2次元ビン・パッキング問題と
真円度問題に関する研究

昭和61年12月

榎原博之

内容梗概

本論文は，著者が大阪大学大学院工学研究科通信工学専攻在学中に行った研究のうち，計算幾何学の分野に属する２次元ビン・パッキング問題と真円度問題に関する研究をまとめたものである．

本論文では，まず，幾何学的最適化問題の１つである２次元ビン・パッキング問題を取り上げ，この問題に制約を加えた場合の厳密解法と近似解法についての研究を，ついで，計算幾何学の近接問題の中心課題の１つであるボロノイ図を応用して，真円度を求める多項式時間の厳密解法についての研究を述べている．

第２章，第３章では，ピースに制約を加えた２次元ビン・パッキング問題について考察している．第２章では，ピースに制約を加えた２次元ビン・パッキング問題に対する３つの厳密解法を提案し，その最適性について論じ，少なくともピースの種類が４ｘ１，１ｘ４，２ｘ２の部分矩形に限定した場合までは，線形時間で最適解が得られることを明らかにしている．

第３章では，ピースに制約を加えた２次元ビン・パッキング問題に対する３つの近似解法を提案し，その良さを評価している．まず，ピースの種類を３ｘ２の部分矩形に限定した場合の近似解法を提案し，その最悪の場合の近似度を理論的に解析している．次に，ピースの種類をｗｘ１およびｗｘ２（ｗ≧５）の部分矩形に限定した場合の近似解法を提案し，その良さを計算機実験によって評価している．

第４章では，ボロノイ図を利用することによって，機械部品の精度として重要な幾何公差の１つである真円度を求める多項式時間の厳密解法が存在することを指摘し，その最適性と時間計算量を明らかにしている．

関連発表論文

Ⅰ．電子通信学会論文誌

（１）　榎原，中野，中西：”ピースに制約を加えた２次元ビン・パッキング問題”，信学論（Ａ），vol.J69-A，no.3，pp.350-358 (Mar. 1986).

（２）　榎原，中野，中西，村松：“ピースに制約を加えた２次元ビン・パッキング問題に対する近似解法”，（ショートノート），信学論（Ａ），vol.J69-A，no.8，pp.1015-1017 (Aug. 1986).

（３）　榎原，中野，中西，真田：“ボロノイ図を応用した真円度を求める解法”，信学論（Ａ）　（掲載予定）．

Ⅱ．電子通信学会研究会資料及び全国大会予稿

（１）　榎原，中野，中西：“ピースに制約を加えた２次元ビン・パッキング問題”，信学技報，ＣＡＳ８４－８２（昭５９－０８）．

（２）　榎原，中野，中西：“ピースの種類を３ｘ２の部分矩形に限定した２次元ビン・パッキング問題”，信学会総合全国大会，４４　（昭６０－０３）．

（３）　榎原，中野，中西，村松：“ピースに制約を加えた２次元ビン・パッキング問題に対する近似解法”，信学技報，ＣＡＳ８５－８８　（昭６０－１０）．

Ⅲ．Proceedings of ISCAS 85

（１）　H.EBARA, H.NAKANO, and Y.NAKANISHI :"Constrained Two-Dimensional Bin Packing Problem", Proceedings of ISCAS 85, pp.965-966 (1985).

目　次

第１章　緒論

近年，コンピュータ・サイエンスの中で急速に発展してきた分野の１つに計算幾何学（computational geometry）[2][9][11][12]と呼ばれる分野がある．これは，幾何学（geometry）に計算の複雑さ（computational complexity）の理論[1]を導入して，幾何学的な問題を計算機で効率良く処理する算法を開発し，その限界等を理論的に究明する学術分野である．計算幾何学は，ＶＬＳＩのＣＡＤ，パターン認識，コンピュータ・グラフィックス，地理情報処理，ロボティクス，データベース理論等応用分野が非常に広く，これら以外の分野にも広く応用される可能性を秘めている．

これまでの幾何学的な問題に対する算法は，人間の直観（パターン認識）によるものが多く，問題の規模が小さければすぐに解を求めることができるが，大規模な問題をしかも計算機を使って解こうとする場合うまくいかないことが多かった．しかし，コンピュータ・サイエンスの分野で算法やデータ構造の研究[1][15]が急速に進歩し，これらの算法やデータ構造を利用することにより，あるいは，これらの算法やデータ構造の考え方を基に新たな算法やデータ構造を開発することにより，幾何学的な情報を計算機で効率的に処理できるようになってきている．

計算幾何学の代表的な問題は大略次のように分類される[11]．

（１）　凸包（convexity）問題

与えられた幾何学的対象物（たとえば，点など）の集合に対してその集合の凸包（convex hull）を求める問題．

（２）　重なり（intersection）問題

与えられた幾何学的対象物（たとえば，線分など）の交差判定をしたり共通部分を求めたりする問題．

（３）　幾何学的探索（geometric searching）問題

ある種の幾何学的な図形（たとえば，地図など）に対してある点を含む領域を決定するとか，ある幾何学的な図形に対してその内部に含まれる幾何学的対象物（たとえば，点など）を列挙するなどという探索問題．

（４）　近接（proximity）問題

ある幾何学的対象物（たとえば，点など）間の近接関係等を調べる問題．この問題にはしばしば計算幾何学の中心課題の１つであるボロノイ図が使われる．

（５）　幾何学的最適化（geometric optimization）問題

与えられた幾何学的対象物に対する各種の最適化問題．

なお，計算幾何学の問題に使用される主な算法には，平面走査法（plane sweep method），分割統治法（divide-and-conquer method），幾何学的変換法（geometric transformation method），バケット法，フィルタリング法等があり，データ構造には，平衡二分木（balanced binary tree），セグメント木，ヒープ探索木（priority search tree）等がある．

本論文では，幾何学的最適化問題に属する２次元ビン・パッキング問題（2-Dimensional Bin Packing Problem）[5]と真円度問題[16]を取り上げている．２次元ビン・パッキング問題については，この問題に制約を加えた場合の厳密解法と近似解法について考察している．２次元ビン・パッキング問題は，ＮＰ完全[7]，あるいはそれ以上のクラスに属する扱いにくい問題であり，問題に制約を加えない限り多項式時間で最適解を得ることは不可能であると考えられている．しかし，もしピースの種類に制約を加えるならば，その扱いにくさ（intractability）[7]が変わるものと考えられる．そこで，この問題の扱いにくさがピースの種類によってどのように変わるかを系統的に追究しやすくするために，ビンの幅，ピースの幅，

高さをすべて整数値とし，かつピースの種類をｗｘｈの部分矩形（幅ｗ以下，高さｈ以下の矩形の集合）に限定する．そして，ｗおよびｈを逐次増加させて，時間計算量がピース数ｎの多項式時間，とくにＯ（ｎ）回のＢＬ（ボトム・レフト）パッキング[3]での厳密解法および極めて良い近似解法の存在するｗおよびｈを見い出すことを試みている．

第２章では，ピースの種類をｗｘｈの部分矩形に限定し，ｗおよびｈを逐次増加させて，時間計算量がピース数ｎの線形時間である厳密解法の存在するｗおよびｈを見い出すことを試み，少なくともピースの種類が４ｘ１，１ｘ４，２ｘ２の部分矩形に限定した場合までは，線形時間で最適解が得られる解法が考えられることを明らかにしている．

第３章では，ピースの種類をｗｘｈの部分矩形としても，ｗおよびｈの値が第２章で見い出した厳密解法で考えられる整数値を越えると，もはや時間計算量がピース数ｎの線形時間となる厳密解法は期待できそうにないとの判断のもとに，近似解法について考察している．このような場合でも，第２章で見い出した厳密解法の考え方や解法そのものを活用することにより，良い近似解法を構成することができる可能性があると思われる．第３章では，まず，近似解法の理論的解析として，ピースの種類を３ｘ２の部分矩形に限定した場合の近似解法を提案し，その最悪の場合の近似度を示している．この近似解法は，第２章のピースの種類が４ｘ１の部分矩形である場合の厳密解法の考え方を基にしたものであり，線形時間で最悪の場合でもたかだか高さが１しか悪くならない近似解が得られることを明らかにしている．次に，ピースの種類をｗｘ１およびｗｘ２（ｗ≧５）の部分矩形に限定した場合の近似解法を提案し，その良さを既存の近似解法であるレベル・アルゴリズム[6]との比較実験を通して評価している．これらの近似解法は，第２章のピースの種類

が４ｘ１の部分矩形である場合の厳密解法を活用したものであり，実験結果から計算時間も妥当であり，性能の点でも良い近似解法であると考えられることを明らかにしている．

真円度問題については，近接問題の中心課題の１つであるボロノイ図[8][13]を利用することによって，多項式時間で真円度[16]を求めることができることを明らかにしている．真円度を求める解法に関しては，今までのところ近似解法に過ぎないとか，厳密解法であっても時間計算量に難点があるものであって，時間計算量が多項式時間である厳密解法は知られていなかった．

第４章では，ボロノイ図を利用することによって，機械部品の精度として重要な幾何公差の１つである真円度を求める多項式時間の厳密解法が存在することを指摘し，その最適性と時間計算量を明らかにしている．この解法は，最近点のボロノイ図（勢力圏図）と最遠点のボロノイ図（非勢力圏図）の結び（Union）をとるという考えに基づいたものである．

第２章　ピースに制約を加えた２次元ビン・パッキング問題の厳密解法

２．１　緒言

本章では，ピースに制約を加えた２次元ビン・パッキング問題に対する３つの厳密解法を提案し，その最適性について議論する．

２次元ビン・パッキング問題[5]とは，

“幅Ｗで，高さが無限大のオープン・エンドの矩形のビンＢと，幅がw_i，高さがh_iのｎ個の矩形のピースp_i（$1 \leq i \leq n$）が与えられたとき（ただし，$w_i \leq W$），すべてのピースp_iを互いに重なることなく，高さが最小となるように詰める問題”

である（図２．１）．なお，ｎ個のピースは，リストＬの中に添字の順に入っているものとし，また，ピースの回転を許さないものとする．この問題は，マルチ・プロセッサのジョブ割当てを始め，型紙のレイアウト等，応用分野も広く，その解法についての研究が活発に進められてきており，これまでにいくつかの近似解法が提案されてきている[3][6][14]．

２次元ビン・パッキング問題は，１次元ビン・パッキング問題[5][10]がＮＰ完全な問題である[7]ことから考え，ＮＰ完全[7]，あるいはそれ以上のクラスに属する問題であると考えられており，一般には扱いにくい問題である．しかし，もしピースの種類が制約されれば，扱いにくさ（intractability）[7]が変わるものと考えられる．本章と次章は，ピースの種類によって扱いにくさがどのようになるかを明らかにすることを目的にしたものであって，このことを系統的に追究しやすくするために，ビンの幅，ピースの幅，高さをすべて整数値とし，ピースの種類をｗｘｈの部分矩形に限定する．本章では，ｗおよびｈを逐次増加させて，時間計算量がピース

数ｎの多項式時間，とくに線形時間となる厳密解法の存在するｗおよびｈを見い出すことを試みている．

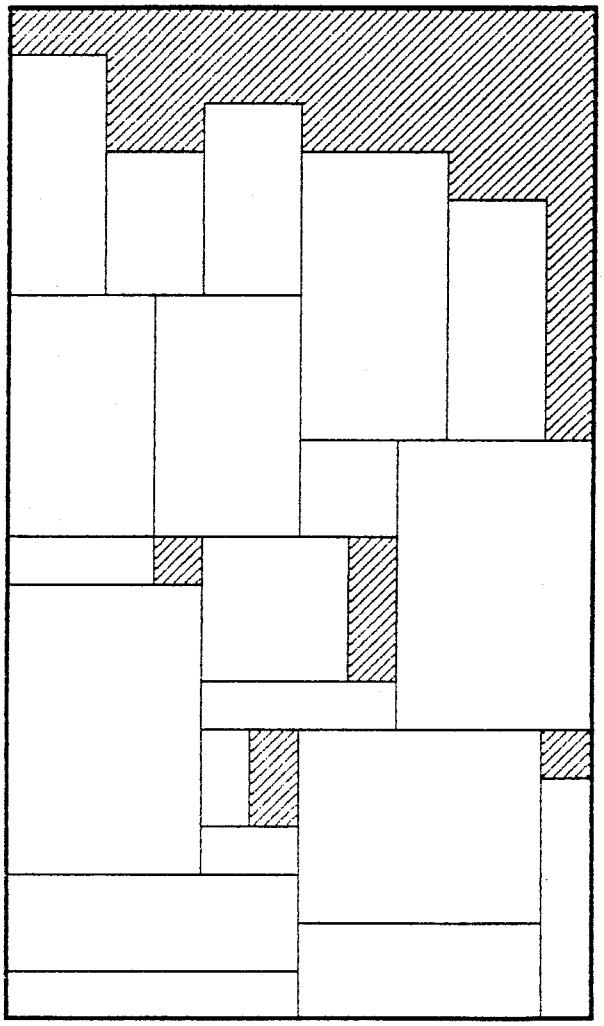


図２．１　２次元ビン・パッキング問題のパッキング例

ここで，本章と次章で使用する特別な用語と記号をまとめて示しておく．

・“ＢＬ（ボトム・レフト）パッキング”……空き地も考慮して，ピースを詰めることができる最も底に近い位置で，かつその中で最も左の位置にピースを詰めること[3]．また，後述の解法で使用している“ＢＬで詰める”とは，この規則に従いピースを詰めることである．

・“空き地”……周囲をピースまたはビンで囲まれたピースの存在しない領域．なお，右端にできる，上にのみ開いた細長いピースの存在しない領域も空き地と呼ぶこととする．

・“ｉｘｊの矩形”……幅ｉ，高さｊの矩形．

・“ｗｘｈの部分矩形”……｛ ｉｘｊの矩形 ｜ $1 \leqq i \leqq w$，$1 \leqq j \leqq h$，ｉ，ｊは整数 ｝．たとえば，３ｘ２の部分矩形とは，３ｘ２，３ｘ１，２ｘ２，２ｘ１，１ｘ２，１ｘ１の６種類の矩形の集合である．

・“タイト”……空き地がなく，ピースがピッタリと詰まっている状態．

・“レベル”……ピースを詰める際に基準となる，ビンの底辺に平行な直線上の，ピースの存在することができる領域[6]．たとえば，高さ１のピースのみを詰めた場合は，高さ１ごとにレベルができる．

・“フリーのピース”……ビンに詰められていないピース．

・p_{ij}……幅ｉ，高さｊの矩形のピース．

・n_{ij}……幅ｉ，高さｊの矩形のピースの個数．

２．２　ピースの高さが一定である場合

ピースの高さが一定であるという制約を加えると，問題は単に１次元ビン・パッキング問題[5][10]となる．この問題において，ピ

ースの種類をｗｘ１の部分矩形に限定し，ｗをパラメータとして多項式時間の厳密解法が考えられるかについて考察する．ピースの種類を４ｘ１の部分矩形に限定した場合の厳密解法を提案し，次の定理を導いている．

［定理２．１］　ピースの種類を４ｘ１の部分矩形に限定する．このとき，ピースをアルゴリズム２．１で詰めると，必ず最適に詰めることができる．また，このアルゴリズムはＯ（ｎ）回のＢＬパッキングで行うことができる．ただし，Ｗはビンの幅であり，ｎはピース数である．　□

アルゴリズム２．１

メイン・ルーチン

```
step１   if  Ｗ＝４  then  手続きＷ４を行う；
step２   if  Ｗ＝５  then  手続きＷ５を行う；
step３   if  Ｗ≧６  then  begin
   3.1  if  W mod ４＝０  then
            手続きＷｍ０を行う；
   3.2  if  W mod ４＝１  then
            手続きＷｍ１を行う；
   3.3  if  W mod ４＝２  then
            手続きＷｍ２を行う；
   3.4  if  W mod ４＝３  then
            手続きＷｍ３を行う；
   3.5  手続きexchangeを行う；
   3.6  タイトに詰まっていないレベルで，
        手続きrepackを行う；
        end
step４   p11をＢＬで詰める；
```

手続きＷ４

step１　p_{41}をＢＬで詰める；

step２　p_{21}をＢＬで詰める；

step３　p_{31}をＢＬで詰める；

手続きＷ５

step１　p_{31}をＢＬで詰める；

step２　p_{21}をp_{31}の最高レベルまでＢＬで詰める；

step３　p_{41}をＢＬで詰める；

step４　残ったp_{21}をＢＬで詰める；

手続きＷｍ０

step１　p_{41}をＢＬで詰める；

step２　p_{21}をＢＬで詰める；

手続きＷｍ１

step１　p_{31}を右端に１列に詰める；

step２　p_{41}を右端の幅５の領域を除いてＢＬで詰める；

step３　p_{31}とp_{41}の間にできた幅２の空き地に，タイトなp_{41}の最高レベルまでp_{21}を詰める；

step４　もしp_{21}がタイトなp_{41}の最高レベルまで詰めることができなければ，以下の操作をタイトなp_{41}の最高レベルまで，あるいは右端に詰まっているp_{31}を上から順に２個取ると，その時点でタイトに詰まっている最高レベルの高さ（H_t）以下になるまで行う；

・タイトでない最低レベルの一番右のp_{41}を取り除き，そこに右端に詰まっているp_{31}を上から順に２個取って詰め，取り除いたp_{41}をstep２の操作に倣って詰め

る；

step5　もしp31がタイトなp41の最高レベルまで詰めることができず，p21がp31の最高レベルまで詰めることができるならば，p31の最高レベルまでp21を詰める；

step6　残ったp21をＢＬで詰める；

手続きＷｍ２

step1　p41をＢＬで詰める；

step2　右端の幅２の空き地に，タイトなp41の最高レベルまでp21を詰める；

step3　もしp21がタイトなp41の最高レベルまで詰めることができなければ，以下の操作をタイトなp41の最高レベルまで，あるいはp31が足りなくなる（１個以下になる）まで行う；

・タイトでない最低レベルの一番右のp41を取り除き，そこにフリーのp31２個を詰め，取り除いたp41をＢＬで詰める；

step4　残ったp21をＢＬで詰める；

手続きＷｍ３

step1　p31を右端に１列に詰める；

step2　p41をＢＬで詰める；

step3　p21をＢＬで詰める；

手続きexchange

step0　この手続きにおいて交換されるp31は，Ｗが偶数のときはフリーのp31であり，Ｗが奇数のときは右端に１列に詰められたp31である；　このピースをpexとする；

