社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－AL－93　(16)
2004／1／30

Reverse-Fit を用いた単純な 2 次元ビンパッキングのアルゴリズム

佐々木 浩高　　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

高さと幅が 1 以下の長方形を辺の長さが 1 の単位正方形に重ならないようにパッキングするとき, 必要な正方形の個数を最小にするという問題が 2 次元ビンパッキング問題である. この問題に対して, 2002 年に発表された Caprara のアルゴリズムは漸近近似率が良くなるように作られたものであり, 現実的な入力においては多くの場合良い結果とならない. 本論文では現実的な入力において良い結果を出せるようなアルゴリズムを提案し, 計算機上に実装して実験的性能の比較評価を行う.

Algorithms with Reverse-Fit for the 2-Dimensional Finite Bin Packing Problem

Hirotaka Sasaki　　Takao Asano

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

Abstract

The 2-dimensional finite bin packing problem is stated as follows : we are given a set of rectangles with height and width at most one and we want to find a minimum number of squares of unit length into which we pack all the rectangles in the set without overlapping. For this problem, Caprara proposed an algorithm with nice asymptotic worst-case ratio in 2002, however, his algorithm seems to perform not so well for practical applications. In this paper, we propose algorithms with good performance for practical applications and, by implementing them and the Caprara's algorithm, we evaluate their experimental performance.

1　はじめに

2 次元ビンパッキング問題 (2-dimensional finite bin packing problem) は n 個の長方形からなる I をインスタンスとする. 長方形 j は幅 $w_j \in (0,1]$ と高さ $h_j \in (0,1]$ をもつ. 長方形の回転は許されない. これらの長方形を重ならないように (辺の長さが 1 の) 単位正方形に詰めていく. このとき, すべての長方形を詰めるのに必要な正方形の個数を最小にする問題が 2 次元ビンパッキング問題である.

この問題に密接に関係する問題として**帯状 2 次元ビンパッキング問題** (2-dimensional strip packing problem) がある. この問題は幅 1 高さが無限大のビンに長方形を詰めていき, すべての長方形を詰めるのに要する高さを最小にする問題である [1].

これらの問題はともに NP-困難であり, 多項式時間で最適解を求めることはできないと信じられている. そこで近似アルゴリズムを用いて近似解を求めることになる. 帯状 2 次元ビンパッキング問題においては [7] によって近似率 2 が達成されている. 一方, 2 次元ビンパッキング問題については, Chung, Garey, Johnson [3] や Caprara [2] によるアルゴリズムがあるものの, **漸近近似率** (asymptotic worst-case ratio) [1]

[1] 最小化問題 P に対するアルゴリズム A の漸近近似率 $R^{\infty}(A)$ は, P のインスタンス全ての集合を $\mathcal{I}_P$ とすれば,

$$R^{\infty}(A) := \lim_{z\to\infty} \sup\left\{ \frac{\mathrm{heur}_A(I)}{\mathrm{OPT}_P(I)} : I \in \mathcal{I}_P, \mathrm{OPT}_P(I) \geq z \right\}$$

として定義される. ここで $\mathrm{OPT}_P(I)$ はインスタンス $I \in \mathcal{I}_P$ に対する最適解の値で $\mathrm{heur}_A(I)$ は I に対するアルゴリズム A の解の値である.

を良くすることを目的としたアルゴリズムであり, 現実的な入力においては多くの場合望ましい結果が得られない.

そこで本論文では, Caprara のアルゴリズムを現実的な入力においても良い結果を出せるように改良すること, また現実的入力においてよい結果を出すことのできる新しいアルゴリズムを提案することを目標としている. そして, それらのアルゴリズムと従来のアルゴリズムを実装し, 計算機実験を通して, 実験的性能の比較評価を行う.

2 Caprara のアルゴリズム [2]

2 次元ビンパッキング問題をそのまま解くのは困難であるので, 単純な問題へと変形する解法を考える. Caprara のアルゴリズムでは 2 次元ビンパッキング問題を**棚作成 2 次元ビンパッキング問題** (2-dimensional shelf bin packing problem) [5] へと変換して解いている. 具体的には, 最初のステージで長方形を横に敷き詰めた棚 (図 1) を作成して, 次のステージでできあがった棚の高さをインスタンスとする (1 次元の) ビンパッキング問題を解いて長方形を正方形に収める (図 2).

