システムＬＳＩ設計技術　101－15
（2001. 5. 18）

O-Treeを拡張したＬ型ブロックパッキング

岡本 悟　藤吉 邦洋

東京農工大学 工学部 電気電子工学科
〒 184-8588 東京都 小金井市 中町 2-24-16
TEL/FAX: 042-388-7250
E-mail: sokamoto@fjlab.ei.tuat.ac.jp, fujiyosi@cc.tuat.ac.jp

矩形パッキングの新しい表現方法として提案されたO-Treeは、矩形を節点に対応させた多分木を用いており、「どんな左下詰めパッキングでも表現可能」で「与えられたO-Treeに基づいたパッキングを線形時間にて求めることが可能」という特長を持っている。一方、Ｌ型ブロックパッキング問題についてはsequence-pairを拡張して表現する方法が提案されており、与えられたsequence-pairに基づいたパッキングを$O(n^3)$時間にて求める手法、後にこれを$O(n^2)$時間に高速化する手法が提案されている。そこで本稿では、O-Treeの特長を保ったまま、これをＬ型ブロックのパッキング表現に拡張する方法を提案する。

キーワード　O-Tree, パッキング, Ｌ型ブロック

L-Shaped Block Packing using an O-Tree Representation

Satoru OKAMOTO　Kunihiro FUJIYOSHI

Department of Electrical and Electronic Engineering,
Tokyo University of Agriculture & Technology
2-24-16 Nakacho Koganei Tokyo, 184-8588, Japan
TEL/FAX: +81-42-388-7250
E-mail: sokamoto@fjlab.ei.tuat.ac.jp, fujiyosi@cc.tuat.ac.jp

O-Tree is one of the representation method of rectangle packing. It can represent any packing where no block can shift leftward and downward, and packing based on any given O-Tree can be obtained in linear time. A method to extend sequence-pair to represent L-shaped block is proposed, and a packing based on a sequence-pair can be obtained in $O(n^3)$ or $O(n^2)$.In this paper, we propose a method to extend O-Tree to represent L-shaped block packing with keeping the features of O-Tree.

key words　O-Tree, packing, L-shaped block

1 まえがき

パッキング問題とは、与えられた全ての図形を重なりなく、なるべく小さい面積の矩形内に配置する問題である。矩形だけのパッキングの表現法として提案されたsequence-pair, BSGは、非スライス構造を含むどんなパッキングでも表現できるという特長から、様々に応用され用いられている。与えられたsequence-pairに基づいたパッキングを求めるのには、矩形数をnとして$O(n^2)$時間の手法が提案され、後に$O(n \log n)$時間に改良された。

近年、矩形パッキングの新しい表現方法として、矩形を節点に対応させた多分木を用いるというO-Treeが提案された[1][2]。O-Treeは、「どんな左下詰めパッキングでも表現できる」という特長と共に、「与えられたO-Treeに基づいたパッキングを線形時間にて求めることができる」という卓越した特長を持っている。

以前に設計されたレイアウトやその一部分を再利用するためには、L型や、より複雑なレクトリニア多角形（水平線分と垂直線分だけにより囲まれた図形）のブロックがパッキングできることが望まれる。そこで、sequence-pairを応用して、どんなL型ブロックのパッキングでも表現できる手法が提案された[3]。この論文では、与えられたsequence-pairに基づいたパッキングを、$O(n^3)$時間にて求める手法が提案されたが、後にこれを$O(n^2)$時間に高速化する手法が提案された[4]。

昨年、多分木であるO-Treeを2分木で表現するというB*-Treeが、レクトリニア多角形のパッキング手法等も含めて提案された[5]。しかし、この手法では表現できないパッキングが多数存在してしまい、最適解を逃す可能性がある。

そこで本稿では、O-Treeの特長である、「どんな左下詰めパッキングでも表現できる」と、「与えられたO-Treeに基づいたパッキングを線形時間にて求めることができる」を保ったまま、L型ブロックのパッキング表現に拡張する方法を提案する。

