アルゴリズム 69－5
（1999．9．2）

ビン・パッキング問題における局所探索法と構築法との関係

恐神 貴行*†　　岡野 裕之‡
日本IBM 東京基礎研究所§

実問題に現れるビン・パッキング問題 (BPP) に応用することを目的に，各種構築法と局所探索法の組み合わせについて，解の質と探索数のトレードオフに着目して比較する．既存のものを含む7種類の構築法に，ベスト改善，ファースト改善，および本稿で提案する優先改善の3つの局所探索法を適用した．計算実験の結果，BPPに対してはアイテム数のバランスした初期解から始め，ランダムな順序で探索するファースト改善が優れているが，制約の付いたBPPに対しては，優先改善が優れていることが明らかになった．

Local Search Algorithms for the Bin Packing Problem and Their Relationships to Various Construction Heuristics

Takayuki Osogami*†　　Hiroyuki Okano‡
IBM Japan, Tokyo Research Laboratory§

Various combinations of construction and local search algorithms for the bin packing problem (BPP) are compared, and the tradeoff between the speed and quality of the solutions is discussed. Their performance for the constrained BPP, of which instances occur in the real world, is also discussed. Three local search algorithms — the best improve, the first improve, and the prioritized improve proposed in this report — were applied to initial solutions obtained by means of seven construction heuristics. Computational experiments show that the first improve that searches in random order is superior to the others for the BPP when it starts with initial solutions that have a balanced number of items in each bin. It is also shown for the constrained BPP that the prioritized improve is superior to the others.

1　はじめに

ビン・パッキング問題 (BPP: bin packing problem) は NP-hard に属する組み合わせ最適化問題として知られ [1]，容量 C の容器にサイズが整数 $s(u_i) \in [0, C]$ のアイテムの有限集合 $U = \{u_1, u_2, ..., u_n\}$ を容器数が最小となるように詰め込む問題である．

BPPに対する研究の多くは構築法の設計とその理論的な性能の解析に対してなされてきた [2]．代表的な構築法に，オンライン・アルゴリズムでは first fit (FF)，オフライン・アルゴリズムでは first fit decreasing (FFD) がある．FFは容器を左から右に並べておき，アイテムを与えられた順に受け入れられる最も左の容器に入れていく．またFFDはアイテムをサイズの大きい順にFFと同様に入れていく．ところで，BPPに対するアルゴリズム A の最悪の場合の性能の評価には漸近的性能比 R_A^∞ が用いられ [2]，FFやFFDは $R_{FF}^\infty = \frac{17}{10}$，$R_{FFD}^\infty = \frac{11}{9}$ であることが知られている [3]．また $R_A^\infty = 1$ の多項式時間オフライン・アルゴリズムが知られており，例えばKarmarkerとKarpによる $O(n^8)$ 時間アル

*E-mail: osogami@jp.ibm.com
†東京大学理学系研究科情報科学専攻今井研究室受託研究員
‡E-mail: okanoh@jp.ibm.com
§〒242-8502 神奈川県大和市下鶴間 1623-14

ゴリズムがある [4].

このように BPP に対しては効率的で性能の良い構築法が提案されてきた. これに対して筆者らは, BPP にモデル化できる実問題に対して効率の良い近似解法を求めることを目指している. その一つとして, BPP のヒューリスティックに基づいた手法を研究中である. 以下では実問題に現れる BPP にモデル化できる問題を実問題 BPP, 純粋な BPP をエレメンタリー BPP と呼ぶ. 筆者らと似たアプローチとして Falkenauer はエレメンタリー BPP をグループ分け問題の一つとして捉え, 一般のグループ分け問題に対して適用可能な遺伝的アルゴリズム (GA: genetic algorithm) を提案し, エレメンタリー BPP に対して適用している [5].