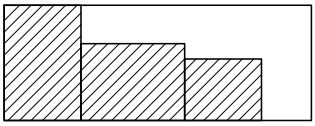


図 1: 棚

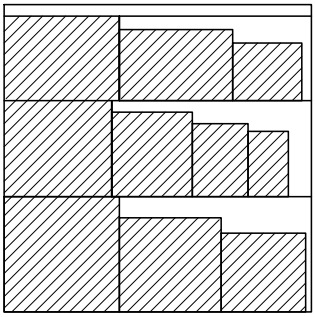


図 2: 棚を正方形に収めたもの

棚の作成に図 3 の調和数列 $1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{k}$ に基づく**幅 k 項調和数列高さ降順アルゴリズム** (Harmonic Decreasing Height$_k$ Algorithm) を, 作成された棚を正方形に収めるのに図 4 の**降順修正 First Fit アルゴリズム** (Modified First Fit Decreasing Algorithm) [4] を用いていて, k を無限大に近づけることにより 2.001... の漸近近似率を達成している[2].

以下, この 2 つのアルゴリズムについて説明する. なお, 本論文での 2 次元ビンパッキングのアルゴリズムでは, n 個の長方形の幅と高さ (長方形 j の幅 w_j と高さ h_j) からなるインスタンス I が入力として与えられるものとしている.

幅 k 項調和数列高さ降順アルゴリズムは, まず 1 次元のビンパッキング近似アルゴリズムである k 項調和数列アルゴリズム (Harmonic$_k$ Algorithm) [6] に基づいて, 長方形 j をその幅 w_j によってタイプ分けを行い, タイプごとに長方形を高さ順にソートし, その順番に従って棚に詰めていく.

1. (タイプ分け)
   for $j := 1$ **to** n **do**:
   $w_j \in (\frac{1}{q+1}, \frac{1}{q}]$ $(q = 1, \ldots, k-1)$ ならば長方形 j はタイプ q とし, $w_j \in (0, \frac{1}{k}]$ ならば長方形 j はタイプ k とする.
2. (タイプごとの棚作成)
   for $q := 1$ **to** k **do**:
   タイプ q の長方形すべてを高さ順に並べ, その順番で棚を作りながら棚に詰めていく. すなわち, 詰めようとしている長方形が棚に収まらなくなったら新しい棚を作って開封し, そこに詰める (タイプ q の長方形は, 1 つの棚に q 個詰められることとなる).

図 3: 幅 k 項調和数列高さ降順アルゴリズム

降順修正 First Fit アルゴリズムも 1 次元ビンパッキング近似アルゴリズムの 1 つであり降順 First Fit アルゴリズムを少し修正したものである.

3 Caprara のアルゴリズムの改良

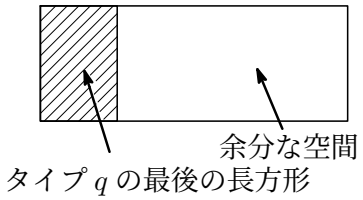
Caprara のアルゴリズムは, 最適解の値が無限大に近づくようなインスタンスの (つまり, 長方形の総面積が無限大に近づく) ときには満足する解を得ることができるが, 一般的な入力においてはかなり悪い解となってしまう. 主な理由として以下の 2 点があげられる.

[2] [2] では特に言及されていないが $\mathrm{OPT}_P(I) >> k$ となるようなインスタンス I のときのみ上記の漸近近似率を達成する.

