アルゴリズム 88－3
（2003．1．20）

オンラインスケジューリングアルゴリズムの実験的性能評価

上ヶ原 誠　　浅野 孝夫
中央大学大学院 理工学研究科 情報工学専攻

概要

スケジューリング問題とは，仕事の機械への割り当てと機械上での処理順序を決定する問題で，コンピュータにおける処理要求のコントロールやプロジェクトの遂行手順の決定など実用上しばしば起こる問題である．ここでは特に，仕事にリリース時刻が付加されており，重みつき完了時刻和の最小化を目的とする単一機械スケジューリング問題を扱う．オンライン環境では，仕事は時間が経過するに従って，それぞれの仕事のリリース時刻に到着する．後に到着する仕事に関する情報は，その仕事が到着するまでわからない．本稿ではこの問題に対して最近提案されたアルゴリズムの概略を示し，コンピュータ上に実装してその実験的性能評価を行う．

Experimental Evaluation of Algorithms for the On-Line Scheduling Problem

Makoto Kamigahara　　Takao Asano

Information and Systems Engineering Course,
Graduate School of Science and Engineering, Chuo University

Abstract

Scheduling is to find a schedule that specifies when and on which machine each job is to be excuted. This problem arises in a variety of settings, for example, to control jobs on the central processing unit of a computer, to decide a plan in what order should tasks be processed, and so on. In this paper, we consider the single machine scheduling problem with release dates in which objective is to minimize a weighted sum of completion times. In an on-line scheduling environment, jobs arrive over time, as defined by their release dates. No information is known about any future jobs. We illustrate several on-line algorithms that were recently proposed for this problem, and try to experimentally evaluate these algorithms.

1　はじめに

本稿では以下のようなスケジューリング問題を扱う．1 台の機械上で処理しなければならない n 個の仕事が与えられる．この仕事の集合を J と書く．仕事にはいくつかの属性が付加されている．仕事 $j \in J$ の処理に必要な時間を p_j，仕事の重要度を表す重みを $w_j > 0$ とする．仕事 j はリリース時刻 $r_j \geq 0$ 以降でなければ処理を開始できない．また，仕事は分割不可能である．すなわち，仕事の処理を途中で中断し，後にその続きから再開することは許されておらず，一度処理を開始した仕事は最後まで連続して処理しなければならない．スケジューリングを評価するための目的関数は重みつき総完了時刻 $\sum_{j=1}^{n} w_j C_j$ であり，この値を最小にするスケジュールを求めることを目指す．ここで C_j は，構成されたスケジュールにおける仕事 j の完了時刻である．この問題は，Graham らによるスケジューリングの表記法 [4] によれば，$1|r_j|\sum w_j C_j$ と書かれる．また，すべての仕事 j に対して $w_j = 1$ であったとしても，強 NP 困難である．

オンラインスケジューリング環境では，仕事は時間が経過するに従って，それぞれの仕事のリリース時刻に到着する．最初仕事の数はわからず，後に到着する仕事に関する情報は何も与えられていない．仕事 j に関する処理時間 p_j，重み w_j は時刻 r_j に仕事が到着したときはじめて明らかになる．

任意のインスタンスに対して，オンラインアルゴリズムによって得られる目的関数値が (オフライン環境における) 最適な目的関数値の ρ 倍以下であるとき，ρ を競合比という．

オンライン環境において，決定性アルゴリズムとしては競合比 2 [7, 1]，ランダム化アルゴリズムとしては競合比 $\mathrm{e}/(\mathrm{e}-1) \approx 1.582$ [10] よりもよいアルゴリズムが存在しないことがわかっている．

