Selected Sequence-Pairのための効率的な隣接解生成手法

藤吉邦洋† 児玉親亮† 清田紘司†

n 個の矩形のパッキングを長さ n の順列の対で表現する sequence-pair に対して，近年，隣接交差と呼ばれる部分列の数を $n-\lfloor\sqrt{4n-1}\rfloor$ 以下に制限した “selected sequence-pair” が提案された．これは sequence-pair と同様にどんな矩形パッキングでも表現可能であるという特長を持ちながら，矩形数の線形時間で，その示唆する制約の下での左下詰めパッキングを得ることができるという優れた特長を持つ．しかし，隣接解やその生成法が提案されていなかったため，Simulated Annealing 法などと組み合わせてのパッキング探索に用いることができなかった．そこで本稿では，selected sequence-pair での到達可能性を保証できる隣接解と，その効率的な生成方法を提案し，計算機実験によってその効率の良さを確認する．

An Efficient MOVE Operation for Selected Sequence-Pair

Kunihiro Fujiyoshi,† Chikaaki Kodama† and Koji Kiyota†

Recently “selected sequence-pair” was proposed, which is a sequence-pair with $n-\lfloor\sqrt{4n-1}\rfloor$ or less sub-sequences called adjacent cross (n is the number of rectangles). In addition to the merit that it can represent any packing, it has a superior feature that a bottom left corner packing can be obtained under the constraints imposed by itself in linear time of the number of rectangles. However, as a method of generating adjacent solution was not proposed, it was impossible to use selected sequence-pair for search of packing by combining with Simulated Annealing, etc. In this paper, we propose adjacent solution which can guarantee reachability in selected sequence-pair and a procedure to generate an adjacent solution and then we confirm its efficiency by experiments.

1. まえがき

VLSI 開発におけるレイアウト設計では，多数の矩形形状のモジュールをいかに密に配置するかという問題がある．Murata らが提案した矩形配置の表現方法である sequence-pair [1] は，どんな矩形パッキングでも表現可能であり，すべての sequence-pair には対応する矩形パッキングが必ず存在するという特長がある．また，水平方向と垂直方向の制約グラフを用いることにより，矩形数を n として $O(n^2)$ 時間で 1 つの sequence-pair に基づいた左下詰めパッキングを求める手法が提案された[1]．

sequence-pair は矩形対の相対位置制約を示唆していることから拡張性が高く，実際，既配置モジュール[2]や，多角形形状の IP モジュールを含んだ自動配置に適用可能なレクトリニア多角形パッキング[3]や，指定モジュールのチップの外周付近への配置[4]，矩形分割への等価変換[5] 等に応用されている．

近年，我々は sequence-pair において隣接交差と呼ばれる部分列の数を $n-\lfloor\sqrt{4n-1}\rfloor$ 以下に制限した “**selected sequence-pair**”（以下 **SSP**）を提案した[6]．これは sequence-pair と同様にどんな矩形パッキングでも表現可能であるという特長を持ちながら，O-tree [7),8)] と同じく任意の SSP に基づいた左下詰めパッキングを矩形数 n の線形時間で求めることができる．また sequence-pair と同じく拡張性が高く，レクトリニア多角形パッキング手法にも応用されている[9]．しかも配置表現の総数は矩形数が同じ sequence-pair より少ないという特長を持つ．

ところで，矩形数 n のパッキングを表現する sequence-pair が持ちうる隣接交差の数は，最大で $\lceil(n-2)/2\rceil\lfloor(n-2)/2\rfloor$ であることが知られている[5]．そのため，これまで sequence-pair で用いられてきた隣接解生成手法を矩形数 n の SSP にそのまま適用して Simulated Annealing 法（SA 法）探索すると，生成された隣接解が $n-\lfloor\sqrt{4n-1}\rfloor$ 個を超える隣接交差を含んでしまう場合があるため，探索の過程で SSP

† 東京農工大学工学教育部電子情報工学専攻
Tokyo University of Agriculture and Technology

ではなくなってしまう可能性がある．

そこで本稿では，サイズ n の SSP に対し隣接交差数を $n-\lfloor\sqrt{4n-1}\rfloor$ 以下に保ちつつ到達可能性を保証できる隣接解生成手法を提案する．この手法ではランダムに SSP の要素の移動先を 1 つ決定した後，SSP の各要素をそこに移動した際に生じる隣接交差数の増減を SSP サイズの線形時間で計算し，隣接交差数を $n-\lfloor\sqrt{4n-1}\rfloor$ 以下に保つことができる要素の 1 つをランダムに選んで移動することとする．また，提案手法の有効性を，計算機実験により確かめる．

2. Selected Sequence-Pair（SSP）

2.1 Sequence-Pair（Seq-Pair）

sequence-pair [1)]（以下 seq-pair）では n 個の矩形の相対位置関係を，矩形名の順列 Γ_+ と Γ_- の対により，$(\Gamma_+;\Gamma_-)$ の形で表す．当然，n 個の矩形の配置について $(n!)^2$ 通りの表現がある．ここで，$\Gamma_+(i)$ は Γ_+ 中で第 i 番目の矩形を指し，$\Gamma_+^{-1}(a)$ は Γ_+ 中で矩形 a が左から何番目かを指す．Γ_- についても同様である．

seq-pair では矩形対の相対位置関係を，以下に示す「上下左右制約」として表す．Γ_+ と Γ_- でともに a が b の前にあるとき，つまり $\Gamma_+^{-1}(a)<\Gamma_+^{-1}(b)$ かつ $\Gamma_-^{-1}(a)<\Gamma_-^{-1}(b)$ であるとき，矩形 a は矩形 b の左に位置する．また Γ_+ では a が b の前にあり Γ_- では a が b の後ろにあるとき，すなわち $\Gamma_+^{-1}(a)<\Gamma_+^{-1}(b)$ かつ $\Gamma_-^{-1}(a)>\Gamma_-^{-1}(b)$ であるとき，矩形 a は矩形 b の上に位置する．

seq-pair はどんな矩形パッキングでも表現可能であるという特長を持ち，1 つの seq-pair からそれが表す左下詰めパッキングを $O(n\log\log n)$ 時間で得ることができる[10)]．

$\Gamma_+=(1\,2\,3\cdots n)$ となるように矩形名を付け替えたときの Γ_- を「**標準化** Γ_-」と呼び，同様に「**標準化** Γ_+」を定義するが，これらはちょうど逆関数の関係にある．たとえば，seq-pair $(abcdef;bfdcae)$ の標準化 Γ_- は $(2\,6\,4\,3\,1\,5)$ で，標準化 Γ_+ は $(5\,1\,4\,3\,6\,2)$ となる．

サイズ n の「**斜格子**」とは，平面上に描いた $+45$ 度の傾きの n 本の平行線 $L_+(1),L_+(2),\cdots,L_+(n)$（X 切片の昇順）と，$-45$ 度の傾きの n 本の平行線 $L_-(1),L_-(2),\cdots,L_-(n)$（X 切片の昇順）による平面上の格子構造をいう．矩形数 n の seq-pair $S=(\Gamma_+;\Gamma_-)$ が与えられたとき，サイズ n の斜格子上で各矩形 i を $L_+(\Gamma_+^{-1}(i))$ と $L_-(\Gamma_-^{-1}(i))$ の交点に対応づけたものを S の**斜格子埋め込み**という．「seq-pair S の斜格子埋め込み」の特長は，S が示唆する上下左右制約がこれから容易に読みとれることである．S において注目している矩形 a の右側 90° の範囲に入っている矩形はすべて，S により a の右側にあると制約されている．左上下側も同様である．

