中央大学理工学研究所論文集　第9号　2004年
Journal of the Institute of Science and Engineering. Chuo University

施設配置問題とスケジューリング問題に対する高性能アルゴリズムの実験的性能評価

浅野孝夫*，上ヶ原誠*，九里史朗*

Experimental Evaluations of Algorithms with Performance Guarantee for the Facility Location and Scheduling Problems

Takao Asano, Makoto Kamigahara, Shiro Kunori

abstract

The facility location problem is to decide which facilities are open to use effectively and the scheduling problem is to find a schedule that specifies when and on which machine each job is to be excuted. These problems appear frequently in real environments. For example, the facility location problem plays a central role in GIS and scheduling arises in a variety of settings to control jobs on the central processing unit of a computer, to decide a plan in what order tasks should be processed. However, these problems are NP-hard, and a lot of researches have been done on approximation algorithms and on-line algorithms. In this paper, we evaluate experimental performance of representative algorithms for the metric uncapacitated facility location problem and for the single machine scheduling problem with release dates in which objective is to minimize a weighted sum of completion times.

1　はじめに

NP-困難な問題に対して多項式時間で最適解を求めるのは困難であるので，最適解に近い近似解を多項式時間で求めて使用することが多い．$\alpha \geq 1$ に対して，最小化問題ならば，いつも最適解の α 倍以下の解を入力サイズの多項式時間で求めるようなアルゴリズムを **α-近似アルゴリズム**という．最小化問題に対する α を**近似率**あるいは**精度保証**という．一方，実時間処理アルゴリズムでは，入力情報が得られた時点で，それ以降の情報がくる前に処理を実行 (確定) することが要求される．入力の情報がすべて得られてから処理を実行する通常のアルゴリズムに対して，このようなアルゴリズムは**オンラインアルゴリズム**と呼ばれる (通常のアルゴリズムをオンラインアルゴリズムに対比するため，以下，**オフラインアルゴリズム**ということもある)．最小化問題に対して，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．

精度保証付きの近似アルゴリズムやオンラインアルゴリズムの競合比の研究は近年精力的に実行され，様々なアルゴリズムが提案されてきている [22]．本論文では，そのようなアルゴリズムから，施設配置問題とスケジューリング問題に対して理論面から最近提案された代表的アルゴリズムを選び，それらを実装して，その実際的性能評価を与える．さらに，実際の状況に適したヒューリスティクスを提案し，その有効性も検証する．

* 中央大学理工学部物理学科（〒112-8551 東京都文京区春日 1-13-27）

2　施設配置問題

容量制限なしメトリック施設配置 (metric uncapacitated facility location) 問題 (以下, 簡略化して施設配置問題と呼ぶ) は, 設立地計画をモデル化した問題であり，OR の分野で幅広く研究されている．n_f 個の開設候補の施設集合 F と n_c 人の利用者集合 C からなる完全 2 部グラフ G が入力として与えられる ($n = n_f + n_c$, $m = n_f \times n_c$ とする)．さらに，施設 i の開設コスト $f_i > 0$ と施設 i と利用者 j との間の (距離の性質を満たす) 接続コスト $c_{ij} \geq 0$ が与えられる. このとき, 適切に施設を開設し, すべての利用者を開設した施設に接続するが，総コスト (開設コストと接続コストの合計) が最小になるような施設の開設と接続を求めることが施設配置問題である. 図 1 に，この問題の具体的な入力例 (すべての開設コスト f_i は一定, 施設数 n_f は 100 個, 利用者数 n_c は 100 人, 接続コストは 2 点間のユークリッド距離とした例) と出力例 (最適解) を示している．なお図において, □は開設候補の施設 (図 1(b) では未開設施設), ■は開設された施設, ○は利用者, そして実線分は接続を表している．

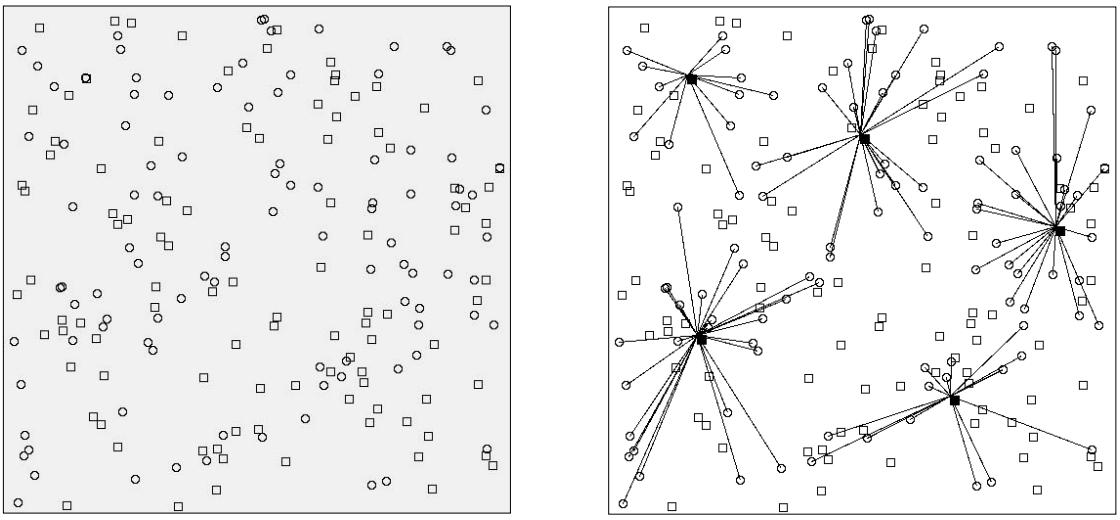

図 1　施設配置問題の入力例 (左図) とその出力例 (右図)

施設配置問題は NP-困難であり, 1997 年に定数近似アルゴリズム [18] が初めて提案されて以来, 様々な手法に基づいて近似アルゴリズムが提案されてきている (表 1)．なお ϵ は十分小さい正数である．