オフラインの問題に対する最初の定数近似アルゴリズムは 1995 年に Phillips, Stein, Wein [8] によって与えられた．このアルゴリズムは，まず仕事の処理の中断を許したスケジュールを構成し，そのスケジュールから得られる情報を用いて，仕事の処理を中断しないスケジュールを構成する．オンラインの問題に対する最初の結果は 1996 年に Hall, Shmoys, Wein [5] によって与えられた．彼らのアルゴリズムは時間軸を区間に分割してそれぞれの区間内でスケジュールを構成していくというものであり，競合比は $4+\epsilon$ である．1997 年，Hall, Schulz, Shmoys, Wein [6] はこの方法に改善を加え，競合比を $3+\epsilon$ とした．同じ年に，Goemans [2] は Phillips らの考えを発展させた α-点の概念に基づき，α の値を $1/\sqrt{2}$ と決めることで，競合比 $1+\sqrt{2} \approx 2.4143$ のアルゴリズムを与えた．また，α の値を一様分布に基づいてランダムに選ぶことにより，競合比 2 のランダム化アルゴリズムが得られることも示した．2002 年，Goemans, Queyranne, Schulz, Skutella, Wang [3] は α の値をある確率分布に基づいてランダムに選ぶことにより競合比 1.6853 を達成した．そして同じく 2002 年，Anderson と Potts [1] は，SWPT (Shortest Weighted Processing Time) 法をオンラインに適応させることにより，競合比 2 の決定性アルゴリズムを発表した．これらの結果を表 1 にまとめる．

本稿では特に，1997 年の Hall らによるアルゴリズム，2002 年の Goemans らによるアルゴリズム，同じく 2002 年の Anderson と Potts によるアルゴリズムについて，計算機実検を行い，実際的な性能を評価する．また，実験結果を観察することにより，近似保証はもたないながらも実用上優れたスケジュールを構成するヒューリスティックを与える．

表 1: 主なオンラインアルゴリズムの競合比

著者	ランダム化	決定性
Hall ら [5]		$4+\epsilon$
Hall ら [6]		$3+\epsilon$
Goemans [2]	2	2.4143
Goemans ら [3]	1.6853	
Anderson ら [1]		2

2 アルゴリズムの概略

この節では，Hall, Schulz, Shmoys, Wein によるアルゴリズム [6]，Goemans, Queyranne, Schulz, Skutella, Wang によるアルゴリズム [3]，Anderson と Potts によるアルゴリズム [1] について，その概略を述べる．

準備として，この節で紹介するアルゴリズムに関して，共通して土台となっている概念について述べる．すべての仕事が同時にリリースされる問題 (Graham らの表記法によれば $1||\sum w_jC_j$ と書かれる問題) は，SWPT 法によって最適に解かれることが 1956 年に Smith によって示されている [9]．SWPT 法では，機械が空になったとき，まだスケジュールされていない仕事の中で w_j/p_j が最大の仕事をスケジュールしていく．

2.1 Hall らのアルゴリズム [6]

Hall らは時間分割の概念を用いたアルゴリズムを提案した．これは，時間軸を長さが幾何的に増加していくような区間に分割し，それぞれの区間の中で，その区間の開始時刻までにリリースされている未処理の仕事をスケジュールするというものである．一連の副問題を解くことによって全体の問題が解かれる．また，それぞれの区間の中ではリリース時刻をもたない仕事をスケジュールすることになる．このアルゴリズムを Greedy-Interval と呼ぶ．このアルゴリズムの競合比は $3+\epsilon$ である．

$\tau_0 = 0$，$\tau_l = 2^{l-1} (l = 1, 2, \ldots)$ としたとき，それぞれの区間は $(\tau_l, \tau_{l+1}]$ と定義される．区間 $(\tau_l, \tau_{l+1}]$ の長さは τ_l であることに注意する ($(\tau_0, \tau_1]$ に限り長さは $\tau_1 (\neq \tau_0)$ である)．

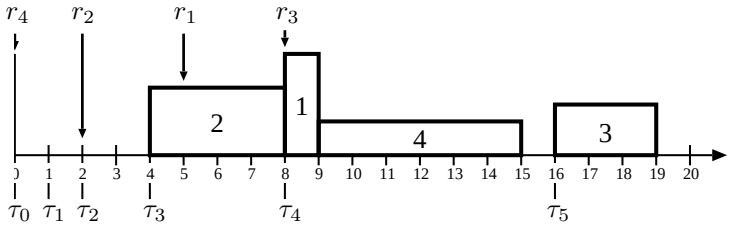

図 1: アルゴリズム Greedy-Interval の出力例

アルゴリズム Greedy-Interval

反復 $l = 1, 2, \ldots$ について，以下を繰り返すことにより，それぞれの区間 $(\tau_l, \tau_{l+1}]$ におけるスケジュールを構成する．