2.2 矩形分割

矩形分割とは，配置問題の前の段階でモジュールと配線チャネルの相対位置関係を求めたものである．具体的には，n 個のモジュール $\{M_1,M_2,\cdots,M_n\}$ を配置する問題に対して，チップ全体を表す矩形を，$M_1,M_2,\cdots,M_n$ と各々名前のついた n 個の重なりのない矩形領域（**部屋**）に，垂直/水平線分（**分割線**）を用いて分割する問題である．ただし，各々の部屋の大きさは，その名前に対応するモジュールが入ることができるように確保する．本稿では，分割線の交点はチップ全体を表す矩形の四隅を除いてすべて T 字状であるとする[11)]．n 部屋の矩形分割は各々の分割線が座標を持つので無限に存在するが，部屋とそれに隣接する分割線の相対位置関係のみに着目すると，これを有限種類に分類することができる．

2.3 隣接交差

seq-pair $\mathcal{S}$ において，矩形 a,b,c,d の位置が，

$\mathcal{S}=(\cdots a\cdots bc\cdots d\cdots;\cdots c\cdots ad\cdots b\cdots)$ または

$\mathcal{S}=(\cdots a\cdots bc\cdots d\cdots;\cdots b\cdots da\cdots c\cdots)$

のとき，「$\mathcal{S}$ は**隣接交差**を持つ」というものとする．またこの隣接交差は，Γ_+ で隣接している対と Γ_- で隣接している対を '/' で区切って，前者は $b,c/a,d$，後者は $b,c/d,a$ と表記する．これは斜格子上では Γ_+ で隣接する b，c を結ぶ線分と Γ_- で隣接する a，d を結ぶ線分の交点となる（**図 7** (b) 参照）．これが隣接交差という名前の由来となっている．

隣接交差を持つサイズ n の seq-pair と同じ相対位置関係の矩形分割は n 部屋では実現できず，矩形が埋め込まれない**空部屋**を隣接交差の数だけ導入しなければならない．サイズ n の seq-pair が含みうる隣接交差数の最大は $\lceil(n-2)/2\rceil\lfloor(n-2)/2\rfloor$ である[5)]．

たとえば seq-pair $(1\,2\,3\,4;2\,4\,1\,3)$ のとき，この相対位置関係すべてを満たす矩形分割は**図 1** のように

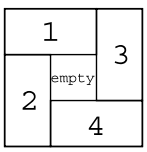


図 1 seq-pair $(1\,2\,3\,4;2\,4\,1\,3)$ に対応した矩形分割
Fig. 1 Rectangular dissection corresponding to seq-pair $(1\,2\,3\,4;2\,4\,1\,3)$.

なり，4 つの部屋の中央に空部屋を導入しなければならないことは容易に確かめられる．

2.4 Selected Sequence-Pair（SSP）[6)]

サイズ n の seq-pair の隣接交差数が $n-\lfloor\sqrt{4n-1}\rfloor$ 以下であるとき，これを **selected sequence-pair**（以下 **SSP**）と呼ぶ．矩形 n 個の任意のパッキングは，$n-\lfloor\sqrt{4n-1}\rfloor$ 以下の隣接交差を持つ seq-pair で表現できることが証明されている[12)]．

2.4.1 SSP からパッキングを求めるアルゴリズム[6)]

k 個の隣接交差を持つ，矩形 n 個の配置を表す SSP から，以下に述べる 3 段階の処理を行うことによって $O(n+k)$ 時間でパッキングを得ることができる．なお，SSP では隣接交差数 k が $O(n)$ なので，このアルゴリズムは全体で $O(n)$ 時間で実行可能であることに注意されたい[6)]．

Step 1：与えられた SSP $\mathcal{S}$ 上のすべての隣接交差（k 個）を列挙する（$O(n+k)$ 時間）．

Step 2：先に得られた隣接交差の位置と列挙順に基づき，ダミー矩形を k 個挿入することにより隣接交差をすべて除去する（$O(n+k)$ 時間）．ここで得られた隣接交差なし SSP $\mathcal{S}'$ のサイズは，ダミー矩形 k 個が加わったので $n+k$ になる．

Step 3：$\mathcal{S}'$ の示唆する相対位置関係に基づく矩形分割を求めてパッキングを得る（$O(n+k)$ 時間）．

2.4.2 Step 1：すべての隣接交差の列挙

k 個の隣接交差を含むサイズ n の seq-pair から，アルゴリズム Adjcross List（**図 2** 参照）は $O(n+k)$ 時間ですべての隣接交差を列挙することが可能である[6)]．このアルゴリズムは本稿で提案するアルゴリズムで用いられているので概観する．

例として，標準化 Γ_- (4 2 6 1 3 5) が入力されたとして Adjcross List の動作を見てみる．**図 3** は双方向リストでの操作である．**(a)** 最も左の要素 4 をみて，要素 5 は未走査なので 4 をリストに挿入．**(b)** 要素 2 をみて，要素 3 は未走査なので 2 をリストに挿入．**(c)** 要素 6 をみて，要素 7 は存在しないので挿入せず．**(d)** 要素 1 をみたときリスト上の 4 を上方向に飛び越えたので隣接交差 $4,5/6,1$ を発見し，同様に 2 も飛び越えたので隣接交差 $2,3/6,1$ を発見する．要素 2 は既走査なので，要素 1 は挿入しない．**(e)** 要素 3 をみるが要素 4 は既走査なので挿入せず，さらにリスト上の要素 2 を削除．**(f)** 要素 5 をみるが，要素 6 は既走査なので挿入せず，さらにリスト上の要素 4 を削除．

以上で左からの走査は終了し，同様に右から走査すると，隣接交差 $3,4/2,6$ を発見する．よって，$4,5/6,1$ と $2,3/6,1$ と $3,4/2,6$ の 3 つの隣接交差が発見された．

アルゴリズム Adjcross List（入力：標準化 Γ_-）
空の双方向リストを用意する；
$p_p = 0$;
foreach ($p = \Gamma_-$ の要素を前から 1 つずつ){
  if ($p < p_p$)
    foreach(k = 大きさが p と p_p の間で双方向リストに現存する要素)
      隣接交差 $k, k+1$ / p_p, p を発見したので登録;
  $p_p = p$;
  if ((p は最大の要素ではない) && ($\Gamma_-^{-1}(p+1) > \Gamma_-^{-1}(p)$))
    // 要素 $p+1$ は未走査
    p を双方向リストに挿入；
  if (($p > 1$) && ($\Gamma_-^{-1}(p-1) < \Gamma_-^{-1}(p)$))
    // 要素 $p-1$ は双方向リスト上に存在
    $p-1$ を双方向リストから削除；
}
$p_p = 0$;
foreach ($p = \Gamma_-$ の要素を後ろから 1 つずつ){
  if ($p < p_p$)
    foreach(k = 大きさが p と p_p の間で双方向リストに現存する要素)
      隣接交差 $k, k+1$ / p, p_p を発見したので登録;
  $p_p = p$;
  if ((p は最大の要素ではない) && ($\Gamma_-^{-1}(p+1) < \Gamma_-^{-1}(p)$))
    // 要素 $p+1$ は未走査
    p を双方向リストに挿入；
  if (($p > 1$) && ($\Gamma_-^{-1}(p-1) > \Gamma_-^{-1}(p)$))
    // 要素 $p-1$ は双方向リスト上に存在
    $p-1$ を双方向リストから削除；
}
（**アルゴリズム** Adjcross List **終.**）

図 2 アルゴリズム Adjcross List
Fig. 2 Algorithm Adjcross List.