Wが偶数のときは，交換することができるフリーのp_{31}がなくなる（２個未満あるいは４個未満になる）まで，以下の操作を行う；　また，Wが奇数のときは，交換するために右端の一番上のp_{31}を取り除くことによって，その時点でタイトに詰まっている最高レベルの高さ（H_t）未満になるならば，交換することは行わず，この手続きは終了する；

step１　同じレベルにあるp_{21}３個をp_{ex}２個と交換し，交換されたp_{21}３個をＢＬで詰める；　この操作を行えば，step１へ；

step２　同じレベルにあるp_{21}とp_{41}をp_{ex}２個と交換する；もし同じレベルにp_{21}がなければ，前もってそのレベルの別のp_{41}と他の同じレベルにあるp_{21}２個を交換しておく；　交換されたp_{41}は，もしあれば，同じレベルにあるp_{21}２個と交換し，その後p_{21}３個をＢＬで詰める；　もし同じレベルにp_{21}２個がなければ，p_{41}，p_{21}をその順にＢＬで詰める；　この操作を行えば，step１へ；

step３　同じレベルにあるp_{41}３個をp_{ex}４個と交換し，交換されたp_{41}３個をＢＬで詰める；　この操作を行えば，step１へ；

手続きrepack

step１　残ったp_{31}が３個以下の場合；

・p_{41}，p_{21}，p_{31}の順にＢＬで詰める；

step２　残ったp_{31}が４個以上の場合；

2.1　p_{21}がなく，p_{41}が２個の場合；

・p_{41}２個をＢＬで詰める；

・p_{41}と同じレベルにp_{31}を詰める；
・もし右端に幅２の空き地ができるならば，そのレベルのp_{41}１個を取り除き，そこにp_{31}２個を詰め，p_{41}をその上のレベルにＢＬで詰める；
・残りのp_{31}をＢＬで詰める；

2.2　その他の場合；
・p_{41}，p_{31}，p_{21}の順にＢＬで詰める；

（証明）　このアルゴリズムがピースを最適に詰めることを示すためには，ピースができる限りタイトに詰められていることを示さなければならない．もしピースが一番上のレベルを除いてタイトに詰まっているならば，このパッキングは最適である．もし一番上のレベル以外にタイトでない複数個のレベルがある場合でも，何回かの交換を行ってもパッキングの高さをこれ以上低くすることができないことを示せば最適である．これを示すためには，一番上のレベル以外にタイトでないレベルがあるすべての場合を１つずつ検討する必要がある．

また，このアルゴリズムでは，p_{11}は最後に空き地も考慮して，ＢＬで詰めることにしている．なぜならば，p_{11}は最小単位のピースであり，このピースより小さい空き地はなく，どんな形，どんな大きさの空き地にもp_{11}は必ずピッタリと詰めることができるからである．よって，最適性を考える場合，p_{11}はないものとして考えることにする．

（１）　Ｗ＝４（手続きＷ４）の場合
この場合は自明である（図２．２）．
（２）　Ｗ＝５（手続きＷ５）の場合
この場合，step２で，p_{21}がp_{31}の最高レベルまで詰めることが

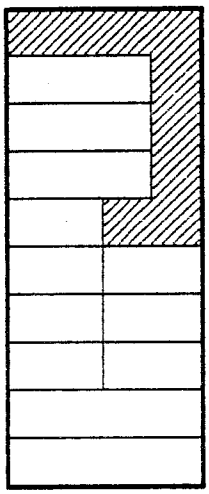


図２．２　アルゴリズム２．１によるパッキング例１
（手続きＷ４）

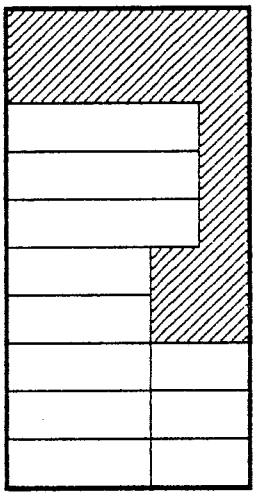


図２．３　アルゴリズム２．１によるパッキング例２
（手続きＷ５）

できない場合でも，幅２以下のピースはないので最適である（図２．３）．

（３）　Ｗ≧６で，Ｗ mod ４＝０（手続きＷｍ０）の場合

この場合はＷが４の倍数であるので，step１，step２で，p_{41}，p_{21}をＢＬで詰めたとき，一番上のレベルを除いてタイトである．

この状態で手続きＷｍ０を終了し，次に手続きexchangeを行う．p_{31}をp_{21}，p_{41}と交換する方法は，p_{31}２個とp_{21}３個，p_{31}２個とp_{21}，p_{41}，p_{31}４個とp_{41}３個の３通りのみが考えられる（図２．４）．手続きexchangeでは，できる限り，かつタイトを崩さずにこの３通りの交換を行い，またp_{41}がp_{21}２個と交換できる場合はその交換を行った後ＢＬで詰める（図２．５）．ただし，step３でp_{41}をp_{21}と交換しないのはできないからである．なぜな

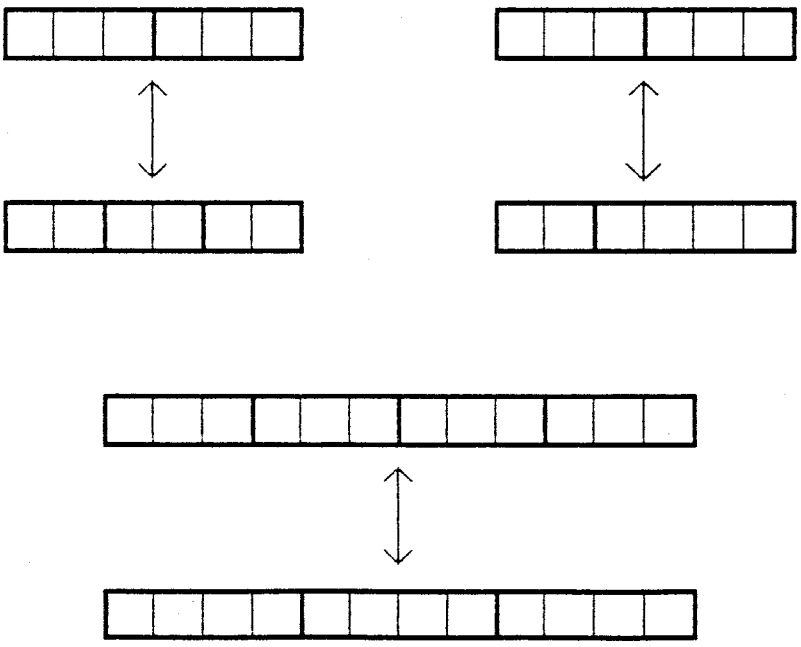


図２．４　p_{31}とp_{21}，p_{41}との交換

らば，もし交換できるならばstep２を行っているはずである．よって，手続きexchange終了後は，一番上のレベルを除いてタイトであり，p_{31}ができる限りp_{21}，p_{41}と交換されている．

ここで，p_{21}，p_{41}が十分あり，p_{31}が手続きexchangeでできる限り交換されるならば，p_{31}は３個以下である．３個以下の場合は，手続きrepackで一番上のタイトでない（唯一の）レベルの続きにp_{31}をＢＬで詰めれば最適となる．なぜならば，p_{31}が１個以下の場合は，明らかに最適に詰めることができる．また，p_{31}が２個あるいは３個ある場合は，p_{31}２個とp_{41}，幅２の空き地とを交換することができるが，$W \geq 9$のときは，p_{31}３個をＢＬで詰めてもた

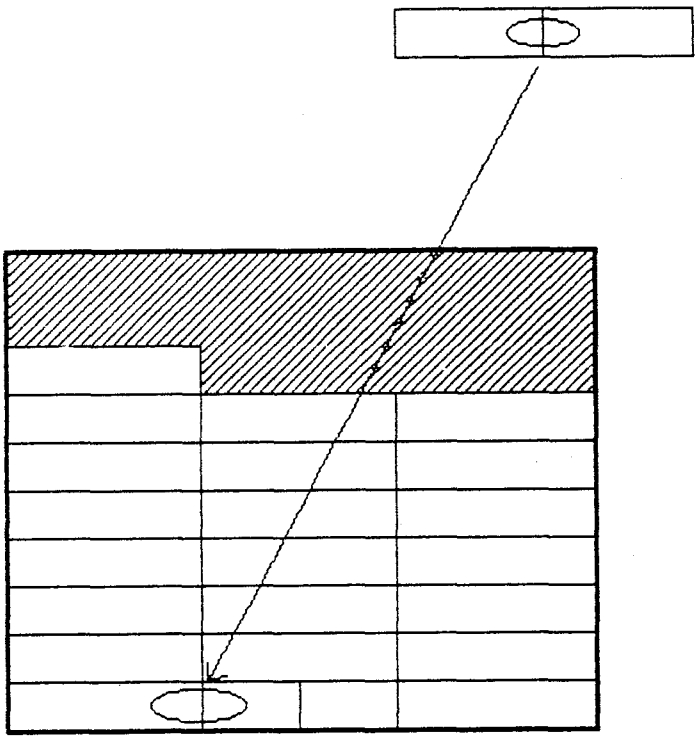


図２．５　アルゴリズム２．１によるパッキング例３
（手続きexchange）

かだかタイトでないレベルまでしか詰められず，この交換を行わなくても最適となる．また，W＝８の場合，幅２の空き地ができ，同じレベルにp_{41}がある場合は起こらない．なぜならば，この場合には必ず同じレベルにp_{21}があり，このp_{21}とp_{41}は既に手続きexchangeでp_{31}２個と交換されているはずである（図２．６）．

次に，手続きrepackでp_{31}が４個以上ある場合を考える．この場合が起こるのはもはやこれ以上p_{31}をp_{21}，p_{41}と交換できない場合であるから，タイトでないレベルにはp_{41}がなくてp_{21}が２個以下，あるいはp_{21}がなくてp_{41}が２個以下の場合のみである．ここで問題となるのはp_{21}がなくてp_{41}が２個の場合である．この場合にはp_{41}を２個先に詰めてしまうと，右端に幅２の空き地ができ反例となることがある（図２．７）．そこで，このように右端に幅２の空き地ができた場合は，p_{41}を１個取り除き幅６の空き地を作り，

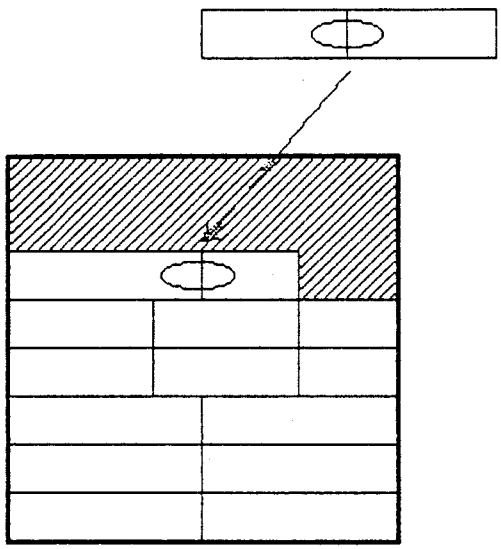


図２．６　アルゴリズム２．１によるパッキング例４
（W＝８の場合）

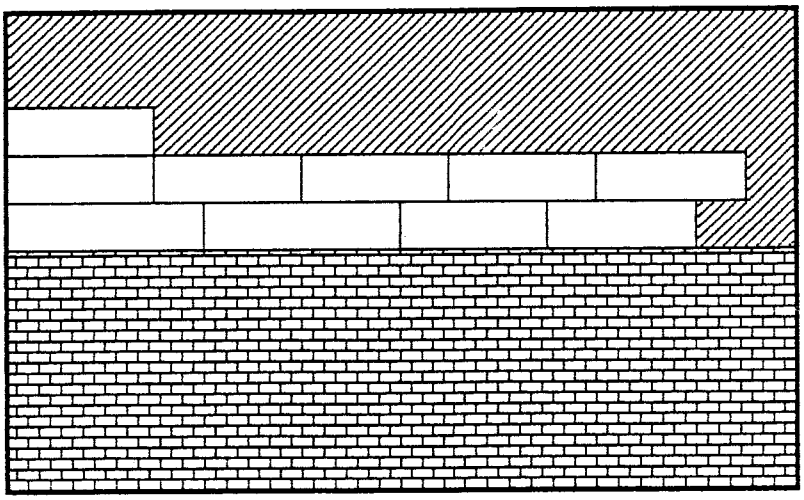


図２．７　アルゴリズム２．１によるパッキング例５
（手続きrepack）

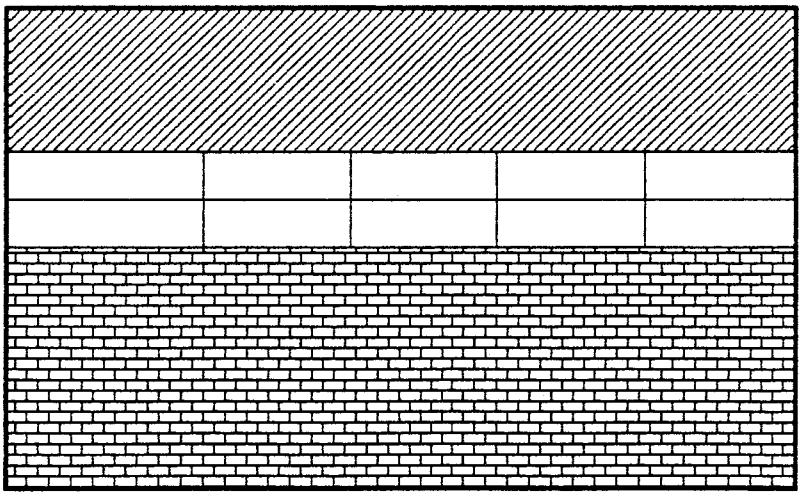


図２．８　アルゴリズム２．１によるパッキング例６
（手続きrepack）

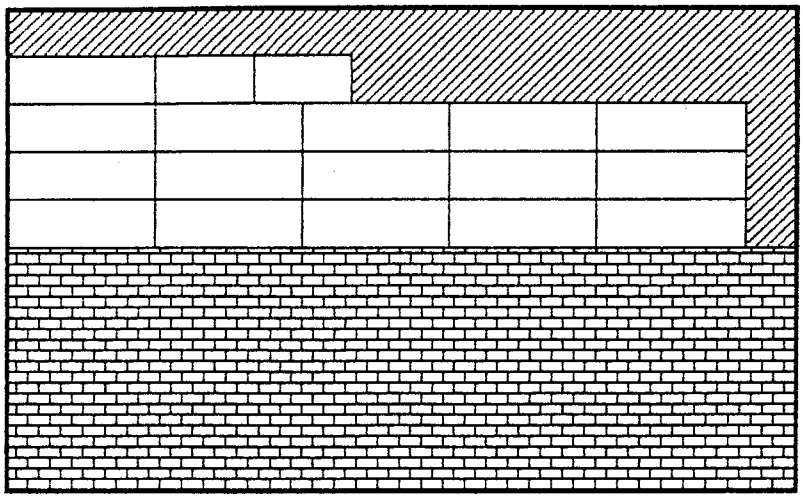


図２．９　アルゴリズム２．１によるパッキング例７
（手続きrepack）

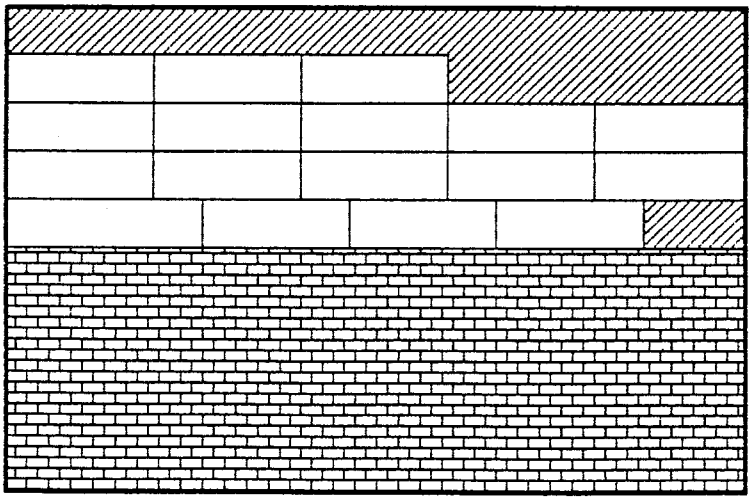


図２．１０　アルゴリズム２．１によるパッキング例８
（手続きrepack）

そこにp_{31}を２個詰めればそのレベルはタイトとなる．この上にp_{41}をＢＬで詰め，残りのp_{31}をＢＬで詰めると明らかに最適となる（図２．８）．次に，p_{41}がなくてp_{21}が２個以下の場合は，p_{31}，p_{21}の順にＢＬで詰めれば明らかに最適となる（図２．９）．また，p_{21}がなくてp_{41}が１個以下の場合も，p_{41}，p_{31}，p_{21}の順にＢＬで詰めれば，空き地ができることもあるが最適となる（図２．１０）．

よって，Ｗ≧６で，Ｗ mod ４＝０の場合は最適に詰めることができる（図２．１１）．

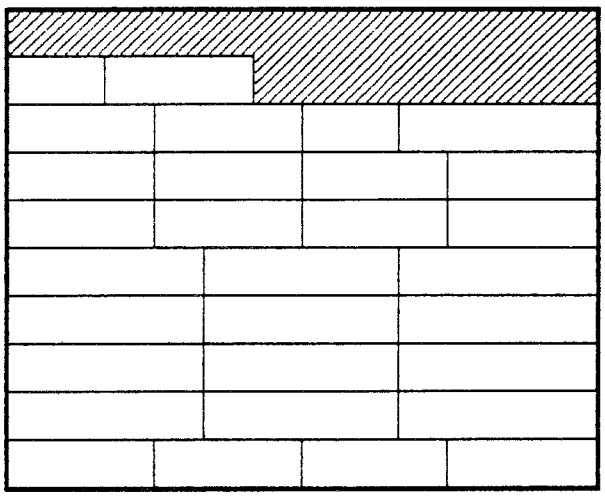


図２．１１　アルゴリズム２．１によるパッキング例９
（手続きＷｍ０）

（４）　Ｗ≧６で，Ｗ mod ４＝１（手続きＷｍ１）の場合

この場合はＷが奇数であるので，タイトに詰めるためには各レベルにp_{31}が必要である．そのためには，step１でp_{31}を右端に１列に詰める．p_{31}の左には幅Ｗ’（Ｗ’ mod ４＝２）のビンがあるこ