BPP にモデル化できる実問題として 2 つ例を挙げる. ある種の配送経路最適化問題 (VRP: vehicle routing problems) の場合, 配送拠点から出発して配送先を巡回し, 元の拠点に戻るような車両への配送先の割り当てと配送経路の生成を, 使用車両数が最少になるように行うことが求められる. またこのとき, 車両数が同じであれば経路長は短い方が良いとする. このような問題に対しては車両を容器, 荷物をアイテムとみて BPP を解けば良いが, このとき目的関数となるのは車両数に加えて各トラックの経路長があり, さらに積載可能なアイテムの組み合わせ等の制約を満たすことが求められる. 各車両の経路長は行商人問題 (TSP: traveling salesman problem) に対する近似アルゴリズムにより求められる. もう一つの例として, ある種の 2 次元や 3 次元の詰め込み問題 [6] が挙げられる. 特にアイテムの向きや相互の位置関係等に制約がある場合は, 最密な詰め込み方を時間をかけて求める以前に, 同じ容器に入れるべきアイテム同士にうまくグループ分けすることが重要である. その際, 詰め込み可能かを判定するフィージビリティ・チェックでは, 実問題の各種制約が考慮される.

これらの実問題 BPP ではエレメンタリー BPP のように効率的な構築法が得られないことが多い. したがって, 構築法で得られた解を局所探索法等により改善する必要がある. エレメンタリー BPP に対して局所探索法を適用した例としてはこれまでに, Kämpke による焼き鈍し法 (SA: simulated annealing) [7], Falkenauer による GA [5], Reeves による GA [8] がある. Kämpke は目標とする値に容器数を固定して, アイテムの交換や移動により各容器の充填率を釣り合わせるという手法を採っている. また Falkenauer はアイテムを詰めた容器の列に対して各遺伝操作を施し, 交配や突然変異の中でさらに局所探索法を適用している. そして Reeves はアイテムの列に対して各遺伝操作を施し, FFD 等の構築法により解を構成して解の質を評価している.

ところで実問題 BPP においてはフィージビリティ・チェックや目的関数の計算に, エレメンタリー BPP に比べて, 多くの計算を必要とする場合が多い. 例えば上で述べた VRP の変種を実問題 BPP として扱うとき, 車両の経路長は TSP の近似アルゴリズムにより求める. L_2 距離の TSP については, 都市数を N として一様分布データに対して実験的に $O(N \log N)$ 時間の効率的な近似アルゴリズムが知られている [9] が, 局所探索法における各近傍の評価に用いるには無視できない計算量である. したがって実問題 BPP では特に解の質と計算量とのトレードオフを考えなければならない. 解の質と計算量とのトレードオフに対する考察は, TSP における構築法と局所探索法の組み合わせについて比較した例がある [10].

そこで本研究では, BPP に対する各種の初期解の構築法と局所探索法の組み合わせについて, 得られる解と探索数を調べ, 解の質と計算量のトレードオフを考えた構築法と局所探索法の組み合わせを考える. また, 局所探索法が良い局所最適解に到達するために必要な初期解の性質を考察する. まず 2 章で以降の計算実験に用いる構築法と局所探索法を示す. 3 章ではエレメンタリー BPP に対して, 4 章では実問題 BPP の一例に対して, 各種構築法と局所探索法を適用した結果を示し, 5 章で全体をまとめる.

2 BPP に対する局所探索法

本章では 3 章以降の計算実験で用いる局所探索法について述べる. 局所探索法はある初期解を構築法により構成し, 定義された近傍の中で目的関数を改善する近傍へ次々と移り, 局所最適解を得る手法

である．局所探索法の性能は，解空間，近傍，目的関数，探索法 (目的関数を改善する近傍が複数ある場合にどの近傍に移るか)，そして初期解の構築法等により決まる．以下ではこれらを定義する．

2.1 解空間と目的関数

解空間は実行可能解の集合と定義する場合と，それに実行不可能解を含める場合とがある．Falkenauer [5] と Reeves [8] は前者，Kämpke [7] は後者を採用している．実行可能解だけでは探索空間が狭く妥当な近傍が定義しにくい時には実行不可能解も含める方が良い場合があるが，その場合は最終的に解が実行可能であることを保証するための工夫が必要である．本研究では解空間は実行可能解の集合と定義する．