2.3 節で述べたように隣接交差はその定義から，$(\cdots a\cdots bc\cdots d\cdots;\cdots c\cdots ad\cdots b\cdots)$ の型と $(\cdots a\cdots bc\cdots d\cdots;\cdots b\cdots da\cdots c\cdots)$ の型に分類される．前者の型の隣接交差 $b,c/a,d$ は斜格子上では必ず，Γ_+ で隣接し $\Gamma_+^{-1}(b)<\Gamma_+^{-1}(c)$ かつ $\Gamma_-^{-1}(b)>\Gamma_-^{-1}(c)$ である b と c を結んだ線と，Γ_- で隣接し $\Gamma_+^{-1}(a)<\Gamma_+^{-1}(d)$ かつ $\Gamma_-^{-1}(a)<\Gamma_-^{-1}(d)$ である d と a を結んだ線の交差となっている．後者の型の隣接交差 $b,c/d,a$ も同様に斜格子上では必ず，Γ_+ で隣接し $\Gamma_+^{-1}(b)<\Gamma_+^{-1}(c)$ かつ $\Gamma_-^{-1}(b)<\Gamma_-^{-1}(c)$ である b と c を結んだ線と，Γ_- で隣接し $\Gamma_+^{-1}(a)<\Gamma_+^{-1}(d)$ かつ $\Gamma_-^{-1}(a)>\Gamma_-^{-1}(d)$ である d と a を結んだ線の交差となっている．前者は b と c の大小関係が Γ_+ と Γ_- で逆順だが a と d の大小関係が Γ_+ と Γ_- で同順であり，後者は b と c の大小関係が Γ_+ と Γ_- で同順だが a と d の大小関係が Γ_+ と Γ_- で逆順であることが，2 つの型の違いであることに注意されたい．

例として，標準化 Γ_- (4 2 6 1 3 5) の持つ隣接交差のうち，前者の型の隣接交差 $3,4/2,6$ を図 3 (h) に，後者の型の隣接交差 $2,3/6,1$ と $4,5/6,1$ を図 3 (g) に，それぞれ線の交差上の丸印で示す．

上述のアルゴリズムの前半は Γ_- の増加順に，すなわち斜格子上で左下から右上に向かって平面を掃くよ

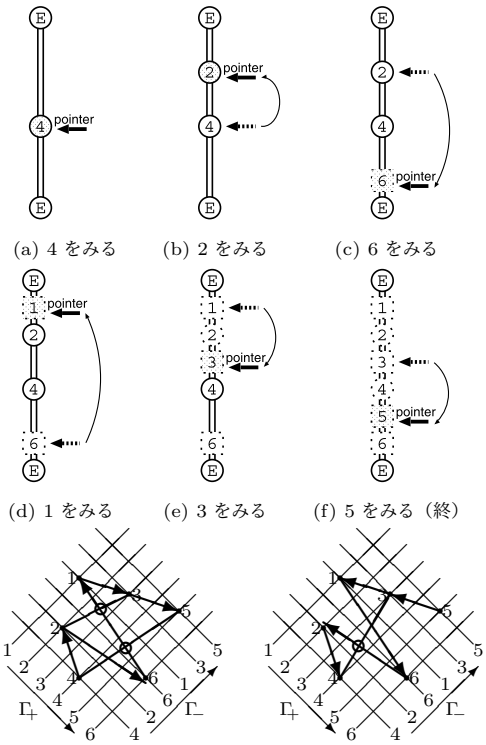


図 3 Adjcross List の標準化 Γ_ (4 2 6 1 3 5) での実行例
Fig. 3 An example of Adjcross List for normalized Γ_ (4 2 6 1 3 5).

うに各要素をたどったとき，Γ_ 増加方向で見ると Γ_+ で増加方向に隣接している 2 要素間を結んだ線分を Γ_+ 減少方向に向かって交差したときの交差点をすべて検出している（図 3 (g) 参照）．したがって，これにより後者の型の隣接交差をすべて列挙できる．アルゴリズムの後半では，Γ_ を減少順に同様にたどっているので，前者の型の隣接交差をすべて列挙できる（図 3 (h) 参照）．

3. 効率的な隣接解生成手法

3.1 到達可能な解空間

本稿では，隣接交差数が $n - \lfloor\sqrt{4n-1}\rfloor$ 以下の seq-pair，すなわち SSP だけを SA 法などで探索することを目的としている．SA 法で探索をするためには，MOVE や perturb などと呼ばれる隣接解移動操作を定義し，これにより解空間を張る必要がある．SA 法は通常，単一の初期解から始めて探索を行うため，任意の SSP から任意の SSP まで，SSP サイズの多項式オーダ回数の隣接解移動操作で到達可能であるという『到達可能性』を保証できることが望まれる．そこで本稿では，以下の『移動操作』を用いる．

【移動操作】 SSP の Γ_+ あるいは Γ_ のどちらか片方の順列において，任意の要素を 1 つ選び（この要素を移動要素と呼ぶ），同じ順列内で移動する． ■

この移動操作は，seq-pair においてよく用いられる隣接解生成操作の 1 つで『挿入変換』とも呼ばれている[13]．

3.1.1 到達可能性の証明

以下での到達可能性の証明は，隣接交差のない seq-pair だけからなる解空間での証明[14] を拡張して得られている．

移動操作は可逆的であり，たとえば要素 a を移動した後で a を元の位置に移動すると，必ず元の SSP に戻る．そのため，基本となる SSP を 1 つ決めておき，これに対して移動操作を n の多項式回だけ行うことにより SSP だけを介してあらゆる SSP に変換可能であることを示せば，到達可能性の証明として十分である．

要素名が辞書順に左から並んだ順列を基本順列 P_B と呼ぶことにする．すると seq-pair $(P_B; P_B)$ は，すべての矩形が左から辞書順に横 1 列に並んだパッキングに対応し，隣接交差を持たないのでこれは必ず SSP である．また，与えられた任意の SSP を $(P_+; P_-)$ とする．

$(P_B; P_B)$ から出発し，移動操作を適用して SSP だけを通り，$(P_+; P_-)$ に変換することは以下の手順で可能である．

Step 1：SSP $(P_B; P_B)$ に対して，Γ_+ と Γ_ に 1 要素ずつ交互に挿入整列法（insertion sort）を適用して $(P_+; P_+)$ とする．

Step 2：SSP $(P_+; P_+)$ に対して Γ_ を挿入整列法により P_- に変換して $(P_+; P_-)$ とする． ■

つまり，SSP は $(P_B; P_B) \rightarrow (P_+; P_+) \rightarrow (P_+; P_-)$ の経路を通って変換される．$(P_B; P_B) = (abcde\,;\, abcde)$ から $(P_+; P_-) = (bcead\,;\, eacdb)$ への変換例を**図 4** に示す．