1° 時刻 τ_l まで待つ．

2° J_l を，時刻 τ_l までにリリースされており，かつその時点でまだスケジュールされていない仕事の集合とする．

3° 集合 J_l に含まれる仕事と与えられた $\epsilon > 0$ に対し，ナップサックのサイズを τ_l，品物 j のサイズを p_j，価値を w_j として，ナップサック問題に対する FPTAS を適用する．得られた $(1+\epsilon)$ 近似解に含まれる仕事の集合を J'_l とする．

4° J'_l に含まれる仕事を，w_j/p_j の非増加順に区間 $(\tau_l, \tau_{l+1}]$ の間にスケジュールする．

アルゴリズム Greedy-Interval は，副問題としてナップサック問題を含んでいる．ナップサック問題に対する FPTAS (fully polynomial time approximation scheme) の概略は以下のとおりである．ナップサックのサイズを D，品物の数を n としたとき，与えられた誤差パラメータ $\epsilon > 0$ に対して，$\delta = \epsilon D/n$ と定義する．次にそれぞれの品物のサイズ s_i を $\overline{s}_i = \lfloor s_i/\delta \rfloor$ に，ナップサックのサイズ D を $\overline{D} = \lfloor D/\delta \rfloor$ に修正し，この修正サイズのもとで，動的計画法を用いて最大価値を達成する品物の集合を求める．このアルゴリズムは $\mathrm{O}(n\overline{D}) = \mathrm{O}(n^2/\epsilon)$ 時間で走る．

入力として処理時間 p_j，重み w_j，リリース時刻 r_j が表 2 のように与えられる 4 つの仕事の集合 $J = \{1, 2, 3, 4\}$ が与えられた場合，アルゴリズム Greedy-Interval は図 1 に示したスケジュールを出力する．このスケジュールの目的関数値は 533 である．この図において，横軸は時間軸であり，長方形の高さは比 w_j/p_j の値を表している．

表 2: 入力例

仕事 j	p_j	w_j	r_j	w_j/p_j
1	1	6	5	6
2	4	16	2	4
3	3	9	8	3
4	6	12	0	2

2.2 Goemans らのアルゴリズム [3]

Goemans らは，中断を許して構成したスケジュールから得られる情報を用いて中断を行わないスケジュールを構成するという Phillips ら [8] の考えを発展させ，競合比 1.6853 のランダム化アルゴリズムを与えた．彼らのアルゴリズムについて述べる前に，必要となるいくつかの概念について説明する．

まず，(オフラインの問題に対する) 次のような線型計画緩和 (LP 緩和) を考える．なお，T は十分大きな正整数であり，$y_{jt} = 1$ ($y_{jt} = 0$) ならば時刻 t に仕事 j の処理が実行されている (いない) と考える．

$$\text{minimize} \quad \sum_{j \in J} w_j C_j^{LP}$$

subject to

$$\sum_{t=r_j}^{T} y_{jt} = p_j \qquad (j \in J)$$

$$\sum_{j \in J} y_{jt} \le 1 \qquad (t = 0, \ldots, T)$$

$$C_j^{LP} = \frac{p_j}{2} + \frac{1}{p_j} \sum_{t=r_j}^{T} y_{jt} \left(t + \frac{1}{2} \right) \qquad (j \in J)$$

$$y_{jt} \ge 0 \qquad (j \in J, t = r_j, \ldots, T)$$

この LP 緩和は組合せ的に解くことができる．任意の時刻において，すでにリリースされておりかつ処理が完了していない仕事の中で，w_j/p_j が最大の仕事を選び，これを中断を許す形でスケジュールしていくことで得られるスケジュールを LP スケジュー

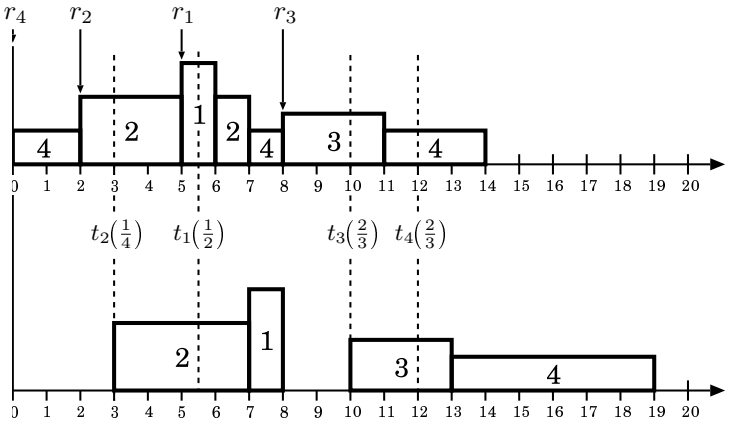

図 2: LP スケジュール (上) とアルゴリズム Random-Alpha の出力例 (下)