本節では, Mahdian-Markakis-Saberi-Vazirani の 1.861-近似アルゴリズム [13], 1.861-近似アルゴリズムの改善である Jain-Mahdian-Saberi の 1.61-近似アルゴリズム [12], 1.61-近似アルゴリズムを用いて初期解を求める Mahdian-Ye-Zhang の 1.52-近似アルゴリズム [16], そして 1.52-近似アルゴリズムを局所改善したもの (以下, 改善案という) の 4 つのアルゴリズムを概観する．その後，4 つのアルゴリズムを C++言語で実装して得られた実験的性能評価を与える．

表 1　施設配置問題に対する代表的アルゴリズム

著者	近似率	主な手法	発表年
Shmoys, Tardos, Aardal [18]	3.16	LP 丸め法	1997
Guha, Khuller [7]	2.408	LP 丸め法, 貪欲改善法	1998
Chudak [3]	1.736	LP 丸め法	1998
Korupolu, Plaxton, Rajaraman [14]	5+ϵ	局所探索法	1998
Jain, Vazirani [13]	3	主双対法	1999
Charikar, Guha [2]	1.853	主双対法, 貪欲改善法	1999
Mahdian, Markakis, Saberi, Vazirani [15]	1.861	双対フィット法	2001
Jain, Mahdian, Saberi [12]	1.61	双対フィット法	2001
Sviridenko [21]	1.582	LP 丸め法	2002
Mahdian, Ye, Zhang [16]	1.52	貪欲改善法	2002

2.1 代表的施設配置アルゴリズムの概略

4 つの近似アルゴリズムを概説する前に, 表 1 の代表的施設配置アルゴリズムは，いずれも整数計画問題 (IP) としての定式化とその線形計画緩和 (LP 緩和) と双対問題を用いているので，それから説明する．

施設 i に利用者 j が接続している ($x_{ij}=1$) かいない ($x_{ij}=0$) かを表す変数 $x_{ij}\in\{0,1\}$ と, 施設 i が開設している ($y_i=1$) かいない ($y_i=0$) かを表す変数 $y_i\in\{0,1\}$ を用いて施設配置問題は以下のように IP として定式化できる．

$$
\begin{aligned}
\min \quad & \sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in C} c_{ij}x_{ij} && && (1)\\
\text{s.t.} \quad & \sum_{i\in F} x_{ij} \;=\; 1 && (\forall j\in C)\\
& x_{ij} \;\le\; y_i && (\forall i\in F, j\in C)\\
& x_{ij} \;\in\; \{0,1\} && (\forall i\in F, j\in C)\\
& y_i \;\in\; \{0,1\} && (\forall i\in F)
\end{aligned}
$$

この定式化 IP(1) で，制約式 $x_{ij}\le y_i$ は，利用者 j が接続されている施設 i は必ず開設されなければならないことを示している．また，制約式 $\sum_{i\in F} x_{ij}=1$ は利用者 j がどれかの施設に必ず接続されなければならないことを示している．したがって，目的関数 $\sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in C} c_{ij}x_{ij}$ は，開設コストと接続コストの合計になる．

IP(1) の整数制約を $0\le x_{ij}\le 1$ と $0\le y_i\le 1$ に緩和すると，$x_{ij}\le 1$ と $y_i\le 1$ は冗長であるので，IP(1) の LP 緩和は以下のように書ける．

$$
\begin{aligned}
\min \quad & \sum_{i\in F} f_i y_i + \sum_{i\in F}\sum_{j\in C} c_{ij}x_{ij} && && (2)\\
\text{s.t.} \quad & \sum_{i\in F} x_{ij} \;=\; 1 && (\forall j\in C)\\
& x_{ij} \;\le\; y_i && (\forall i\in F, j\in C)\\
& x_{ij} \ge\; 0 && (\forall i\in F, j\in C)\\
& y_i \;\ge\; 0 && (\forall i\in F)
\end{aligned}
$$

LP 緩和 (2) の双対問題は次のように書ける．

$$
\begin{aligned}
\max \quad & \sum_{j\in C} \alpha_j && && (3)\\
\text{s.t.} \quad & \sum_{j\in C} \beta_{ij} \;\le\; f_i && (\forall i\in F)\\
& \alpha_j - \beta_{ij} \le\; c_{ij} && (\forall i\in F, j\in C)\\
& \alpha_j \;\ge\; 0 && (\forall j\in C)\\
& \beta_{ij} \ge\; 0 && (\forall i\in F, j\in C)
\end{aligned}
$$

なお，α_j は利用者 j の全負担費用で，β_{ij} は施設 i を開設するときの j の貢献分 (負担額) である．すなわち $\beta_{ij}=\max(\alpha_j-c_{ij},0)$ と考えることができる．

2.1.1 1.861 近似アルゴリズム [15]

図 2 の Mahdian-Markakis-Saberi-Vazirani の 1.861-近似アルゴリズム [15] は主双対法に基づいているといえる．アルゴリズムは，双対問題の解 $(\boldsymbol{\alpha},\boldsymbol{\beta})$ を時刻ともに更新しながら，主問題の実行可能整数解 (施設配置問題の解)$(\boldsymbol{x},\boldsymbol{y})$ を求める．得られる双対問題の解 $(\boldsymbol{\alpha},\boldsymbol{\beta})$ は必ずしも双対問題の実行可能解にならない

ので，双対フィット法を用いて解析をする．すなわち，双対問題の解 $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ をある定数 $R(\geq 1)$ 倍縮小して $(\boldsymbol{\alpha}/R, \boldsymbol{\beta}/R)$ が実行可能解になるようにする．するとこの縮小する値 R が近似率になる．なお，この双対フィット法は，アルゴリズムで得られた解の近似率の解析のみに用いられる手法である．

未接続の利用者の費用 α_j をすべて同じ割合で増加しながら，$\beta_{ij} = \max\{\alpha_j - c_{ij}, 0\}$ と定め，以下の (a), (b) を繰り返す．すべての利用者が接続されたら終了とする．

(a)　**if** (ある未開設施設 i とすべての未接続の利用者 j に対して, $\sum_j \beta_{ij} = f_i$ が成立) **then**
施設 i を開設し, $\alpha_j \geq c_{ij}$ であるすべての未接続の利用者 j を施設 i に接続する．