ここで，**Step 1** の変換は隣接交差を生ずることなく $2(n-1)$ 回以下の移動操作で実行できることを，以下の補題 1 および補題 2 により示す．

【補題 1】 Γ_+ と Γ_ が等しい seq-pair S に移動操作を 1 回適用しても，隣接交差は生じない．

（証明） a，b，c，d が隣接交差をなす seq-pair では，一般性を失わずに $\Gamma_+ = (\cdots a \cdots bc \cdots d \cdots)$ とできる．すると，Γ_ $= (\cdots b \cdots da \cdots c \cdots)$ あるいは Γ_ $= (\cdots c \cdots ad \cdots b \cdots)$ である．前者の Γ_ では b，a と

Step 1

$(P_B; P_B) = (\underline{a}bcde; \underline{a}bcde)$
$(\underline{ba}cde; \underline{ba}cde)$
$(\underline{bac}de; \underline{bac}de)$
$(\underline{bacd}e; \underline{bacd}e)$
$(\underline{bcad}e; \underline{bcad}e)$
$(\underline{bcad}e; \underline{bcad}e)$
$(\underline{bcad}e; \underline{bcad}e)$
$(\underline{bcead}; \underline{bcad}e)$
$(P_+; P_+) = (\underline{bcead}; \underline{bcead})$

Step 2

$(P_+; P_+) = (\underline{bcead}; \underline{b}cead)$
$(\underline{bcead}; \underline{cb}ead)$
$(\underline{bcead}; \underline{ecb}ad)$
$(\underline{bcead}; \underline{eacb}d)$
$(P_+; P_-) = (\underline{bcead}; \underline{eacdb})$

図 4 SSP $(P_B; P_B) = (abcde; abcde)$ から $(P_+; P_-) = (bcead; eacdb)$ への変換例．下線付きは整列済みの部分を表す

Fig. 4 An example of converting from SSP $(P_B; P_B)$ to $(P_+; P_-)$. Elements lined below were already sorted.

d，c の独立な 2 対の要素が，後者の Γ_- では c，a と d，b の独立な 2 対の要素が各々 Γ_+ での順序に対して逆順である．よって明らかに，S に移動操作を 1 回適用しただけでは隣接交差は生じない． ■

【補題 2】 **Step 1** は変換途中で隣接交差を生じることなく，$2(n-1)$ 回以下の移動操作で必ず実行可能である．

（証明） **Step 1** は Γ_+ と Γ_- を 1 要素ずつ交互に変換するので，途中の seq-pair では Γ_+ と Γ_- は等しいか，等しいものに移動操作を 1 回適用したものである．すると補題 1 により，変換の途中で隣接交差を生じることはない．また，P_B の最も左の要素を除いた各要素に対して Γ_+ と Γ_- で 1 回ずつ移動を行うので，移動操作の回数は $2(n-1)$ 以下である． ■

続いて，**Step 2** の変換は隣接交差数が $n - \lfloor\sqrt{4n-1}\rfloor$ を超えることなく，$(n-1)$ 回以下の移動操作で実行できることを，以下の補題 3 および補題 4 により示す．

【補題 3】 **Step 2** の変換途中で隣接交差数は減少しない．

（証明） **Step 2** の変換途中で隣接交差が減少したと仮定し，このとき消滅した隣接交差は a，b，c，d がなすものとし，一般性を失わずに $\Gamma_+ = (\cdots a \cdots bc \cdots d \cdots)$ とする．すると，$\Gamma_- = (\cdots c \cdots ad \cdots b \cdots)$ あるいは $\Gamma_- = (\cdots b \cdots da \cdots c \cdots)$ である．

Step 2 開始時の SSP $(P_+; P_+)$ には隣接交差が存在しないので，この隣接交差は **Step 2** の途中で生じたことになる．a，b，c，d からなる隣接交差があると Γ_- 上で d は b もしくは c より前に位置し，**Step 2** では b や c が d の後ろに移動することはない（挿入整列では前方向に移動）ので，この隣接交差は d が挿入整列の対象となったときもしくはその後に生じている．

d が移動した後，a，b，c，d の相対位置関係は P_- と等しく，以後，相対位置関係は変わらない．このため，**Step 2** 途中で隣接交差が消滅したとなると，後の整列でその隣接交差の Γ_- 上での隣接対の間，すなわち a と d の間に適当な要素 x が挿入されたことになる．明らかに x は P_+ において d より後ろにあるから，x の挿入により a，b，c，d による隣接交差が解消すると同時に a，b，c，x により新たな隣接交差が生じる．したがって，隣接交差は 1 つ消滅すれば必ず 1 つ生成され，**Step 2** の変換途中で隣接交差数が減少することはない． ■

【補題 4】 **Step 2** は変換途中で隣接交差数が $n - \lfloor\sqrt{4n-1}\rfloor$ を超えることなく，$(n-1)$ 回以下の移動操作で必ず実行可能である．

（証明） 補題 3 により，変換途中で隣接交差数は減少することがないので，変換途中の隣接交差数は必ず $(P_+; P_-)$ の隣接交差数以下である．また，P_+ の最も左の要素を除いた各々の要素に対して Γ_- 上で 1 回ずつ移動を行うので，移動操作の回数は $(n-1)$ 以下である． ■

補題 2，4 から次の定理が得られる．

【定理 1】 移動操作で得られる提案隣接解により張られる解空間の直径は $6(n-1)$ 以下である． ■

任意の SSP $(P_+^1; P_-^1)$ から任意の SSP $(P_+^2; P_-^2)$ までの距離は，P_B を P_+^2 とすることにより，$(P_+^1; P_-^1) \to (P_+^1; P_+^1) \to (P_+^2; P_+^2) \to (P_+^2; P_-^2)$ と変換できるので，直径の上界は $4(n-1)$ に減らすことができる．ここで，$(P_+^1; P_-^1) \to (P_+^1; P_+^1)$ の変換は前述 **Step 2** の逆方向の変換であり，補題 4 からこの過程で隣接交差数が $n - \lfloor\sqrt{4n-1}\rfloor$ を超えることはない．

3.2 隣接解生成の効率的な手法

移動操作に基づき，すべての隣接解（隣接解中で許容なもの）がランダムに選ばれうる手法を考える．もし，つねに同じ隣接解が選ばれたり，まったく選ばれない隣接解が存在すると，到達可能性が失われたり，解空間の直径が大きくなったりする可能性があるので，すべての隣接解が必ず選ばれうる必要がある．このため，すべての隣接解が選ばれうる手法としては，

(1) 許容解が得られるまで，ランダムに隣接解を作る操作を繰り返す．
(2) 非許容解に非常に大きなコストを課し，非許容解も含めてランダムに隣接解を作る．
(3) 非許容解を，似た許容解に変換する[2]．
(4) すべての隣接解について許容か否かを調べて，許容なものからランダムに 1 つ選び出す（清田らが提案[14]）．

というものが用いられてきた．

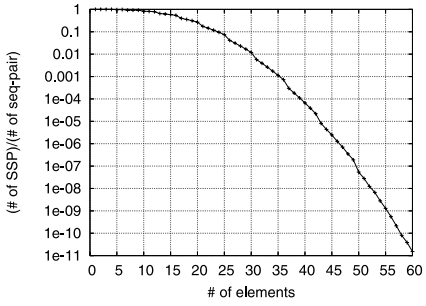


図 5 Seq-pair に対する SSP の割合：サイズが 17 以下ではすべての seq-pair の中での割合．サイズが 18 以上 49 以下ではランダムに生成した 10^9 個の，50 以上では 10^{10}〜約 $2*10^{12}$ 個の seq-pair の中での割合
Fig. 5 The ratio of SSP to a variety of seq-pair.