ルと呼ぶ．LP スケジュールは上の LP 緩和の最適解のひとつであることが Goemans [2] によって示されている．なお，LP スケジュールはオンライン環境でも構成することができる．

次に仕事の α_j-点について述べる．ある $0 < \alpha_j \leq 1$ に対し，仕事 j の α_j-点とは，その仕事の処理のうち，割合 α_j が完了した時刻，すなわち $\alpha_j p_j$ 単位の処理が完了した時刻のことである．仕事 j の α_j-点を $t_j(\alpha_j)$ と書く．

Goemans らのアルゴリズムは次のとおりである．

アルゴリズム Random-Alpha

次の a)，b)，c) を並列に行っていく．

a) 仕事 j が到着したとき，その仕事に対する α_j の値を確率密度関数

$$f(\alpha) = \begin{cases} (c-1)\cdot \mathrm{e}^{\alpha} & (\alpha \leq \delta \text{ のとき}) \\ 0 & (\text{それ以外}) \end{cases}$$

にしたがってランダムに選ぶ．ここで，δ と c は，γ を式

$$\gamma + \ln(2-\gamma) = \mathrm{e}^{-\gamma}\big((2-\gamma)\mathrm{e}^{\gamma} - 1\big)$$

の $0 < \gamma < 1$ を満たす解 $\gamma \approx 0.4835$ としたとき，$\delta := \gamma + \ln(2-\gamma) \approx 0.8999$, $c := 1 + \mathrm{e}^{-\gamma}/\delta < 1.6853$ である．

b) LP スケジュールを構成しながら，α_j-点が現れた順に仕事を待ち行列に入れていく．

c) 機械が空ならば，待ち行列に入っている仕事を非中断にスケジュールする．

アルゴリズム Random-Alpha の動作例を示す．まず，表 2 の入力に対する LP スケジュールと，このスケジュールにおいて $\alpha_1 = 1/2$, $\alpha_2 = 1/4$, $\alpha_3 = 2/3$, $\alpha_4 = 2/3$ としたときの α_j-点は図 2 の上側のようになる．このとき，Random-Alpha の出力は図 2 の下側のようになり，目的関数値は 505 である．

LP スケジュールは先に示した LP 緩和の最適解のひとつである．図 2 の例において，LP 緩和の 3 つ目の制約式にしたがってそれぞれの仕事の完了時刻を計算すると $C_1^{LP} = 6$, $C_2^{LP} = 25/4$, $C_3^{LP} = 11$, $C_4^{LP} = 65/6$ となり，LP 緩和の最適値は $\sum w_j C_j^{LP} = 365$ であることがわかる．

2.3 Anderson, Potts のアルゴリズム [1]

2002 年，Anderson と Potts は競合比 2 の決定性アルゴリズムを与えた．このアルゴリズムは，SWPT 法をオンラインに適用し，修正を加えたものである．

アルゴリズム Delayed-SWPT

以下の 1°，2° を繰り返す．

1° 到着している仕事のうち，比 w_j/p_j が最大の仕事 j を選ぶ (処理可能な仕事がなければ，仕事が届くまで待つ)．

2° $p_j \leq t$ ならば仕事 j をスケジュールする．そうでなければ，他の仕事が届くか，$t = p_j$ になるまで待つ．

表 2 で示した入力に対し，アルゴリズム Delayed-SWPT は図 3 に示すようなスケジュールを出力す

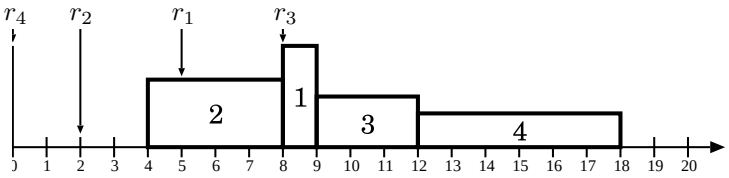

図 3: アルゴリズム Delayed-SWPT の出力例

る．このスケジュールの目的関数値は 506 である．

3 実験的性能評価

ここでは前節で示したアルゴリズムについて計算機実験を行い，実際的な性能を評価する．また，その結果に基づいて，理論的な精度保証はもたないながらも，実用上優れたスケジュールを構成するいくつかのヒューリスティックアルゴリズムを与える．