(b)　**if** (ある開設施設 i とある未接続の利用者 j に対して, $\alpha_j = c_{ij}$ が成立) **then**
利用者 j を施設 i に接続する．

図 2　Mahdian-Markakis-Saberi-Vazirani の 1.861-近似アルゴリズム [15]

このアルゴリズムの (a) は, 幾人かの未接続の利用者による未開設施設 i への貢献の合計が i の開設コストと等しくなったら, i を開設として, i に貢献している ($\alpha_j \geq c_{ij}$ となる) 未接続の利用者 j は i に接続することを意味する．(b) は, 未接続の利用者 j の全費用が j とある開設施設との接続コストに等しくなったら, j をこの施設に接続することを意味する．また, (a), (b) から一度接続された利用者の接続は不変であることがわかる．

図 8 は，(a) と (b) の実行例である．なお, 以下のアルゴリズムの実行例を示す図において, 左が実行前, 右が実行後を表していて, 明記していない接続コストは, 距離の性質を満たす大きい値としている．

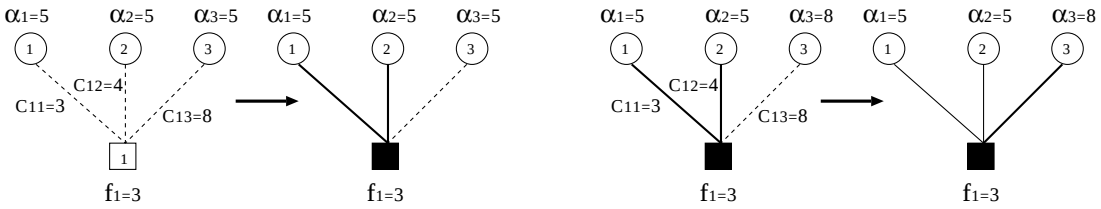

図 3　図 2 のアルゴリズムの実行例 ((a)(左図), (b)(右図))

図 2 のアルゴリズムでは, ある利用者 j はある施設に一度接続すると他の施設への影響は考えないものとしている．具体的には, (a) の条件において $\sum_j \beta_{ij} = f_i$ となる j を未接続の利用者のみから考えている．そのため, アルゴリズムの最後において施設 i に接続していない j の β_{ij} を含む $\sum_{j \in C} \beta_{ij}$ を考えると,

$$\sum_{j \in C} \beta_{ij} = \sum_{j \in C} \max(\alpha_j - c_{ij}, 0) > f_i$$

となり，双対問題 (3) の 1 番目の制約式は成立しない．よって, $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ は双対問題の実行不可能解である．もしインスタンスに依存せず,

$$\sum_{j \in C} \max\{(\frac{\alpha_j}{R} - c_{ij}), 0\} \leq f_i$$

を満たす最小の $R \geq 1$ を求めることができれば, $(\boldsymbol{\alpha}/R, \boldsymbol{\beta}/R)$ は実行可能解となり, $\frac{\sum_{j \in C} \alpha_j}{R}$ は最適解のコストの少なくとも $\frac{1}{R}$ 倍である．そのため近似率は R となる．しかし, この R は近似率が最悪となるインスタンスがわからないかぎり求められない．そのため, Mahadian らは factor-revealing LP という定式化を用いて近似率 1.861 を求めている．

2.1.2 1.61-近似アルゴリズム [12]

Mahdian-Markakis-Saberi-Vazirani の 1.861-近似アルゴリズムでは一度決めた接続は不変であったが, 総コストが減少するなら**接続を変更する**としたのが Jain-Mahdian-Saberi の 1.61-近似アルゴリズムである [12](図 4). アルゴリズムの流れは Mahdian らの 1.861-近似アルゴリズムと同じであるが, 利用者 j から施設 i への貢献 β_{ij} の定義を以下のように変更している.

- **if** (利用者 j は未接続) **then** $\beta_{ij} = \max(\alpha_j - c_{ij}, 0)$ とする.
- **if** (利用者 j は既に施設 $i^{'}$ と接続) **then** $\beta_{ij} = \max(c_{i^{'}j} - c_{ij}, 0)$ とする.

この定義から, 既に施設に接続している利用者においても貢献を考えることがわかる. これに伴い, 1.861-近似のアルゴリズムの (a) の実行において, 既に接続している利用者 j も考える. すなわち, j が既に別の施設 $i^{'}$ に接続しているなら, j の接続を $i^{'}$ から i に変更する. この操作では $\beta_{i^{'}j}$ は不変であり, β_{ij} のみが変わる. なお, 接続を変更することで $i^{'}$ へ接続している利用者がいなくなったら, $i^{'}$ は未開設とする (本節のこれ以降のアルゴリズムでも同様).

未接続の利用者の費用 α_j をすべて同じ割合で増加しながら, 利用者 j が未接続なら $\beta_{ij} = \max\{\alpha_j - c_{ij}, 0\}$ と定め, 利用者 j が既に施設 $i^{'}$ と接続なら $\beta_{ij} = \max\{c_{i^{'}j} - c_{ij}, 0\}$ と定めて, 以下の (a), (b) を繰り返す. すべての利用者が接続されたら終了とする.

(a) **if** (ある未開設施設 i とすべての利用者 j に対して, $\sum_j \beta_{ij} = f_i$ が成立) **then**
施設 i を開設し, $\alpha_j \geq c_{ij}$ であるすべての利用者 j を施設 i に接続する.

(b) **if** (ある開設施設 i とある未接続の利用者 j に対して, $\alpha_j = c_{ij}$ が成立) **then**
利用者 j を施設 i に接続する.

図 4 Jain-Mahdian-Saberi の 1.61-近似アルゴリズム [12]

図 5 に接続を変更する実行例を示している.

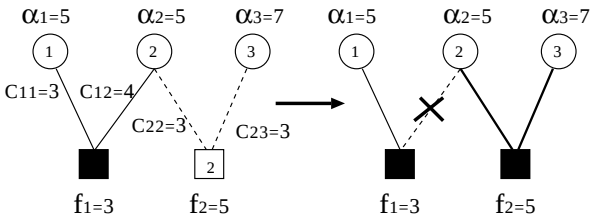

図 5 接続を変更する実行例

このアルゴリズムの解も 1.861-近似アルゴリズムと同様実行不可能である. 双対フィット法を用いて近似率が 1.61 と解析されている [12].