とになる．step2でp_{41}を詰め，step3でその右にできた幅2の空き地にp_{21}を詰める．もし，p_{31}，p_{21}がタイトなp_{41}の最高レベルまで詰めることができるならば，手続きWm1を行った後は，幅W'の一番上のレベルを除いてタイトである．

ここで，もしp_{21}がタイトなp_{41}の最高レベルまで詰めることができなければ（step4），一番右のp_{41}1個を取り除き，幅6の空き地を作り，そこにp_{31}を2個詰めることによってタイトとなるようにする（図2．12）．この操作をタイトなp_{41}の最高レベルまで，あるいは，右端に詰まっているp_{31}を上から順に2個取ると，その時点でタイトに詰まっている最高レベルの高さ（H_t）以下になるまで行う．この操作を行うごとに，その時点でタイトに詰まっている最高レベルの高さ（H_t）が1つ上がる．もしp_{31}が2個取

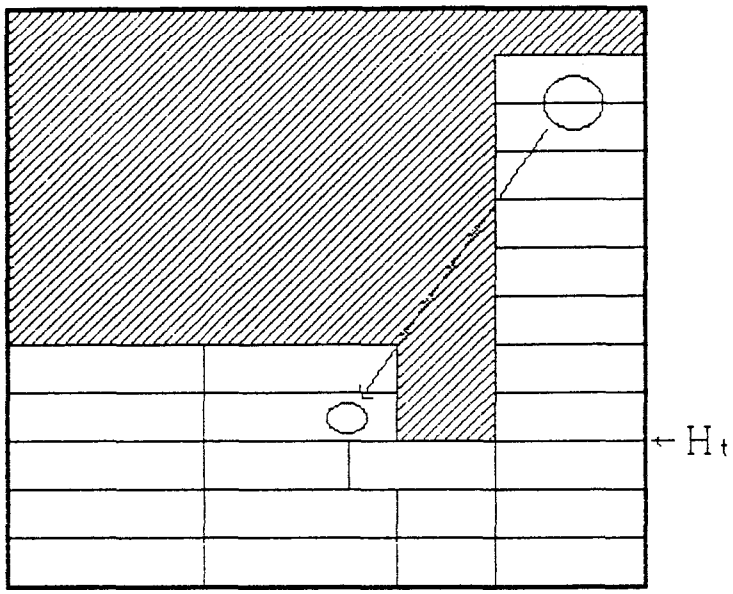


図2．12　アルゴリズム2．1によるパッキング例10
（手続きWm1）

るとタイトに詰まっている最高レベルの高さ以下になるならば，タイトに詰めることはできないので，上の複数個のレベルがタイトでないまま手続きＷｍ１を終了する．ここで，タイトに詰まっていないレベルのp_{31}はたかだか２個である．なぜならば，p_{21}はないので，p_{41}とp_{31}のみで１つのレベルをタイトにするためには，Ｗ mod ４＝１の場合，p_{31}が少なくとも３個必要だからである．次に，手続きexchangeを行う．この場合には，たかだかp_{31}は２個しかないので，交換はたかだか１回しか起こらない．次に，手続きrepackを行う．p_{31}は２個以下であるので，step１でp_{41}，p_{31}，p_{21}の順にＢＬで詰める．この場合は，右端に空き地ができるが，p_{41}とp_{31}２個以下とではタイトに詰めることができないので最適である（図２．１３）．もしp_{31}が十分あり，p_{41}を１個取り除くことによってできた幅６の空き地にp_{31}をタイトなp_{41}の最高レベルまで詰めることができるならば，手続きＷｍ１を行った後は，幅Ｗ’の一番上のレベルを除いてタイトである．

次に，もしp_{31}がタイトなp_{41}の最高レベルまで詰めることができず，p_{21}がp_{31}の最高レベルまで詰めることができるならば（step５），p_{31}の最高レベルまでp_{21}を詰める．step６で残ったp_{21}をＢＬで詰めて手続きＷｍ１を終了する．ただし，右端に空き地が残る（図２．１４）ので手続きrepackで詰め直す．p_{31}はないので手続きexchangeは行われない．次に，手続きrepackで上のタイトでないレベルに詰まっているp_{41}，p_{21}がその順でＢＬで詰められる．これは右端に幅１の空き地ができるが，p_{31}がないので最適なパッキングである（図２．１５）．

最後に，step６で残ったp_{21}をＢＬで詰めるときに，p_{31}が少なくp_{31}の上にp_{21}が詰められる場合がある．この場合も右端に幅１の空き地ができるが，p_{31}がないので手続きrepackで最適に詰めることができる（図２．１６）．

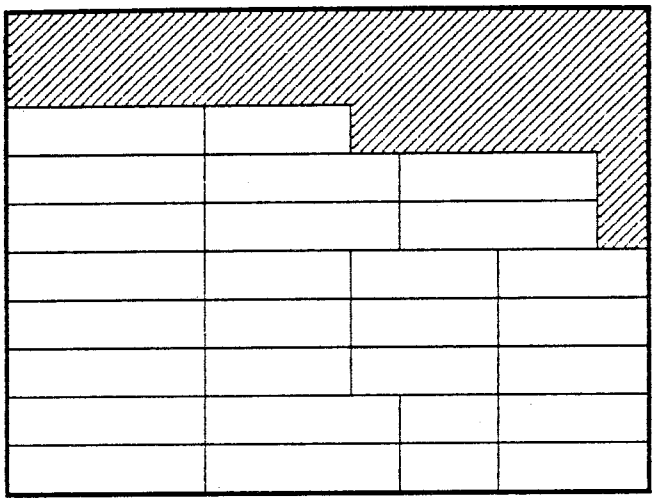


図２．１３　アルゴリズム２．１によるパッキング例１１
（手続きＷｍ１）

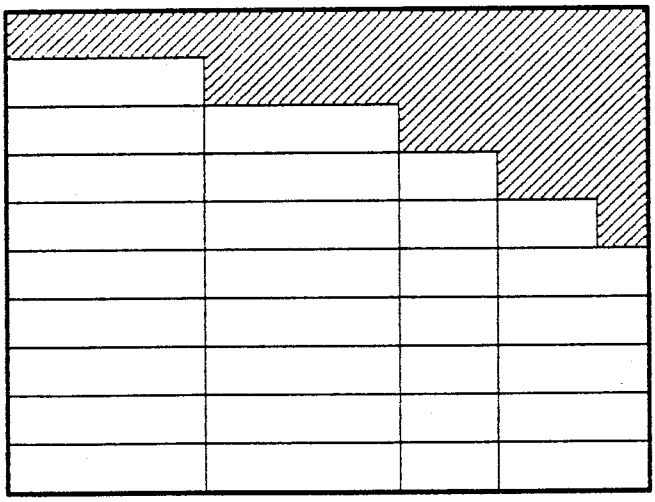


図２．１４　アルゴリズム２．１によるパッキング例１２
（手続きＷｍ１）

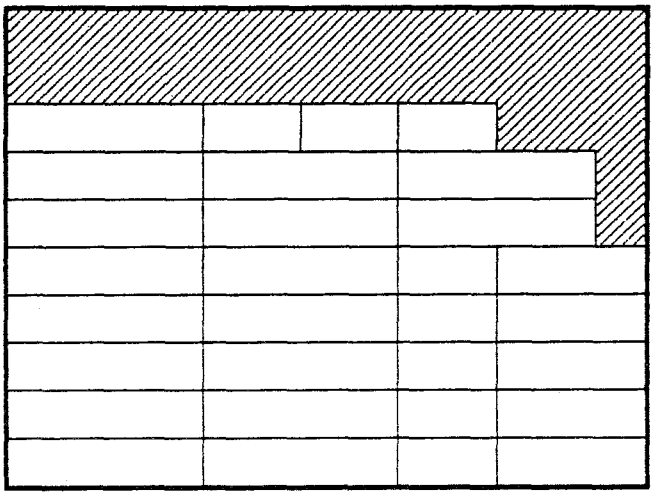


図２．１５　アルゴリズム２．１によるパッキング例１３
（手続きＷｍ１）

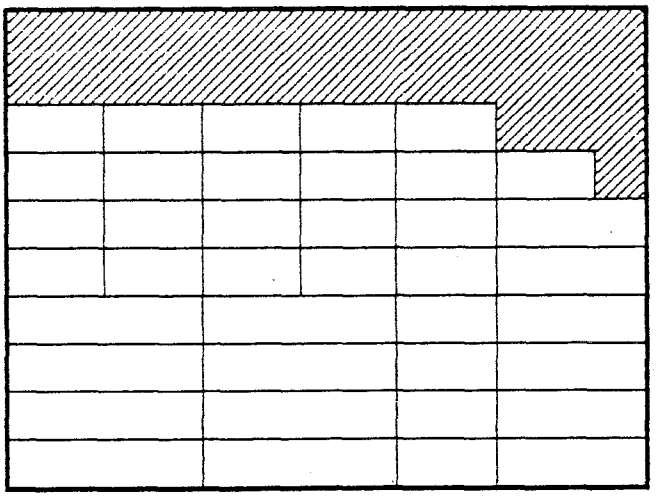


図２．１６　アルゴリズム２．１によるパッキング例１４
（手続きＷｍ１）

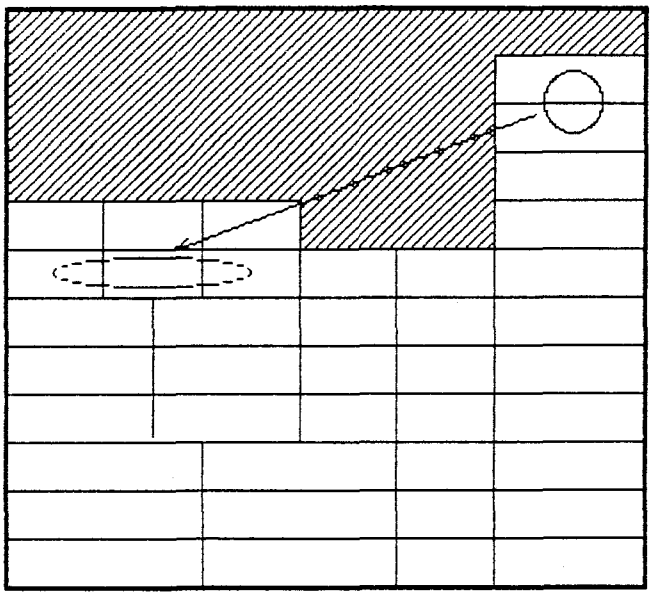


図２．１７　アルゴリズム２．１によるパッキング例１５
（手続きexchange）

幅W’の一番上のレベルを除いてタイトである状態で手続きWm１を終了した場合は，手続きexchangeを行う（図２．１７）．この場合も（３）の場合と同様に，手続きexchange終了後は，幅W’の一番上のレベルを除いてタイトであり，p_{31}ができる限りp_{21}，p_{41}と交換されている．

ここで，p_{21}，p_{41}が十分あり，p_{31}が手続きexchangeでできる限り交換されるならば，p_{31}は３個以下である．３個以下の場合も（３）の場合と同様に，手続きrepackで一番上のタイトでない（唯一の）レベルの続きにp_{31}をＢＬで詰めれば最適となる．また，手続きrepackでp_{31}が４個以上ある場合も（３）の場合と同様に，step２に従って詰めれば最適となる．

よって，W≧６で，W mod ４＝１の場合は最適に詰めることができる（図２．１８）．

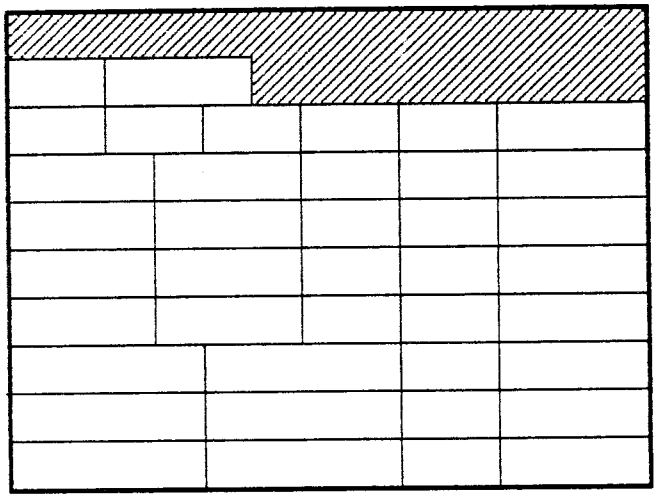


図２．１８　アルゴリズム２．１によるパッキング例１６
（手続きWm１）

（５）　W≧６で，W mod ４＝２（手続きWm２）の場合

この場合，Wは偶数であるが４の倍数ではないので，step１でp_{41}を詰めたとき，右端に幅２の空き地ができる．step２でこの空き地にp_{21}を詰める．ここで，もしp_{21}がタイトなp_{41}の最高レベルまで詰めることができなければ（step３），一番右のp_{41}１個を取り除き，幅６の空き地を作り，そこにp_{31}を２個詰めることによってタイトとなるようにする（図２．１９）．この操作をタイトなp_{41}の最高レベルまで，あるいはp_{31}が足りなくなる（１個以下になる）まで行う．もしp_{31}も足りない（１個以下）ならば，タイトに詰めることはできないので，上の複数個のレベルがタイトでないまま手続きWm２を終了する．次に，手続きexchangeを行う．この

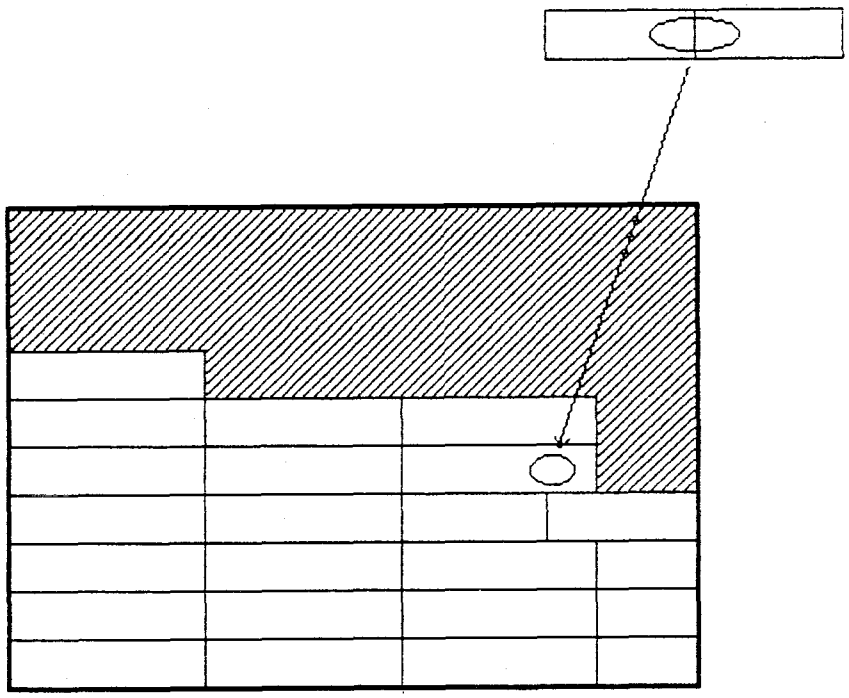


図２．１９　アルゴリズム２．１によるパッキング例１７
（手続きＷｍ２）

場合には，p_{31}はたかだか１個しかないので，交換は起こらない．次に，手続きrepackを行う．p_{31}は１個以下であるので，step１でp_{41}，p_{31}の順にＢＬで詰める．この場合は，右端に空き地ができるが，p_{41}とp_{31}１個以下とではタイトに詰めることができないので最適である（図２．２０）．もしp_{31}が十分あり，p_{41}を１個取り除くことによってできた幅６の空き地にp_{31}をタイトなp_{41}の最高レベルまで詰めることができるならば，手続きＷｍ２を行った後は，一番上のレベルを除いてタイトである．

また，p_{21}が十分あり，タイトなp_{41}の最高レベルまで詰めることができるならば，step４で残ったp_{21}をＢＬで詰める．この場合も手続きＷｍ２を行った後は，一番上のレベルを除いてタイトである．