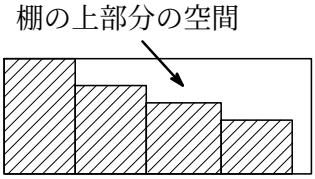
1. 各アイテムを降順に並び替える.
2. サイズが $\frac{1}{2}$ より大きいアイテムをビンに詰める.
3. $(\frac{1}{6}, \frac{1}{3}]$ のサイズのアイテムで最も小さいサイズのアイテムと 2 番目に小さいアイテムが 2. で作られたいずれかのビンに同時に収めることができるかを調べる.
   - 可能な場合, 最も小さいアイテムと入りうる最大のサイズのアイテムを同時に収め 3. に戻る.
   - 不可能な場合, あるいはこの範囲のアイテムが 2 個未満となったら 4. へ進む.
4. 残ったアイテムを First Fit アルゴリズムに従って詰めていく.

図 4: 降順修正 First Fit アルゴリズム

(a) タイプの異なる長方形は異なる棚へと詰められるため棚に余分な空間ができるときがある.

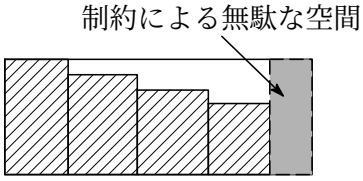


(b) 長方形を横に並べるだけで縦には積まずに棚を構成するため, 棚の上部分に無駄な空間ができてしまう.



これら余分な空間を利用することにより一般的な入力においても極端に悪い結果とならないように改善する.

空間の利用方法として以下の 3 つの方法を提案する. 3 つの提案手法はどれも, (a) タイプの異なるときの余分な空間, (b) 棚の上部分の空間, さらに, (c)“タイプ q の長方形は棚に q 個までしか収められない” という制約による無駄な空間部分, を区別せず同等の空間として扱う.



いずれの方法でもインスタンス I の n 個の長方形は幅 w_j により k 個のタイプへと分類されていると仮定する.

方法 1

幅 k 項調和数列高さ降順アルゴリズムに従って構築される棚の空間部分から, 面積が最大となるような長方形を取り出し, その部分を**補助棚**とする. 入力の長方形は First Fit アルゴリズムに従い, 補助棚にも長方形を詰めていく (図 5). 補助棚にはタイプの異なる長方形が混在してもかまわない. 詳細は図 6 のように書ける.

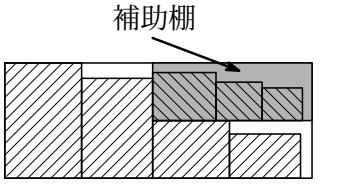


図 5: 方法 1 の補助棚

for $q := 1$ **to** k **do**:

1. タイプ q のすべての長方形を高さ順にソートし, $R_1^{(q)}, R_2^{(q)}, \ldots, R_q^{(q)}$ とする.
2. **for** $l := 1$ **to** n_q **do**:

   長方形 $R_l^{(q)}$ がいずれかの補助棚に収められるか調べる. 収まる場合, $R_l^{(q)}$ を First Fit アルゴリズムで補助棚に収める. 収まらない場合, あるいは補助棚が存在しない場合は, タイプ q の棚でタイプ q の長方形が 1 個以上で $q-1$ 個以下のものがあれば, $R_l^{(q)}$ をその棚に詰め込み, そのような棚がなければ $R_l^{(q)}$ の高さ $h_l^{(q)}$ の棚を新しく作りその棚に $R_l^{(q)}$ を詰め込む. さらにこのとき, タイプ q の長方形が q 個詰め込まれた棚が生じたとき, あるいは $l = n_q$ のときは, その棚の空間部分から面積が最大となるように長方形を取り出しそれを補助棚とする.

図 6: 方法 1

方法 2

幅 k 項調和数列高さ降順アルゴリズムに従って構築されるタイプ q の長方形が詰められた棚を幅 $1/q$ ずつに分け, それぞれを一つのビンとする. 入力の長方形は First Fit アルゴリズムに従い, ビン上部の空間にも長方形を詰めていく (図 7). ビンにはタイプの異なる長方形が混在してもかまわない. 方法 1 と方法 2 の違いは, 方法 1 の補助棚の役割を方法 2 ではビンが果たす点のみであることに注意すれば, 詳細は図 8 のように書ける.