2.1.3 1.52-近似アルゴリズム [16]

Mahdain-Ye-Zhang のアルゴリズム [16](図 6) は, 開設コストを $\delta(> 1)$ 倍してから Jain らの 1.61-近似アルゴリズムを用いることで, 貢献を効果的に集めることのできる施設のみをまず開設しておき, その後の貪欲改善により, 総コストをできるだけ減少する未開設施設を開設していくアルゴリズムである.

(初期解)　スケーリングパラメータを $\delta = 1.504$ とする．このとき, 開設コストを一様に δ 倍した問題に対して, 図 4 の 1.61-近似アルゴリズムを適用する．

(貪欲改善)　開設コストを一様に $\frac{1}{\delta}$ 倍した問題 (つまりもとの問題) に対して，総コストが減少する ($gain(i) > 0$ である) かぎり以下を繰り返す．

- **if** (ある未開設施設 i の $\frac{gain(i)}{f_i}$ が最大) **then**
  i を開設して, 利用者は最も近い開設施設に接続を変更する．

図 6　Mahdian-Ye-Zhang の 1.52-近似アルゴリズム [16]

このアルゴリズムでは, 総コストの減少を $gain$ として表している．未開設施設 i を開設することで接続コストが c から c' になるなら, $gain(i) = c - c' - f_i$ と定義する．なお，貪欲改善の条件において, $gain(i)$ ではなく $\frac{gain(i)}{f_i}$ を用いているが，$gain(i)$ が大きいだけでなく f_i も小さい施設の開設を強調しているといえる．

図 7 は貪欲改善の実行例を示している (左は初期解を求めた直後を表し, 便宜上 $\delta = 1.5$ としている)．図 7 における $gain(2)$ は $(4-3)+(8-3)-5 = 1$ である．

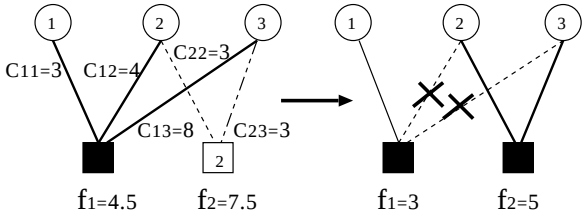

図 7　貪欲改善の実行

2.1.4　改善案 (1.52 近似の局所改善)

図 6 の 1.52-近似アルゴリズムと同じ方法で初期解を求め, 1.52-近似アルゴリズムの貪欲改善において施設を開設することだけではなく**開設施設を削除する (未開設にする)** ことも考慮するヒューリスティックを導入する (この考え方は [2] でも提案されている)．すなわち, もし開設施設 i を削除すると総コストが減少するなら, この減少する値も, $gain(i)$ と定義する (未開設施設 i' の $gain(i')$ は Mahdian-Ye-Zhang のものとする)．このような $gain$ をすべての施設 i で考え, $gain(i) > 0$ であるかぎり, $\frac{gain(i)}{f_i}$ が最大の i に対して開設か削除, もしくはその両方を実行するものを改善案と呼ぶことにする．

2.2　計算機実験

1.52-近似アルゴリズムと改善案のパラメータ δ の値は, 解析において近似率が最も良くなる $\delta = 1.504$ を用いているが, δ を変化した場合の実験的性能を観察するため以下の実験を行った．入力データとして, 施設と利用者を座標 $(0,0)$ と $(500,500)$ で定まる正方形内の整数格子点をそれぞれランダムに 100 個ずつ ($n_f = n_c = 100$) 発生し, 開設コストはすべての施設で一定 ($f_i = 1000$), 接続コストは施設と利用者のユークリッド距離としたものを用いている．図 8 は，それぞれの δ において実験を 20 回行ったときの平均値を近似率に用いている (横軸が δ で，縦軸が $\delta = 1.504$ の近似率に対する比率である)．なお, 改善案は図では improved と表記している．

解析では $\delta = 1.504$ とした場合に近似率が最も良くなるが, 図 8 の入力データでは，δ が 1.504 より小さい $\delta = 1.3$ の場合の方が近似率の良いことが観察された．改善案については, $\delta = 1.0$ とした場合, つまりコストスケーリングを行わずに初期解を求めた場合の方が近似率は良くなることが観察された．

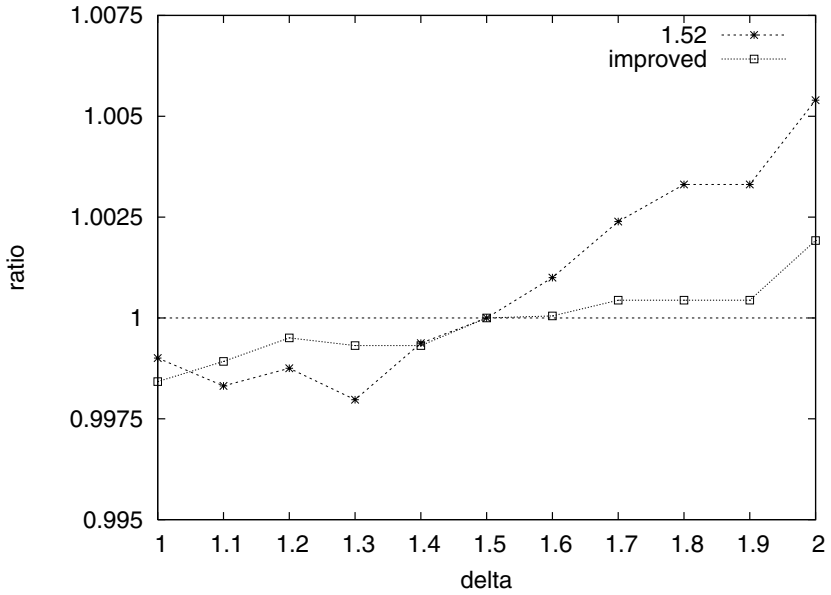

図 8 パラメータ δ による性能評価

2.2.1 実験的性能評価

上記の 4 つの近似アルゴリズムに対して様々な計算機実験を行い, それぞれの近似アルゴリズムの性能を観察した. 最適解の値は, LP 緩和 (2) を線形計画ソフト XPRESS-MP で解いて得られた目的関数値を用いた. 入力データは,

- 施設と利用者は, 座標 $(0, 0)$ から $(500, 500)$ の間でランダムに 100 個ずつ発生,
- 開設コストは, 1 から 1000 の間のランダムな正整数,
- 施設と利用者の接続コストは, 2 点間 (施設と利用者の間) のユークリッド距離

としたものを用い, この入力データ (以下, **標準入力データ**という) の一部を変更することで様々な実験を行った. 以下の図の縦軸は, それぞれの横軸の値で実験を 20 回行ったときの近似率の平均値を表している.