一番上のレベルを除いてタイトである状態で手続きＷｍ２を終了した場合は，手続きexchangeを行う（図２．２１）．この場合も

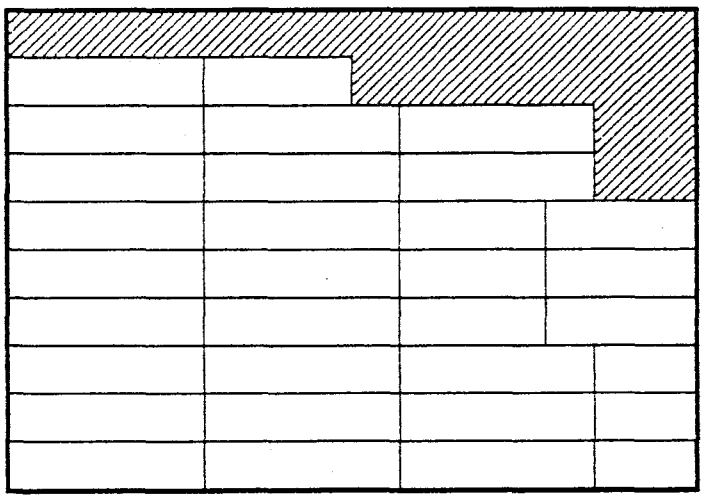


図２．２０　アルゴリズム２．１によるパッキング例１８
（手続きＷｍ２）

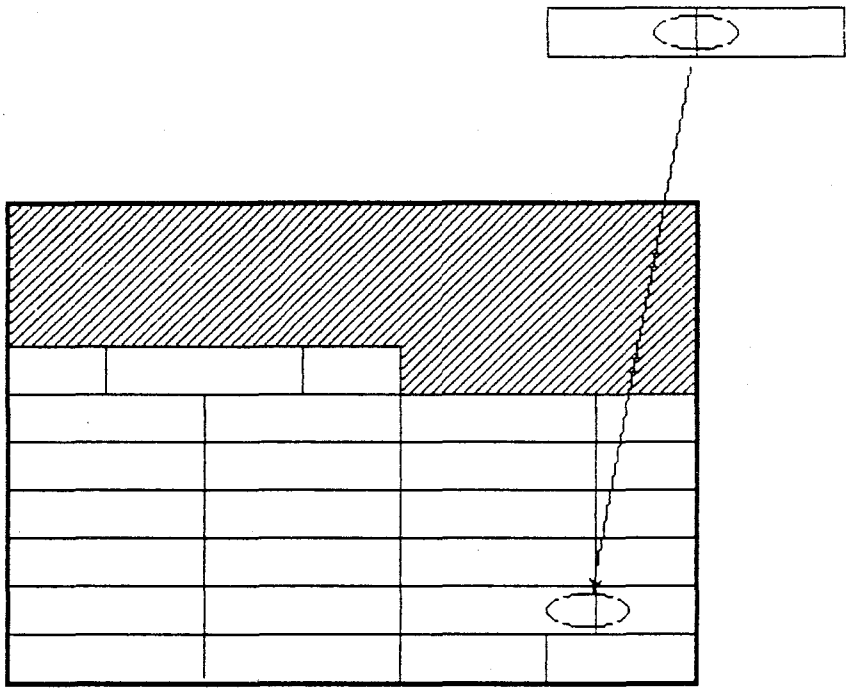


図２．２１　アルゴリズム２．１によるパッキング例１９
（手続きexchange）

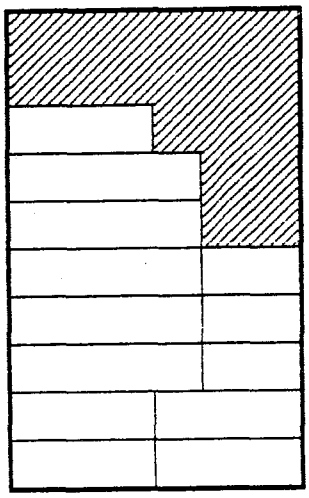


図２．２２　アルゴリズム２．１によるパッキング例２０
（Ｗ＝６の場合）

（３）の場合と同様に，手続きexchange終了後は，一番上のレベルを除いてタイトであり，p_{31}ができる限りp_{21}，p_{41}と交換されている．

ここで，p_{21}，p_{41}が十分あり，p_{31}が手続きexchangeでできる限り交換されるならば，p_{31}は３個以下である．３個以下の場合も（３）の場合と同様に，Ｗ≧９のときは，手続きrepackで一番上のタイトでない（唯一の）レベルの続きにp_{31}をＢＬで詰めれば最適となる．また，Ｗ＝６の場合も，Ｗが３の倍数でp_{31}がタイトに詰まるので，p_{31}を後で詰めても最適となる（図２．２２）．また，手続きrepackでp_{31}が４個以上ある場合も（３）の場合と同様に，step２に従って詰めれば最適となる．

よって，Ｗ≧６で，Ｗ mod ４＝２の場合は最適に詰めることができる（図２．２３）．

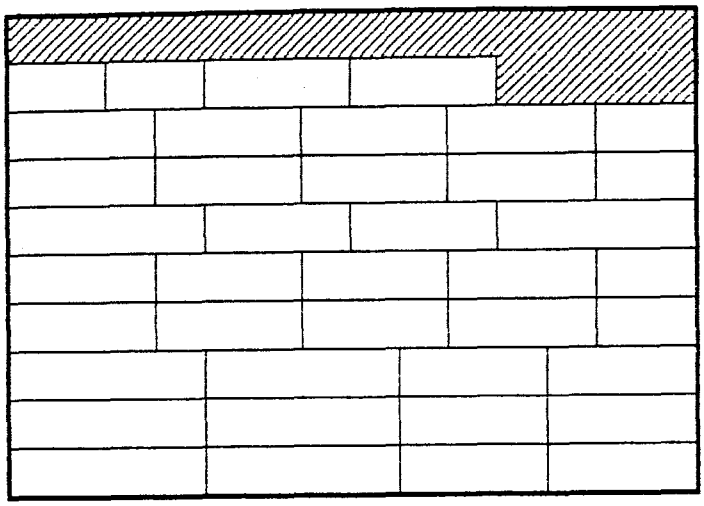


図２．２３　アルゴリズム２．１によるパッキング例２１
（手続きＷｍ２）

（６）　Ｗ≧６で，Ｗ mod ４＝３（手続きＷｍ３）の場合

この場合はＷが奇数であるので，タイトに詰めるためには各レベルにp_{31}が必要である．そのためには，step１でp_{31}を右端に１列に詰める．p_{31}の左には幅Ｗ’（Ｗ’ mod ４＝０）のビンがあることになる．次に，step２，step３で，p_{41}，p_{21}をＢＬで詰める．もし，p_{31}が十分あり，タイトなp_{41}，p_{21}の最高レベルまで詰めることができるならば，手続きＷｍ３を行った後は，幅Ｗ’の一番上のレベルを除いてタイトである．

次に，もしp_{31}がタイトなp_{41}の最高レベルまで詰めることができないならば，step３で右端の幅３の空き地にp_{21}をＢＬで詰める．この場合，右端に幅１の空き地ができるが，p_{31}がないので最適である．また，もしp_{21}もタイトなp_{41}の最高レベルまで詰めることができない場合も，右端に幅３の空き地ができるがp_{41}しかないの

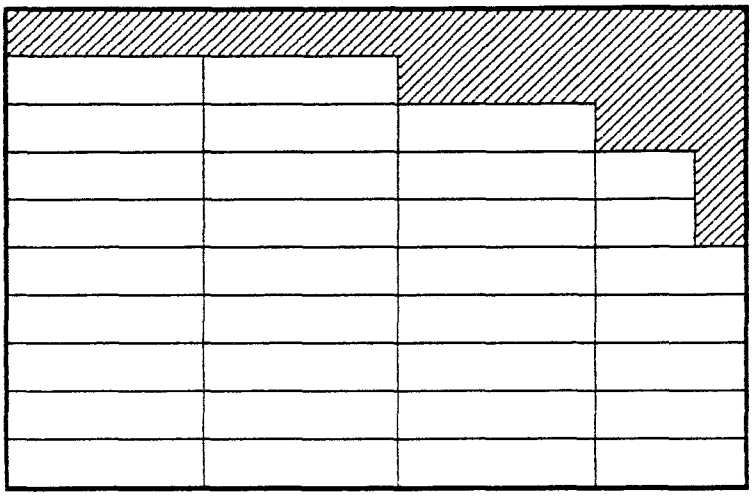


図２．２４　アルゴリズム２．１によるパッキング例２２
（手続きＷｍ３）

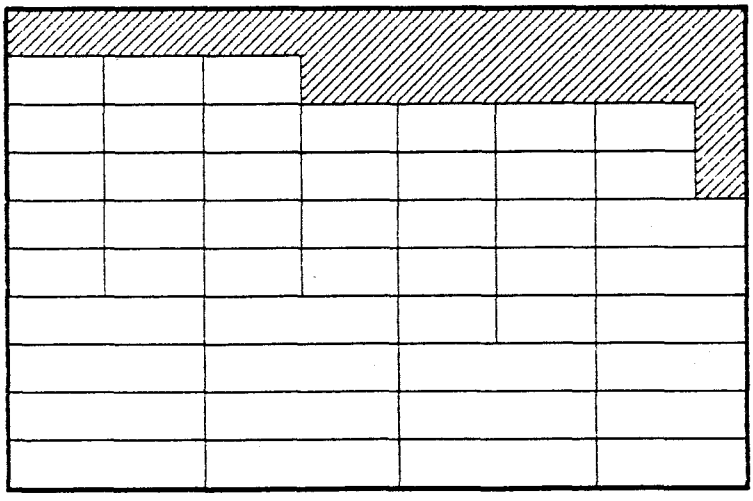


図２．２５　アルゴリズム２．１によるパッキング例２３
（手続きＷｍ３）

で最適である（図２．２４）．また，p_{31}がタイトなp_{41}の最高レベルまで詰めることができたが，step３を実行中にp_{21}がp_{31}のレベルを越えてしまった場合でも，右端に幅１の空き地ができるがp_{31}がないので最適である（図２．２５）．

幅W’の一番上のレベルを除いてタイトである状態で手続きWm３を終了した場合は，手続きexchangeを行う（図２．２６）．この場合も（３）の場合と同様に，手続きexchange終了後は，幅W’の一番上のレベルを除いてタイトであり，p_{31}ができる限りp_{21}，p_{41}と交換されている．

ここで，p_{21}，p_{41}が十分あり，p_{31}が手続きexchangeでできる限り交換されるならば，p_{31}は３個以下である．３個以下の場合も（３）の場合と同様に，W≧９のときは，手続きrepackで一番上の

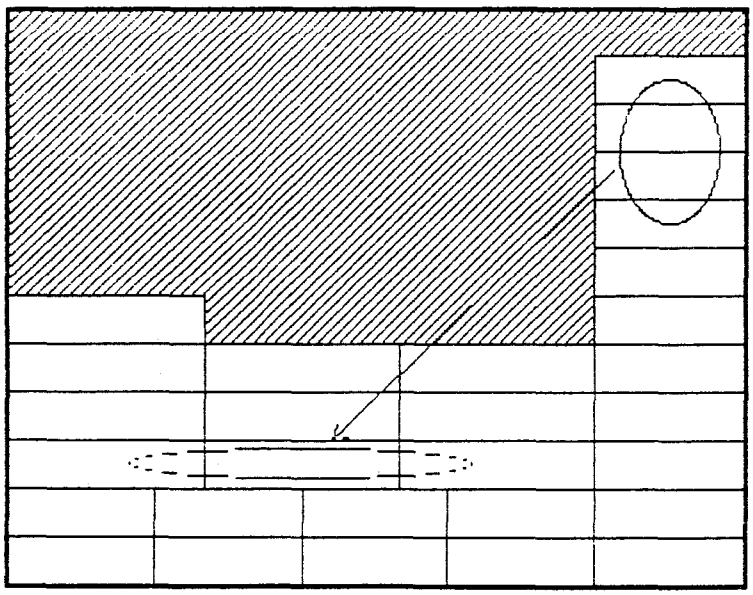


図２．２６　アルゴリズム２．１によるパッキング例２４
（手続きexchange）

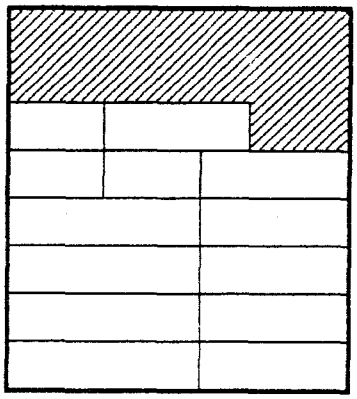


図２．２７　アルゴリズム２．１によるパッキング例２５
（Ｗ＝７の場合）

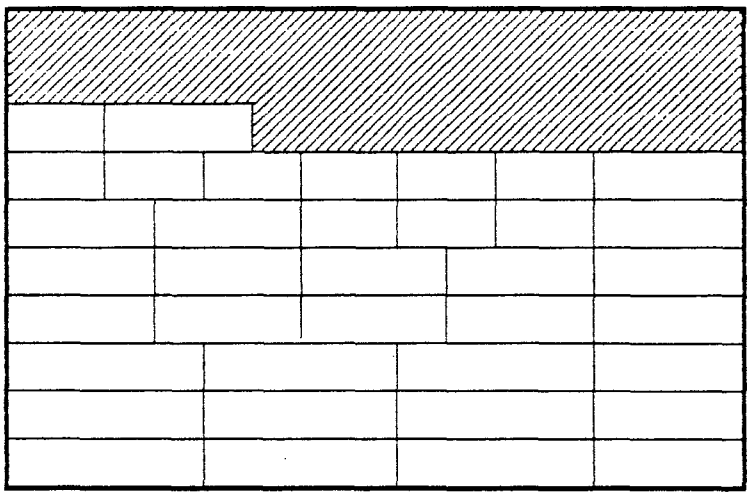


図２．２８　アルゴリズム２．１によるパッキング例２６
（手続きＷｍ３）

タイトでない（唯一の）レベルの続きにp_{31}をＢＬで詰めれば最適となる．また，Ｗ＝７の場合も，幅２の空き地ができ同じレベルにp_{41}がある場合は起こらないので，最適となる（図２．２７）．また，手続きrepackでp_{31}が４個以上ある場合も（３）の場合と同様に，step２に従って詰めれば最適となる．

よって，Ｗ≧６で，Ｗ mod ４＝３の場合は最適に詰めることができる（図２．２８）．

（７）　アルゴリズムの時間計算量

このアルゴリズムがＯ（ｎ）回のＢＬパッキングで行うことができることを示す．まず，Ｗ＝４，Ｗ＝５の場合は詰め直すことを行わないので，ｎ回のＢＬパッキングで行うことができる．Ｗ≧６の場合は，まず，手続きＷｍ１，手続きＷｍ２で，p_{41}を詰め直すことがあるが，p_{31}２個とp_{41}を交換するので，この回数は１／２・n_{31}回以下である．また，手続きexchangeでp_{21}，p_{41}を詰め直すことがあるが，これはたかだかp_{41}とp_{21}２個との交換，p_{41}，p_{21}とp_{31}２個との交換，そしてp_{41}とp_{21}２個との交換だけであるから，この回数は７／２・n_{31}回以下である．また，手続きrepackではすべてのピースをもう１度だけ詰め直すことがある．なお，ピースを交換する際，交換すべきピースを見つける操作は各幅のピースが何個，どのレベルにあるかを覚えておくことにより，定数時間でできる．以上のことから，ＢＬパッキングはたかだか（２・ｎ＋４・n_{31}）回以下である．よって，このアルゴリズムはＯ（ｎ）回のＢＬパッキングで行うことができる．　□

定理２．１から，ピースの大きさを４以下に限定した場合の１次元ビン・パッキング問題は，多項式時間で解けることになる．

２．３　ピースの幅が一定である場合

ピースの幅が一定であるという制約を加えると，問題はスケジュ

ーリング問題における最大滞留時間最小問題[4]となる．この問題に関しては次の定理が導ける．

［定理２．２］　ピースの種類をｈｘ１の部分矩形に限定した場合の厳密解法が存在するとする．このとき，この解法を利用して，ピースの種類を１ｘｈの部分矩形に限定した場合の最適解を求めるアルゴリズム（アルゴリズム２．２）が存在する．ただし，ｈｘ１の部分矩形に対する厳密解法の時間計算量がＯ（f(n)）であるならば，この厳密解法の時間計算量はＯ（h･f(n)）となる．なお，ｎはピース数である．　□

アルゴリズム２．２

step０　このアルゴリズムにおいてｈｘ１の部分矩形に対する厳密解法を行う場合は，１ｘｈの部分矩形であるすべてのピースを，幅と高さを交換してｈｘ１の部分矩形であるとする；

step１　ピースの面積の総和をビンの幅Ｗで割ると下界が求まる；　それをＬとする；

step２　repeat

・Ｌをビンの幅と考えて，ｈｘ１の部分矩形に対する厳密解法を行う；

・もし最適解H_{opt}が実際のビンの幅Ｗを越えるならば，Ｌを１つ増やす；

until　最適解H_{opt}が実際のビンの幅Ｗ以下となる；

（証明）　このアルゴリズムで求まる解は，ｈｘ１の部分矩形に対する厳密解法を使って，下界からＬ（実際のビンの高さ）を１つずつ増やしていくので，明らかに最適である（図２．２９）．

また，１ｘｈの部分矩形の最適なパッキングの中には，最適解

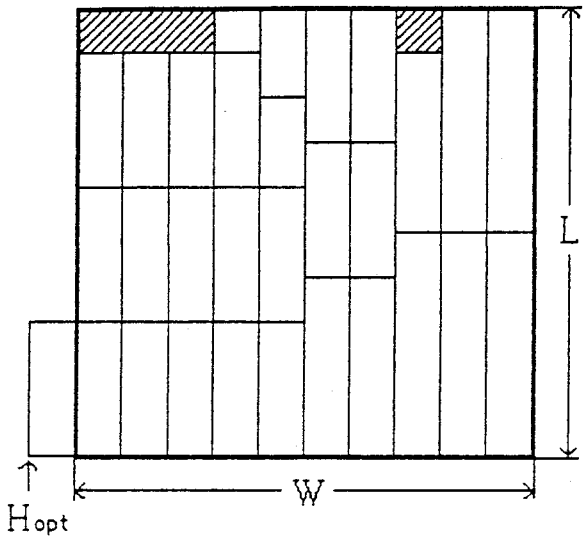


図２．２９　アルゴリズム２．２によるパッキング例

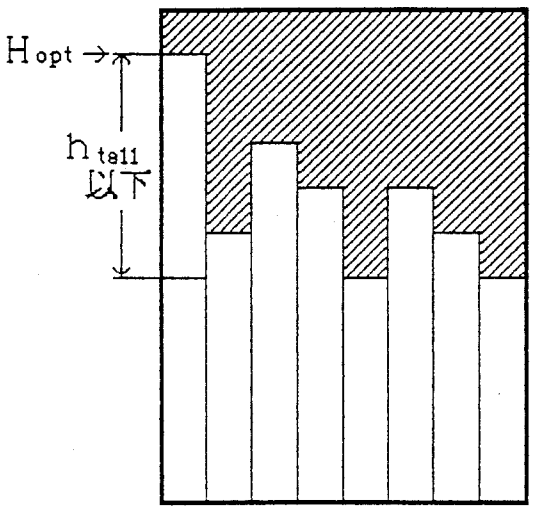


図２．３０　１ｘｈの部分矩形の最適なパッキング例

H_{opt}よりたかだかh_{tall}だけ低い部分を除いてタイトに詰まっているものが，少なくとも１つある（図２．３０）．ただし，h_{tall}は最も高いピースの高さである．したがって，最適解は，下界＋h_{tall}未満である．ゆえに，１ｘｈの部分矩形に限定した場合，step２の操作はたかだかｈ回しか行われない．よって，このアルゴリズムの時間計算量はＯ（h･f(n)）となる．　□