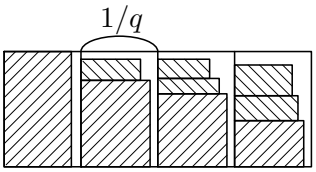


図 7: 方法 2 のビン

for $q := 1$ **to** k **do**:

1. タイプ q のすべての長方形を高さ順にソートし, $R_1^{(q)}, R_2^{(q)}, \ldots, R_q^{(q)}$ とする.
2. **for** $l := 1$ **to** n_q **do**:
   長方形 $R_l^{(q)}$ がいずれかのビンに収められるか調べる. 収まる場合, $R_l^{(q)}$ を First Fit アルゴリズムでビンに収める. 収まらない場合, あるいはビンが存在しない場合は, タイプ q の棚でタイプ q の長方形が 1 個以上で $q-1$ 個以下のものがあれば, $R_l^{(q)}$ をその棚に詰め込み, そのような棚がなければ $R_l^{(q)}$ の高さ $h_l^{(q)}$ の棚を新しく作りその棚に $R_l^{(q)}$ を詰め込む. さらにこのとき, タイプ q の長方形が q 個詰め込まれた棚が生じたとき, あるいは $l = n_q$ のときは, その棚を $1/q$ ずつに切り分けそれらを新しいビンとする.

図 8: 方法 2

方法 3

幅 k 項調和数列高さ降順アルゴリズムに従って構築されるタイプ q の長方形が詰められた棚を幅 $1/q$ ずつに分けた後, 空間部分から面積が最大となるような長方形を取りだし補助棚とする. さらに $1/q$ ずつに切り分けられて補助棚が含まれない部分をビンとする. 入力の長方形は First Fit アルゴリズムに従い, 補助棚およびビン上部の空間にも長方形を詰めていく (図 9). 補助棚とビンにはタイプの異なる長方形が混在してもかまわない. 方法 3 が方法 1, 方法 2 と異なる点は補助棚およびビンの両方を用いるということのみであるので, 詳細は図 10 のように書ける.

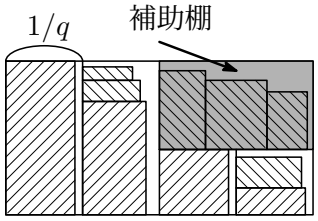


図 9: 方法 3 の補助棚とビン

for $q := 1$ **to** k **do**:

1. タイプ q のすべての長方形を高さ順にソートし, $R_1^{(q)}, R_2^{(q)}, \ldots, R_q^{(q)}$ とする.
2. **for** $l := 1$ **to** n_q **do**:
   長方形 $R_l^{(q)}$ がいずれかの補助棚またはビンに収められるか調べる. 収まる場合, $R_l^{(q)}$ を First Fit アルゴリズムで補助棚またはビンに収める. 収まらない場合, あるいは補助棚もビンも存在しない場合は, タイプ q の棚でタイプ q の長方形が 1 個以上で $q-1$ 個以下のものがあれば, $R_l^{(q)}$ をその棚に詰め込み, そのような棚がなければ $R_l^{(q)}$ の高さ $h_l^{(q)}$ の棚を新しく作りその棚に $R_l^{(q)}$ を詰め込む. さらにこのとき, タイプ q の長方形が q 個詰め込まれた棚が生じたとき, あるいは $l = n_q$ のときは, その棚を $1/q$ ずつに切り分け補助棚とビンを作成する.

図 10: 方法 3

4 Reverse-Fit を用いたアルゴリズム

この節では, 帯状 2 次元ビンパッキング問題を解く近似率 2 のアルゴリズムと, Reverse-Fit アルゴリズム [7] の考え方を用いた, 棚作成の新しいアルゴリズム (図 11, 以下, ASRF と表記) について述べる. さらにこのアルゴリズムによって作成された棚を降順修正 First Fit アルゴリズムを用いて正方形にパッキングし, 2 次元ビンパッキング問題の解を得る.