実験 1: 施設数と利用者数による影響

- 施設数 n_f, 利用者数 n_c を増加する場合
  (標準入力データの n_f, n_c をそれぞれ同じ割合で増加した場合の性能評価)
- 利用者数 n_c のみを増加する場合
  (標準入力データの n_f を $n_f = 100$ と固定して, n_c のみを増加した場合の性能評価)

1.861-近似アルゴリズムは利用者数 n_c が多い場合, 性能がわずかに悪くなるが, その他のアルゴリズムは施設数と利用者数による影響を受けないことが観察された.

実験 2: 開設コストと接続コストによる影響

- 開設コスト f_i を増加する場合
  (標準入力データの開設コスト f_i を, ランダムではなくすべての施設で一定とし, この値を増加した場合の性能評価) 結果を図 9 に示している (横軸は開設コスト).

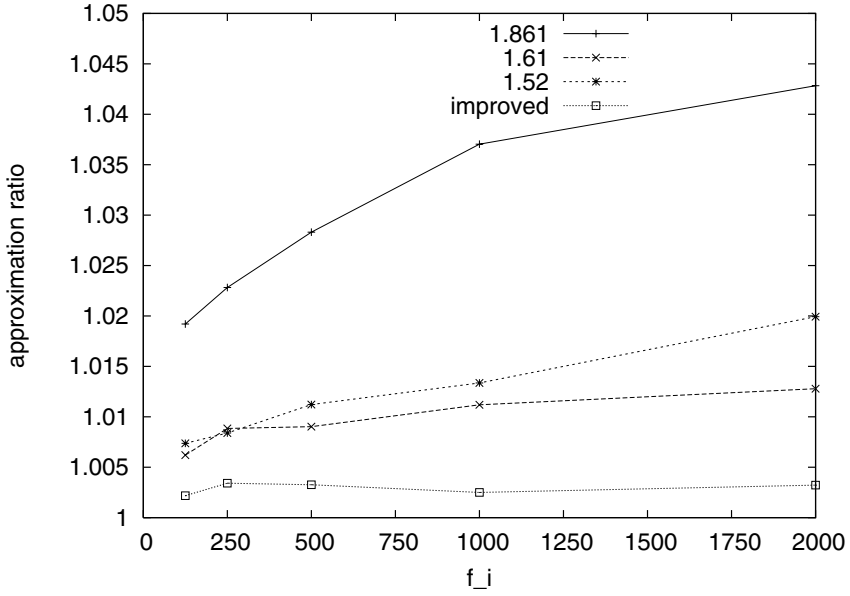

図 9　開設コスト f_i を増加した結果

- 接続コストを増加する場合

  (標準入力データの開設コストを一定 ($f_i = 1000$) として, 最大座標 (x, x) を $x = 500$ からさらに増加した場合の性能評価) 結果を図 10 に示している (横軸は最大座標).

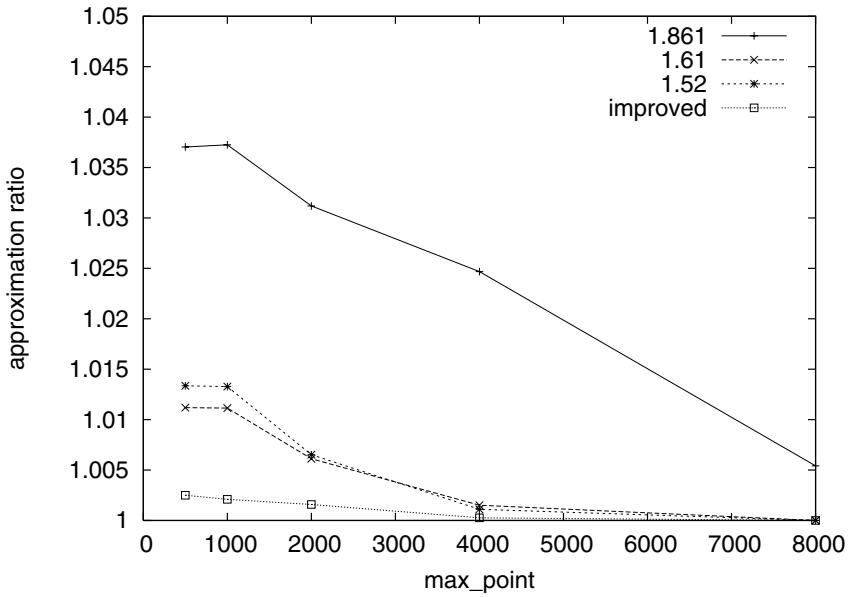

図 10　接続コストを増加した結果

図 9,10 より, 開設コストによる影響は接続コストの影響に比べて大きいことが観察された.

実験 3: 接続コストを最短パスの長さとする場合の影響

接続コストを, ユークリッド距離以外の距離とした場合を考える．ここでは，施設と利用者の 2 部グラフのそれぞれの辺に 0 から 707($500 \times \sqrt{2} \approx 707$) の間のランダムな整数を与え, 2 部グラフの最短パス長を接続コストとしている．このように標準入力データの接続コストを変更し, n_f, n_c を同じ割合で増加した場合の性能評価を図 11 に示している (横軸は n_f, n_c を表している).