3.1 データ生成

実験のインスタンスは次のように発生させた．

- 仕事数 n は 10, 20, 50, 100, 200 から選ぶ．
- 仕事の処理時間 p_j と重み w_j はそれぞれ
  - $[1, 100]$ の範囲の一様分布，
  - 平均 $\mu = 50$, 分散 $\sigma^2 = 25$ の正規分布，
  - 最頻値を 2 つもつ分布 (確率 0.5 で平均 $\mu = 25$, 分散 $\sigma^2 = 5$ の正規分布，確率 0.5 で平均 $\mu = 75$, 分散 $\sigma^2 = 5$ の正規分布をとる)

  の 3 通りの確率分布に従って，乱数を用いて発生させる．
- リリース時刻は $\left[1, \lambda \sum_{j=1}^{n} p_j\right]$ の範囲の一様分布に従い，乱数を用いて発生させる．ここで，λ の値としては 0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5 の 7 つから選ぶ．

以上のすべての組合せ 315 通りについて，それぞれ 20 ずつ，合計 6300 のインスタンスを生成する．

3.2 実験結果

アルゴリズムの評価は，構成されたスケジュールを (オフラインの問題に対する) 下界と比較して行う．より正確にいえば，アルゴリズムによって構成されたスケジュールの目的関数値を ALG，2.2 節で示した LP 緩和の最適値を LP と書くとき，ALG/LP の値を考え，これが 1 に近いほどよいアルゴリズムであるとする．

なお，Greedy-Interval について，ラウンディングを行う際の ϵ の値は 0.1 に固定して実験した．

3 つのアルゴリズムについての実験結果を表 3 に示す．この表は ALG/LP の値の平均と標準偏差，そして最大値を，仕事数 n に関して分類したものである．最下行には全インスタンスに対する平均と標準偏差，最大値を示す．また，ALG/LP の値の具体的な分布を図 4 に示す．

Random-Alpha と Delayed-SWPT が理論的な競合比よりもかなりよい結果を出したのと比べると，Greedy-Interval の性能はややもの足りなさを感じる．また，Random-Alpha と Delayed-SWPT に関しては仕事数が増えるにしたがって ALG/LP の値が 1 に近づくことが観察されるが，逆に Greedy-Interval が構成するスケジュールの ALG/LP の平均値はゆるやかな増加傾向をみせる．

3.3 実験結果から導かれるヒューリスティック

Random-Alpha と Delayed-SWPT は非常に優れたスケジュールを構成するが，ある特殊な場合にはやや悪いスケジュールを構成することがある．ここでは，アルゴリズムを少し修正することで，実用的な視点からみてよい結果を出すヒューリスティックアルゴリズムを導く．

Goemans らのアルゴリズムではそれぞれの仕事 j に対する α_j の値を確率分布に基づいてランダムに選んでいる．この場合，平均的にみればよい結果を導くのであるが，その性能は α_j の選び方に大きく左右されることになる．w_j/p_j の値が相対的に小さな仕事に対しては大きな α_j を，逆に w_j/p_j が

表 3: 仕事数 n で分類した Greedy-Interval，Random-Alpha，Delayed-SWPT の性能

n	Greedy-Interval			Random-Alpha			Delayed-SWPT		
	平均	標準偏差	最大	平均	標準偏差	最大	平均	標準偏差	最大
10	1.36524	0.00181	1.76611	1.16099	0.00356	1.41672	1.08506	0.00232	1.31914
20	1.36525	0.00166	1.70256	1.10476	0.00205	1.37430	1.04630	0.00149	1.17816
50	1.37721	0.00251	1.71783	1.05667	0.00129	1.14670	1.02091	0.00097	1.07958
100	1.38915	0.00427	1.67192	1.03419	0.00114	1.09962	1.01103	0.00086	1.04772
200	1.39713	0.00304	1.66524	1.02007	0.00096	1.04565	1.00572	0.00081	1.01803
全体	1.37902	0.00148	1.76611	1.07550	0.00096	1.41672	1.03396	0.00050	1.31914