1. 幅 1/2 以上の各長方形 j に対して, 高さ h_j の棚を作成し長方形 j をその棚に詰め込む.
2. 幅 1/2 未満の長方形を高さの高い順番にソートし, 順番に従って 1. で作られた棚に First Fit アルゴリズムで収める.
3. 残った長方形を高さの順に底辺をそろえ, 幅 1 の長さをこえないように左からそろえていく.
4. 続いて, 右から上辺をそろえて幅が 1/2 をこえるまで長方形をそろえていく.
5. 4. を上辺をそろえたまま, 3. の上部に乗せる (図 12). そして, 4. の上辺部分をラインとする.
6. この時点で高さが 1 を超えた場合, 4. で用いた長方形すべてを保留し, 4. へ戻る.
7. もし 4. と 3. の接触した辺が 4. の左端の長方形のものならば, 左端の長方形以外の長方形を上辺をそろえたままいずれかの辺が接触するまで下におろし, その上辺をラインとする (図 13).
8. 3. の左端の長方形の上辺がラインよりも高く, かつその差が次に詰める長方形の高さより大きいかを調べる.
   - 条件を満たすとき, ラインの高さに沿って左詰めで長方形が入らなくなるまで詰めていく (図 14). ここで詰めた左端の長方形の上辺を新たなラインとし, 8. へ戻る.
   - 条件を満たさないとき, ここまで構築したものを新たな棚とする. 残った長方形に 6. で保留した長方形を加え 3. に戻る.

図 11: Rivers-Fit を用いた棚作成アルゴリズム ASRF

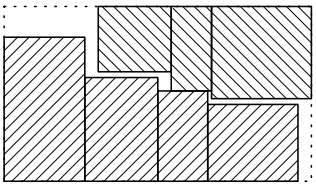


図 12: 3. と 4. を組み合わせたところ

5 計算機実験 1

Caprara のアルゴリズムとその 3 つの改良方法, ASRF および高さ降順 First Fit アルゴリズムを用いて棚を作成し, 降順修正 First Fit アルゴリズムによって正方形へと詰めるアルゴリズム (以下, FFDH と表記) の 6 つのアルゴリズムで計算機実験を行い, それぞれのアルゴリズムの性能評価を行った.

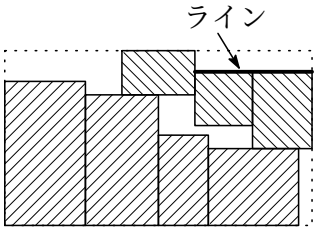


図 13: 4. 左端の長方形が 3. と接触したとき

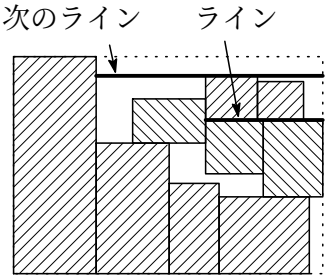


図 14: ラインの高さに沿って長方形を詰める

高さ降順 First Fit アルゴリズム

1. 各長方形を高さが降順になるように並び替える.
2. この順番に従って First Fit アルゴリズムを適用し, 棚を構成していく.

下界は, 入力の長方形の総面積を整数に切り上げたものを用いた. 入力する長方形の幅と高さは以下のような乱数で発生させた.

- $(0, 1]$ の一様分布.
- 平均 $\mu = 0.5$ 分散 $\sigma = 1/6$ の正規乱数. ただし, 0 以下, または 1 より大きい数が出たときは再び乱数を発生させる.

インスタンスのサイズである長方形の数 n は, 100, 500, 1000 の 3 通りを用いた. これらを 10 回ずつ試行しその平均を求めることで最終的な結果とした. なお, Caprara のアルゴリズムにおいて 2.001... の漸近近似率を達成するには k を無限大に近づける必要があるが今回の実験ではインスタンス数が有限であるため, 十分に大きな数 (500) を用いて実験を行った.

$(0,1]$ の一様分布

$(0,1]$ の一様分布のときの結果を図 15 に示す. また, それぞれのアルゴリズムの結果が下界の何倍になったかを表 1 に示す.

