レクトリニア図形配置問題に対する構築型解法

胡　艷楠

2 次元配置問題は 2 次元の図形を長方形の容器に効率よく配置する問題であり，古典的な組合せ最適化問題の一つである．2 次元配置問題の中でも工学的に重要な意味をもつ長方形配置問題に対する研究が盛んである．著者の博士論文では垂直もしくは水平線分で描かれるレクトリニア図形を対象としている．レクトリニア図形は長方形を一般化した図形であるため長方形配置問題よりも応用の幅が広い．本稿では，長方形配置問題に対する構築型解法をレクトリニア図形配置問題に拡張し，その効率的な実装を説明した後に，新たな配置戦略の構築型解法と一般の 2 次元図形への適用を紹介する．

キーワード：配置問題，長方形，レクトリニア図形，2 次元図形

1. はじめに

2 次元配置問題は古典的な組合せ最適化問題の一つであり，NP 困難であることが知られている．2 次元配置問題においては，長方形はその扱いが容易であり工学的に重要な意味をもつことから，これまでさまざまな研究が行われてきた．一方，曲線を含む一般の図形の配置問題は非常に多くの応用をもつが，形状の取り扱いが難しい．たとえば，一般の図形の配置問題では，図形の配置に対して重なりがあるか否かを判定する際，計算コストの高い計算幾何的な処理が必要となり，さらに，それに起因する数値誤差が問題となる．

本稿では，垂直もしくは水平線分で描かれる図形であるレクトリニア図形を対象とする．レクトリニア図形を用いれば，一般の図形をビットマップ図形として近似的に扱うことができる（図 1）．また，レクトリニア図形は相対位置が固定された長方形の集合として表現できるため，長方形と似た構造をもち，図形の重なりを容易に判定することができる．

長方形配置問題に対して，代表的な構築型解法に bottom-left 法（BL 法）[1] と best-fit 法（BF 法）[2] がある．BL 法も BF 法も容器に何も配置されていない状態から始めて，長方形を一つずつ配置していき，最後に解を得るという構築タイプの解法である．BL 法では，長方形の順序があらかじめ与えられ，その順序で長方形を一つずつ配置する．BF 法はすでに配置されている状態を見て，次に配置する長方形を動的に選ぶため，BL 法より少し複雑な解法である．どちらの手法においても新たな長方形を配置する位置は，すでに容器内に配置されている長方形と重なりなく配置可能な位置の中で最も下の位置で，そのような位置が複数存在する場合にはその中で最も左の位置である．

一般の図形　ビットマップ図形　レクトリニア図形

図 1　一般の図形に対するレクトリニア図形としての近似

本稿では，長方形配置問題に対する BL 法と BF 法をレクトリニア図形配置問題に拡張し，その効率的な実装を説明した後に，新たな配置戦略の構築型解法と一般の 2 次元図形への適用を紹介する．

2. レクトリニア図形配置問題

レクトリニア図形配置問題は，与えられた n 個のレクトリニア図形を幅 W の長方形の容器に重なりなく配置するとき，その容器の高さ H をできるだけ小さくする問題である．レクトリニア図形は長方形の集合として表されるものとし，レクトリニア図形 i を表現する長方形の数を r_i とおく．r_i はレクトリニア図形の頂点の数より小さい正整数で抑えられる．本節ではレクトリニア図形に対して BL 法と BF 法を拡張し，その実装の考え方を紹介する．本稿では簡単のため図形の回転を考慮しないこととする．回転を考慮する場合については，回転した各形状に対して表現できる長方形の集合を準備することで 90°, 180°, 270° の角度の回転を許すケースに拡張できるが，紙面の都合上，ここでは割愛する．