エレメンタリー BPP に対する目的関数は，必要な容器数と定義するのが自然であるが，このような目的関数に対して有効な近傍を定義することは難しい．つまりほとんどの場合に容器数を減らす近傍操作が存在せず，局所探索法がうまく働かない．そこで Kämpke は目標とする値に容器数を固定して，全体の充填率を釣り合わせることを目的関数としている．この方法は解空間に実行不可能解を含める場合は良いが，実行可能解の集合を解空間とする場合にはうまくいかない．そこで Falkenauer は次の関数 f の最大化を目的関数とすることを提案している．

$$f = \frac{\sum_{i=1}^{N} F_i^k}{N} \qquad (1)$$

ここで N は容器数，F_i は容器 i にあるアイテムのサイズの総和である．$k = 1$ のときは，右辺の分子はアイテムのサイズの総和となり一定であるから，式 (1) は容器数の最小化と同義である．$k > 1$ のときには，容器数が同じ場合には満杯や空に近い容器が多いほど f の値は大きくなる．k の値が 1 から大きくなるほど満杯や空に近い容器をより極端に好むことになる．式 (1) は $k > 2$ では容器数が多くても満杯や空に近い容器が多い解の方が，容器数が少ない解よりも良いという場合が生じる．Falkenauer は以上のことと経験的なことから $k = 2$ が最も良いとしており，Reeves も $k = 2$ とした式 (1) を目的関数として用いている [8]．そこで、本稿でも式 (1) で $k = 2$ としたものを目的関数とする．

2.2 近傍

Kämpke の近傍操作は充填率が最大の容器を含む 2 つの容器をランダムに選択し，さらにそれぞれの容器からアイテムを 1 つずつランダムに選択して交換する．またこのとき，ある容器の最上部のアイテムの底が充填率が最小の容器の最上部のアイテムの天井よりも上に位置するならば，前者のアイテムを後者の容器に移動して滑らかにする．また Falkenauer の GA において突然変異や交配の中で適用する局所探索法は，容器の外に取り出されたアイテムと，ある別の容器内のアイテムとを交換する．例えば突然変異はランダムに選ばれた数個の容器からアイテムを全て取り出し，取り出されたアイテムのうちの 3 個までとある容器内の 2 個までのアイテムとを交換する．目的関数が向上するアイテムの組み合わせがなくなるまで繰り返し，そのとき容器内にないアイテムは FFD により詰め直す．本研究では Kämpke の近傍操作を修正した「m-m-交換」を考える．「m-m-交換」は任意の 2 つの容器の間でそれぞれ m 個までのアイテムと m 個までのアイテムを交換する操作と定義する．

2.3 探索法

次に目的関数を改善する近傍が複数ある場合にどの近傍に移るかを定義する．探索の一つの方法は近傍に含まれる全ての解の目的関数値を評価し，その値の最も良い解に移る方法であり，この方法をベスト改善 (BI: best improve) と呼ぶ．Falkenauer の GA において突然変異や交配の中で適用する局所探索法ではベスト改善を採用している．別の方法として現在の解を改善する解が見つかったらすぐにその解に移る方法があり，ファースト改善 (FI: first improve) と呼ぶ．ところでファースト改善の場合，近傍の探索順序を定義する必要があるが，ランダムに探索する方法がしばしば用いられている．Kämpke の SA ではファースト改善を採用し，探索順序はランダムとしている．

本研究ではベスト改善といくつかのファースト改善について 3 章以降で比較する．ベスト改善は任意の 2 つの容器の組み合わせの中で最も目的関数を向上させる「m-m-交換」を適用する．ランダムな探索順序のファースト改善は，ランダムに選択され

た2つの容器の間で解を改善する「m-m-交換」があれば, そのうち最も目的関数を改善する「m-m-交換」を適用する.

2.4 新しい探索法

ファースト改善の変種として探索順序を予め定めておく方法 (優先改善, PI: prioritized improve) を提案する. BPP における優先改善としては充填率の低い容器から順に探索する方法を考える. 充填率の低い容器は中にあるアイテムの量が少ないため近傍操作により空になる確率が高く, また空いているスペースが大きいため他のアイテムを受け入れる確率も高い. したがって充填率の低い容器から探索することにより, 容器数を減らせるような操作を優先的に行える. ここで充填率の低い容器から探索する方法として次の2つが考えられる. 1つは充填率の低いもの同士を優先的に探索する方法 (方式 1) で, もう一つは探索すべき容器の中で最も充填率の低い容器から探索する方法 (方式 2) である:

方式 *1*

```
1  for n = 3 to N × 2 − 1 do          N: 容器数
2     for i = max{n − N, 1} to ⌈n/2 − 1⌉ do
3        j = n − i
4        容器 i と容器 j の間の近傍を全て探索する
```

方式 *2*

```
1  for i = 1 to N −1 do               N: 容器数
2     for j = i + 1 to N do
3        容器 i と容器 j の間の近傍を全て探索する
```

これらの間では実験的に方式 1 がほぼ良い結果を与えるので, 優先改善は方式 1 を指すことにする. ただし, 次章以降の計算機実験において, 方式 1, 2 の性質は同様であったことに注意する. また, 探索順序をランダムとしたファースト改善を単にファースト改善と呼ぶ.

2.5 各探索法の最適性判定計算量

ところで近傍の大きさは探索時間に影響を与える. 目的関数を改善する近傍が存在するか判定するため, 全ての近傍を走査することをその近傍における最適性判定と呼ぶが, この最適性判定の時間計算量は, m をアイテム数, n を近傍操作を適用する時点での容器数として, 「k-k-交換」では $O(\frac{m^{2k}}{n^{2k-2}})$ となる. ただし今考えている近傍操作の場合, ベスト改善においては最初は全ての容器の組み合わせについて探索する必要があるが, 一度近傍操作を適用してからはその必要はない. 近傍操作は2つの容器の間での操作と考えられるが, これら以外の容器間の近傍操作はこれら2つの容器間の近傍操作とは独立であり, 直前に適用された近傍操作に関わっていない容器間の近傍を改めて探索する必要はない. このことを考慮すると実用上は1回の探索の時間計算量は「k-k-交換」では $O(\frac{m^{2k}}{n^{2k-1}})$ となる. ファースト改善, 優先改善も, 既に探索された近傍のうちそれぞれの探索後に適用された近傍操作に関わっていない容器の間には改善する操作はないので改めて探索する必要はない.

2.6 初期解

初期解の構築法として, 1章で述べた FF, FFD の他に次の3つを考える. 1つは next fit (NF) として知られている構築法である. NF はアイテムを与えられた順に蓋の開いている唯一の容器に詰めていくが, その容器に入らない場合はその容器の蓋を閉めてそれ以降はその容器は用いず, 新たに用意した容器に詰めていく. 残りの2つは実験用に新たに考案した構築法で, 1つは容器あたりのアイテム数を k 個までに制限して FFD を適用する C_kFFD, もう1つは容器の容量を予め δ だけ減らしておいて FFD を適用する R_δFFD を考える.

3 エレメンタリー BPP への局所探索法の適用

本章では2章で述べた局所探索法をエレメンタリー BPP に対して適用し, 各種構築法と改善法の組み合わせについて解の質と探索数を計算実験により比較する. その結果から局所探索法が良い局所最適解に到達するための条件を考察する.

3.1 実験の設定

計算実験には OR-Library[†] にあるテスト・インスタンスのうちユニフォーム・クラスのインスタンスを用いる. これらのインスタンスは 20 から 100 の間で一様に分布した整数のサイズのアイテムの集合と容量が 150 の容器とからなる. アイテム数に

[†] http://mscmga.ms.ic.ac.uk/info.html

よって4種類のクラスがあり，それぞれアイテム数は120, 250, 500, 1000である．これら4種類のクラスについて20個ずつのインスタンスが用意されており，以下の議論ではこれら20個のインスタンスについての平均値を用いる．さらにアイテム数が5000, 10000のインスタンスをユニフォーム・クラスと同様な方法でそれぞれ20個ずつ作り，後の議論で用いる．以降ではアイテム数が120, 250, 500, 1000, 5000, 10000のインスタンスをそれぞれu120, u250, u500, u1000, u5000, u10000と書く．

後の議論では計算時間ではなく近傍の探索数すなわち目的関数の評価回数を基準とする．これは計算時間はコンピューターやプログラム技術などアルゴリズムの本質とは関係のない要因に影響されやすいことを考慮したためでもあるが，実問題BPPに対して局所探索法を適用する場合には目的関数の評価やフィージビリティ・チェックにかかる時間は各問題によって大きく異なるが探索数はその局所探索法の性質に依存する割合が大きいと考えられるからである．