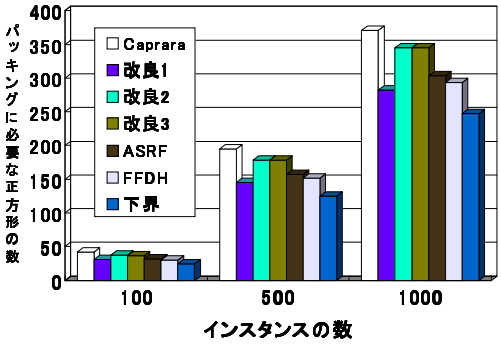


図 15: $(0,1]$ の一様分布

表 1: アルゴリズムの解/下界 ($(0,1]$ の一様分布のとき)

アルゴリズム	100	500	1000
Caprara	1.69	1.56	1.50
改良 1	1.24	1.16	1.14
改良 2	1.50	1.42	1.40
改良 3	1.46	1.42	1.40
ASRF	1.30	1.26	1.23
FDDH	1.23	1.21	1.19

Caprara のアルゴリズムの改良は改良 1 が 2 割程度, 改良 2 と 3 が 1 割程度改善した. FFDH は単純なアルゴリズムでありながら手の込んだアルゴリズムである ASRF の結果を上回る結果となった. インスタンスのサイズ増減による大きな変化はどのアルゴリズムでも見られない.

平均 $\mu = 0.5$ 分散 $\sigma = 1/6$ の正規乱数

結果及び下界の何倍になったかをそれぞれ図 16 と表 2 に示す.

Caprara のアルゴリズムの改良 1 は一様分布のときとほぼ変わらず 2 割程度の改善が見られる. 比して, 改良 2 と 3 はほとんど改善が見られない結果となった. 改良 2 と 3 はタイプ q の棚は $1/q$ ずつ切り分けられそれぞれをひとつのビンとして扱う. その際幅が $1/q$ より大幅に小さい長方形を収めたとき, 大きな空間ができてしまう. 特に, 正規乱数では幅の大きさが 0.5 付近の長方形が集中して入力されるため幅が 0.5 をわずかに超えた長方形が多く入力されることとなり改良 2 と 3 の結果が悪くなったと考えられる.

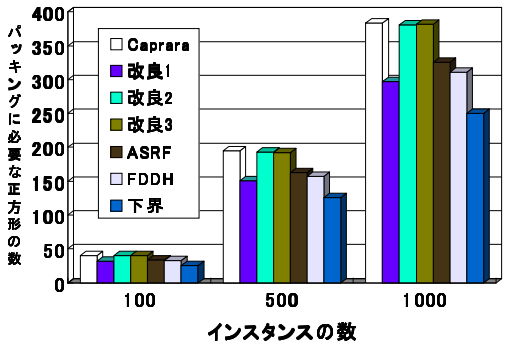


図 16: 平均 $\mu = 0.5$ 分散 $\sigma = 1/6$ の正規乱数

表 2: アルゴリズムの解/下界 (平均 $\mu = 0.5$ 分散 $\sigma = 1/6$ の正規乱数のとき)

アルゴリズム	100	500	1000
Caprara	1.61	1.55	1.53
改良 1	1.27	1.20	1.19
改良 2	1.58	1.54	1.52
改良 3	1.58	1.53	1.52
ASRF	1.35	1.30	1.30
FDDH	1.29	1.25	1.24

ASRF はこちらでも奮わない結果となった. インスタンス数がどのときでも FDDH にかなわない結果となってしまっている.

計算機実験 1 のまとめ

Caprara のアルゴリズムの改良は 3 通りとも “無駄なスペースの利用” を考えたものだが改良 2 と 3 では, ビンの作成の際に改良 1 には生じい無駄なスペースができたために改良 1 に劣る結果となった. この点をさらに改善すれば改良 1 なみの結果にはなると考えられる.

それ以上の改善には,

- より有効な空間の利用法を考える.
- 無駄な空間を一ヶ所に集め, より大きな空間を形成する.

といった工夫が必要になってくると考える.