まずは図形を配置する位置を決定する際に重要となる BL 安定点と BL 点の概念を説明する．いくつかの図形が配置された容器に新たな図形を配置することを

こ　えんなん
東京理科大学理学部第一部応用数学科
〒 162–8601　東京都新宿区神楽坂 1–3
yannanhu@rs.tus.ac.jp

考える．新たな図形を容器の中の位置 p に配置したとき，既配置の図形と重複なく，かつ位置 p から下または左に動かそうとしても既配置の図形と重なるか，または容器からはみ出るため動かすことができない位置を BL 安定点と呼ぶ．一般に BL 安定点は複数存在するが，その中で最も下で最も左にある BL 安定点を BL 点と呼ぶ．レクトリニア図形においても，BL 法と BF 法はどちらも新たな図形をその BL 点に配置する．

BL 法は，n 個のレクトリニア図形とその図形に対する順序が与えられたとき，その順序に従って図形を一つずつ配置する手法である．新たな図形を配置する位置は，その図形を配置する直前の状態における BL 点である．

BF 法は次に配置する図形を動的に計算して選ぶ．n 個のレクトリニア図形とそれぞれの図形に対する優先度が与えられたとき，以下のルールに従って次に配置する図形を選ぶ．まず，すべての未配置図形に対して BL 点を計算する．計算された BL 点の中で最も下の位置で最も左にある BL 点をもつ図形を選ぶ．そのような図形が複数存在する場合には，優先度が最も高い図形を選ぶ．このルールに基づいて選んだ図形をその BL 点に配置するという操作を，配置していない図形がなくなるまで繰り返す．

2.1 BL 点の探索の実装

BL 法と BF 法において最も計算に手間がかかるのは BL 点の探索である．それに対して，no-fit polygon (NFP) [3] という概念を利用し，図形同士の重なりを判定することで，探索の効率化を行う．

2.1.1 no-fit polygon

多角形 B と B' が与えられ，B の配置が固定されているとする．このとき，B と B' が重なりをもつような B' の配置位置の集合を B に対する B' の NFP と呼び，NFP(B, B') と表す．本稿では図形を包含する最小の長方形の左下の頂点の座標をその図形の参照点とする．B と B' がともに長方形の場合，NFP(B, B') は B' を B と接するように平行移動させたときの B' の参照点の軌跡の内部領域であり，形状は長方形となる（図 2）．

レクトリニア図形は長方形の集合として表現できるため，レクトリニア図形同士の NFP も相対位置が固定された長方形の集合として表せる．したがって，レクトリニア図形同士の重なりを判定する際には参照点が NFP を構成するいずれかの長方形の内部に入っているか否かを判定すればよい．

2.1.2 平衡探索木を用いた BL 点の発見法

Find2D-BL 法 [4] は，長方形配置問題に対して NFP を用いて BL 点を発見するアルゴリズムである．以下では Find2D-BL 法をレクトリニア図形に拡張し，BL 点を発見する時間計算量を解説する．

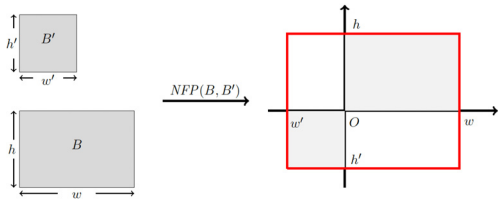

図 2　NFP の例

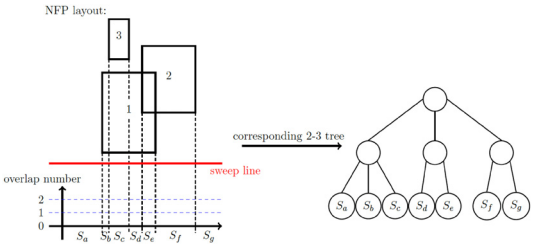

図 3　NFP 長方形と走査線上の重複度

既配置のレクトリニア図形すべてに対して，新たに配置したい図形 i の NFP を作成する．図形 i の参照点がそれら NFP のいずれの内部にも含まれないならば，既配置の図形のいずれとも重ならずにその位置に i を置くことができる．Find2D-BL 法は，走査線を用いてこのような点を発見するという考え方に基づいている．x 軸に平行な走査線を容器の底から y 軸の上方向に向かって動かす．このとき，走査線上の任意の点に対して，その点を含む NFP 長方形の数（重複数）を平衡探索木に保存し，維持しておく．走査線上で NFP の重複数が 0 になる位置が初めて現れたとき，走査線上のそのような位置の中で x 座標の値が最も小さい位置が i の BL 点となる．$R = \sum_{j=1}^{n} r_j$ とすると，この方法により平衡探索木を生成し新たに配置する図形 i の BL 点を計算するのにかかる時間は $O(r_i R \log R)$ となる．図 3 の左の図は容器の中の NFP 長方形と走査線を表し，右の図は走査線上の区間における重複度を保存する 2-3 木を示している．