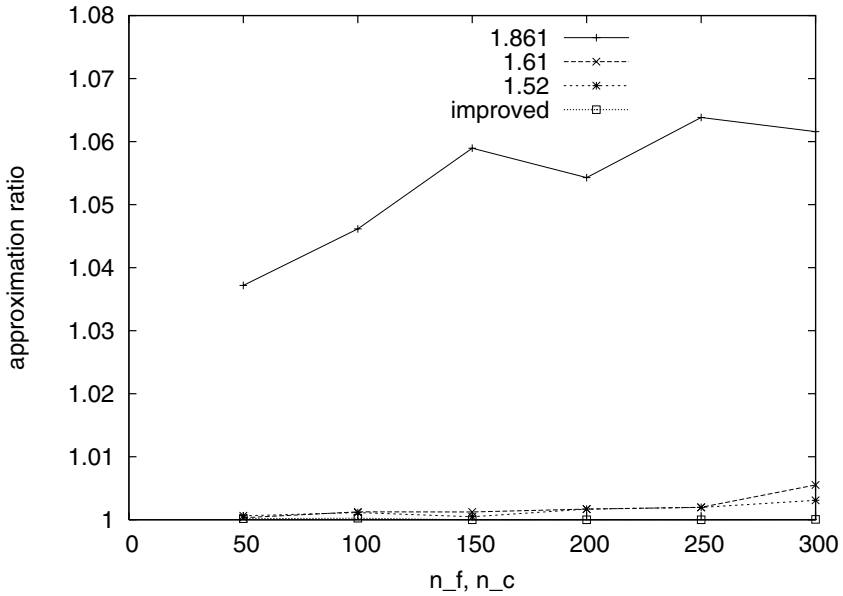

図 11 接続コストを最短パス長とした結果

図 11 より, 1.861-近似アルゴリズムはかなり性能が悪くなることが観察された．これは, 接続を変更できないことが最大の理由であると考えられる.

2.2.2 計算機実験のまとめ

計算機実験では, 理論的な近似率よりもかなり良い近似率になることが観察できた．また, 削除を加えた改善案は, 他のアルゴリズムよりもかなり性能が良く, 最適解もしくはそれに近い解を求めることが観察できた.

1.52-近似アルゴリズムは, 解析では 1.61-近似アルゴリズムより良い近似率となるが, 実験では 1.61-近似アルゴリズムよりも近似率が悪くなる場合があった．これは, 1.52-近似アルゴリズムの初期解で開設される施設が, 最適解の開設施設とは大きく異なることが原因であると思われる.

4 つの近似アルゴリズムの実験的性能の観察から，入力データに対する近似率は, 解析による値よりもかなり良くなってしまったため, 最悪の近似率となるインスタンスを考え, このインスタンスに対しても性能評価を行うことが今後の課題であると思われる．また, 近似率が 1.463 より良いアルゴリズムは存在しない [7] と考えられているため, 1.463 により近い近似率を達成する新たな手法のアルゴリズムの提案も今後の重要な課題であろう.

3 オンラインスケジューリング問題

本節では，重みつき総完了時刻を最小化する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 個の仕事の集合 J が与えられる．各仕事 $j \in J$ には，処理時間 p_j，重み $w_j > 0$ およびリリース時刻 $r_j \geq 0$ が付随して与えられる．仕事 $j \in J$ は，リリース時刻 r_j 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

表 2　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	発表年
Hall, Shmoys, Wein [8]		$4+\epsilon$	1996
Hall, Schulz, Shmoys, Wein [9]		$3+\epsilon$	1997
Goemans [4]	2	2.4143	1997
Goemans, Queyranne, Schulz, Skutella, Wang [5]	1.6853		2002
Anderson, Potts [1]		2	2002

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が問題である．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 [6] 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．これに対して，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 $1||\sum w_j C_j$ と書かれる問題) は，SWPT (Shortest Weighted Processing Time) 法によって最適解の得られることが 1956 年に Smith によって示されている [19]．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 j をスケジュールしていく方法が SWPT 法である．

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．

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 [1, 11]，ランダム化アルゴリズムとしては競合比 $\mathrm{e}/(\mathrm{e}-1) \approx 1.582$ [20] 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 [17] 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 [8] 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 [9] 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 [4] 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 [5] 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 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 (表 2)．

本節では特に，Hall, Schulz, Shmoys, Wein によるアルゴリズム [9]，Goemans, Queyranne, Schulz, Skutella, Wang によるアルゴリズム [5]，Anderson と Potts によるアルゴリズム [1] 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，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ヒューリスティクスを与える．

3.1　代表的アルゴリズムの概略

この節では，Hall, Schulz, Shmoys, Wein のアルゴリズム [9], Goemans, Queyranne, Schulz, Skutella, Wang のアルゴリズム [5]，Anderson と Potts のアルゴリズム [1] について，その概略を述べる．

3.1.1 Hall, Schulz, Shmoys, Wein のアルゴリズム [9]

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．以下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)．

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 $O(n\overline{D}) = O(n^2/\epsilon)$ 時間で走る [22]．

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 3 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 12 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 3 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

3.1.2 Goemans らのアルゴリズム [5]

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 [17] 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．

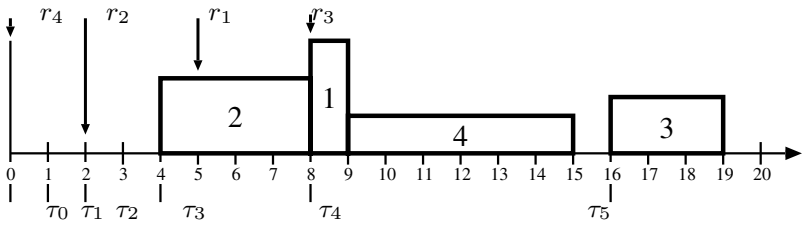

図 12　アルゴリズム Greedy-Interval の出力例

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．

$$\begin{aligned}
\min \quad & \sum_{j \in J} w_j C_j^{LP} & \\
\text{s.t.} \quad & \sum_{t=r_j}^{T} y_{jt} = p_j & (j \in J) \\
& \sum_{j \in J} y_{jt} \leq 1 & (t = 0, \ldots, T) \\
& C_j^{LP} = \frac{p_j}{2} + \frac{1}{p_j} \sum_{t=r_j}^{T} y_{jt} \left(t + \frac{1}{2} \right) & (j \in J) \\
& y_{jt} \geq 0 & (j \in J, t = r_j, \ldots, T)
\end{aligned}$$

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 [4] 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

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

に従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 3 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 13 の上側のようになる．このとき，Random-Alpha の出力は図 13 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 13 の例において，LP 緩和の 3 つ目の制約式に従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.