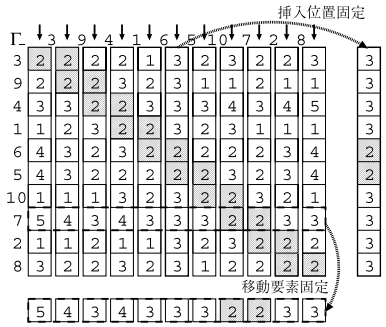


図 6 標準化 Γ_ (3 9 4 1 6 5 10 7 2 8) のテーブル[14] の概念図．下側に取り出したように 1 行だけに注目するのが「移動要素固定」，右側に取り出したように 1 列だけに注目するのが「挿入位置固定」．各マス内の数値は，その行の Γ_ の要素をその列の Γ_ の要素間に移動したときに生成される隣接解において，隣接交差数がいくつであるかを表す．網掛けのマスに対応する隣接解は元の解と同じになるので，隣接交差数も変わらない
Fig. 6 Conceptual diagram of table for normalized Γ_ (3 9 4 1 6 5 10 7 2 8).

(1) の手法は最もよく使われているが，解の中で非許容なものの割合が多い場合には，1 つの許容な隣接解を得るのに多数の非許容解を生成することになり非効率である．(2) の手法を用いた場合は，SA 法で最終的に許容解に収束する保証がない．(3) の手法は，解によって得られる確率が違ってしまうことや，対称性がない（1 回の隣接解移動により解 a から解 b を得られても，b から a は得られない場合がある）ために，解空間が歪んでしまう．(4) の手法は，すべての隣接解を列挙するので必ず相当量の時間がかかってしまう．

本稿で対象としている SSP は，同サイズの seq-pair と比べて，**図 5** に示すようにサイズが大きくなるのにともなって劇的にその割合が少なくなるので，(1)，(2)，(3) の手法を用いると非常に効率が悪くなってしまう．

(4) の手法は，全隣接解の中で SSP の割合が多い場合には相対的に効率が悪いが，SSP の割合が少ないときも効率が変わらないため，このときには相対的に効率が良い．そこで本稿では，原則的には清田ら[14] と同じに (4) の手法を用いることにする．

3.2.1 隣接解列挙テーブルの縮小

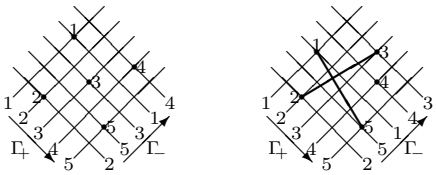
清田らはすべての隣接解について許容か否かを調べるために隣接解列挙テーブルを作成した[14]．説明のために，隣接交差数が 2 の標準化 Γ_ (3 9 4 1 6 5 10 7 2 8) について作成したテーブルの概念図を**図 6** に示す．テーブルの各行はテーブルの左側に示した Γ_ の各要素に，各列はテーブルの上側に示した Γ_ の各要素間に対応する．テーブルの各マス内の数値は，その行の Γ_ の要素をその列の Γ_ の要素間に移動したときに生成される隣接解において，隣接交差数がいくつであるかを表している．たとえば，要素 3 を要素 6 と 5 の間に移動した場合，該当するマスには数値 3 が示されているので，得られる標準化 Γ_ の隣接解 (9 4 1 6 3 5 10 7 2 8) の隣接交差数が 3 であることが分かる．

移動操作で生成されうる隣接 seq-pair の数は移動要素と移動先の組合せにより Γ_+ と Γ_- を合わせて約 $2n^2$ ある[14] ので，上述のようなテーブルを作成しようとするとテーブルのリセットだけでサイズの 2 乗時間が必要となる．SSP 表現に基づいてパッキングを SA 法探索するときは隣接 SSP を次々と作成して評価することになるが，もしも隣接 SSP 作成にサイズの 2 乗時間を使うと，この部分が解探索のボトルネックとなってしまい，「線形時間でパッキングに変換することができる」という SSP の特長を損なってしまう．このため，隣接解生成の時間複雑度には，ほぼ線形時間で可能であることが要求される．

そこで，テーブルの大きさを $O(n)$ に減らすため，移動要素をランダムに決定してテーブルを作ってから移動先を決める**「移動要素固定」**，もしくは，移動先をランダムに決定してテーブルを作ってから移動要素を決める**「挿入位置固定」**方法を用いることにする．また同時に，Γ_+ と Γ_- のどちらで移動するかも決めることにする．移動要素固定は，テーブルのいずれか 1 行だけを作成し，その中から挿入位置を選び出すことを意味する．また挿入位置固定は，テーブルのいずれか 1 列を作成し，その中から移動要素を選び出すことを意味する．たとえば図 6 で，移動要素固定方法で移動要素を 7 としたとき，テーブルの下側に示した 1 行だけを作成すればよい．また，挿入位置固定方法

表 1 必ず隣接交差数が増加する SSP の割合の比較結果
Table 1 Average number of unsuccessful times in generating adjacent solution of SSP.

サイズ (n)		4	5	6	7	8	9	10	11	12	13	14
移動要素固定 ($\times 10^{-3}$)	必ず非 SSP 化	0	0	0	0.863	0.725	0.558	1.52	1.42	1.36	2.46	49.4
	必ず隣接交差数増加	0	3.33	7.41	10.9	13.4	15.0	15.8	16.2	16.1	15.8	15.3
挿入位置固定 ($\times 10^{-9}$)	必ず非 SSP 化	0	0	0	0	0	0	0	0	0	0.351	5.28
	必ず隣接交差数増加	0	0	0	0	0	0	200	434	586	635	602



(a) 移動前は隣接交差なし (b) 太線の交差が隣接交差

図 7 $S = (1\,2\,3\,4\,5\,;\,2\,5\,3\,1\,4)$ での要素 3 の移動による隣接交差増加
Fig. 7 An adjacent cross generated by moving element 3 in $S = (1\,2\,3\,4\,5\,;\,2\,5\,3\,1\,4)$.