BL 法の計算時間は，上に説明した手法で図形の BL 点を計算し，与えられた図形を一つずつ配置するため，$\sum_{i=1}^{n} O(r_i R \log R) = O(R^2 \log R)$ となる．

BF 法は一つの図形を配置する際に，すべての未配置の図形の形状に対して，BL 点を計算する．相異なる形状の数を t とし，すべての相異なる形状に対する r_j の合計を r とすると，一つの図形を配置するのに

$\sum_{j=1}^{t} O(r_j R \log R) = O(rR \log R)$ 時間かかる．したがって，BF 法の計算時間は $O(nrR \log R)$ となる．

2.2 計算過程を利用したより効率的な BL 点の探索

前節で紹介した BL 点の探索法の実装では，新しい図形を配置する際に毎回平衡探索木を生成して BL 点を計算していたが，以前の計算過程を利用することで BL 法と BF 法の計算時間を改善することができる [5]．以下ではその高速な実装の概略を示す．

BL 法と BF 法は構築型解法であるため，容器にいったん置いた図形を消したり移動したりすることはない．この性質を利用し，相異なる形状ごとに，現在構築中の解に対応する NFP の集合を保持し，解の更新に応じて逐次修正していく．各形状 j の NFP 集合に対して，容器に図形 i を配置した際に図形 i に対する j の NFP 長方形の集合を j の NFP 集合に加えることで更新できる．計算手順の概略は以下のとおりである．前節と同様に NFP 集合の区間の情報を平衡探索木に保存する．ここで，いずれの形状に対しても，走査線をその形状の前回の BL 点の高さから上に向かって走査すればよいことに注意する．走査線の現在の位置をヒープで管理し，図形を配置するたびに各形状の走査線を更新する．

すべての相異なる形状に対して，平衡探索木とヒープを更新する時間の合計は $\sum_{j=1}^{t} O(r_j R \log R) = O(rR \log R)$ となる．走査線上に BL 点が存在するか否かを判定するのに $O(\log R)$ 時間かかり，形状 j に対して走査線は $O(r_j R)$ 回移動する．よって，すべての相異なる形状に対して，BL 点を探索する時間の合計は $\sum_{j=1}^{t} O(r_j R \log R) = O(rR \log R)$ となる．したがって，BL 法と BF 法の計算時間は $O(rR \log R)$ となる．

図形の数が 1 万個程度のベンチマーク問題例に対して，C 言語により実装されたプログラムを Intel Core i5，2.3 GHz，3 MB cache，4 GB memory の PC 上で実行したところ 2 秒以内に解を得ることを確認した．

2.3 新たな配置戦略の構築型解法

本節では，BL 法と BF 法で得られる解を分析し，その考察に基づいて新たな配置戦略を提案する [6]．

計算機実験により，BL 法と BF 法によって得られる解の充填率について次のことが確認できた．BF 法は多くの問題例においてより高い充填率を得ることができる．その一方，BL 法でよりよい解を得るケースも無視できない数がある．そこで，配置結果を分析し，多くの問題例において BF 法の性能がよりよい理由と BL 法がよりよい配置を得る問題例の特徴をまとめる．

BF 法は一つの図形を配置する際に，すべての未配置の図形の形状を試し，その時点で最も下に配置できる図形を選ぶ．そのため，容器の中の既配置の図形に対して，最も噛み合う図形を選び，配置後の余剰スペースが小さくなる傾向がある．BL 法は図形の配置順序が与えられるものであるため，その時点の配置と合わない図形を配置することになる．そのため，BF 法の方が多くの問題例において，よりよい配置を得ることができる．