ピースの種類を４ｘ１の部分矩形に限定した場合の厳密解法（アルゴリズム２．１）が存在するので，この定理から次の系が導かれる．

［系２．１］　ピースの種類を４ｘ１の部分矩形に限定した場合の厳密解法（アルゴリズム２．１）を利用して，ピースの種類を１ｘ４の部分矩形に限定した場合の最適解を求めるアルゴリズムが存在する．また，このアルゴリズムはＯ（ｎ）回のＢＬパッキングで行うことができる．ただし，ｎはピース数である．　□

系から，ジョブの処理時間を４以下に限定した場合の最大滞留時間最小問題は，多項式時間で解けることになる．

２．４　ピースの種類が２ｘ２の部分矩形である場合

本節では，ピースの種類をｗｘｈの部分矩形に限定し，ｗおよびｈをパラメータとして多項式時間の厳密解法が考えられるかについて考察する．そして，ピースの種類を２ｘ２の部分矩形に限定した場合の厳密解法を提案し，次の定理を導いている．

［定理２．３］　ピースの種類を２ｘ２の部分矩形に限定する．このとき，ピースをアルゴリズム２．３で詰めると，必ず最適に詰めることができる．また，このアルゴリズムはＯ（ｎ）回のＢＬパッキングで行うことができる．ただし，Ｗはビンの幅であり，ｎはピース数である．　□

アルゴリズム２．３

step１　p_{22}をＢＬで詰める；　もし右端（Ｗが奇数のときは右端の１つ左）まで詰めることができなければ，そのレベルのピースはすべて詰めないでおく；　すなわち，p_{22}を$\lfloor n_{22}/\lfloor W/2 \rfloor \rfloor$列をＢＬで詰め，その高さは$2\cdot\lfloor n_{22}/\lfloor W/2 \rfloor \rfloor$である；

step２　p_{21}をＢＬで詰める；　もし右端（Ｗが奇数のときは右端の１つ左）まで詰めることができなければ，そのレベルのピースはすべて詰めないでおく；　また，もし詰めたレベルが奇数の場合は，一番上のレベルのピースはすべて詰めないでおき，高さを偶数にする；　すなわち，p_{21}を$2\cdot\lfloor n_{21}/(2\cdot\lfloor W/2 \rfloor)\rfloor$列詰め，その高さは$2\cdot\lfloor n_{21}/(2\cdot\lfloor W/2 \rfloor)\rfloor$である；

step３　もしＷが奇数ならば，p_{12}を右端の幅１の空き地に左のピースの高さまで詰める；

step４　残ったp_{22}をＢＬで詰める；

step５　p_{12}をＢＬで詰める；

step６　残ったp_{21}をＢＬで詰める；

step７　p_{11}をＢＬで詰める；

（証明）　このアルゴリズムでは，［定理２．１］の証明の場合と同様に，p_{11}はないものとして考える．

step１，step２では，Ｗが偶数のときはタイトなパッキングである．Ｗが奇数のとき，step３でp_{12}を右端の幅１の空き地に詰める．左に詰められているピースの高さは偶数であるので，もしp_{12}が十分あり，左のピースの高さまで詰めることができるならば，step３終了後はＷが奇数の場合もタイトに詰められる．もしp_{12}が左のピースの高さまで詰めることができないならば，右端に幅１の空き地

ができるが，幅１のピースがないので最適なパッキングである（図２．３１）．

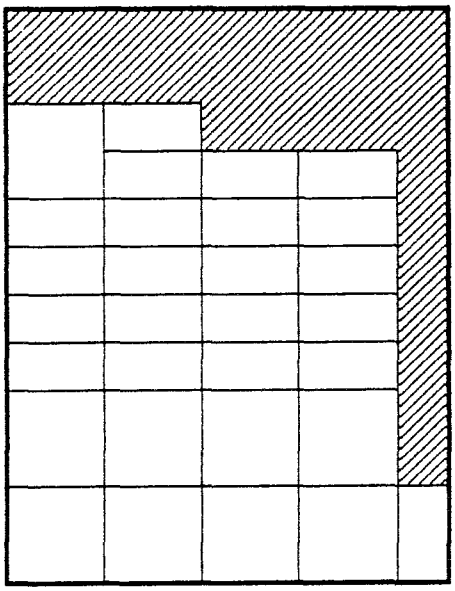


図２．３１　アルゴリズム２．３によるパッキング例１

step４では，１列に満たない（たかだか⌊Ｗ／２⌋－１個の）p_{22}をＢＬで詰め，step５では，その続きにp_{12}をＢＬで詰める．ここまでは明らかにタイトなパッキングである．次に，（たかだか２・⌊Ｗ／２⌋－１個の）p_{21}をＢＬで詰めるのであるが，このときＷが偶数，奇数であるに関わらず，右端に１ｘ２の空き地ができる可能性がある．しかし，残ったp_{21}を詰めたときの高さは，たかだか２であるので（∵p_{21}はたかだか２・⌊Ｗ／２⌋－１個），p_{12}を１つ取り除き，そこにp_{21}を２つ詰め，空き地をなくしてしまい，p_{12}をその上のレベルに詰めるという操作をとる必要はない．ゆえに，この場合も最適なパッキングである（図２．３２）．

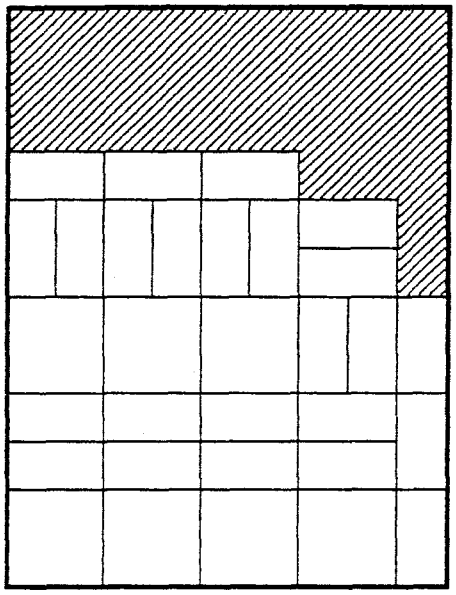


図２．３２　アルゴリズム２．３によるパッキング例２

このアルゴリズムでは，step１で残った，たかだか⌊Ｗ／２⌋－１個のピースの詰め直しと，step２で残った，たかだか２・⌊Ｗ／２⌋－１個のピースの詰め直しがある（これは，前もって，ピースの数からレベルの数を計算することによって，避けることができる）だけで，それ以外に詰め直しは起こらない．したがって，このアルゴリズムはＯ（ｎ）回のＢＬパッキングで行うことができる．　□

２．５　結言

本章では，ビンの幅，ピースの幅，高さを整数値に制約した２次元ビン・パッキング問題を取り上げ，その厳密解法を考察した．この制約のもとで，ピースの種類をｗｘｈの部分矩形に限定すること

によって，線形時間で最適解が得られるかどうかについて考察した．その結果，少なくともピースの種類が４ｘ１，１ｘ４，２ｘ２の部分矩形である場合までは，線形時間で最適解が得られることを明らかにした．

第３章　ピースに制約を加えた２次元ビン・パッキング問題の近似解法

３．１　緒言

本章では，ピースに制約を加えた２次元ビン・パッキング問題に対する３つの近似解法を提案し，その良さを評価する．ｗｘｈの部分矩形においてｗおよびｈの値が前章で見い出した厳密解法の整数値を越えると，もはや時間計算量がピース数ｎの線形時間となる厳密解法は期待できそうにない状況にあり，近似解法を主題とせざるをえないと思われる．このような場合でも，前章で見い出した厳密解法の考え方や解法そのものを活用することにより，良い近似解法を構成することができる可能性があると思われる．

まず，近似解法の最悪の場合の理論的解析を考え，ピースの種類を３ｘ２の部分矩形に限定した場合の近似解法を提案する．この近似解法は前章のピースの種類が４ｘ１の部分矩形である場合の厳密解法の考え方を基にしたものであり，線形時間で最悪の場合でもたかだか高さが１しか悪くならない近似解が得られることを明らかにする．

次に，ピースの種類をｗｘ１およびｗｘ２（ｗ≧５）の部分矩形に限定した場合の近似解法を提案し，その良さを既存の近似解法であるレベル・アルゴリズムとの比較実験を通して評価する．ピースの種類がｗｘ１の部分矩形である場合の近似解法は，前章のピースの種類が４ｘ１の部分矩形である場合の厳密解法を活用したものであり，ピースの種類がｗｘ２の部分矩形である場合の近似解法は，前述のピースの種類が３ｘ２の部分矩形である場合の近似解法の考え方を基に，ピースの種類がｗｘ１の部分矩形である場合の近似解法を活用したものである．

３．２　近似解法の理論的解析

～ピースの種類が３ｘ２の部分矩形である場合

本節では，ピースの種類を３ｘ２の部分矩形に限定した場合について議論する．この場合には常に最適解が得られるとは限らないが，最悪の場合でもたかだか高さが１しか悪くならない近似解法を提案し，次の定理を導いている．

［定理３．１］　ピースの種類を３ｘ２の部分矩形に限定する．このとき，ピースをアルゴリズム３．１で詰めると，近似度は次のようになる．

$H_A \leqq H_{opt} + 1$

また，このアルゴリズムは０（ｎ）回のＢＬパッキングで行うことができる．ここで，H_Aは近似解法でのパッキングの高さ，H_{opt}は最適なパッキングの高さである．ただし，Ｗはビンの幅であり，ｎはピース数である．　□

アルゴリズム３．１

メイン・ルーチン

step１　幅２以上の高さ１のピース（p_{31}，p_{21}）を縦に２つ並べて，高さ２のピースとする；　ただし，n_{31}，n_{21}が奇数となり，ピースが１つ余る場合は，余ったピースはstep７で詰めるので，そのまま残しておく；

step２　if　Ｗ＝偶数　then　　手続きevenを行う；

step３　if　Ｗ＝奇数　then　　手続きoddを行う；

step４　if　（Ｗが偶数のとき，Ｗ≧６）or
（Ｗが奇数のとき，Ｗ≧９）then
手続きexchangeを行う；

step５　タイトに詰まっていないレベルで，
手続きrepackを行う；

step6　p_{12}をＢＬで詰める；

step7　残ったピース（たかだか，p_{31}とp_{21}が１ずつ）をＢＬで詰める；

step8　p_{11}をＢＬで詰める；

手続きeven

step1　p_{22}をＢＬで詰める；

手続きodd

step1　p_{32}を右端に１列に詰める；

step2　p_{22}をＢＬで詰める；

手続きexchange

step0　この手続きにおいて交換されるp_{32}は，Ｗが偶数のときはフリーのp_{32}であり，Ｗが奇数のときは右端に１列に詰められたp_{32}である；　このピースをp_{ex}とする；
Ｗが偶数のときは，交換することができるフリーのp_{32}がなくなる（２個未満になる）まで，以下の操作を行う；　また，Ｗが奇数のときは，交換するために右端の一番上のp_{32}を取り除くことによって，その時点でタイトに詰まっている最高レベルの高さ（H_t）未満になるならば，交換することを行わず，この手続きは終了する；

step1　同じレベルのp_{22}３個をp_{ex}２個と交換し，交換されたp_{22}３個をＢＬで詰める；

手続きrepack

step1　残ったp_{32}が１個以下ある場合は，p_{22}，p_{32}の順に

ＢＬで詰める；

step２　残ったp_{32}が２個以上ある場合は，p_{32}，p_{22}の順にＢＬで詰める；

（証明）　このアルゴリズムでは，［定理２．１］の証明の場合と同様に，p_{11}はないものとして考える．

（１）　Ｗ＝偶数（手続きeven）の場合

この場合はＷが偶数であるので，手続きevenのstep１で，p_{22}を

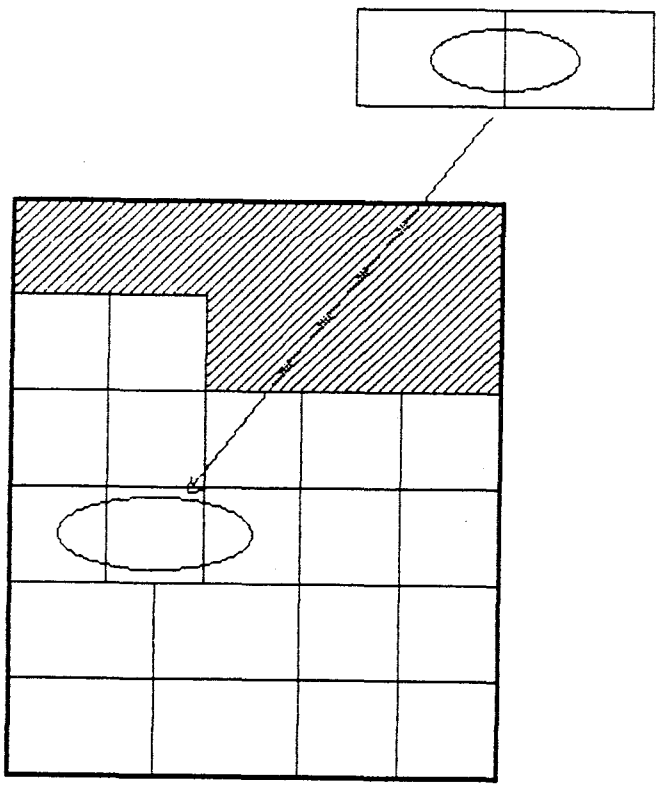


図３．１　アルゴリズム３．１によるパッキング例１
（手続きexchange）

ＢＬで詰めたとき，一番上のレベルを除いてタイトである．

次に，Ｗ≧６のとき手続きexchangeを行い，できる限り，かつタイトを崩さずに，p_{32}をp_{22}と交換する（図３．１）．p_{32}をp_{22}と交換する方法は，p_{32}２個とp_{22}３個との交換のみが考えられる．よって，手続きexchange終了後も，一番上のレベルを除いてタイトであり，p_{32}ができる限りp_{22}と交換されている．

ここで，p_{22}が十分あり，p_{32}が手続きexchangeでできる限り交換されているならば，手続きrepackで残ったp_{32}は１個以下である．p_{32}が１個以下ならば，手続きrepackで一番上のタイトでない（唯一の）レベルの続きにp_{32}をＢＬで詰める．この場合は，一般には，一番上の高さ２のレベルを除いてタイトとなるが，上の２つの高さ２のレベルがタイトとならない場合がある（図３．２）．しかし，

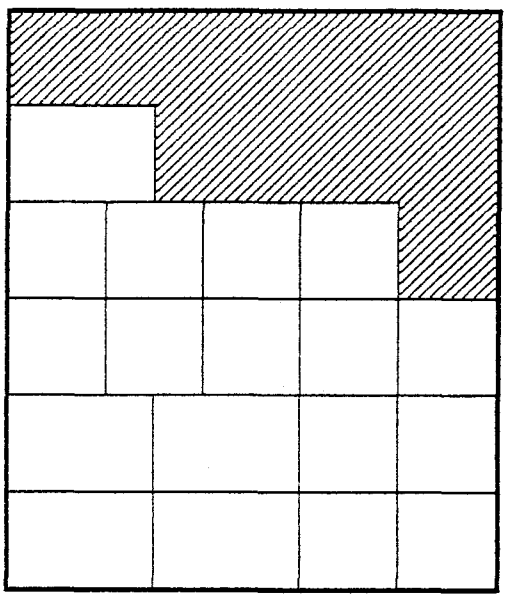


図３．２　アルゴリズム３．１によるパッキング例２
（手続きrepack）

この場合でも面積の下限から考えて，この解法によるパッキングの高さは最適解に比べてたかだか１しか悪くならないので近似度は保たれる．その後，step６でp_{12}がＢＬで詰められ，step７で残っている（たかだか１個の）p_{31}，p_{21}がＢＬで詰められる．この場合も，一般に，一番上の高さ２のレベルを除いてタイトである．ただし，step７でp_{31}，p_{21}を詰めるとき，高さ３のタイトでない領域を作る可能性がある（図３．３）．しかし，この場合も面積の下限から考えて，この解法によるパッキングの高さは最適解に比べてたかだか１しか悪くならないので近似度は保たれる．

次に，手続きrepackで残ったp_{32}が２個以上ある場合を考える．この場合が起こるのは，もうこれ以上p_{32}をp_{22}と交換できない場合であり，上のタイトでないレベルにはp_{22}は２個以下である．こ

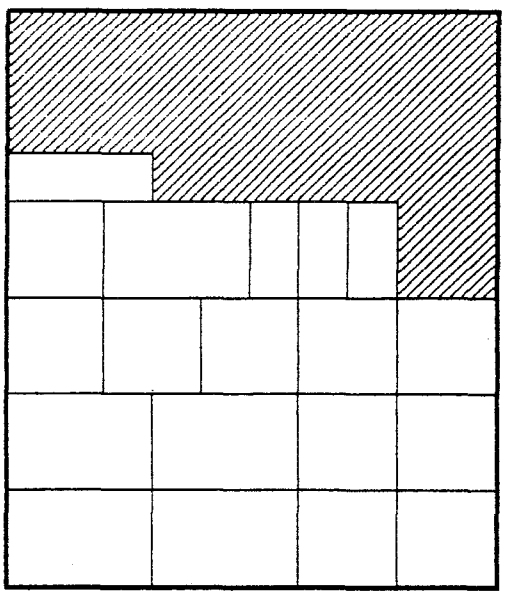


図３．３　アルゴリズム３．１によるパッキング例３
（手続きeven）

の場合は，たとえすべてのピースの高さが１であると考えても，たかだか１回を除いて幅３のピースと幅２のピースの交換はできないので，手続きrepackでp_{32}，p_{22}を順にＢＬで詰めれば近似度は保たれる．また，p_{21}２個とp_{31}，幅１の空き地との交換がたかだか２回考えられるが，これを行ってもパッキングの高さは最適解に比べてたかだか１しか悪くならないので近似度は保たれる．その後，step６でp_{12}がＢＬで詰められ，step７で残っている（たかだか１個の）p_{31}，p_{21}がＢＬで詰められる．この場合，p_{31}とp_{21}，幅１の空き地を交換して空き地を減らすことができるが，それを行わなくてもパッキングの高さは最適解に比べてたかだか１しか悪くならないので近似度は保たれる（図３．４）．