で挿入位置を要素 6 と 5 の間にしたとき，テーブルの右側に示した 1 列だけを作成すればよい．

これによりテーブルの大きさが $O(n)$ になるので高速化が期待できるが，欠点が 2 つある．1 つは，隣接 SSP ごとに選ばれる確率が異なってしまうことである．もう 1 つは，テーブルを作って調べた結果，隣接交差数が $n - \lfloor\sqrt{4n-1}\rfloor$ 以下になるものが存在しなかった場合，移動要素もしくは移動先を変更してテーブルを再作成する必要が生じてしまい，多くの時間が必要になってしまうことである．以下では後者の欠点に基づき，移動要素と移動先のどちらを先に決定した方が好ましいかを考える．

本章では以降，説明の簡単のため要素の移動は Γ_- で行われるものとし，標準化 Γ_-（$\Gamma_+ = (1\,2\,\cdots)$）を用いて説明するが，移動が Γ_+ で行われたとしても同様である．

まず，移動要素を決定してテーブルを作ってから移動先を決めることを検討する．いま，標準化 Γ_- が $(2\,5\,3\,1\,4)$ である seq-pair S を考える（図 7 (a) 参照）．S には隣接交差はないが，もし標準化 Γ_- 上で要素 3 が移動要素として選ばれると，隣接交差数は必ず 1 増える．すると，このパターンを含んだ seq-pair で，しかも隣接交差数が上限の $n - \lfloor\sqrt{4n-1}\rfloor$ であるものについては，この標準化 Γ_- での要素 3 に該当する要素を移動要素として選んだ場合には，どの移動先に行っても隣接交差数が $n - \lfloor\sqrt{4n-1}\rfloor$ を超えてしまい，SSP サイズが大きければ結構な確率でテーブルの再作成の必要が生じてしまうと予想される．

たとえば，標準化 Γ_- が $(2\,5\,3\,1\,4\,9\,11\,7\,12\,6\,10\,8)$ である seq-pair を考える．この Γ_- では，前側の要素 $1, 2, \cdots, 5$ が前述 S の Γ_- と同じ位置関係で連続している．後ろ側には要素 $6, 7, \cdots, 12$ が，隣接交差を 6 個持つサイズ 7 の標準化 Γ_- $(4\,6\,2\,7\,1\,5\,3)$ と同じ相対位置関係で連続しており，全体での隣接交差数は要素 12 個の SSP の上限と同じ 6 となっている．この SSP で要素 3 を移動要素として選択すると，どこに移動しても隣接交差数は上限の 6 を超えてしまい，SSP ではなくなってしまう．

そこでこの移動要素固定方法により隣接解を生成した際に，どのくらいの確率でテーブルの再作成が必要となるのかをサイズ 14 以下の seq-pair について全探索して調べた結果を**表 1** の「移動要素固定」の 2 行に示した．ここで「必ず非 SSP 化」の行では，「SSP とその中から選択された移動要素との組合せ」の中で，この移動要素をどこに移動しても隣接交差数が $n - \lfloor\sqrt{4n-1}\rfloor$ を超えてしまい非 SSP となってしまうものが占める割合を表している．また参考までに「必ず隣接交差数増加」の行では先に決定した移動要素をどこに移動しても必ず隣接交差数が増えてしまう「seq-pair と移動要素の組合せ」の割合を表している．表 1 から，この方法ではサイズ 5 以上ではどこに移動しても必ず隣接交差数が増えてしまう seq-pair が存在し，さらにサイズ 7 以上ではどこに移動しても非 SSP となってしまう SSP が存在することが分かる．

次に，移動先を決定してテーブルを作ってから移動要素を決めることを検討する．こちらについては前述の場合とは異なり，移動先が決まると必ず隣接交差数が増える seq-pair を考え出すのは困難である．そこで，移動先を先に決定した際にどのくらいの確率でテーブルの再作成が必要となるのかを先述の「移動要素固定」の場合と同様にサイズ 14 以下の seq-pair について全探索し，この結果を表 1 の「挿入位置固定」の 2 行に示す．これによると，必ず隣接交差数が増加する seq-pair はサイズが 9 以下ではまったく存在しなかったが，サイズが 10 では逆順も含めて 8 種存在し，たとえば標準化 Γ_- $(3\,9\,4\,1\,6\,5\,10\,7\,2\,8)$ でその中央を移動先に決めると，隣接交差数が現在の 2 から，必ず 3 に増えてしまう（図 6 参照）．しかしサイズ 10 では，隣接交差数が 2 から 3 に増える場合と 1 から 2 に増える場合だけで $n - \lfloor\sqrt{4n-1}\rfloor$ を超えることは

なく，サイズ 13 以上の SSP でないとテーブルの再作成に至ることはないことが分かり，しかもそれは先述の移動要素固定の場合と比べてきわめて少なく，稀であった．そこで本稿では，移動先を先に決定することにする．

なお，サイズの大きな SSP においてどのくらい『稀』であるかについては，4.1 節において実験的に確かめるが，『ほぼ線形時間で隣接解生成が可能』といえると思われるほどに『稀』である．

3.3 各々の移動要素に対する，隣接交差の増減数の算出

前述のように，本稿で提案する隣接解生成手法ではまず移動先をランダムに決定し，これに基づいて移動要素ごとに隣接交差数の増減を求めたテーブルを作成し，隣接交差数が $n-\lfloor\sqrt{4n-1}\rfloor$ 以下になる移動要素の中からランダムに 1 つを選択する．この中で最も難しいのがテーブルの作成である．ここでは隣接交差が消滅する場合と生成する場合に分けて考えていくが，隣接交差数の増減を求める際，「1 つの消滅を必ずともなう 1 つの生成」や「1 つの生成を必ずともなう 1 つの消滅」は相殺されるので無視する．たとえば，標準化 Γ_- (2 4 1 3 5) は隣接交差 $2,3/4,1$ を持つが，4 と 1 の間に 5 が移動すると，(2 4 5 1 3) となり，隣接交差 $2,3/4,1$ は消滅するが，代わりに隣接交差 $2,3/5,1$ が生じるので，このような増減については数えないことにする．

まず，隣接交差 $a,b/c,d$ が消滅するのはどういう場合かを考える．標準化 Γ_- 上での移動操作であるとすると，隣接交差の条件から明らかに，4 つの要素のうちの 1 つが移動する場合と，Γ_- 上で隣接している c と d の間に他の要素（e とする）が移動してくる場合だけである．ところが後者は前節で述べた例のように必ず代わりの隣接交差 $a,b/e,d$ もしくは $a,b/c,e$ が生じてしまうので「1 つの生成を必ずともなう 1 つの消滅」であり，ここでは隣接交差の消滅として数えないことになる．したがって，隣接交差が消滅するとして数えなくてはならないのは，隣接交差を構成する 4 つの要素のうちの 1 つが移動する場合だけである．これについては，3.3.1 項で述べる．

次に，どういうときに隣接交差 $a,b/c,d$ が生成されるかを考えると，消滅の場合の話しを逆にすれば，a，b，c，d のいずれかの要素が移動してくる場合と，Γ_- 上で c と d の間にあった要素が移動することにより c と d が隣接する場合の 2 通りだけであるが，後者は明らかに「1 つの消滅を必ずともなう 1 つの生成」であり，考慮すべきなのは前者だけである．これはさらに，生じる隣接交差において移動要素がその内側となるか外側となるかにより分類して算出することにし，それぞれを 3.3.2 項と 3.3.3 項で述べる．

3.3.1 隣接交差の消滅

標準化 Γ_- 上での移動操作により隣接交差が消滅するのは，前述のように隣接交差を構成する 4 つの要素のうちの 1 つが移動する場合だけである．そこで 2.4.2 項で詳述した，与えられた seq-pair のすべての隣接交差を $O(n+k)$ 時間で列挙するアルゴリズム Adjcross List（n と k は各々，seq-pair のサイズと隣接交差数）を利用する．そしてすべての隣接交差について，その構成要素が移動要素になったときにはこの隣接交差が消滅するので，構成要素に対して隣接交差数が 1 だけ減ると記録していく．