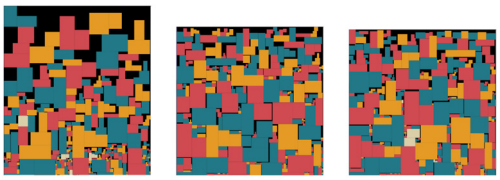

図 4 BF 法，BL 法，PBF 法で得られる配置図

その一方で，図形のサイズが大幅に異なる問題例に対して，BL 法がよりよい解を得ることが多い傾向がある．BL 法では，図形のサイズが大きい順で図形を配置すると性能がよいということが知られている．ここでは与えられる図形の順序が面積の大きい順の場合を考える．図形のサイズが大幅に異なる問題例に対して，BL 法はサイズが大きい図形を先に配置する．その後に配置する小さい図形は既配置の大きい図形の隙間に配置することが可能であるため，小さい図形は最終的な容器の高さに影響することが少ない．次に BF 法では，小さい図形の BL 点は通常大きい図形の BL 点より低くなるため，それらを先に配置する傾向があり，最後に残る大きい図形は小さい図形の上に置くことになる．大きい図形の隙間は最終的な配置の充填率を大きく棄損する．図 4 はそのような問題例に対して，BF 法（左）と BL 法（中）で得られる配置である．

以上の考察に基づいて新しい構築型解法 Partition-based best-fit 法（PBF 法）を提案する．この解法は，与えられるレクトリニア図形をグループに分けて，グループの順序を決める．グループの順序に従って，各グループに含まれるレクトリニア図形を BF 法で容器に配置する．この手法は，図形のグループ分けの仕方によって BL 法と BF 法の両方を特殊な場合として含む枠組みである．PBF 法の計算時間については，2.2 節の実装を利用すると BF 法と BL 法と同じ $O(rR \log R)$ となる．図 4 の右の図は PBF 法によって得られた配置である．

3. 2 次元図形配置への応用

前節で紹介したレクトリニア図形配置問題に対する高速な解法を利用してビットマップ図形配置問題を求解する [7]．

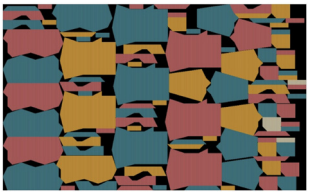

図 5　BL 法によるベンチマーク問題例の配置図

曲線を含む一般の図形を精度高くビットマップ図形に近似すると，図形を表現するためのデータ量が膨大になる．この問題を解消するため，ビットマップ図形を効率的に扱うための次の三つの手法を提案する．

表現法 1 では，ビットマップ図形に対して各行が高さ 1 の長方形で構成されていると捉え，それらの集合としてその図形を表現する．ビットマップ図形が y 単調の図形（y 軸に垂直な任意の直線との交わりが一つの区間か空であるような連結な図形）の場合は，高さ h_i のビットマップ図形 i は h_i 個の長方形の集合で表現できる．

表現法 2 は，集合被覆問題に対するアルゴリズムを利用して，できるだけ少ない数の長方形の集合でそのビットマップ図形を表す手法である．

表現法 3 は，あらかじめビットマップ図形の形状のすべてのペアに対して NFP を準備しておく手法である．与えられた二つのビットマップ図形 i（高さ h_i）と j（高さ h_j）がともに y 単調の図形であれば，NFP(i, j) は $h_i + h_j - 1$ 個の長方形の集合で表現できる．

多角形配置問題に対する代表的なベンチマーク問題例と，それらをもとに生成した大規模な問題例に対して計算機実験を行った．その結果，いくつかの問題例に対しては充填率が 80% を超えるよい解を短い時間で構築できた．既存研究では 100 程度の図形の配置に 1,200 秒以上かかってしまうのに対し，本研究の手法は，図形が約 1 万個ある問題例に対しても 100 秒以内に良質の解を構築できることを確認した．図 5 は BL 法によるベンチマーク問題例の配置図である（多角形配置問題の慣例に従い時計まわりに 90 度回転して描画している）．