2 従来手法

2.1 O-Tree

O-Tree はy軸を示す仮想の矩形を根とする多分木で配置を表現する。各矩形は根以下の節点集合として表現され、座標は次の制約に基づいて決定される。ここで、深さ優先(根を左に置いたとき下側を優先)探索(DFS)順で節点aが節点bよりが前に出現する場合、「DFSでaがbより前にある」もしくは「DFSでbがaより後にある」という。また、節点iが矩形iに対応するものとし、今後節点と矩形を混同のない範囲で同一視する。

O-Treeの座標制約

x座標 矩形pの左辺x座標は、節点pの親節点に対応する矩形の右辺x座標に等しい。

y座標 DFSで節点pより前にあり、x座標の範囲が矩形pと重なる矩形の集合をT_iとする。矩形pの下辺y座標は、T_iに対応する矩形集合の内で上辺y座標が最も大きい矩形の上辺のy座標と等しい。

O-Treeは次の特長を持つ[1][2]。

特長1 O-Tree表記からパッキングを求めるのに必要な計算時間は、矩形数に対して線形である。

特長2 O-Treeは、各矩形を左下詰めにしたどんな配置でも表現できる。

2.2 B*-Tree

B*-Tree はO-Treeの多分木を、一般的な方法[6]により二分木で表現したものであり、性質はO-Treeと同じである。B*-Treeでのレクトリニア多角形の扱い方は以下のようになる。

まず、凹多角形に対してはくぼみを埋めて凸多角形に変換する。その後、凸多角形を垂直線分により矩形に分割する。矩形集合に分割されたブロックは左側の矩形から順に親、子、孫...となる関係を保ち、後にそれぞれのy座標の調整を行うが、この操作を$O(n)$で実現することは簡単ではない。この手法では、後述のように表現できないパッキングが存在し、最適な配置を失うことがある。

3 L型ブロックに拡張したO-Tree

3.1 L型ブロックの種類

L型ブロックは、矩形の4隅のいずれかに矩形の切り欠きを設けたものとみなして、切り欠きの位置により図1に示す4種類に分類できる。

図 1: L型ブロックの4種類

3.2 L型ブロックの切断

O-Treeは元来、矩形のみしか扱うことができない。よって、[3][4][5]と同様、L型ブロックを垂直線分で2つの矩形に分割することでO-Treeに対応させる。L型ブロックaを縦切り分割して作った矩形対(L型矩形対と呼ぶ)は左側をL型左部分矩形、右側をL型右部分矩形と呼び、それぞれa_l、a_rと表記する。各々の矩形対は、元のL型ブロックへ復元が可能である接した位置に配置しなくてはならない。

3.3 ブロックの領域拡大

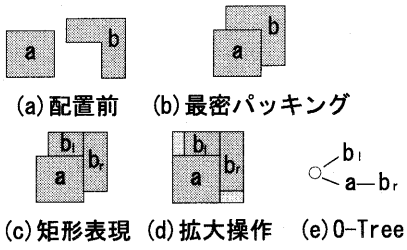

図 2: O-Tree で表現できない配置

図2(a) に示した矩形 a と L 型ブロック b に対して、最密パッキングは (b) となる。ところが、(b) のパッキングを矩形分割で表した配置 (c) は、L 型左部分矩形 b_l が左詰めになってなく、また、L 型右部分矩形 b_r は下詰めになってない。この矩形パッキングは左下詰めでない矩形があるため、O-Tree では表現できない。

B*-Tree では b_r のような下詰めでない L 型ブロックは、配置後に位置調整操作を施すことで対応している。しかし、a_l のような左下詰めでない L 型ブロックは O-Tree と同様、表現することができない。

よって、このような L 型ブロックを含む配置を O-Tree で表現するために、O-Tree の定義を緩和する。

ブロックの配置領域 ブロックを囲む長方形の領域を配置領域と呼び、拡張した O-Tree では矩形を配置する替わりに配置領域を配置する。配置領域の大きさは各ブロックが格納するのに十分なものとし、原則では矩形と同じ大きさとするが、L 型部分矩形では必要に応じて領域の拡大を許す。以後、領域 i についても混同のない範囲で節点 i、矩形 i と同一視する。

O-Tree の節点 O-Tree や B*-Tree では、木の節点はブロックに対応していた。しかし、拡張した O-Tree では、木の節点をブロックの配置領域に対応させる。