ASRF に関しては一様分布, 正規乱数どちらの入力に対しても満足する結果を得ることができなかった. そこで, 次節にて改善案を示す.

6 ASRF の改善

2 次元ビンパッキング問題を棚作成とそのパッキングという 2 つのステージに分けて行う場合, 良い結果を収めるには棚が作成されるときに以下の 2 つを満たすことが重要と考えられる.

- すべての棚の高さの合計ができるだけ小さい.
- 個々の棚の高さができるだけ小さい.

ASRF は 1 つ目の条件はある程度満たしているが, 互い違いに並べた 2 段の長方形を組み合わせて棚を作成するために 2 つ目の条件を満たさないことが多い. そこで個々の棚の高さに制限を加えることでこの問題を解決する.

作られた棚をビンに詰めるアルゴリズムに k 項調和数列アルゴリズムを用いることで制限する棚の高さを求めた. k 項調和数列アルゴリズムではアイテム i のサイズ s_i が $1/(q+1) < s_i \leq 1/q$ のときタイプ q に分類されパッキングされることとなるが, このときサイズが $1/q$ に近いほど無駄なスペースができなくなる (図 17).

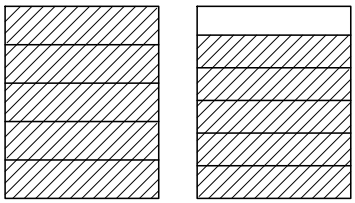


図 17: サイズの大きさとスペース

そこで ASRF の手順 3. の時点での高さ h がタイプ q に属すとき, ASRF 適用後の高さを $1/q$ までに制限することにする. つまり手順 6. を以下のように書き換える.

6. 3. の左端の長方形の高さ h が $1/(q+1) < h \leq 1/q$ のとき, この時点で高さが $1/q$ を超えた場合, 4. で用いた長方形すべてを保留し, 4. へ戻る.

さらに, 高さが $1/q$ に達するまでなるべく長方形を詰めこむようにするため手順 7. と 8. も以下のように変更する.

7. もし 4. と 3. の接触した辺が 4. の左端の長方形のものならば, 左端の長方形以外の長方形を上辺をそろえたままいずれかの辺が接触するまで下におろし, その上辺あるいは 3. の左端の長方形の上辺のうち高さの低いほうをライン, 高さの高いほうを予備のラインとする.
8. ($1/q$ − ラインの高さ) 以下の高さを持つまだ詰められていない長方形があるかどうかを調べる.
   - 条件を満たす長方形がある限り, ラインの高さに沿って左詰めで長方形が入らなくなるまで詰めていく. ここで詰めた左端の長方形の上辺と, 予備のラインのうち高さの低いほうをライン, 高いほうを予備のラインとし 8. へ戻る.
   - 条件を満たす長方形がないとき, ここまで構築したものを新たな棚とする. 残った長方形に 6. で保留した長方形を加え 3. に戻る.

また k 項調和数列アルゴリズムも以下のように変更する.

k 項調和数列アルゴリズム (変更)

n 個のアイテム (アイテム i のサイズ s_i) からなるインスタンス I が入力として与えられる.

1. (タイプ分け)
   for $i := 1$ **to** n **do**:
   $s_i \in (\frac{1}{q+1}, \frac{1}{q}]$ $(q = 1, \ldots, k-1)$ ならばアイテム i はタイプ q とし, $s_i \in (0, \frac{1}{k}]$ ならばアイテム i はタイプ k とする.
2. (タイプごとのビン作成)
   for $q := 1$ **to** k **do**:
   まだ詰められていないタイプ q のアイテムが q 個未満あるいは $q = k$ であって, かつ封のないビンが存在するならば, First Fit アルゴリズムでそれらのビンにアイテムを収める. アイテムが収まらないときは新たなビンを作成しそこに詰めてビンは封をしない. そうでないときは新たなビンを作成しアイテムを q 個まとめてそこに収め, ビンには封をする.