以上のことから，Ｗ＝偶数の場合にこのアルゴリズムで詰めると，

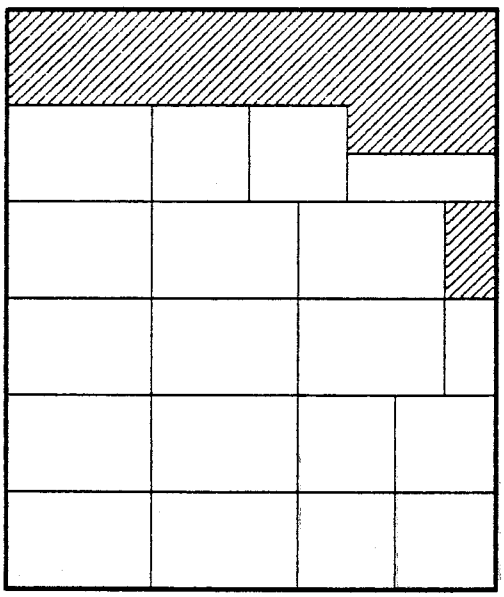


図３．４　アルゴリズム３．１によるパッキング例４
（手続きeven）

近似度は$H_A \leqq H_{opt} + 1$となる（図３．５）．

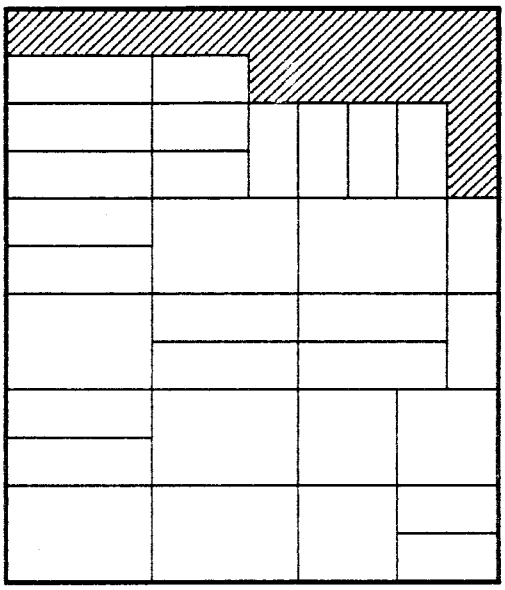


図３．５　アルゴリズム３．１によるパッキング例５
（手続きeven）

（２）　W＝奇数（手続きodd）の場合

この場合にはWが奇数であるので，タイトに詰めるためには各レベルにp_{32}が必要である．そのために手続きoddのstep１でp_{32}を右端に１列に詰める．p_{32}の左には，幅W’（W’＝偶数）のビンがあることになる．手続きoddのstep２では，p_{22}をＢＬで詰める．ここで，p_{32}が十分あり，タイトなp_{22}の最高レベルまで詰めることができるならば，手続きoddを行った後は，幅W’の一番上のレベルを除いてタイトである．

次に，もしp_{32}がタイトなp_{22}の最高レベルまで詰めることができないならば，手続きoddのstep２でp_{32}の上にp_{22}を詰めること

になり，右端に幅１の空き地ができる．しかし，p_{32}はないのでp_{12}以外で空き地をなくすることはできない．この場合，手続きexchangeは行われずに，手続きrepackへ進む．手続きrepackでも，p_{22}しかないので右端に幅１の空き地ができたまま詰められる．step６でp_{12}がＢＬでまず右端の幅１の空き地に詰められ，step７で残っているp_{31}，p_{21}がＢＬで詰められる．この場合，p_{31}とp_{21}，幅１の空き地を交換して空き地を減らすことができるが，それを行わなくてもパッキングの高さは最適解に比べてたかだか１しか悪くならないので近似度は保たれる（図３．６）．

幅Ｗ’の一番上のレベルを除いてタイトである状態で手続きoddを終了した場合は，Ｗ≧９のとき手続きexchangeを行う（図３．７）．この場合も（１）の場合と同様に，手続きexchange終了

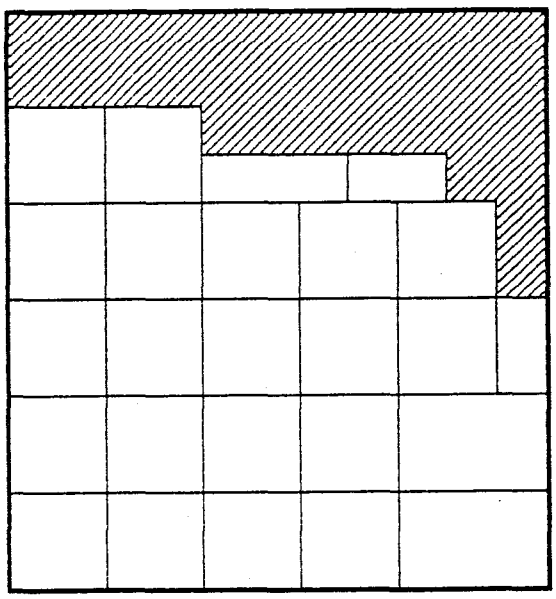


図３．６　アルゴリズム３．１によるパッキング例６
（手続きodd）

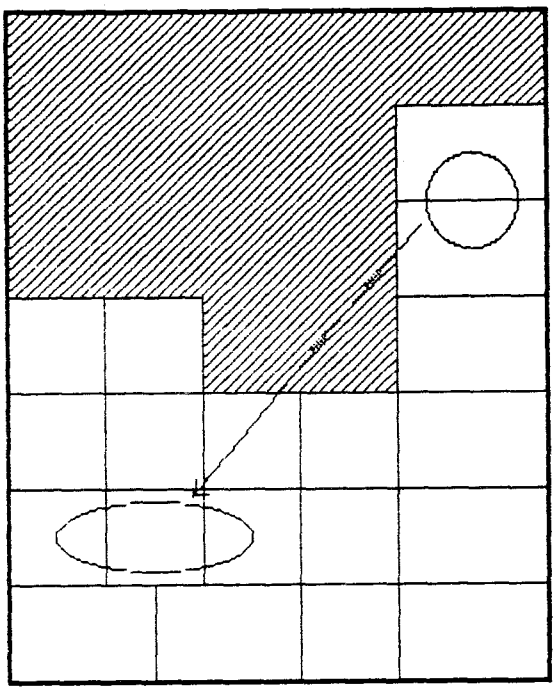


図３．７　アルゴリズム３．１によるパッキング例７
（手続きexchange）

後は，幅W’の一番上のレベルを除いてタイトであり，p_{32}ができる限りp_{22}と交換されている．

ここで，p_{22}が十分あり，p_{32}が手続きexchangeでできる限り交換されているならば，手続きrepackで残ったp_{32}は１個以下である．１個以下の場合も（１）の場合と同様に，手続きrepackで一番上のタイトでない（唯一の）レベルの続きにp_{32}をＢＬで詰めれば近似度は保たれる．また，手続きrepackでp_{32}が２個以上ある場合も（１）の場合と同様に，p_{32}，p_{22}の順にＢＬで詰めれば近似度は保たれる．

以上のことから，W＝奇数の場合にこのアルゴリズムで詰めると，近似度は$H_A \leqq H_{opt} + 1$となる（図３．８）．

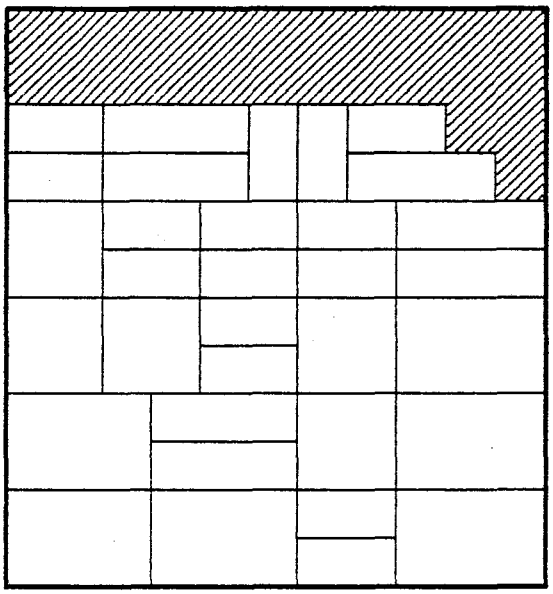


図３．８　アルゴリズム３．１によるパッキング例８
（手続きodd）

（３）　アルゴリズムの時間計算量

［定理２．１］の証明の場合と同様に，このアルゴリズムは，１つのピースをたかだか定数回しか詰め直すことを行わないので，０（ｎ）回のＢＬパッキングで行うことができる．　□

３．３　近似解法の計算機実験

本節では，ピースの種類をｗｘ１およびｗｘ２（ｗ≧５）の部分矩形に限定した場合について，前章のピースの種類が４ｘ１の部分矩形である場合の厳密解法を利用した近似解法を提案し，その良さ

を既存の近似解法であるレベル・アルゴリズム[6]との比較実験を通して評価している．

3.3.1　ピースの種類がｗｘ１およびｗｘ２の部分矩形である場合

まず，ピースの種類をｗｘ１（ｗ≧５）の部分矩形に限定した場合を問題にする．この場合には，ピースをｗが５以上のものと４以下のものに大別して処理する手法が考えられる．アルゴリズム３．２はこの考えに基づいて提案したものである．

アルゴリズム３．２

step１　幅ｗから幅５までのピースを幅の大きなピースから順にＢＬで詰める；

step２　step１で詰められた高さまで，幅４から幅１までのピースをその順にＢＬで詰める；

step３　その上に残った幅４から幅１までのピースをアルゴリズム２．１で詰める；

次に，ピースの種類をｗｘ２（ｗ≧５）の部分矩形に限定した場合を問題にする．この場合には，高さ１のピースを縦に２つ重ねてすべて高さ２のピースとして扱うものと，高さ１のまま残るたかだかそれぞれ１個のピースに大別して処理する手法が考えられる．アルゴリズム３．３は，この考えに基づいて，高さが２のピースに対してアルゴリズム３．２を適用するとともに，高さ１のピースを常にＢＬで詰めるのではなく，より良い解にするための交換手続きを工夫したものである．

アルゴリズム３．３

step１　幅２以上で高さ１のピースについて，同じ幅のピースを

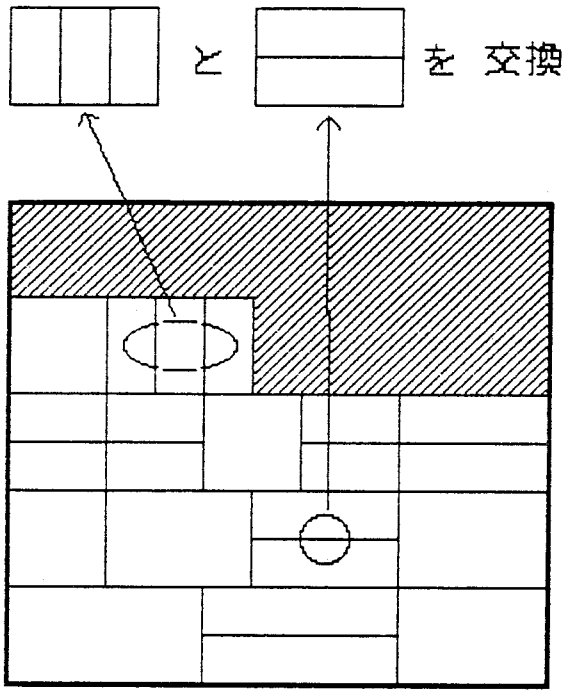


図３．９　アルゴリズム３．３のstep３の操作

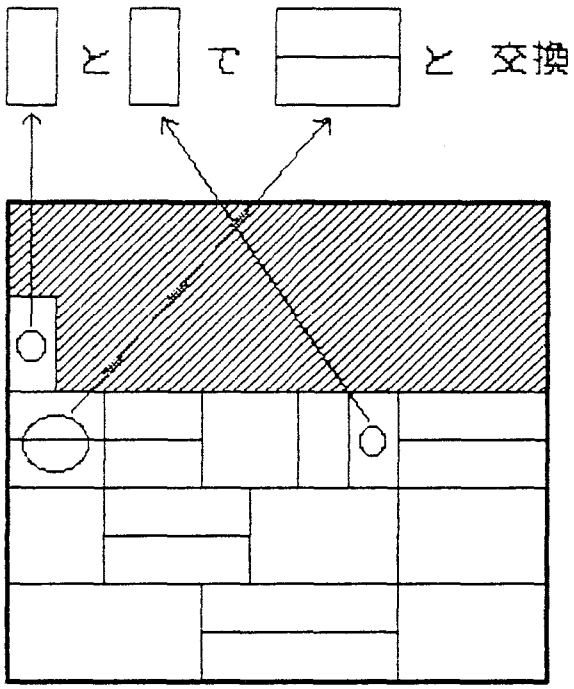


図３．１０　アルゴリズム３．３のstep４の操作

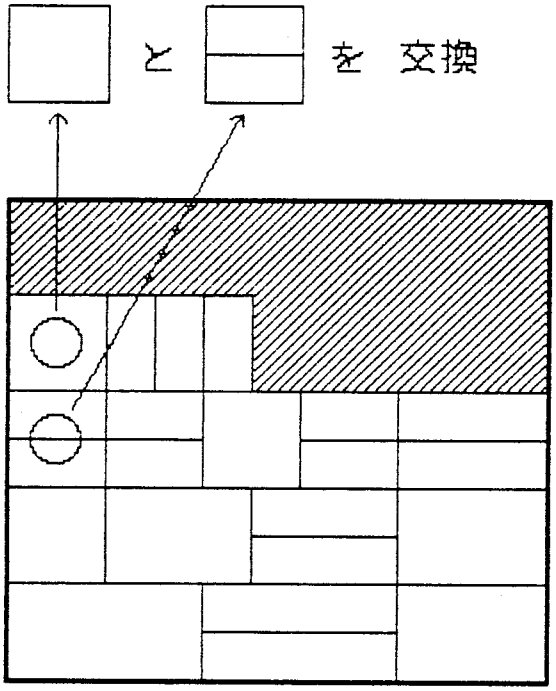


図３．１１　アルゴリズム３．３のstep５の操作

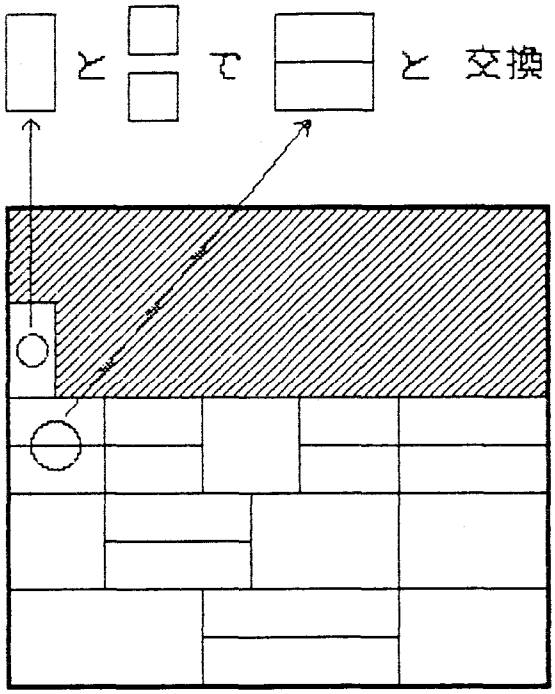


図３．１２　アルゴリズム３．３のstep６の操作

縦に２つ並べて高さ２のピースとする；　ただし，ピースが１つ余る場合はそのまま残しておく；

step２　高さ２のピースを前述のアルゴリズム３．２で詰める；

step３　タイトでない領域のｍ個（ｍ≧２）のp_{12}を高さ１のピースを重ねて作られたp_{m2}と交換する（図３．９）；

step４　タイトでない領域のp_{12}が１つのときは，タイトな領域からp_{12}を１つ取ってきて，それらのピースを高さ１のピースを重ねて作られたp_{22}と交換する（図３．１０）；

step５　タイトでない領域の高さ２のピースで作られたピースと，タイトな領域の高さ１のピースを重ねて作られた同一幅のピースを交換する（図３．１１）；

step６　タイトでない領域にp_{12}がまだ１つだけ残っている場合は，p_{11}２個よりp_{12}を作り，これらのp_{12}２個を高さ１のピースを重ねて作られたp_{22}と交換する（図３．１２）；

step７　残っている高さ１のピースと交換された高さ１のピースを幅の大きなピースから順にＢＬで詰める；

step８　p_{11}をＢＬで詰める；

3.3.2　計算機実験による評価

前項で提案した近似解法を，計算機実験によって既存の近似解法であるＦＦＤＨ－レベル・アルゴリズム[6]との比較で評価する．比較基準として，ＦＦＤＨ－レベル・アルゴリズムを取り上げたのは，このアルゴリズムが最悪の場合の解の振舞いの点から考えて，最も良い近似解法の１つであることが知られているからである．

ふつう，近似解法での評価基準は，“近似解／最適解の平均値”である近似度を用いる．しかし，本節では，“すべての近似解の中で最適解と一致した解の割合（％）”を最適率と呼び，これを評価

基準に用いる．この理由は，アルゴリズム３．２，アルゴリズム３．３で求まった近似解がすべて最適解＋１で収まり，近似度に差異が少なかったからである．

計算機実験にあたっては，問題例を（１）矩形分割法（ＲＤＧ），（２）乱数ピース発生法（ＲＰＧ）によって生成するものとし，ビンの幅を１０から４０の範囲で，ピース数を４０から４００の範囲で変更して試行し，各試行に対しては１００個の問題例を生成している．なお，ＲＰＧ法により生成される問題例に対しては，ピースの面積の総和から下界を求め，それを最適解の代わりに用いる．

１）アルゴリズム３．２の評価

アルゴリズム３．２とレベル・アルゴリズムの実験による最適率を表３．１に示す．ピースの幅の最大値が５の場合は，ＲＤＧもＲＰＧも１６試行（１６００例）行い，ピースの幅の最大値が８と１５の場合は，いずれも，ＲＤＧもＲＰＧも８試行（８００例）行った．アルゴリズム３．２は問題例６４００すべてに対して最適解を得ている．

表３．１からわかるように，ＲＰＧよりＲＤＧで問題例を生成した方が解が悪い傾向にある．これはＲＤＧで生成した最適解はすべてタイトな最適解であり，最適解の自由度が少ないためであると考えられる．また，表３．１ではわからないが，各試行による最適率の差は大きく，ピース数が多くなればなるほど最適率が良くなる傾向にあった．