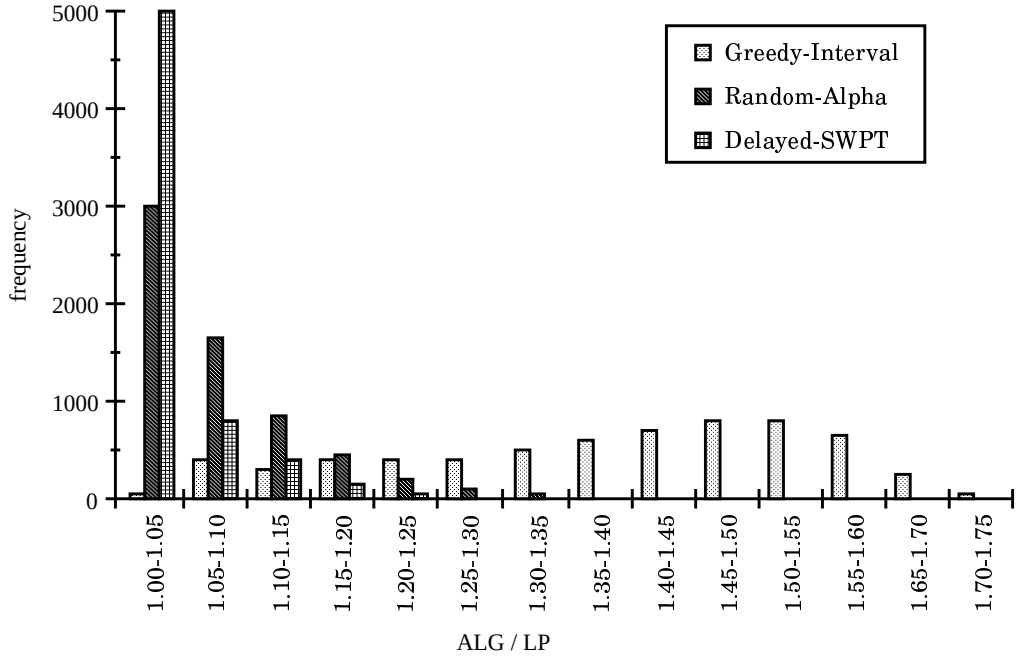

図 4: 全 6300 インスタンスに対する ALG/LP の値の分布

相対的に大きな仕事に対しては小さな α_j を選ぶことが理想的である．そこで，次のような α_j の選び方を考えてみる．j 番目にリリースされた仕事 j の w_j/p_j の値が，それまでに到着している j 個の仕事の中で k 番目に大きいとするとき，$\alpha_j = k/(j+1)$ と決める．このヒューリスティックアルゴリズムを Greedy-Alpha と呼ぶことにする．

Greedy-Alpha についての実験結果を表 4 に示す．Random-Alpha と比べると，Greedy-Alpha は平均的な性能としてはかなり改善されているが，最悪の場合の値については大きく劣っている．

次に SWPT アルゴリズムをオンラインに適用させることを考える．機械が空になったとき，既にリリースされており，かつまだスケジュールされていない仕事の中から w_j/p_j が最大の仕事を選び，スケジュールする．このヒューリスティックアルゴリズムを Online-SWPT と呼ぶことにする．

表 4: アルゴリズム Greedy-Alpha の性能

n	平均	標準偏差	最大
10	1.09516	0.00284	1.72783
20	1.06273	0.00198	1.38715
50	1.03664	0.00129	1.24911
100	1.02152	0.00105	1.12016
200	1.01279	0.00090	1.05883
全体	1.04593	0.00064	1.72783

Delayed-SWPT が構成するスケジュールについて考えてみる．Delayed-SWPT は次のような場合にあまりよくない結果を示す．時刻 t に処理時間の長い仕事 j の処理を開始したとする．その直後の時刻 $t+1$ に $w_k/p_k \gg w_j/p_j$ であるような仕事 k が