ただし，相殺になって減らない場合がある．隣接交差 $a,b/c,d$ で Γ_- で隣接している c，d について，Γ_- で c の直前の要素 x が $\Gamma_+^{-1}(x)<\Gamma_+^{-1}(a)$ かつ $\Gamma_+^{-1}(c)<\Gamma_+^{-1}(a)$ であるか，$\Gamma_+^{-1}(x)>\Gamma_+^{-1}(a)$ かつ $\Gamma_+^{-1}(c)>\Gamma_+^{-1}(a)$ であるなら，c が移動して隣接交差 $a,b/c,d$ が消滅しても隣接交差 $a,b/x,d$ が生成されて相殺されるので，この場合には c が移動要素となっても隣接交差数は減らない．d についても同様である．

たとえば，隣接交差 $3,4/5,1$ を持つ標準化 Γ_- (3 5 1 2 4) において，1，3，4，5 が移動要素となるとこの隣接交差は消滅する．このうち，3，4，5 の場合には隣接交差数が 1 減少するが，1 の場合には隣接交差 $3,4/5,2$ が代わりに生成されるため，隣接交差数は相殺されて変化しない．

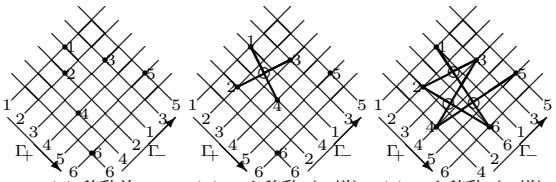
この手順の計算複雑度は Adjcross List と同じ $O(n+k)$ である．なお「隣接交差の消滅」での減数は，移動先がどこに指定されたかには依存しない．

3.3.2 移動要素が内側になっての隣接交差の生成

もし移動先が Γ_- の端なら，移動要素が内側な隣接交差は生成しえないので，この場合の増数はすべて零になる．以下ではこれ以外の場合を考え，移動先の左右の要素が必ず存在するので各々を ℓ と r とし，移動要素を q とする．

q の値が ℓ と r の間であるなら，q の移動により隣接交差 $a,b/\ell,q$ もしくは $a,b/q,r$ が生成されたとしても，それ以前に隣接交差 $a,b/\ell,r$ が必ず存在しており，しかもこれは移動により消滅する．したがって，このときには増減は相殺される．

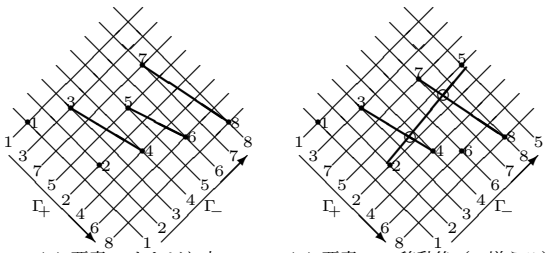
q が ℓ と r のどちらよりも大きいなら，ℓ と r のどちらよりも大きいか等しくて q 未満で大きさの差が 1 で移動先をまたいだ要素対を考えると，この要素対は必ず ℓ と q，もしくは q と r の要素対と隣接交差を



(a) 移動前 (b) 4 を移動（1 増） (c) 6 を移動（3 増）

図 8 標準化 Γ− (6 4 2 1 3 5) で中央の 2，1 間を移動先としての隣接交差増加．太線の交わったところの丸印が生成した隣接交差を表す

Fig. 8 Adjacent crosses generated by moving element 4 or 6 into inside of element 2 and 1. Circled cross points show adjacent crosses generated by moving elements.



(a) 要素 2 をたどり中 (b) 要素 5 の移動後（2 増える）

図 9 標準化 Γ+(1 3 7 5 2 4 6 8;1 2 3 4 5 6 7 8) に対し，この Γ− の右端が移動先のときの，要素 5 の移動による隣接交差増加．丸印が生成された隣接交差を表す

Fig. 9 Adjacent crosses generated by moving element 5 into rightmost in Γ− of normalized Γ+(1 3 7 5 2 4 6 8; 1 2 3 4 5 6 7 8). Circled cross points show adjacent crosses generated by moving elements.

生成する．したがって，要素対の数だけ隣接交差が増える．q が ℓ と r のどちらよりも小さい場合にも同様である．

この増加数の計算は，以下のアルゴリズムに示すように，線形時間で効率的に求めることができる．

for $(c = 0,\ p = \max(\ell, r);\ p \leq n-2;\ p++)${
  if (要素 p と要素 p+1 が移動先をまたいでいる) $++c$;
  移動要素が $p+2$ なら c だけ増えると記録; }
for $(c = 0,\ p = \min(\ell, r);\ p > 2;\ p--)${
  if (要素 p と要素 p+1 が移動先をまたいでいる) $++c$;
  移動要素が $p-2$ なら c だけ増えると記録; } ■

たとえば，標準化 Γ− (6 4 2 1 3 5) において（**図 8** 参照）2 と 1 の間が移動先として指定された場合，2 と 1 のどちらよりも小さいか等しい要素は 1 だけで対はなく，どちらよりも大きいか等しい要素 2，3，4，5，6 の中で大きさの差が 1 で移動先をまたいでいる対は 2，3 と 4，3 と 4，5 と 6，5 である．したがって，3 より大きい要素 q が移動要素なら隣接交差 $2,3/2,q$ もしくは $2,3/q,1$ が生成され（実際は後者），4 より大きい要素 q が移動要素なら加えて隣接交差 $3,4/2,q$ もしくは $3,4/q,1$ が生成され（実際は前者），5 より大きい要素 q が移動要素ならさらに加えて隣接交差 $4,5/2,q$ もしくは $4,5/q,1$ が生成され（実際は後者），6 より大きい要素はない．これらにより，移動要素が 3 なら 0，4 なら 1，5 なら 2，6 なら 3 だけ隣接交差が増えることが分かる．

3.3.3 移動要素が外側になっての隣接交差の生成

標準化 Γ− から標準化 Γ+ を求め，これに対して 2.4.2 項で詳述した Adjcross List を応用して適用する．標準化 Γ+ に対しての隣接交差列挙のため，ポインタや双方向リストには Γ− で何番目の要素かが入ることになる．Adjcross List で標準化 Γ+ の要素 p をたどっているとする．すなわち，双方向リスト上で p にポインタがある．

p が Γ− 上で移動先よりも前側にあるなら，標準化 Γ+ で p よりも 1 つ前の要素 p_p（すなわち $\Gamma_+^{-1}(p_p) = \Gamma_+^{-1}(p)-1$）が移動要素になったとき，双方向リスト上で p と移動先の 1 つ手前との間にある要素（複数ありうるが，そのうちの 1 つを x とする）は Γ− で x の直後の要素（y とする，つまり $\Gamma_-^{-1}(y) = \Gamma_-^{-1}(x)+1$）と，隣接交差 $p_p, p/x, y$ を生成する．ただし，p_p が x と一致していた場合には，この隣接交差が生じることはないが，もし Γ− で x（すなわち p_p）の直前の要素（z とする，つまり $\Gamma_-^{-1}(z) = \Gamma_-^{-1}(x)-1$）が標準化 Γ+ 上で x よりも前側（つまり $\Gamma_+^{-1}(z) < \Gamma_+^{-1}(x)$）であったなら，隣接交差 $p_p, p/z, y$ を生成する．