3.3.1 幅方向の領域拡大

L 型矩形対の左矩形 a_l に限り、その配置領域の幅方向への拡大を必要に応じて許す。拡大する幅は a_l の配置領域の右辺 x 座標が対応する a_r の左辺 x 座標と一致するように決定され、当然、拡大幅が負になることは許されない。

図3において、(a) の最適パッキングに対する矩形分割表現は (b) であるが、(b) の L 型左矩形 b_l に対応する領域 b_l は (c) のように拡大される。これに対応する O-Tree は (d) である。

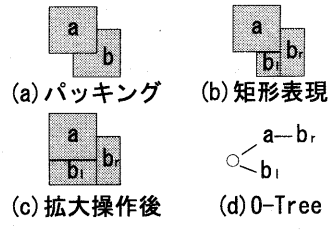

図 3: 幅方向の領域拡大

3.3.2 高さ方向の領域拡大

L 型矩形対のどちらか一方の矩形に限り、その配置領域の高さ方向への拡大を必要に応じて許す。図4において、(a) の最適パッキングに対する矩形分割表現は (b) であるが、(b) の L 型右矩形 b_r に対応する領域は (c) のように拡大される (対応する O-Tree は (d))。この場合のように、拡大後の L 型ブロックの配置領域が L 型にならない場合もある。

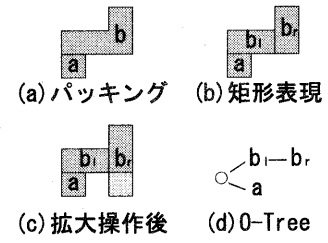

図 4: 高さ方向の領域拡大

図2(c) の場合、一つの L 型ブロックに対して高さ方向の領域拡大と幅方向の領域拡大の両方が行われ、(d) のようになる (対応する O-Tree は (e))。

3.4 許容 O-Tree

幅と高さの領域拡大操作を必要に応じて許して拡張した O-Tree において、対応するパッキングが存在する O-Tree を「許容 O-Tree」と呼ぶ。これは、言い換えれば次の 2 つの条件の両方を満たす O-Tree のことである。

1. L 型左部分矩形 a_l の幅方向領域拡大操作により、各 L 型ブロックの a_l と a_r について、配置領域 a_l の右辺と配置領域 a_r の左辺の x 座標が等しくできる
2. a_l または a_r の高さ方向領域拡大操作により、領域 a_l と領域 a_r の y 座標範囲に重なりを作れる

逆に、上記の許容 O-Tree に該当しないものは非許容 O-Tree とし、領域の拡大幅が負となるもの (条件 1 に反する) や図4のように L 型矩形対が分離されるもの (条件 2 に反する) がこれにあたる。

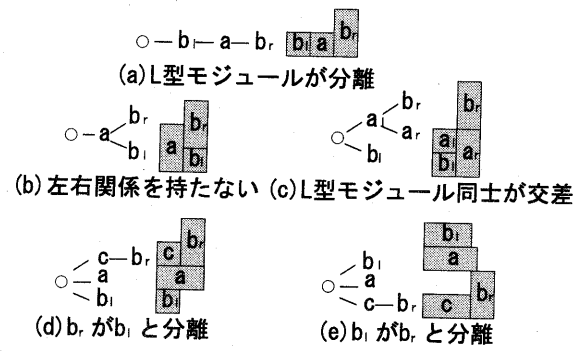

図 5: 非許容 O-Tree の例。(a)(b) は条件 1 に、(c)(d)(e) は条件 2 に反している。

3.5 対象とする O-Tree

次節にて L 型ブロックパッキングを求めるアルゴリズムを提案するが、以下の条件に該当しない O-Tree だけをアルゴリズムの入力対象とする。なお、この条件で制限してもすべての左下詰めのパッキングをもれなく表現できることについては後に説明する。

