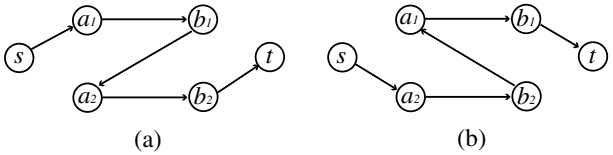


図 6: ジョブ数が 2 の場合の $G'$

**証明**: ジョブ数が 2 の時，ステップ 2 で得られる $G'$ は図 6(a)(b) に示した 2 つのうちどちらか一方しかあり得ない．よって，常にサイクルが存在しないので，最適解を得る． □

# 5 計算機実験

実験環境を以下に示す.

**Machine:** Dell Precision 650

**CPU:** Intel Xeon 3.06GHz × 2

**OS:** Microsoft Windows 2003 Server

**Memory:** 4GB

**Compiler:** Microsoft Visual C++ 6.0

前章で提案したヒューリスティックアルゴリズムの性能を評価するために，アルゴリズムを実装して上に示した環境で計算機実験を行った．実装の際，C++のクラスライブラリである LEDA を用いてプログラムを作成した．特に最小費用流の計算は，LEDA で用意されている関数 MIN_COST_FLOW() を利用した．また，機械数 $m = 1$，タイムスロット長 $L = 20$ の下で，以下に示すランダムな入力データを対象とした．

- $0 <$ 各納期 $d_j \leqq L$
- $0 <$ 各処理時間 $p_j \leqq d_j$
- $0 \leqq$ 各セットアップタイム $s_{jk} \leqq L$

ヒューリスティックアルゴリズムのステップ 2 で $G'$ にサイクルが存在しなければ，そこで終了して最適解を得ることになるが，100 回の試行を行いその割合を調べた．実験結果を表 1 に示す．各行はそれぞれ入力されるジョブの個数，及びヒューリスティックアルゴリズムのステップ 2 で終了する割合を表している．表 1 から分かるように，ジョブの個数が比較的少ない時は最適解を得る確率が高いが，逆にジョブの個数が 320 では，100%の確率でサイクルを生じる．

| ジョブ数 | 最大値 | 平均値 | 分散 | 理論的な最大値 |
|---|---|---|---|---|
| 100 | 8 | 3.56 | 2.4264 | 49 |
| 200 | 10 | 4.08 | 3.1136 | 99 |
| 400 | 12 | 4.57 | 4.8051 | 199 |
| 800 | 12 | 5.14 | 4.3804 | 399 |
| 1600 | 11 | 5.8 | 4.46 | 799 |

表 2: Heuristic Algorithm ステップ 2 におけるサイクルの個数

| ジョブ数 | 近似比の最悪値 | 近似比の平均 | 分散 |
|---|---|---|---|
| 3 | 1 | 1 | 0 |
| 4 | 1.5 | 1.00978 | 0.00300402 |
| 5 | 1.33333 | 1.01522 | 0.00366418 |
| 6 | 1.5 | 1.01932 | 0.00393428 |
| 7 | 1.4 | 1.0224 | 0.00393347 |
| 8 | 1.4 | 1.02461 | 0.00370951 |
| 9 | 1.4 | 1.0237 | 0.00326279 |
| 10 | 1.4 | 1.02431 | 0.00301326 |

表 3: 計算機実験による Heuristic Algorithm の近似比

それでは，どのくらいの個数のサイクルが生じているのであろうか．そのことを調べるために，100 回の試行を行った結果を表 2 に示す．各列はそれぞれ入力されるジョブの個数，得られたサイクル数の最大値，サイクル数の平均，分散，そして理論的なサイクル数の最大値を表している．ここで，理論的なサイクル数の最大値は，ジョブ数 $n$ に対して，$\lfloor \frac{n-1}{2} \rfloor$ である．表 2 から分かるように，理論的な最大値と実験による最大値に大きな開きを確認した．また，平均値はジョブの個数が増えるに従ってわずかに増加するが，理論的な最大値の増加量に比べると少ない．これらのことから，ジョブ数がサイクル数に与える影響はほとんど無視できるであろう．

次に，ステップ 2 でサイクルを生じた場合，ヒューリスティックアルゴリズムから求まるタイムスロット数と全探索アルゴリズムから求めた最適なタイムスロット数との近似比を調べた．10000 回の試行を行った結果を表 3 に示す．各列はそれぞれ入力されるジョブの個数，得られた近似比の最悪値，近似比の平均，そして分散を表している．表 3 から分かるように近似比の最悪値と比べて，平均値の方はほぼ 1 であり良好な結果となった．分散についてもかなり小さな値となった．