まず初期解をFFD, FF, NF, R_5FFD, R_{15}FFD, C_4FFD, C_2FFDで構成し，改善法としてベスト改善，ファースト改善，優先改善のそれぞれを適用した場合の解の質と探索数を調べる．ここで解の質とは，u120, u250, u500, u1000については得られた局所最適解の最適解[11]からの差，u5000, u10000については解の下限からの差とする．解の下限N_{LB}は次式により求める．

$$N_{LB} = \left\lceil \frac{\sum_{i=1}^{N} s_i}{C} \right\rceil \qquad (2)$$

また一度満たされた容器は，目的関数の改善に寄与することはないので，探索の対象から省く．本章の実験においては探索が進むにつれて大部分の容器が満たされるため，満たされた容器を探索の対象から省くことは探索数に非常に大きな影響を与える．

3.2 結果

図1に，それぞれ各構築法とベスト改善，ファースト改善，優先改善を組み合わせたときの探索数と解の質との関係を示す．インスタンスはu1000を用いて，近傍操作としては「1-1-交換」を採用したものである．「2-2-交換」や「3-3-交換」を採用した場合，得られる解の質は「1-1-交換」とほとんど変わらなかったが，探索数は増加し，その増加率はベスト改善の場合に最も大きかった．またその他のインスタンスについても，u1000と同じ傾向を示した．これらのインスタンスについての比較結果は，構築法がFFD, FF, C_2FFD，改善法がベスト改善，ファースト改善，優先改善の場合の解の質と探索数をそれぞれ表1，表2に示す．ただし表1で改善法を「なし」としたものは，構築法による解を表す．

図1よりC_2FFDとファースト改善の組み合わせが得られる解の質が最も良く，かつ収束するまでの探索数も最も少ないことがわかった．ここで最適解からの差が1を下回るには少なくとも半数のインスタンスにおいて最適解まで到達することが必要であるから，1を下回るかどうかは重大な差であることに注意する．よって本章の以下で良い解とは最適解からの差が1を下回ること，悪い解とは1を上回ることを意味するものとする．ファースト改善の場合，C_4FFD, C_5FFDと容器あたりのアイテム数の制限を緩めると，得られる解の質は悪くなり，また収束するまでの探索数が増えていくことも確認した．ファースト改善によって良い局所最適解まで到達するのはC_2FFD, C_3FFDのように容器あたりのアイテム数がバランスした状態から始めたときだけで，その他のどの構築法を用いた場合にも良い解に到達しなかった．

ベスト改善は，ファースト改善とは全く異なる傾向があり，FFやNFで初期解を構築した場合に良い局所最適解まで到達した．またC_nFFDにおいて容器あたりのアイテム数の制限nを大きくしていくと，得られる解の質は悪くなるが，収束するまでの探索数は減っていくことも確認した．

優先改善の場合は，C_2FFD, C_3FFDのように容器あたりのアイテム数がバランスした状態から始めたときに良い解まで到達するファースト改善に見られた性質は保存され，またC_4FFD, C_5FFDと容器あたりのアイテム数の制限を緩めると得られる解の質が悪くなる点も同様であった．しかしアイテム数の制限を緩めたときにも探索数はそれほど変わらなかった．またFFやNFで初期解を構成した場合にも良い解まで到達し，この点ではベスト改善と性質が似ていた．さらにR_5FFDやR_{15}FFDで初期解

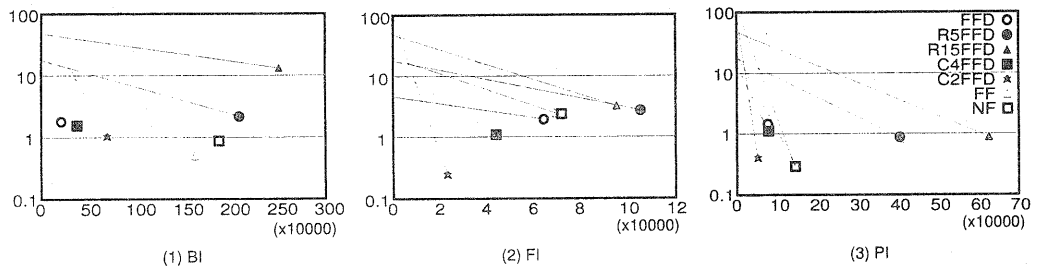

図 1: u1000 に対して各局所探索法を適用したときの, 探索数 (横軸) と得られる解の最適解からの差 (縦軸)