入力対象としない O-Tree の条件

次の条件を持つ L 型ブロックが 1 つでも存在する

対象外条件 1 右側の欠けた L 型ブロックで、a_r の親が a_l でない

対象外条件 2 左側の欠けた L 型ブロックで、a_l は子を持ち、その子に a_r を含まない

対象外条件 3 DFS 走査での順で a_l と a_r の間にある矩形集合の中に、左辺 x 座標が a_r の左辺 x 座標と等しい矩形が存在

対象外条件 3 に該当するが、許容な O-Tree の例を図 6に示す。

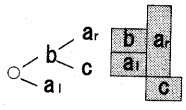

図 6: 対象外条件 3 に該当する許容 O-Tree の例

3.6 配置アルゴリズム

拡張した O-Tree 表現からパッキングを得るアルゴリズムを次に示す。なお、領域 i の左下座標を $x[i]$、$y[i]$、幅を $w[i]$、高さを $h[i]$ で表す ($i = 0$ は根を表す仮想矩形の領域)。また、$sy[i]$ は対応する L 型左部分矩形の下辺を基準とした L 型右部分矩形の相対的な y 座標を持つ定数とする。つまり、元の L 型ブロックの形状が左上または右上欠けの場合に 0 であり、左下欠けの場合に負、右下欠けの場合に正の値を持つ。

Step 0: 対象外条件 1・2 のチェック

```
for(全ての右側の欠けた L 型ブロック a)
  if(a_r の親が a_l でない)
    exit(対象外 O-Tree1);
for(全ての左側の欠けた L 型ブロック a)
  if(a_l は子を持つが、その子に a_r を含まない)
    exit(対象外 O-Tree2);
```

Step 1: 各領域の x 座標の計算

```
x[0] = 0;
w[0] = 0;
for(木のすべての節点 i について DFS 順に) {
  x[i] = x[iの親] + w[iの親];
  w[i] = iの幅;
}
```

Step 2: 幅方向の領域拡大

```
for(全ての L 型ブロック a について) {
  if(a_r の親が a_l でない) {
    if(x[a_r] − x[a_l] < 矩形 a_l の幅)
      exit(非許容 O-Tree);
    else
      w[a_l] = x[a_r] − x[a_l];
  }
}
```

Step 3: 非許容 O-Tree と対象外条件 3 のチェック

```
スタック st の初期化;
for(木のすべての節点 i について DFS 順に) {
  if(i が後に出現した L 型部分矩形)
    st.pop();
  else if(スタックの先頭値 < i の右 x 座標) {
    if(スタックの先頭値 == (領域 i の左 x 座標))
      exit(対象外 O-Tree3);
    else
      exit(非許容 O-Tree);
  }
  else if(領域 i が L 型部分矩形) {
    if(i が L 型左矩形)
      st.push(i の右辺 x 座標);
    else
      st.push(i の左辺 x 座標);
  }
}
```

Step 4: 親の付け換え

```
for(全ての L 型右矩形 a_r) {
```

if(節点 a_r が対応する節点 a_l の子ではない)
節点 a_r を節点 a_l の下に付け換え、
節点 a_l の子とする;
}

Step 5: 高さ方向に領域を拡大しながら配置

floor 構造の初期化;
for(木のすべての節点 i について DFS 順に) {
$y[i]$ を floor 構造より得る;
$h[i]$ = 矩形 i の高さ;
if(i が L 型右矩形) {
ℓ = i が属する L 型ブロックの左部分矩形;
// 左の領域が拡大される場合
if($y[\ell] + sy[i] < y[i]$) {
$h[\ell] += y[i] - y[\ell] - sy[i]$;
領域 ℓ の x 座標範囲における floor 構造の高さを
$y[\ell] + h[\ell]$ に変更;
}
// 右の領域が拡大される場合
else if($y[\ell] + sy[i] > y[i]$) {
$h[i] += y[\ell] - y[i] + sy[i]$;
}
}
領域 i の x 座標範囲における floor 構造の高さを
$y[i] + h[i]$ に変更;
}

Step 6: 領域内にブロックを挿入

各領域に対応するブロックを入れる;