表 5: Online-SWPT と Modified-SWPT ($\epsilon = 1/4, 1/2$) の性能

n	Online-SWPT			Modified-SWPT ($\epsilon = 1/4$)			Modified-SWPT ($\epsilon = 1/2$)		
	平均	標準偏差	最大	平均	標準偏差	最大	平均	標準偏差	最大
10	1.05554	0.00170	1.38804	1.05589	0.00171	1.38804	1.06056	0.00183	1.33937
20	1.03321	0.00125	1.18807	1.03278	0.00124	1.21947	1.03555	0.00130	1.17953
50	1.01605	0.00088	1.09354	1.01614	0.00088	1.08839	1.01693	0.00090	1.07023
100	1.00853	0.00083	1.03477	1.00860	0.00083	1.03485	1.00905	0.00083	1.03833
200	1.00440	0.00080	1.01782	1.00445	0.00080	1.01914	1.00463	0.00080	1.01674
全体	1.02371	0.00037	1.38804	1.02373	0.00037	1.38804	1.02551	0.00040	1.33937

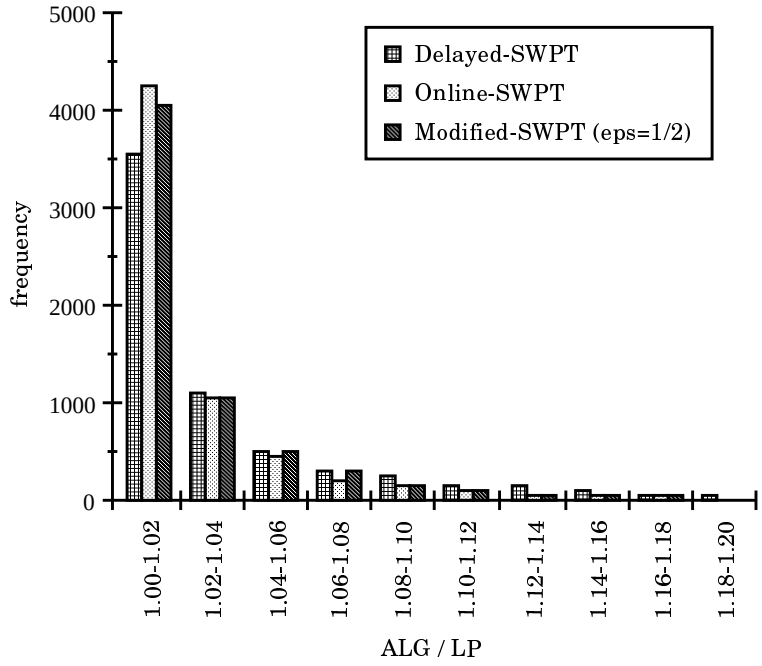

図 5: Online-SWPT，Modified-SWPT と Delayed-SWPT との ALG/LP の値の分布の比較

到着したとすると，仕事 k は仕事 j の処理が完了するまで待たなければならない．この場合，アルゴリズムは時刻 $t+1$ まで待って先に仕事 k を処理し，その後仕事 j を処理した方がよい．

そこで以下のような，仕事の処理時間に応じて処理の開始可能時刻を遅らせるヒューリスティックアルゴリズム Modified-SWPT を考える．ある ϵ に対し，それぞれの仕事 j のリリース時刻 r_j を $r'_j = \max\{r_j, \epsilon p_j\}$ と修正し，この修正インスタンスに Online-SWPT を適用する．

Online-SWPT と Modified-SWPT についての実験結果を表 5 に示す．Online-SWPT は平均的な性能としては Delayed-SWPT を上回っているが，最悪の場合についてはやや劣っている．一方，$\epsilon = 1/2$ の場合の Modified-SWPT は平均的な性能として Delayed-SWPT を上回っており，最悪の場合についてもほとんど遜色ない．特に仕事数が 50 以上では，最悪の場合も Delayed-SWPT を上回っている．

全 6300 インスタンスに対する Online-SWPT と Modified-SWPT の ALG/LP 値の分布を Delayed-SWPT と比較したものが図 5 である．Modified-SWPT ($\epsilon = 1/2$) は Delayed-SWPT と Online-SWPT の中間的な性能を示すことがわかる．