最後に，ヒューリスティックアルゴリズムの計算時間を LEDA で用意された関数 used_time() を用いて計測した結果を図 7 に示す．ジョブ数が $100, 200, 300, \cdots, 2000$ のそれぞれに対して 100 回の試行を行い平均を取った．図

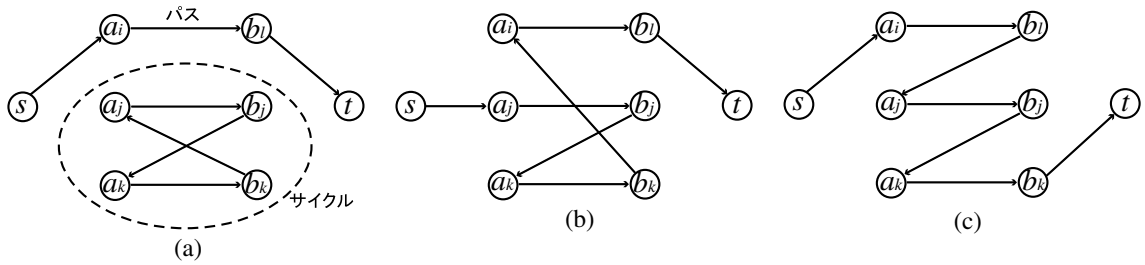


図 5: 枝の付け替え

| ジョブ数 | 5 | 10 | 20 | 40 | 80 | 160 | 320 |
|---|---|---|---|---|---|---|---|
| ステップ 2 で終了する割合 (%) | 35 | 21 | 5 | 4 | 5 | 2 | 0 |

表 1: Heuristic Algorithm がステップ 2 で終了する割合

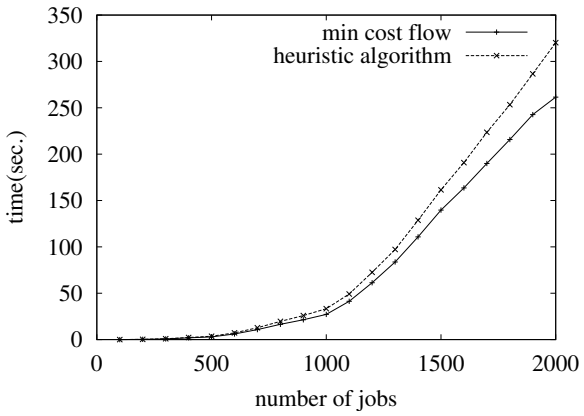


図 7: Heuristic Algorithm の計算時間

7 で実線は最小費用流を求める（ヒューリスティックアルゴリズムのステップ 2) のに要する計算時間で，点線は枝の付け替え処理も含めた（ヒューリスティックアルゴリズムのステップ 2,3,4,5）計算時間を示している．図 7 から分かるように，計算時間は最小費用流を求めることが支配的であり，高速である．

# 6 近似比

本章では，近似比の観点から述べる．そのために，まず以下の二つの変数を導入する．

$$\delta_{k*} := \max_{k \neq j \ \wedge \ j \in I_n} w(b_k, a_j) - \min_{k \neq j \ \wedge \ j \in I_n} w(b_k, a_j)$$

$$\delta_{*j} := \max_{j \neq k \ \wedge \ k \in I_n} w(b_k, a_j) - \min_{j \neq k \ \wedge \ k \in I_n} w(b_k, a_j)$$

**補題 2.** $\delta := \max_{k \neq j \ \wedge \ j,k \in I_n} \min\{\delta_{k*}, \delta_{*j}\}$，$r^*$ が最適スケジュールに必要なタイムスロット数であるとすると，ヒューリスティックアルゴリズムで求められるスケジュールのタイムスロット数は高々

$$r^* + \delta \cdot \left\lfloor \frac{n-1}{2} \right\rfloor$$

である．

**証明**. 入力されるジョブの個数を $n$ とする．ヒューリスティックアルゴリズムのステップ 2 で求められる $G'$ において，サイクルの個数は高々$\lfloor \frac{n-1}{2} \rfloor$ であり，ヒューリスティックアルゴリズムのステップ $3,4,5$ のループ回数に等しい．また，枝の付け替えに伴うコスト増加量は高々 $\delta$ であることを考慮して主張が導かれる． □