表 1: 各局所探索法により得られる解の最適解または下限との差

構築法	FFD				FF				C_2FFD			
改善法	なし	BI	FI	PI	なし	BI	FI	PI	なし	BI	FI	PI
u120	0.70	0.25	0.35	0.25	3.15	0.15	0.60	0.20	10.95	0.25	0.25	0.30
u250	1.55	0.75	0.75	0.75	6.55	0.50	1.10	0.35	23.40	0.6	0.45	0.55
u500	2.70	1.20	1.35	1.15	11.55	0.50	1.70	0.25	48.80	0.85	0.20	0.45
u1000	4.85	1.85	2.05	1.50	20.95	0.50	2.50	0.30	99.45	1.05	0.25	0.40
u5000	20.9	6.25	5.40	5.75	95.85	1.20	9.95	0.45	500.40	3.90	0.05	0.90
u10000	40.2	12.9	9.65	11.55	187.00	2.15	20.45	0.60	995.30	4.40	0.15	1.70

を構成した場合にも, 探索数は多くなるがある程度良い解まで到達した.

次に構築法に注目すると, FFD により構成した解はどの探索法を用いても良い解まで到達しなかった. これに対して構築法としての性能は FFD に劣る FF や NF から始めたときにはベスト改善や優先改善によって良い解まで到達した. ところが単に初期解を悪く作れば探索法により良い解まで到達するわけではなく, このことは R_δFFD から始めた場合にはどの探索法でも探索数が多く, 得られる解も良くなかったことからわかる. 局所探索法をある初期解に対して適用する場合, C_nFFD のようにアイテム数がバランスしている方が良いと予想される. そこでそれぞれの構築法について得られる解における容器あたりのアイテム数と充填率の平均と標準偏差を調べ, 表 3 に u1000 に対する結果を示す. 表 3 より良い局所最適解に到達しやすい C_nFFD, FF, NF は容器あたりのアイテム数のばらつきが小さく, 良い局所最適解に到達しにくい R_δFFD や FFD はそのばらつきが大きいことがわかる. また充填率やそのばらつきは良い局所最適解への到達のしやすさとはあまり関係がないことがわかる.

表 3: 各構築法による解における容器あたりのアイテム数及び充填率の平均値及び標準偏差

	アイテム数		充填率 (%)	
構築法	平均値	標準偏差	平均値	標準偏差
FFD	2.467	0.125	98.67	2.906
R_5FFD	2.389	0.122	95.54	2.404
R_{15}FFD	2.227	0.096	89.07	2.304
C_4FFD	2.411	0.050	96.42	3.200
C_2FFD	2.000	0.000	80.00	2.386
FF	2.373	0.059	94.90	1.650
NF	1.897	0.032	75.85	1.370

4 実問題 BPP への局所探索法の適用

3 章ではエレメンタリー BPP に対して局所探索法を適用したが, 本章ではそれに制約の付いた実問題 BPP に対して局所探索法を適用し, 3 章同様, 各種構築法と改善法の組み合わせについて, 解の質と探索数の関係を計算実験により調べる. その結果からエレメンタリー BPP と実問題 BPP とでの局所探索法の振る舞いの違いについて考察する.

4.1 実験の設定

本章では実問題 BPP の一例として次のような詰み合わせ制約の付いた問題を考える. 容量 C の容器にサイズが整数 $s(u_i) \in [0, C]$ の白アイテムとサイズが整数 $s(u_i) \in [0, a]$ の黒アイテムの有限集合

表 2: 各局所探索法が収束するまでの近傍の探索数 (単位:1000 回)

構築法	FFD			FF			C_2FFD		
改善法	BI	FI	PI	BI	FI	PI	BI	FI	PI
u120	6.8	4.2	3.0	35.8	7.0	9.6	17.5	5.7	3.7
u250	20.4	12.0	8.4	136.5	16.2	27.4	64.3	11.1	10.9
u500	69.8	32.6	26.2	461.4	34.1	63.3	212.9	16.7	21.6
u1000	212.7	64.0	78.0	1,614.2	69.5	147.8	696.1	23.1	52.4
u5000	4,258.5	462.1	1,691.2	31,867.2	481.0	1,471.4	13,863.9	72.2	825.6
u10000	15,703.5	1,328.0	6,931.7	119,959.8	1,402.7	4,942.0	47,766.3	130.4	3,664.1

$U = \{ u_1, u_2, ..., u_n\}$ を, 各容器内の黒アイテムのサイズの総和が a を超えないようにかつ容器数が最小となるように詰め込む.