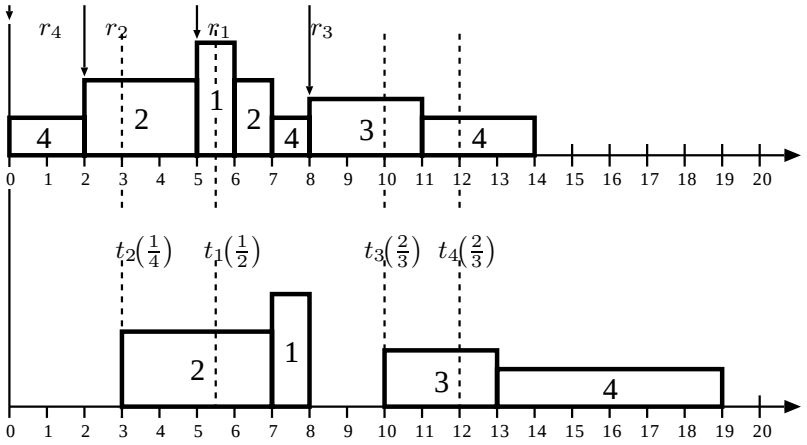

図 13 LP スケジュール (上) とアルゴリズム Random-Alpha の出力例 (下)

3.1.3 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 3 で示した入力に対し，アルゴリズム Delayed-SWPT は図 14 に示すような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 506 である．

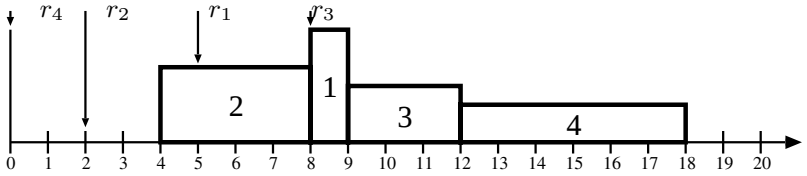

図 14 アルゴリズム Delayed-SWPT の出力例

3.2 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．

3.2.1 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．

3.2.2 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, 3.1.2 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 4 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 15 に示す．

表 4 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

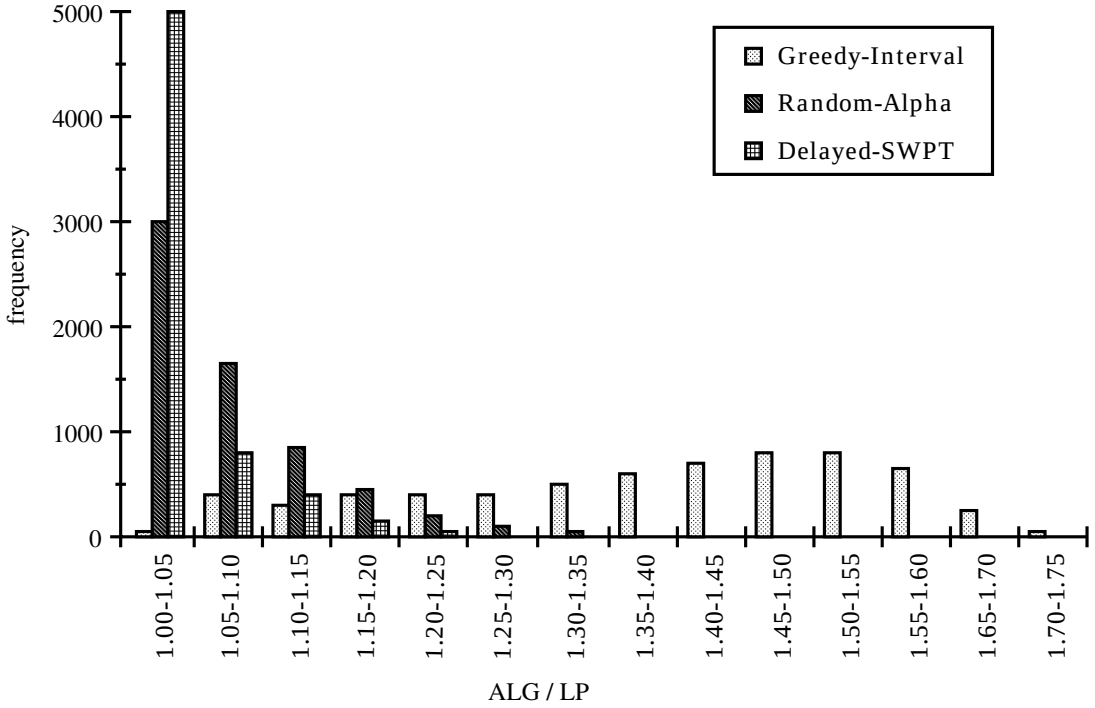

図 15 全 6300 インスタンスに対する ALG/LP の値の分布

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従って 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クス

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

表 5　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

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 5 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．

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．

表 6　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

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 6 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 16 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