Modified-SWPT に対して，$\epsilon = 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1, 2, 4$ と動かしたときの，ALG/LP の平均値と最大値の変化を調べた結果が図 6 である．これより，最悪の場合の値を小さくするためには ϵ を 1/2 前後に選ぶとよいことがわかる．なお，ϵ の値を小さくしていくと Online-SWPT の解に近づく ($\epsilon = 0$ の場合は Online-SWPT に等しい)．

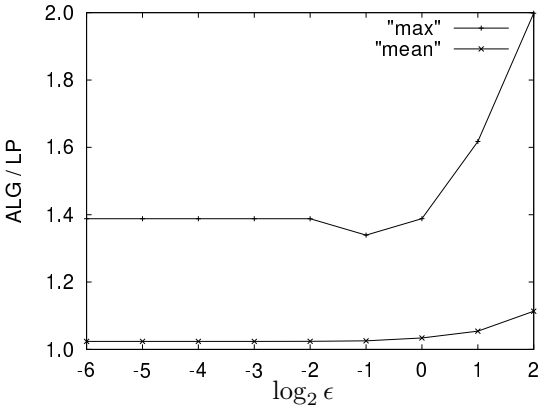

図 6: ϵ の与え方による Modified-SWPT の性能

4 おわりに

重みつき完了時刻和の最小化を目的とする単一機械スケジューリング問題に対して最近提案された 3 つのオンラインアルゴリズムを概説し，その実際的な性能の評価を行った．また，実験結果を分析することにより，実際的な視点からみて優れたヒューリスティックアルゴリズムを導いた．

今後の課題として，機械が複数台与えられた場合や，仕事に先行制約や納期などの条件が付加された場合に，よいスケジュールを構成するアルゴリズムの研究があげられる．

謝辞

本研究は, 一部, 中央大学理工学研究所，21 世紀 COE プログラム，文部省科学研究費補助金，および電気通信普及財団からの援助のもとで行われたものである．

参考文献

[1] E.J. Anderson and C.N. Potts, “On-line scheduling of a single machine to minimize total weighted completion time”, *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms*, 548–557, 2002.

[2] M.X. Goemans, “Improved approximation algorithms for scheduling with release dates”, *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, 591–598, 1997.

[3] M.X. Goemans, M. Queyranne, A.S. Schulz, M. Skutella and Y. Wang. “Single machine scheduling with release dates”, *SIAM Journal on Discrete Mathematics* **15**, 165–192, 2002.

[4] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnoy Kan, “Optimization and approximation in deterministic sequencing and scheduling: A survey”, *Annals of Discrete Mathematics*, **5**, 287–326, 1979.

[5] L.A. Hall, D.B. Shmoys and J. Wein, “Scheduling to minimize average completion time: Off-line and on-line algorithms”, *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, 142–151, 1996.

[6] L.A. Hall, A.S. Schulz, D.B. Shmoys and J. Wein, “Scheduling to minimize average completion time: Off-line and on-line approximation algorithms”, *Mathematics of Operations Research*, **22**, 513–544, 1997.

[7] J.A. Hoogeveen and A.P.A. Vestjens, “Optimal on-line algorithms for single-machine scheduling”, in W.H. Cunningham, S.T. McCormick and M. Queyranne (eds.), *Integer Programming and Combinatorial Optimization* (Proceedings of the 5th International IPCO Conference), Lecture Notes in Computer Science, vol. 1084, Springer, Berlin, 404–414, 1996.

[8] C. Phillips, C. Stein and J. Wein, “Minimizing average completion time in the presence of release dates”, *Mathematical Programming*, **82**, 199–223, 1998. An Extended abstract appeared under the title “Scheduling jobs that arrive over time” in S.G. Akl, F. Dehne, J.-R. Sack, N. Santoro (eds.), *Algorithms and Data Structures* (Proceedings of the 4th International WADS), Lecture Notes in Computer Science, vol. 955, Springer, Berlin, 86–97, 1995.

[9] W.E. Smith, “Various optimizers for single-stage production”, *Naval Research and Logistics Quarterly*, **3**, 59–66, 1956.

[10] L. Stougie and A.P.A. Vestjens, “Randomized on-line scheduling: How low can’t you go?”, Manuscript, 1997.