本章で用いるインスタンスは, u1000 に対して各アイテムを $\frac{3}{5}$ の確率で白アイテム, $\frac{2}{5}$ の確率で黒アイテムとラベル付けして, さらに容器の容量を 200, すなわち最大のアイテムの 2 倍, としたものである. また黒アイテムのサイズの総和に対する制約 a は容器の容量の半分の 100 とした.

各構築法と探索法は実問題 BPP に対応して修正する必要がある. 各構築法は, 黒アイテムを詰めようとしたときに制約を満たさない場合には次の容器に詰めることを除いては, エレメンタリー BPP に対するものと同じであるとする. また各改善法は, 黒アイテムを含む近傍操作を適用したときに制約を満たさなくなる場合にはその操作は適用しないことを除いては, エレメンタリー BPP に対するものと同じであるとする.

4.2 結果

まず初期解を FFD, FF, NF, R_5FFD, R_{15}FFD, C_4FFD, C_2FFD で構成した場合に, 改善法としてベスト改善, ファースト改善, 優先改善を適用した場合の解の質と探索数を調べる. ここで解の質とは得られた局所最適解の下限からの差とするが, 解の下限 N_{LB} は式 (2) により求める. 図 2 には, それぞれ各構築法とベスト改善, ファースト改善, 優先改善を組み合わせたときの探索数と解の質との関係を示す. 近傍操作としては「1-1-交換」を採用した.

まず構築法による初期解について, エレメンタリー BPP では最も性能の良かった FFD よりも FF の方が良い解が得られ, 今の問題に対しては FF による初期解が最も良い解となった. FF による解の下限からの差は 18 程度であるから, 以降ではその半分である下限からの差が 9 以下の解を良い解と呼ぶ.

エレメンタリー BPP の場合に最も性能の良かった C_nFFD とファースト改善の組み合わせは, ここでは良い結果が得られなかった. またその他の構築法により初期解を構成した場合も, ファースト改善により良い解が得られることはなかった. 今のような制約が付いた場合にはベスト改善や優先改善において良い解が得られ, 特に FF により初期解を構築するのが良かった. 優先改善はベスト改善と傾向が似ているが, 最も良い解が得られる FF を構築法とした場合に, 探索数は $\frac{1}{3}$ 以下に抑えられた.

実問題 BPP の場合, 初期解の持つ性質が局所探索法により得られる解の質に与える影響はエレメンタリー BPP 程大きくないことがあり, 容器数の少ない解を初期解とするのが良いことがわかった. またファースト改善では良い解に到達する以前に収束してしまうが, ベスト改善, 優先改善ではより良い解まで到達することがわかった.

5 おわりに

本研究では, BPP に対する局所探索法を探索数と解の質に注目して計算実験により解析した. 実問題に現れる BPP にモデル化できる問題は, 1 回の目的関数の計算やフィージビリティ・チェックに要する計算量が大きく, 探索数を少なく抑えることが重要である. 計算実験の結果, 各種構築法及び局所探索法が持つ探索数と解の質に関するトレードオフが明らかになり, 実問題 BPP の解法を設計する上で有用な知見が得られた.