3.7 提案アルゴリズムの実行可能性

Step 0 が対象外条件 1,2 に該当する O-Tree を線形時間にて発見できることは、容易に分かるであろう。

Step 1 では各々の配置領域の左辺 x 座標を求め、**Step 2** では配置領域の幅の拡大を必要に応じて行なっている。ここの終了後に

$$x[a_l] + w[a_l] = x[a_r] \tag{1}$$

が必ず満たされることは、節点 a_r が節点 a_l の子であれば幅方向の拡大は必要なく、節点 a_r が節点 a_l の子でないときには対象外条件 2 により節点 a_l は子をもたないので、L 型ブロック毎に 1 回の操作により全ての L 型ブロックについて式 (1) を満たすことができる。単純な 1 重ループであることから線形時間にて実行可能である。

Step 3 では O-Tree の y 座標の制約を満たせない非許容 O-Tree でないか、もしくは入力対象外条件 3 に該当しないかを調べている。これらは共に、DFS 走査での順で L 型矩形対 a_l と a_r の間に出てきた他の矩形の右辺 x 座標が、a_r の左辺 x 座標、つまり a の分割線の x 座標をよりも大きかったときに該当する。複数の L 型矩形対が入れ子になっている可能性もあるが、条件を満たした分割線 x 座標は必ず単純に減少するので、スタックを使ってこれをチェックしている。明らかに、線形時間にて実行可能である。

Step 4 では、y 座標の決定を線形時間に行なえるよう、節点 a_r が節点 a_l の子でないときに、付け替えを行なっている（この操作なしでもパッキングを求めることはできるが、線形時間では困難である）。これが線形時間にて行なえることは明らかである。この付け替え操作により、入力された O-Tree とは異なるパッキングを表すことにならないことは、**Step 3** において、DFS 走査での順で L 型矩形対 a_l と a_r の間に出てきた他の矩形の右辺 x 座標が、a の分割線の x 座標をよりも大きいことがないことをチェック済みなので、保証できる。L 型ブロック数の線形時間にて実行可能である。

Step 5 では、配置領域の高さ方向の領域拡大を行ないながら各々の下辺 y 座標を求めている。この step が線形時間にて可能なことについては、O-Tree に基づいた矩形パッキングを求めるアルゴリズム [1] と全く同様の理由により可能である。高さ方向の領域拡大により y 座標制約が満たされることは容易に分かるであろう。この領域拡大において他の矩形と重なることがないことは、各々の L 型矩形対 a について、**Step 4** の付け換え操作により a_r は a_l の子であるため、必ず配置領域 a_r よりも前に配置領域 a_l の y 座標が決定されると共に、a_r を見ているときにはまだ a_l の上側の矩形の y 座標が決定されてないためである。

Step 6 で各々の配置領域の中にブロックが入れられることと、線形時間にて実行可能なことは明らかであろう。

また、提案アルゴリズムは、非許容もしくは入力対象外な O-Tree を入力すれば、必ずそれを指摘して停止する。

この理由は、非許容 O-Tree については、x 座標制約を満たせないのであれば **Step 2** で、y 座標制約を満たせないのであれば **Step 3** にて必ず発見されて停止する。入力対象外な O-Tree については、入力対象外 1,2 については **Step 1** で、入力対象外 3 については **Step 3** にて必ず発見されて停止する。

以上から以下の定理が得られるであろう。

【定理 1】 提案アルゴリズムは、入力対象外ではない許容 O-Tree を入力すると、O-Tree サイズの線形時間にて、拡張した制約を守ったパッキングを得ることができる。また、入力対象外もしくは非許容 O-Tree を入力すると、O-Tree サイズの線形時間にて、それを指摘して停止する。

3.8 任意の左下詰めパッキングを表現できることの証明

提案した拡張を行なった O-Tree は、入力対象条件により制限されても、どんな左下詰めパッキングでも表現できることが証明できる。これにより、入力対象条件を満たした許容 O-Tree を全探索すれば最密なパッキングを得ることができると保証可能である。

【定理2】　L型ブロックを含んだパッキング問題に対するどんな左下詰めのパッキングでも、提案した表現方法により、入力対象外となってないO-Treeにより表現可能である。

（証明）　任意のパッキングPが与えられたとしてそれを表すO-Tree、それも入力対象外でないものを求める手順を述べることにより証明する。

Pの中の全てのL型ブロックを垂直線分により分割して2つの矩形にし、Pの最左にPよりも高さが十分にある仮想矩形rを入れる。切り欠きが左側のL型ブロックaのL型左部分矩形a_lが左詰めでないなら、a_lをその左辺が他の矩形に接するまで左方向に拡大する。これにより、仮想矩形rを除いた全ての矩形が左詰めである矩形パッキングP'を得る。