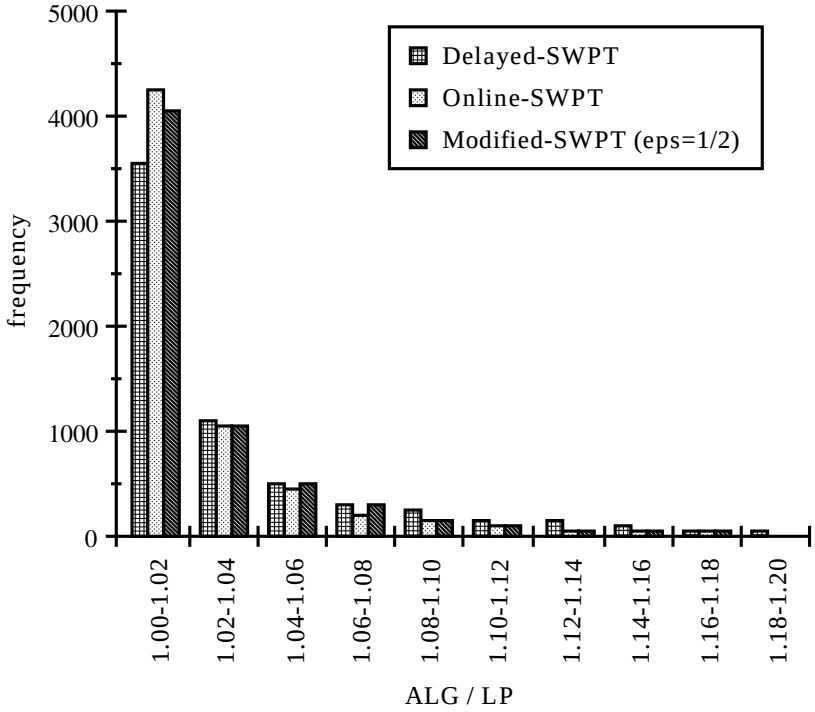

図 16 Online-SWPT，Modified-SWPT と Delayed-SWPT との ALG/LP の値の分布の比較

の平均値と最大値の変化を調べた結果が図 17 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).

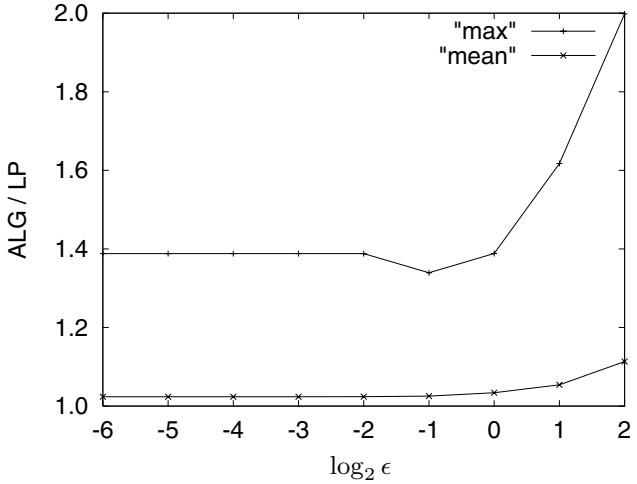

図 17 ϵ の与え方による Modified-SWPT の性能

3.4 計算機実験のまとめ

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．

4 おわりに

施設配置問題とスケジューリング問題に対して理論面から最近提案された代表的アルゴリズムを選び，それらを実装して，その実際的性能評価を与えた．さらに，実際の状況に適したヒューリスティクスを提案し，その有効性を検証した．精度保証付きの近似アルゴリズムやオンラインアルゴリズムの研究は理論的な研究が主体で，得られたアルゴリズムの理論的性能は，実際的性能と異なることもしばしば生じている．実際的性能を伴って初めて実際に有用なアルゴリズムといえるので，理論的なアルゴリズムの実験的性能評価の研究は今後ますます活発に行われるものと考える．

謝辞

本研究は, 一部, 中央大学理工学研究所, 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*, 2002, pp.548–557.

[2] M. Charikar and S. Guha, Improved combinatorial algorithms for facility location and k-median problems, *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, 1999, pp. 378–388.

[3] F.A. Chudak, Improved approximation algorithms for uncapacitated facility location, *Proceedings of the 6th International Integer Programming and Combinatorial Optimization Conference*, 1998, pp. 180–194.

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

[5] 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(2002), 165–192.

[6] 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(1979), pp.287–326.

[7] S. Guha and S. Khuller, Greedy strikes back: Improved facility location algorithms, *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998, pp. 649–657.

[8] 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*, 1996, pp.142–151.

[9] 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(1997), pp.513–544, 1997.

[10] D.S. Hochbaum, Approximation algorithms for the set covering and vertex cover problems, *SIAM Journal on Computing*, 11(1982), pp.555-556.

[11] J.A. Hoogeveen and A.P.A. Vestjens, Optimal on-line algorithms for single-machine scheduling, *Proceedings of the 5th International Integer Programming and Combinatorial Optimization*

Conference, 1996, pp.404–414.

[12] K. Jain, M. Mahdian, and A. Saberi, A new greedy approach for facility location problems, *Proceedings of the 34th ACM Symposium on Theory of Computing*, 2002, pp. 731–740.

[13] K. Jain and V.V. Vazirani, Primal-dual approximation algorithms for metric facility location and k-median problems, *Proceedings of the 40th Symposium on Foundations of Computer Science*, 1999, pp. 2–13.

[14] M. Korupolu, G. Plaxton, and R. Rajaraman, Analysis of a local search heuristic for facility location problems, *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998, pp.1–10.

[15] M. Mahdian, E. Markakis, A. Saberi, and V.V. Vazirani, A greedy facility location algorithm analyzed using dual fitting, *Proceedings of the 5th International Workshop on Randomization and Approximation Techniques in Computer Science*, 2001, pp. 127–137.

[16] M. Mahdian, Y. Ye, and J. Zhang, Improved approximation algorithms for metric facility location problems, http://www.mit.edu/˜mahdian/pub.html, 2002.

[17] C. Phillips, C. Stein, and J. Wein, Minimizing average completion time in the presence of release dates, *Mathematical Programming*, 82(1998), pp.199–223. An Extended abstract appeared under the title "Scheduling jobs that arrive over time" in *Proceedings of the 4th International Workshop on Algorithms and Data Structures*, 1995, pp.86–97.

[18] D.B. Shmoys, E. Tardos, and K.I. Aardal, Approximation algorithms for facility location problems, *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997, pp. 265–274.

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

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

[21] M. Sviridenko, An Improved Approximation algorithm for the metric uncapacitated facility location problems, *Proceedings of the 9th International Integer Programming and Combinatorial Optimization Conference*, 2002, pp. 230–239.

[22] V.V. Vazirani: *Approximation Algorithms*, Springer, 2001 (邦訳: 浅野孝夫: 近似アルゴリズム, シュプリンガー・フェアラーク東京, 2002).