4.　その後の研究

本節では，博士課程修了後に携わった配置問題に関する研究について紹介する．

4.1　レクトリニア図形配置問題に対する厳密解法

レクトリニア図形の形状の特性を分析し定式化することで，厳密解法を開発した [8]．

各図形を幅 1 の複数の長方形へと分割し，y 軸上の相対位置に関する制約を緩和する緩和問題を整数計画問題として定式化する．整数計画ソルバーで緩和問題を解くことで元の問題の下界と図形の x 座標を得ることができる．図形の x 座標を固定し，図形の配置を決定する問題をもう一つの整数計画問題として定式化し，元の問題の上界を得ることができる．下界と上界に対して，探索空間を絞っていく制約を加えていくことで下界と上界の質の向上を行い，下界が上界と同等もしくはそれ以上の値となったときに厳密な最適解が得られる．

また，新しい解表現法を提案し，さまざまな定理や補題を導き出すとともに，もう一つの厳密解法を提案した．分枝限定法を導入することで，効率的に探索空間を絞ることに成功し，整数計画問題として定式化した場合と比較し，特に複雑な形をした図形を含む問題例においては，最大で 35 倍の計算時間の大幅な短縮に成功した．

4.2　3 次元配置問題へ応用

4.2.1　コンテナ配置問題

本問題は国際会議 ESICUP が自動車メーカー Renault と共同で実施したコンペティションの課題で，実社会に現れるさまざまな制約条件を含むコンテナ配置問題である．コンテナ配置問題とは，直方体の形状をもつ複数のコンテナと直方体の荷物が与えられたとき，すべての荷物をコンテナ内に配置し，使用するコンテナの数の最小化を目的とする問題である．直方体をコンテナに配置する際に，スタック，レイヤー，ローと呼ばれる形態を構成しなければならない．まずいくつかの直方体を x 軸または y 軸方向に並べてローと呼ばれる直方体の列を構成し，次はいくつかのローを x 軸または y 軸方向に並べてレイヤーと呼ばれる直方体の層を構成し，最後にいくつかのレイヤーを z 軸方向に積み上げてスタックと呼ばれる直方体の積み重ねを構成する．最終的にスタックをコンテナの xy 平面に配置する．配置の仕方に関する条件以外にもさまざまな制約条件，たとえば，直方体の材質や，重さ，方向などに関する制約がある．

この問題に対して，2 段階の貪欲戦略 [9] を提案した．まず，ナップサック問題を用いて効率的な直方体の積み重ね（スタック）を求める．生成したスタックをコンテナに配置する問題は 2 次元の長方形配置問題として扱うことができるため，BF 法を複数回実行してスタックをコンテナに効率よく配置した．

予選の結果は 12 チーム中 3 位という結果であった．20 個の問題例に対して，提案手法は四つの問題例で最

良解を得ることができた．

4.2.2 レクトリニア多面体配置問題

3 次元多面体で各面が xy 平面，yz 平面，zx 平面のいずれかと平行であるものを 3 次元レクトリニア多面体と呼び，そのような多面体を直方体の容器に詰め込む問題を 3 次元レクトリニア多面体配置問題という．レクトリニア多面体は相対位置が固定された直方体の集合として表現できる．レクトリニア図形に対する BL 法，BF 法と PBF 法を 3 次元レクトリニア多面体へ拡張し，3 次元レクトリニア多面体に対する deepest-bottom-left 法（DBL 法），3D best-fit 法（3BF 法）と 3D partition-based best-fit 法（3PBF 法）を提案した [10]．

多面体を配置する位置を deepest-bottom-left 戦略（DBL 戦略）で選ぶ．新たな多面体をいくつかの多面体が配置された容器の中の位置 p に配置したとき，既配置の多面体と重複なく配置できる位置の中で，最も奥行きの小さい面の最も低い位置のうち，できるだけ左の位置を DBL 安定点と呼ぶ．DBL 戦略では，多面体を配置する際に常にその時点の DBL 点に配置する．