次にある仮定の下で近似比を導出する．

**定理 2.** ヒューリスティックアルゴリズムのステップ 1 で得られるグラフ $G$ について，$w(b_j, a_k) \in \{1, 2\}$（ただし，$j \neq k \wedge j, k \in I_n$）である時，ヒューリスティックアルゴリズムは近似比 1.5 を保証する．

**証明**: $r^*$ を最適スケジュールに必要なタイムスロット数とする．また，$r_f^*$ を最小費用流から求まるタイムスロット数とする．この時，$r_f^* \leqq r^*$ が成り立つ．また，入力となるジョブの個数を $n$ とすると，$w(b_j, a_k) \in \{1, 2\}$ より，$n$ は $r^*$ の下界である．すなわち，$n \leqq r^*$ が成り立つ．以上から，$r$ をヒューリスティックアルゴリズムから求められるタイムスロット数とすると，

$$r \leqq r_f^* + \left\lfloor \frac{n-1}{2} \right\rfloor \leqq r^* + \frac{n-1}{2}$$

が成り立つので，以下の関係式が導かれる．

$$\frac{r}{r^*} \leqq 1 + \frac{(n-1)/2}{r^*} \leqq 1 + \frac{(n-1)/2}{n} < 1.5$$

よって，近似比 1.5 が導かれた． □

次に，一般の場合の近似比に関して議論する．そのために，次の変数

$$w_j^* = \min_{k \in I_n} \{w(b_j, a_k)\},$$
$$w^* = \max_{j \in I_n} \{w_j^*\},$$
$$c = \frac{(\sum_{j=1}^{n} w_j^*) - w^*}{n-1}$$

を導入する．すると，$c(n-1)+1$ は，$r_f^*$ の下界である．よって，以下の関係が成り立つ．

$$\begin{aligned} \frac{r}{r^*} &\leqq 1 + \frac{(n-1)(\Delta-1)/2}{r^*} \\ &\leqq 1 + \frac{(n-1)(\Delta-1)/2}{c(n-1)+1} \\ &\leqq 1 + \frac{(n-1)(\Delta-1)/2}{c(n-1)} \\ &= 1 + \frac{\Delta-1}{2c} \end{aligned} \tag{1}$$

ただし，$\Delta := \max_{j \neq k \wedge j,k \in I_n} \{w(b_j, a_k)\}$ である．ヒューリスティックアルゴリズムのステップ 4 で，フローの更新に伴うコスト増加量が高々$\Delta - 1$ であることは，以下の事実から導かれる．

- ステップ 3 でコスト 0 の枝を選択すると，コスト増加量は高々$\Delta$ である．
- どのサイクルもコストが正の枝が存在するので，その枝をステップ 3 で選択すると，増加量は $\Delta$ よりも小さい．
- ステップ 3 での枝の選択のやり方より，フローの変更に伴うコスト増加量は高々$\Delta - 1$ である．

以上の議論から次の様な系を得る．これは，$\Delta = 1$ の時は，式 (1) より近似比が 1 になることから導かれる．

**系 1.** $w(b_k, a_j) \in \{0, 1\}$ ならば，ヒューリスティックアルゴリズムは最適解を返す．

## 7 結論と今後の課題

周期的なタイムスロットという現実的な仮定を設けたジャストインタイムスケジューリング問題について，機械数が 1 である場合の解法を示した．今後は $m$ 機械の場合への拡張が課題となる．今回はセットアップタイムが単純に非負であるという場合を取り扱ったが，三角不等式を満たす様なセットアップタイムを仮定（具体的には，それぞれの $i$ に対して $s_{jk} \leqq s_{ji} + s_{ik}$ を満たす）すると，3 章に記述した証明手法が通用しない．また，今回はジョブは一つのタイムスロット内で処理される場合を取り扱ったが，タイムスロットをまたいで処理することを許した場合は議論が少し異なってくる．上述の様に問題の定義を変更した場合の解法を今後検討すると同時に，計算機実験によりアルゴリズムの有用性を示していきたい．

## 参考文献


[1] K. Hiraishi, E. Levner, and M. Vlach, *Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs*, Computers & Operations Research, 29, (2002), pp.841-848.

[2] O. Čepek and S. C. Sung, *Just in time scheduling with periodic time slots*, Proc. 5th Czech-Japan Seminar on Data Analysis and Decision Maling under Uncertainty, (2001), pp.27-29.