２）アルゴリズム３．３の評価

アルゴリズム３．３とレベル・アルゴリズムの実験による最適率を表３．２に示す．ピースの幅の最大値が５の場合は，ＲＤＧは２０試行（２０００例），ＲＰＧでは１６試行（１６００例）行い，ピースの幅の最大値が８と１５の場合は，いずれも，ＲＤＧもＲＰＧも８試行（８００例）行った．アルゴリズム３．３はいずれ

表３．１　アルゴリズム３．２の実験結果（最適率）

ピースの幅の	レベル・アルゴリズム		アルゴリズム３．２	
最大値（w）	ＲＤＧ	ＲＰＧ	ＲＤＧ	ＲＰＧ
５	69.9%	94.0%	100.0%	100.0%
８	24.6%	75.3%	100.0%	100.0%
１５	15.4%	69.3%	100.0%	100.0%

表３．２　アルゴリズム３．３の実験結果（最適率）

ピースの幅の	レベル・アルゴリズム		アルゴリズム３．３	
最大値（w）	ＲＤＧ	ＲＰＧ	ＲＤＧ	ＲＰＧ
５	4.6%	46.1%	100.0%	100.0%
８	0.6%	19.8%	99.6%	98.9%
１５	0.3%	12.3%	96.9%	97.3%

の場合にも９５％以上の最適率を得ている．

アルゴリズム３．２の場合と同様に，ＲＰＧよりＲＤＧで問題例を生成した方が解が悪い傾向にある．また，表３．２ではわからないが，アルゴリズム３．２の場合と同様に，各試行による最適率の差は大きく，ピース数が多くなればなるほど，最適率は良くなる傾向にあった．また，ピースの幅の最大値ｗが大きくなればなるほど，

すなわち，ピースの種類が多くなればなるほど，最適率は悪くなる傾向にあった．

３．４　結言

本章では，ビンの幅，ピースの幅，高さを整数値に制約した２次元ビン・パッキング問題を取り上げ，その近似解法を考察した．まず，近似解法の理論的解析として，ピースの種類を３ｘ２の部分矩形に限定した場合の近似解法を提案し，その解法が線形時間で最悪の場合でもたかだか高さが１しか悪くならない近似解が得られることを示した．次に，ピースの種類をｗｘ１およびｗｘ２（ｗ≧５）の部分矩形に限定した場合の近似解法を提案し，その良さを既存の近似解法であるレベル・アルゴリズムとの比較実験を通して評価した．実験結果から，提案した２つの近似解法は，計算時間も妥当であり，性能の点でも良い近似解法であると評価している．

第４章　ボロノイ図を応用した真円度を求める解法

４．１　緒言

本章では，ボロノイ図を応用した真円度を求める多項式時間の厳密解法を提案し，その最適性と時間計算量を明らかにする．

真円度[16]は，機械部品の精度として重要な幾何公差（形状公差）の１つであり，“円筒断面の輪郭を２つの同心円ではさんだとき，両円の間隔が最小となる両円の半径差”と定義される（図４．１）．

真円度を求める手法としては，これまでのところ，最小自乗法による解法，Min-Max法を適用した解法，シンプレックス法を用いた解法等が考えられているが，近似解法に過ぎないとか，厳密解法で

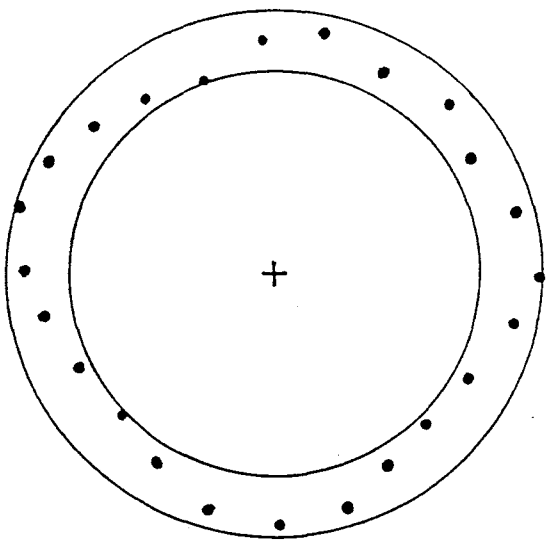


図４．１　　真円度

あっても時間計算量に難点があるものであって，時間計算量が多項式時間である厳密解法は知られていない．

４．２　最近点および最遠点のボロノイ図とそれらの構成算法

4.2.1　最近点および最遠点のボロノイ図

ボロノイ図（勢力圏図）[8][13]とは，次のように定義されるものである．

“平面上のｎ個の点$p_i=(x_i, y_i)$（$i=1,2,\cdots,n$）に対して，点p_iの‘勢力圏’$V(p_i)$を

$$V(p_i)=\bigcap_{j=1}^{n}\{p=(x,y)\mid d(p_i,p)\leqq d(p_j,p)\}$$

と定め，$V(p_i)$（$i=1,2,\cdots,n$）による平面分割をボロノイ図と呼

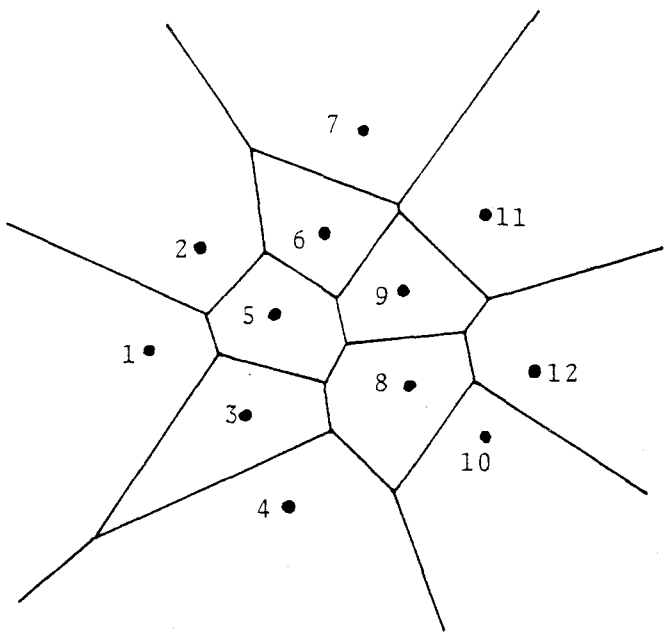


図４．２　最近点のボロノイ図

ぶ（図４．２）”．ここで，$d(p_i, p)$は２点p_i，pのユークリッド距離，すなわち，$d^2(p_i, p)=(x-x_i)^2+(y-y_i)^2$である．

ボロノイ図において，‘勢力圏’$V(p_i)$を点p_iのボロノイ領域と呼び，これは凸多角形である．また，ボロノイ領域の頂点をボロノイ点，辺をボロノイ辺と呼ぶ．

以上は，最近点（勢力圏）を表すボロノイ図であるが，これに対して，最遠点（非勢力圏）を表すボロノイ図[13]も考えられ，次のように定義される．

“平面上のｎ個の点$p_i=(x_i, y_i)$ $(i=1,2,\cdots,n)$ に対して，点p_iの‘非勢力圏’$U(p_i)$を

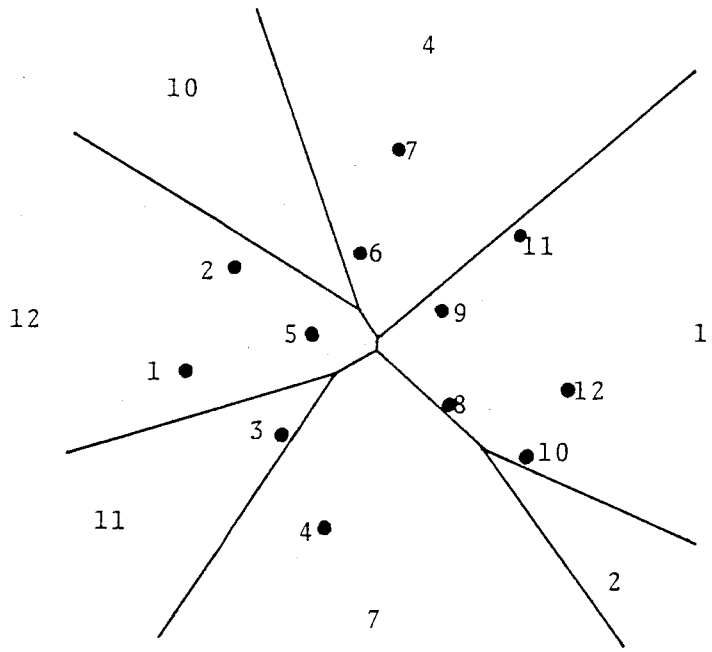


図４．３　最遠点のボロノイ図

$$U(p_i)=\bigcap_{j=1}^{n}\{p=(x,y)\mid d(p_i,p)\geqq d(p_j,p)\}$$

と定め，$U(p_i)$ $(i=1,2,\cdots,n)$ による平面分割を最遠点のボロノイ図と呼ぶ（図４．３）”．

最遠点のボロノイ図の場合はすべての点が領域を持つわけではなく，与えられたｎ個の点の凸包上の点のみ領域を持つ．

4.2.2 最近点および最遠点のボロノイ図の構成算法

ボロノイ図の構成算法は，分割統治法を用いる手法が考えられている．最近点のボロノイ図の構成算法をアルゴリズム４．１に示す[13]．

アルゴリズム４．１

step１　ｎ個の点をｘ座標の値に関してソートし，その順にp_1，…，p_nとする；

step２　これらの点を左側の点の集合Ｌ＝$\{p_1, \cdots, p_{[n/2]}\}$と右側の点の集合Ｒ＝$\{p_{[n/2]+1}, \cdots, p_n\}$に二等分し，それぞれに対するボロノイ図Ｖ（Ｌ），Ｖ（Ｒ）を再帰的に構成する；

step３　点集合Ｌ，Ｒから等距離な点の集合の軌跡（折れ線）を求め，その線より左にあるＶ（Ｒ），その線より右にあるＶ（Ｌ），それぞれのボロノイ点，ボロノイ辺を除くことによって，２つのボロノイ図Ｖ（Ｌ），Ｖ（Ｒ）を併合する（図４．４）；

step３の併合は併合する図の点の数に比例する手間で行えるので，アルゴリズム４．１の時間計算量はＯ（ｎlogｎ）である[13]．

次に，最遠点のボロノイ図の構成にあたっては，アルゴリズム

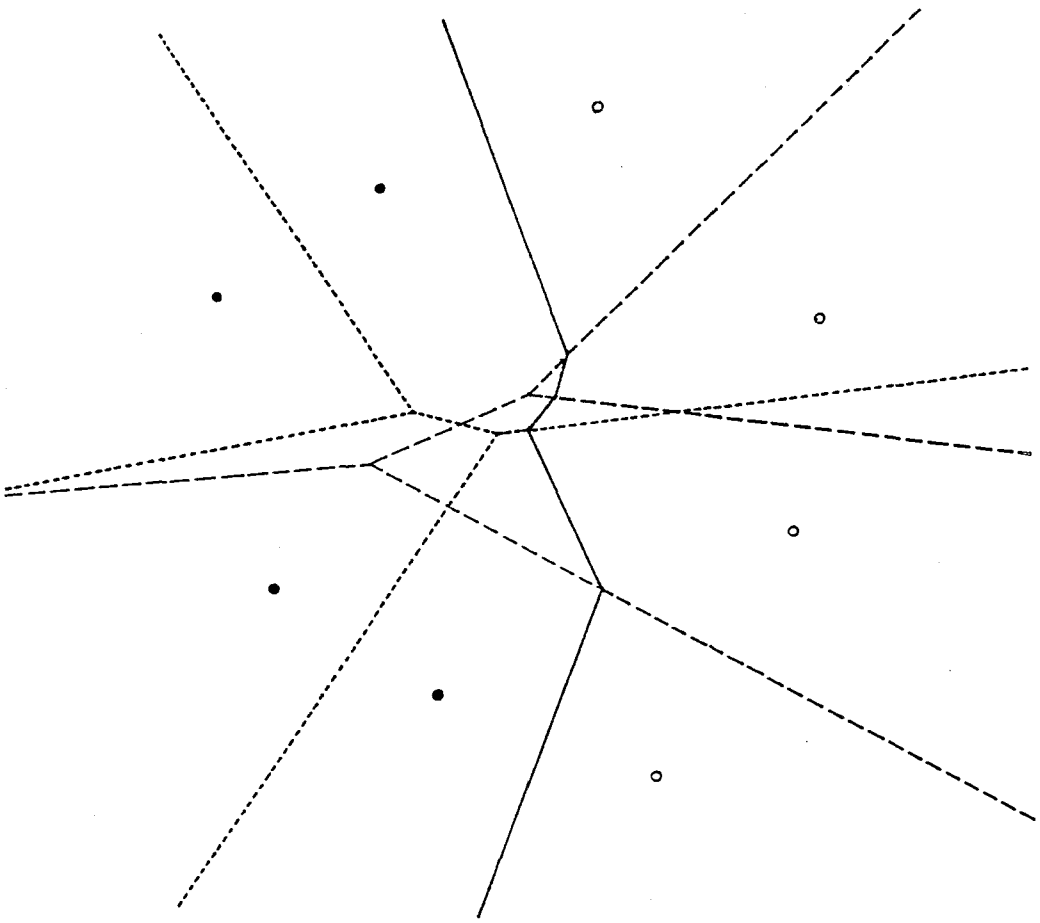


図４．４　最近点のボロノイ図の併合

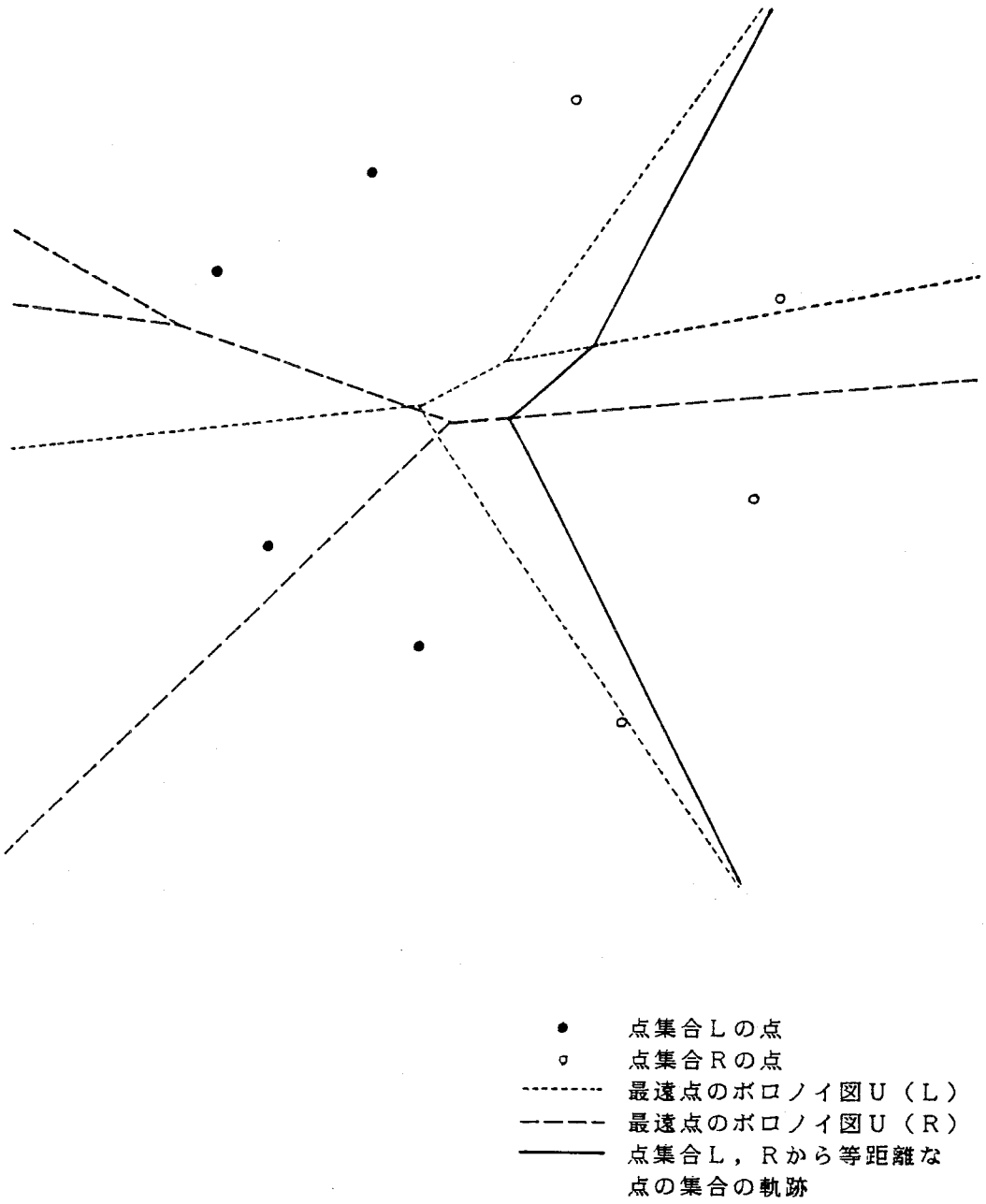


図４．５　最遠点のボロノイ図の併合

４．１で等距離な点の集合の軌跡より遠い側を取り除いたのに対して，近い側を取り除けば済む．したがって，最遠点のボロノイ図の構成算法はこの点を除いてアルゴリズム４．１と同じである．この算法をアルゴリズム４．２に示す[13]．

アルゴリズム４．２

step１　ｎ個の点をｘ座標の値に関してソートし，その順にp_1，…，p_nとする；

step２　これらの点を左側の点の集合Ｌ＝｛p_1，…，$p_{[n/2]}$｝と右側の点の集合Ｒ＝｛$p_{[n/2]+1}$，…，p_n｝に二等分し，それぞれに対するボロノイ図Ｕ（Ｌ），Ｕ（Ｒ）を再帰的に構成する；

step３　点集合Ｌ，Ｒから等距離な点の集合の軌跡（折れ線）を求め，その線より左にあるＵ（Ｌ），その線より右にあるＵ（Ｒ），それぞれのボロノイ点，ボロノイ辺を除くことによって，２つのボロノイ図Ｕ（Ｌ），Ｕ（Ｒ）を併合する（図４．５）；