矩形iを節点i、仮想矩形を根節点vとして、矩形iの左辺が矩形jの右辺と接しているなら節点iは節点jの子とする。但し、節点jが左方向に拡大したL型左部分矩形であるなら、これの子とはせずに他に接する矩形に対応する節点の子とする。この条件は、L型ブロックaはPでは左詰めであるがaを切断したa_lは左詰めでないことから明らかに、a_rの左辺はa_l以外の矩形に接していることから、守ることができる。

これらの親子関係は接する矩形同士でだけ結ばれるため、明らかに平面上で交差なく多分木にて表すことができる。そこで、これをvを根とする多分木Tとする。

得られたTは明らかにP'を表す許容O-Treeである。入力対象外なO-Treeでないかを考えると、対象外条件1については、右側の欠けたL型ブロックaではa_rの左辺が接するのはa_l以外にはなく、節点a_rは必ず節点a_lの子になるので該当しない。対象外条件2については、左側の欠けたL型ブロックaではa_lの右辺に接するのはa_r以外にはなく、節点a_lの子はないか節点a_rだけなので該当しない。対象外条件3については、平面上での位置関係から、DFS走査順でa_lとa_rの間にある矩形の右辺x座標がa_rの左辺x座標より大きいことはあり得ないので該当しない。

従って、上記手順により得られた多分木Tは入力対象外でない許容O-Treeである。■

4　まとめ

矩形パッキングの表現方法であるO-Treeを拡張して、L型ブロックパッキングを表現可能にする方法を提案した。提案方法は、どんな左下詰めパッキングでも表現可能であり、それに基づいたパッキングを線形時間にて得ることが可能であるという、O-Treeの特長をそのまま受け継いでいる。

L型ブロックを含むことにより、対応するパッキングが存在しない「非許容O-Tree」が生じてしまう。そして、線形時間にてパッキングを求めることを可能にするため、ある条件により限定された許容O-Treeだけをパッキング対象とし、この条件の下でも、どんな左下詰めパッキングでも表現可能であることを示した。これにより、全探索を用いて入力対象である許容O-Treeを全て調べれば最密パッキングが得られることを保証できる。

提案したアルゴリズムに入力するO-Treeには条件を設けたが、右上切り欠きのL型ブロックaならa_rはa_lの子の中でも最もDFS順で前にあるものに限定でき、同様に右下切り欠きであれば子の中でも最もDFS順で後ろにあるものに限定することにより、冗長なものを更に除くことが可能である。

今後の課題としては、まず、計算機上にて実際にパッキング性能を調べることが挙げられる。また、どんなパッキングでも表現でき、しかも高速にてO-Treeからパッキングを得ることができることを保ちつつ、より複雑なレクトリニア多角形のパッキングを表現する様に拡張することも挙げられる。

参考文献

[1] Pei-Ning Guo, Chung-Kuan Cheng, Takeshi Yoshimura, "An O-Tree Representation of Non-Slicing Floorplan and its Applications", DAC, pp.268–273, 1999.

[2] Toshihiko Takahashi, "A New Encoding Scheme for Rectangle Packing Problem", ASP-DAC, pp.175–178, 2000.

[3] 藤吉邦洋, 村田洋, "L型モジュールを含んだ矩形パッキング問題に対する、sequence-pairを用いたアルゴリズム", 第11回 回路とシステム(軽井沢)ワークショップ, pp.113–118, 1998.

[4] 齋藤宏明, 藤吉邦洋, "L型ブロックパッキングの高速化に関する研究", 第13回 回路とシステム(軽井沢)ワークショップ, pp.245–250, 2000.

[5] Yun-Chih Chang, Yao-Wen Chang, Guang-Ming Wu, and Shu-Wei Wu, "B*-Trees: A New Representation for Non-Slicing Floorplans", DAC, pp.458–463, 2000.

[6] R. Sedgewick, "Algorithms", Addison-wesley publishing company, 1989.