改善法としては一般によく用いられるベスト改善, ファースト改善に加え, 充填率の低い容器から順に探索する優先改善を考えた. 初期解の構築法としてはよく知られた FFD, FF, NF に加え, FFD の変種である C_nFFD, R_δFFD を考え, エレメンタリー BPP においては容器あたりのアイテム数が釣

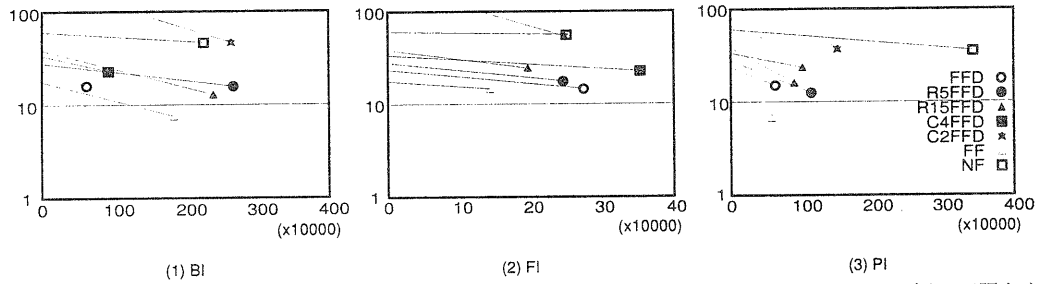

図 2: 実問題 BPP の一例に対して各局所探索法を適用したときの, 探索数 (横軸) と得られる解の下限からの差 (縦軸)

り合った初期解からは局所探索法により比較的良い局所最適解に到達することを示した. またエレメンタリー BPP に対しては, C_nFFD とファースト改善の組み合わせが解の質, 探索数の両方の観点から優れていることを示した. さらに優先改善は, ベスト改善と似た性質と, ファースト改善に似た性質の両方を持ちあわせていることがわかった.

実問題 BPP の一例に対して同じ局所探索法を適用した場合には, エレメンタリー BPP の場合には良かった C_nFFD とファースト改善の組み合わせでは良い結果が得られず, ファースト改善は制約が付いた場合には必ずしも優れているとは限らないことがわかった. ここではむしろベスト改善や, ベスト改善と似た性質を持つ優先改善が良く, 特に FF との組みあわせが良いことが示された. また探索数は優先改善ではベスト改善の $\frac{1}{3}$ 以下に抑えられた. 以上の結果から, 探索数を少なく抑え, かつ良い解の質を得るには本稿で提案した優先改善のような探索法が良いことが判明した. またそのとき, 初期解の構築法との組み合わせでは, 性能の影響がエレメンタリー BPP 程には大きくないことが判明した.

今後の課題として, 得られた知見に基づき, 様々な実問題 BPP の解法を設計することがある. さらに 2 次元及び 3 次元の実問題 BPP に関しても同様の解析を行う予定である.

参考文献

[1] M. R. Garey and D. S. Johnson, "Computers and Intractability — A Guide to the Theory of NP-Completeness," W. H. Freeman and Company (1979).

[2] E. G. Coffman, et al., "Approximation Algorithms for Bin Packing: a Survey," Approximation Algorithms for NP-Hard Problems, Chapter 2, PWS Publishing Company (1997).

[3] D. S. Johnson, "Near-Optimal Bin packing Algorithms," Ph.D. thesis, MIT, Department of Mathematics, Cambridge (1973).

[4] N. Karmarker and R. M. Karp, "An Efficient Approximation Scheme for the One-Dimensional Bin Packing Problem," In Proc. 23rd FOCS, pages 312–320 (1982).

[5] E. Falkenauer, "A Hybrid Grouping Genetic Algorithm for Bin Packing," J. Heuristics, **2**(1):5–30 (1994).

[6] T. Osogami, "Approaches to 3D Free-Form Cutting and Packing Probles and Their Applications: a Survey," IBM Res. Report, RT0285 (1998).

[7] T. Kämpke, "Simulated Annealing: Use of a New Tool in Bin Packing," Ann. of Oper. Res., **16**:327–332 (1988).

[8] C. R. Reeves, "Hybrid Genetic Algorithms for Bin-Packing and Related Problems," Ann. of Oper. Res., **63**:371–396 (1996).

[9] J. J. Bentley, "Fast Algorithms for Geometric Traveling Salesman Problems," ORSA J. Comput., **4**(4):387–411 (1992).

[10] H. Okano, et al., "New TSP Construction Heuristics and Their Relationships to the 2-Opt," J. Heuristics, **5**(1):71–88 (1999).

[11] V. Carvalho, "Exact Solution of Bin-Packing Problems Using Column Generation and Branch-and-Bound," Ann. of Oper. Res., **86**:629–659 (1999).