7 計算機実験 2

この節では改良した ASRF (以下, 改良 ASRF と表記) と改良前の ASRF の結果の違いを調べる. さらに比較参考として, FFDH に Caprara のアルゴリズムの改良 1 を施したもの (以下, 改良 FFDH と表記) を加え実験を行う. 下界と入力は計算機実験 1 と同様のものを用いた.

$(0,1]$ の一様分布

結果をそれぞれ図 18 と表 3 に示す．改良 ASRF が ASRF に比べ大きく改善された．改良 FFDH よりもわずかながらではあるが良い結果となっている．

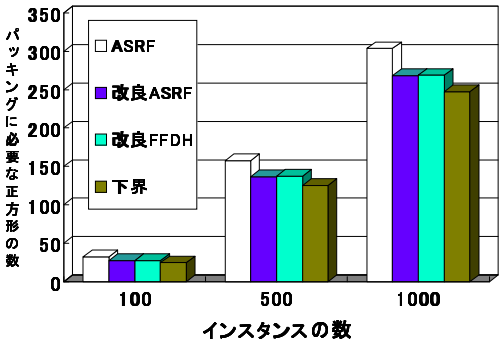


図 18: $(0,1]$ の一様分布 2

表 3: アルゴリズムの解/下界 2 ($(0,1]$ の一様分布のとき)

アルゴリズム	100	500	1000
ASRF	1.30	1.26	1.23
改良 ASRF	1.10	1.09	1.08
改良 FFDH	1.10	1.10	1.09

平均 $\mu = 0.5$ 分散 $\sigma = 1/6$ の正規乱数

結果及び下界の何倍になったかをそれぞれ図 19 と表 4 に示す．こちらも結果が大きく改善したのがわかる．

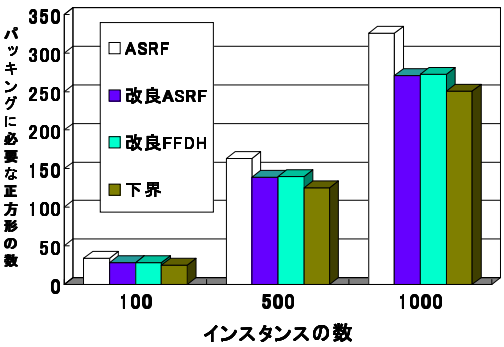


図 19: 平均 $\mu = 0.5$ 分散 $\sigma = 1/6$ の正規乱数 2

表 4: アルゴリズムの解/下界 2 (平均 $\mu = 0.5$ 分散 $\sigma = 1/6$ の正規乱数のとき)

アルゴリズム	100	500	1000
ASRF	1.35	1.30	1.30
改良 ASRF	1.14	1.11	1.08
改良 FFDH	1.14	1.11	1.09

計算機実験 2 のまとめ

一様分布, 正規乱数どちらの入力においても改良 ASRF が ASRF から大きく改善された．誤差程度ではあるが改良 FDDH よりもよい結果を収めた．

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, 文部科学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] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest, “Orthogonal packing in two dimensions”, *SIAM Journal on Computing*, 9, pp. 846–855, 1980.

[2] A. Caprara, “Packing 2-Dimensional Bins in Harmony”, *Proceedings of the 43-rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002), IEEE Computer Society Press*, pp. 490–499, 2002.

[3] F. R. K. Chung, M. R. Garey, and D. S. Johnson, “On packing two-dimensional bins”, *SIAM Journal on Algebraic and Discrete Methods*, 3, pp. 66–76, 1982.

[4] M. R. Garey and D. S. Johnson, “A 71/60 theorem for bin packing”, *Journal of Complexity*, 1, pp. 65–106, 1985.

[5] P. C. Gilmore and R. E. Gomory, “Multistage cutting problem of two and more dimensions”, *Operations Research*, 13, pp. 94–119, 1965.

[6] C. C. Lee and D. T. Lee, “A simple on-line bin packing algorithm”, *Journal of the ACM*, 32, pp. 562–572, 1985.

[7] I. Schiermeyer, “Reverse-fit: A 2-Optimal algorithm for packing rectangles”, *Lecture Notes in Computer Science*, 855, pp. 291–299, 1994.