アルゴリズム４．２の時間計算量も同様にＯ（ｎｌｏｇｎ）である[13]．

４．３　真円度を求める解法

円筒断面の輪郭がｎ個の点で与えられているとする．このとき，真円度を決める同心円の中心はその点からの最も近い点（１点とは限らない）と最も遠い点（１点とは限らない）の距離の差が最小となる点である．ところで，ｎ個の点についてのボロノイ図を考えると，最近点のボロノイ領域内の点はその領域に対応する点が最近点であり，最遠点のボロノイ領域内の点はその領域に対応する点が最

遠点になっている．ここで，最近点のボロノイ図と最遠点のボロノイ図の結び（Union）をとると，平面はたかだか$O(n^2)$個の領域に分割され，各領域内の点は最近点と最遠点が唯一に決められている．

なお，最近点と最遠点のボロノイ領域はいずれも凸領域であり，２つの凸領域の交わり（Intersection）は凸領域であることから，２つのボロノイ図の結びとして作られる各領域は凸領域であることに注意しておく．

以上のことから，真円度を決める同心円の中心は，与えられたｎ個の点に対する２つのボロノイ図の結びの領域について，その領域内の点で最近点および最遠点への距離の差を最小とする点を見い出せばよいことになる．すなわち，結びの領域ｒ内の点をｐ（x_p，y_p），その領域に対する最近点および最遠点をそれぞれｖ（x_v，y_v），ｕ（x_u，y_u）とすれば，問題は領域ｒ内で距離の差の関

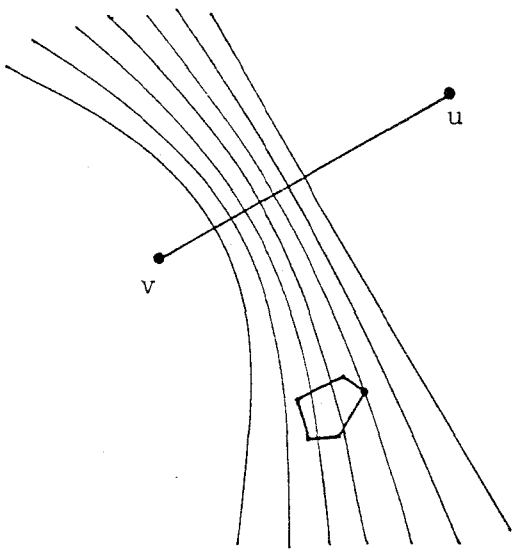


図４．６　距離の差の関数と凸領域

数

$$f = \sqrt{(x_u - x_p)^2 + (y_u - y_p)^2} - \sqrt{(x_v - x_p)^2 + (y_v - y_p)^2}$$

を最小とすると定式化できる．

関数ｆは双曲線となるので，結びの領域側（最近点側）からみると凹関数であり，結びの領域は先に指摘したように凸領域であるので，最適解は必ず結びの領域の頂点にある（図４．６）．したがって，問題の最適解の探索は，２つのボロノイ図の結びについて，最近点のボロノイ点，最遠点のボロノイ点，および，最近点のボロノイ辺と最遠点のボロノイ辺との交点のみを調べることで済む．

上述の論拠に基づいて真円度を求める解法としてアルゴリズム４．３を提案する．

アルゴリズム４．３

step１　アルゴリズム４．１を行い，最近点のボロノイ図を構成する；

step２　アルゴリズム４．２を行い，最遠点のボロノイ図を構成する；

step３　step１，２で求めた２つのボロノイ図の結び（Union）をとる（図４．７）；

step４　２つのボロノイ図の結びの図の各点（最近点のボロノイ点，最遠点のボロノイ点，最近点のボロノイ辺と最遠点のボロノイ辺との交点）に対して，その点からの最近点と最遠点の距離の差を求める；

step５　最近点と最遠点の距離の差が最小となる点とその値を求める；　最小となる距離の差が真円度であり，最小となる点が真円度の同心円の中心である（図４．８）；

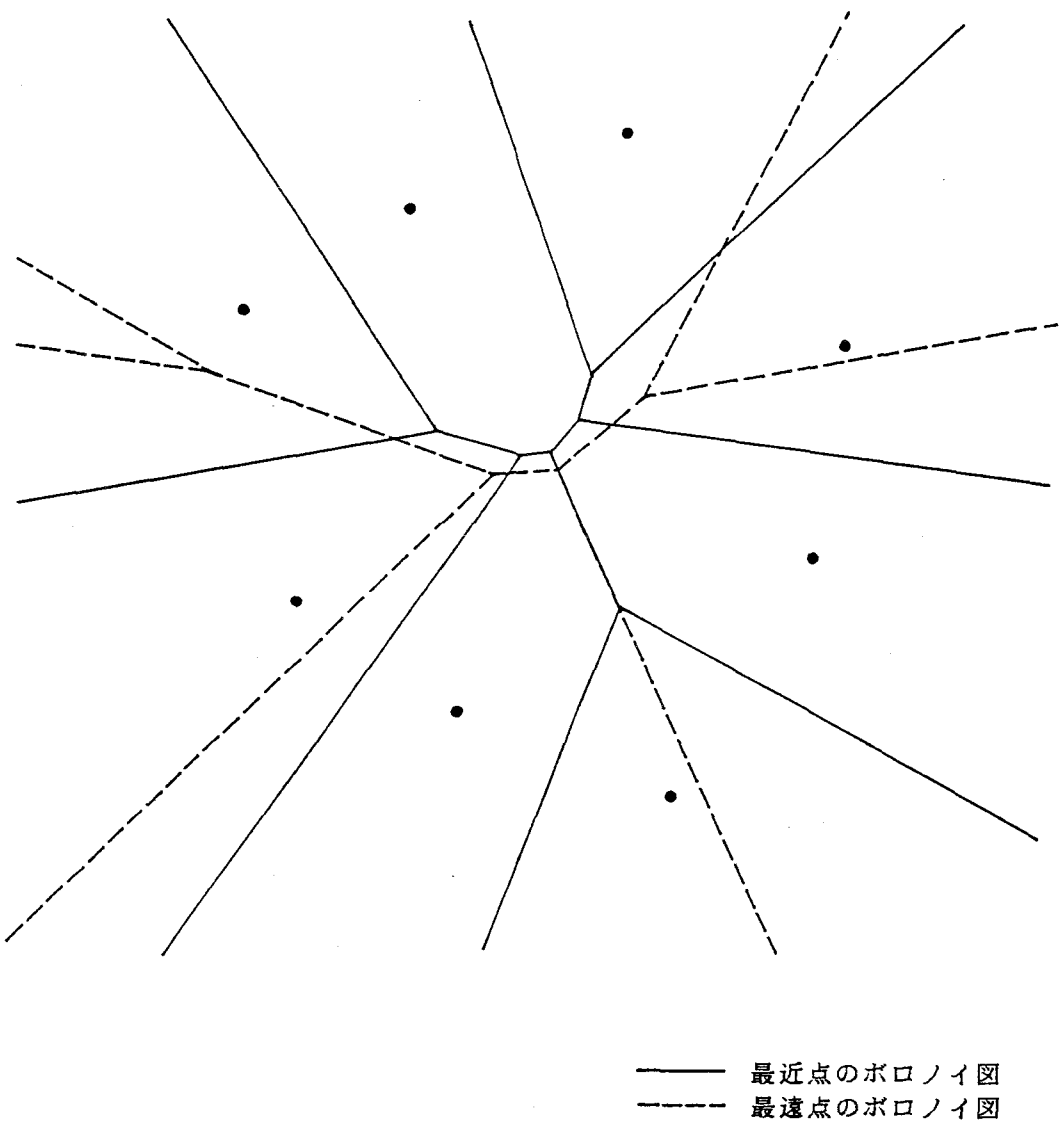
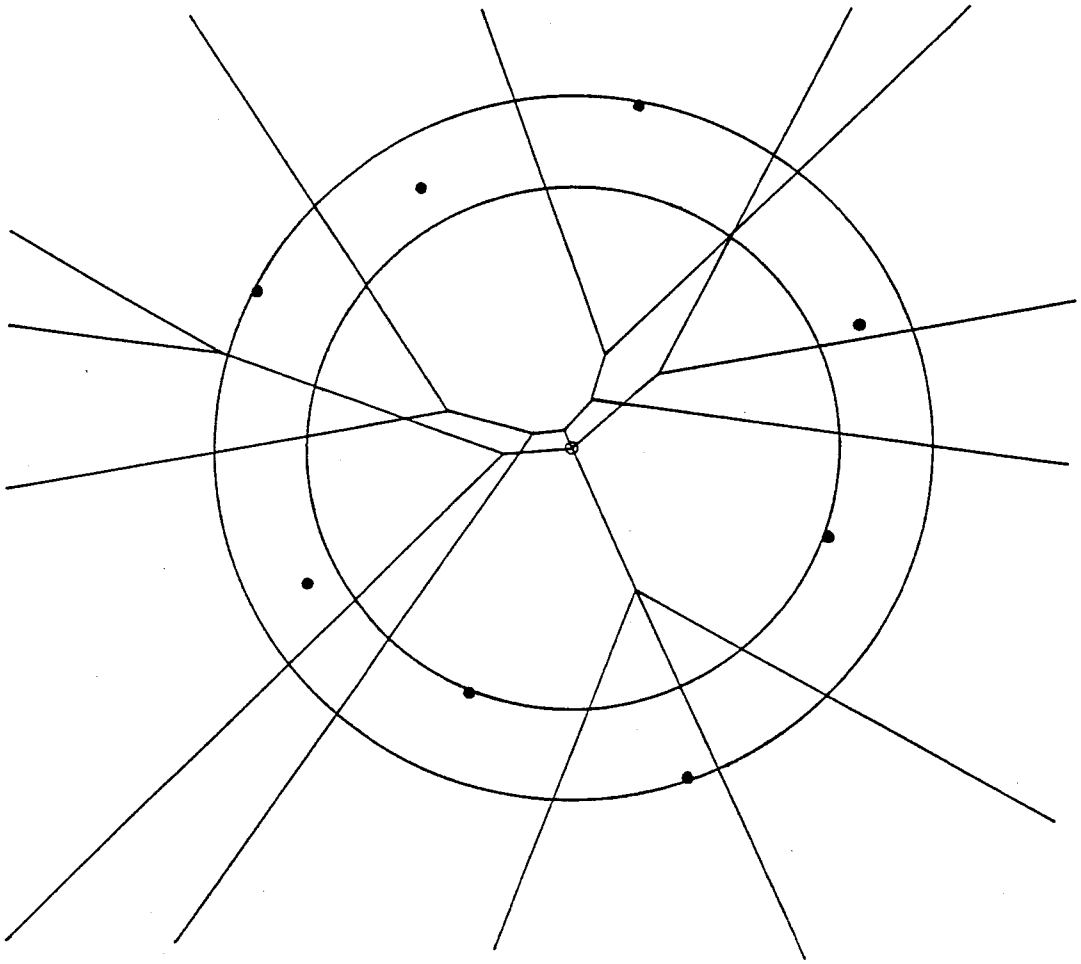


図４．７　最近点と最遠点のボロノイ図の結び



○　真円度の同心円の中心

図４．８　真円度の同心円

アルゴリズム４．３の時間計算量

アルゴリズム４．３におけるstep１，step２の最近点と最遠点のボロノイ図の構成は，前節で示したように０（ｎｌｏｇｎ）の時間計算量でできる．２つのボロノイ図は平面グラフであり，頂点の数，辺の数，領域の数はすべて０（ｎ）である．したがって，step３での２つのボロノイ図の結びは０（n^2）の手間で作ることができ，できあがった結びの図の頂点，辺，領域の数はたかだか０（n^2）である．step４では，たかだか０（n^2）個の各点に対してその点からの最近点と最遠点の距離の差を求めるが，これは結び（Union）をとるときに各点の最近点と最遠点を１つずつ記憶しておくことによって各点に対して０（１）の手間でできる．step５は，step４で計算された各点についての最近点と最遠点の距離の差のうちで，最小となる点を見い出す手続きであり，これに要する手間はたかだか０（n^2）である．

以上のことから，アルゴリズム４．３全体の時間計算量は，０（n^2）ということになる．

４．４　結言

本章では，機械部品の精度にとって重要な因子の１つである真円度を求める解法として，最近点のボロノイ図と最遠点のボロノイ図の結び（Union）をとるという考えに基づいた解法を提案し，その時間計算量が０（n^2）であることを明らかにした．

本解法は，計算幾何学の中心課題の１つであるボロノイ図の１つの応用を指摘するだけでなく，真円度の決定問題に対して実用的にも有力な解法になるものと評価している．

また，本問題を３次元に拡張すると，真円度は真球度となり，真球度も３次元のボロノイ図を構成することによって，同様に多項式時間で求めることができる．

第５章　結論

本論文では，コンピュータ・サイエンスの中で急速に発展してきた計算幾何学の分野に属する２つの問題，２次元ビン・パッキング問題と真円度問題を取り上げた．

２次元ビン・パッキング問題については，ビンの幅，ピースの幅，高さをすべて整数値とし，ピースの種類をｗｘｈの部分矩形にするという制約を加えた場合について，この問題の扱いやすさ（tractability）を考察し，次の成果を得た．

（１）　ピースの種類を４ｘ１，１ｘ４，２ｘ２の部分矩形に限定した場合の解法を考案し，それらの解法が線形時間で最適解が得られるものであることを明らかにした．この結果は，２次元ビン・パッキング問題に対して線形時間の解法が存在する限界を明らかにしたものと考えている．

（２）　ピースの種類を３ｘ２の部分矩形に限定した場合の解法を考案し，この解法が線形時間で最悪の場合でもたかだか高さが１しか悪くならない近似解が得られるものであることを示した．

（３）　ピースの種類をｗｘ１およびｗｘ２（ｗ≧５）の部分矩形に限定した場合の近似解法を考案し，その良さを既存の近似解法であるレベル・アルゴリズムとの比較実験を通して評価した．その結果，考案した２つの近似解法が，計算時間も妥当であり，性能の点でも既存の解法よりも良い近似解法であることを示した．

真円度問題については，計算幾何学の中心課題の１つであるボロノイ図を利用することによって，機械部品の精度として重要な幾何公差の１つである真円度を求める多項式時間の厳密解法が存在することを指摘し，その最適性と時間計算量を明らかにした．この解法

は，最近点のボロノイ図（勢力圏図）と最遠点のボロノイ図（非勢力圏図）の結び（Union）をとるという考えに基づいたものであり，その時間計算量は$O(n^2)$である．

謝　　辞

本研究の全過程を通じて，終始懇切丁寧な御指導，御激励を戴いた大阪大学工学部通信工学科中西義郎教授に衷心より感謝の意を表する．

大阪大学大学院在学中に，通信工学一般に関して御指導，御教示を戴いた大阪大学工学部通信工学科滑川敏彦名誉教授，熊谷信昭教授（現，大阪大学総長），手塚慶一教授，倉薗貞夫教授に深謝する．

本研究を遂行するにあたり，終始直接の御指導，御助言を戴いた大阪大学工学部通信工学科中野秀男助手に心より感謝の意を表する．

日頃より，貴重な御意見，御討論を戴いた神戸大学工学部電気工学科岡田博美助教授，琉球大学工学部電子情報工学科川口剛助手，カノープス電子株式会社稲村浩氏に深く感謝する．

真円度問題に関して，有益な御意見を戴いた新日本電工株式会社橋本良夫氏，株式会社東京精密真田友宏氏に感謝する．

日頃から，様々な面で御協力戴いた大阪大学工学部通信工学科浜富士雄技官を始めとする中西研究室の諸兄に御礼申し上げる．

参考文献

[1] A.V.Aho, J.E.Hopcroft and J.D.Ullman :"Data Structures and Algorithms", Addison-Wesley (1982).

[2] 浅野孝夫：" 計算幾何学とその応用" , 情報処理, vol.25, no.3, pp.208-221 (Mar. 1984).

[3] B.S.Baker, E.G.Coffman Jr. and R.L.Rivest :"Orthogonal Packings in Two Dimensions", SIAM J. Comput., vol.9, no.4, pp.846-855 (Nov. 1980).

[4] K.R.Baker :"Introduction to Sequencing and Scheduling", John Wiley & Sons. Inc. (1974).

[5] E.G.Coffman Jr., M.R.Garey and D.S.Johnson :"Approximation Algorithms for Bin-Packing : An Updated Survey", Algorithm Design for Computer System Design (edited by G.Ausiello, M.Lucertini and P.Serafini), Springer-Verlag, pp.49-106 (1984).

[6] E.G.Coffman Jr., M.R.Garey, D.S.Johnson and R.E.Tarjan :"Performance Bounds for Level Oriented Two-Dimensional Packing Algorithms", SIAM J. Comput., vol.9, no.4, pp.808-826 (Nov. 1980).

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

[8] P.J.Green and R.Sibson :"Computing Dirichlet tessellations in the plane", The Computer Journal, vol.21, no.2, pp.168-173 (May 1978).

[9] 伊理正夫監修／腰塚武志編集：" 計算幾何学と地理情報処理" , ｂｉｔ別冊, 共立出版 (Sep. 1986).

[10] D.S.Johnson, A.Demers, J.D.Ullman, M.R.Garey and R.L. Graham :"Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms", SIAM J. Comput., vol.3, no.4, pp.299-325 (Dec. 1974).

[11] D.T.Lee and F.P.Preparata :"Computational Geometry-A Survey", IEEE Trans. on Computers, vol.c-33, no.12, pp.1072-1101 (Dec. 1984).

[12] F.P.Preparata and M.I.Shamos :"Computational Geometry : An Introduction", Springer-Verlag (1985).

[13] M.I.Shamos and D.Hoey :"Closest-Point Problems", Proc. 16th IEEE Symp. Foundations of Computer Science, pp.151-162 (Oct. 1975).

[14] D.D.K.D.B.Sleator :"A 2.5 Times Optimal Algorithm for Packing in Two Dimensions", Info. Proc. Letters, vol.10, no.1, pp.37-40 (Feb. 1980).

[15] R.E.Tarjan :"Data Structures and Network Algorithms", Society for Industrial and Applied Mathematics (1983).

[16] 塚田忠夫，金田徹，奥田謙造：”最適化技法を用いた最小領域法真円度の評価法”，精密機械，vol.49，no.10，pp.1351-1357 (Oct. 1983).