多面体同士の重なりを容易に判定するために，no-fit cube (NFC) という考え方を提案し，2.2 節で紹介したレクトリニア配置問題に対する実装方法と同様に，高度なデータ構造，および効率よく NFC を保持する手法を組み込むことにより計算の高速化を行った．

多面体数が約 3,000 個ある問題例に対しても 72 秒以内に解を構築できることを確認した．

5. おわりに

本稿では，著者が博士課程の間に研究したレクトリニア図形配置問題と博士課程修了後に携わった配置問題に関する研究について紹介した．

レクトリニア図形配置問題に関する研究では，大規模な問題を高速に解く手法に焦点を絞った．これらの手法はレクトリニア図形配置問題に対して提案された手法であるが，そのアイデアは一般の 2 次元図形や 3 次元の物体の配置にも適用することができる．

著者は博士前期課程のときに日本に留学してきた．所属していた研究室では，論文の探し方，計算機の使い方，プログラミング，プレゼンテーションの仕方など基礎から研究する方法を指導していただいた．配置問題は解（図形の配置）を描画することで視覚的にその状況を把握することができるため，最初からとても興味をもって研究に取り組むことができた．研究の中で，とても複雑なデータ構造をプログラミングすることになり，プログラミング初心者の著者にとってつらいときもあったが，先生方や研究室の方々に色々教えていただき，問題を一つ一つ乗り越えた．問題を解決するときの喜びは一生の宝と記憶する．

今後も，配置問題に関する研究を引き続き実施し，さらに掘り下げた研究を行う予定である．さまざまな実用的な問題に対して，よい内部構造を見いだし，それをうまく利用した数理的な手法を用いることで，効率のよい解法を開発したいと考えている．また，研究活動を通して，現実社会の生産，流通などに現れる問題を解決するための最適化手法を提供できれば幸いである．

参考文献

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

[2] E. K. Burke, R. Hellier, G. Kendall and G. Whitwell, “A new placement heuristic for the orthogonal stock-cutting problem,” *Operations Research*, **54**, pp. 587–601, 2006.

[3] R. C. Art, “An approach to the two dimensional irregular cutting stock problem,” Ph.D. Thesis, Massachusetts Institute of Technology, 1966.

[4] S. Imahori, Y. Chien, Y. Tanaka and M. Yagiura, “Enumerating bottom-left stable positions for rectangle placements with overlap,” *Journal of Operations Research Society of Japan*, **57**, pp. 45–61, 2014.

[5] Y. Hu, H. Hashimoto, S. Imahori and M. Yagiura, “Efficient implementations of construction heuristics for the rectilinear block packing problem,” *Computers & Operations Research*, **53**, pp. 206–222, 2015.

[6] Y. Hu, H. Hashimoto, S. Imahori, T. Uno and M. Yagiura, “A partition-based heuristic algorithm for the rectilinear block packing problem,” *Journal of the Operations Research Society of Japan*, **59**, pp. 110–129, 2016.

[7] Y. Hu, S. Fukatsu, H. Hashimoto, S. Imahori and M. Yagiura, “Efficient overlap detection and construction algorithms for the bitmap shape packing problem,” *Journal of the Operations Research Society of Japan*, **61**, pp. 132–150, 2018.

[8] K. Matsushita, Y. Hu, H. Hashimoto, S. Imahori and M. Yagiura, “Exact algorithms for the rectilinear block packing problem,” *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, **12**, JAMDSM0074, 2018.

[9] H. Iwasawa, Y. Hu, H. Hashimoto, S. Imahori and M. Yagiura, “A heuristic algorithm for the container loading problem with complex loading constraints,” *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, **10**, JAMDSM0041, 2016.

[10] 梅田知樹，胡艶楠，柳浦睦憲，“三次元レクトリニア多面体配置問題に対する構築型解法の配置戦略および効率的実現法，”情報処理学会研究報告，2019-AL-172(6), pp. 1–8, 2019.