たとえば，標準化 Γ− (1 5 2 6 4 7 3 8) で移動先が右端だとする．この標準化 Γ− から得られる標準化 Γ+ は (1 3 7 5 2 4 6 8) である．今，標準化 Γ+ を要素 2（標準化 Γ− での 5）までたどったとき，要素 2 は Γ− で移動先よりも前なので，Γ− で 1 つ前の要素 5 が移動したときのことを考える．このとき双方向リストには 1，3，5，7 が入っているので，2 と右端との間にある 3，5，7 により 3 つの隣接交差が生成されそうにみえるが，標準化 Γ+ で 2 の直前にある 5 については隣接交差を生成しないので，要素 5（標準化 Γ− での 4）を移動したときの隣接交差増加数は 2 つ（5,2/3,4 と 5,2/7,8）である（**図 9** 参照）．

p が Γ− 上で移動先よりも後ろ側にあるなら，標準化 Γ+ で p よりも 1 つ後ろの要素 p_a が移動要素になったとき，双方向リスト上でポインタのある p と移動先の間にある要素（x とする）は Γ− で x の直後の要素（y とする）と，隣接交差 $p, p_a/x, y$ を生成する．ただし，p_a が y と一致していた場合には，この

隣接交差が生じることはないが，もし Γ_- で y の直後の要素（z とする）が標準化 Γ_+ 上で y よりも後ろ側であったなら，隣接交差 $p, p_a/x, z$ を生成する．

上述のように，双方向リスト上で移動先との間にある要素の数を定数時間で得るため，Adjcross List で標準化 Γ_+ の要素を 1 つずつたどる際，双方向リスト上で移動先とポインタとの間にいくつの要素が挿入されているかを記憶させておく（ただし，移動先のすぐ左の要素は移動操作により，Γ_- 上でその直後の要素が移動要素に変わってしまうので除外する）．これは双方向リスト上を移動したり，挿入/削除する際にカウントしたりしておけば，全体で $O(n+k)$ 時間で可能である．

この後，隣接交差を列挙するアルゴリズム Adjcross List と同じに，標準化 Γ_+ を逆順にたどって同様の処理をする必要があるが，割愛する．ここまでに詳述した，標準化 Γ_+ の順にたどるという前半のアルゴリズムを疑似コードにしたものを**図 10** に示す．

アルゴリズム 移動要素が外側での隣接交差生成数（入力：標準化 Γ_+, 挿入位置（$\Gamma_-(w)$ と $\Gamma_-(w+1)$ の間）
空の双方向リストを用意する；
foreach ($p = \Gamma_+$ の要素を前から 1 つずつ){
  if ($p < w$){
    $p_p = \Gamma_+(\Gamma_+^{-1}(p)-1)$;
    foreach (x| $p < x$ && $x < w$ && x は双方向リスト上に現存)
      if ($x \neq p_p$)
        移動要素が p_p だと隣接交差 p_p, p / $x, \Gamma_-(\Gamma_-^{-1}(x)+1)$ が生じると登録；
      else if ($\Gamma_+^{-1}(\Gamma_-(\Gamma_-^{-1}(x)-1)) < \Gamma_+^{-1}(x)$)
        移動要素が p_p だと隣接交差 p_p, p / $\Gamma_-(\Gamma_-^{-1}(x)-1), \Gamma_-(\Gamma_-^{-1}(x)+1)$ が生じると登録；
  }else if ($p > w+1$){
    $p_a = \Gamma_+(\Gamma_+^{-1}(p)+1)$;
    foreach (x| $w+1 < x$ && $x < p$ && x は双方向リスト上に現存)
      if ($\Gamma_-(\Gamma_-^{-1}(x)+1) \neq p_a$)
        移動要素が p_a だと隣接交差 p, p_a / $x, \Gamma_-(\Gamma_-^{-1}(x)+1)$ が生じると登録；
      else if ($\Gamma_+^{-1}(\Gamma_-(\Gamma_-^{-1}(p_a)+1)) > \Gamma_+^{-1}(p_a)$)
        移動要素が p_a だと隣接交差 p, p_a / $x, \Gamma_-(\Gamma_-^{-1}(p_a)+1)$ が生じると登録；
  }
  if (p は最大の要素ではない && $\Gamma_+^{-1}(p+1) > \Gamma_+^{-1}(p)$)
    // $p+1$ は未走査
    p を双方向リストに挿入；
  if ($\Gamma_+^{-1}(p-1) < \Gamma_+^{-1}(p)$) // 要素 $p-1$ は双方向リスト上に存在
    $p-1$ を双方向リストから削除；
}　（**アルゴリズム 移動要素が外側での隣接交差生成数 終.**）

図 10 アルゴリズム 移動要素が外側での隣接交差生成数
Fig. 10 Algorithm of counting the number of adjacent crosses generated by MOVE operation where a move element will be one of outside elements of every new adjacent cross.

4. 実験結果

4.1 隣接 SSP 生成における，失敗回数と計算時間の比較

提案した隣接 SSP 生成手法を，C 言語で計算機実装した（以下では提案手法と呼ぶ）．また，比較として，最も一般的と思われる「(1) 隣接交差数が $n-\lfloor\sqrt{4n-1}\rfloor$ 以下の seq-pair が得られるまで，ランダムに隣接 seq-pair を作成する」手法についても計算機実装した（以下では単純手法と呼ぶ）．これらの手法により次々と隣接 SSP へと移っていく際に，隣接 SSP 生成で失敗した回数や計算時間を測定して比較した．なお，これらの手法での効率は隣接交差数に依存するため，$\Gamma_+ = \Gamma_-$ な SSP から始めて隣接交差数が $n-\lfloor\sqrt{4n-1}\rfloor-n/100$ 以上で $n-\lfloor\sqrt{4n-1}\rfloor$ 以下の隣接交差を持つ SSP を得て，これを初期 SSP として測定を行った．

まず，提案手法（挿入位置固定）において仮決定した移動先では SSP を得ることができずにテーブルを再作成した回数の平均を**表 2** の上段に示した．ここでの単位は，ppb（parts per billion）を用いている．また，単純手法において，1 つの隣接 SSP を得るのに何回その生成に失敗したかを調べて，その平均を表 2 の中段に示した．さらに参考のために比較手法として，移動要素固定手法において，その移動要素では SSP を得ることができずにテーブルを再作成した回数の平均を表 2 の下段に示した．ここでの単位は，ppth（parts per thousand）を用いている．これらを見比べると，提案手法においてテーブルを作り直すことは非常に稀であるのに対して，単純手法はもちろん比較手法では

表 2 隣接 SSP 生成における失敗（提案手法ではテーブルの再作成，単純手法では隣接 seq-pair の再生成）回数の平均
Table 2 Average number of unsuccessful times in generating adjacent solution of SSP.

SSP サイズ (n)		8	16	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384
隣接交差最大数		3	9	21	49	106	225	467	961	1,958	3,969	8,011	16,129
提案手法 挿入位置固定	平均失敗回数 ($\times 10^{-9}$)	0	0.25	9.75	31.3	71.5	103	121	139	145	139	133	120
	試行回数	$4*10^9$	$4*10^9$	$4*10^9$	$4*10^9$	$4*10^9$	$4*10^9$	$4*10^9$	$4*10^9$	$4*10^9$	$2*10^9$	$2*10^9$	$2*10^9$
単純手法	平均失敗回数	0.0539	0.196	0.749	1.92	4.42	9.44	19.6	39.8	80.2	160	319	599
	試行回数	10^8	10^8	10^8	10^8	10^8	10^8	10^8	10^8	10^8	10^7	10^7	10^6
比較手法 移動要素固定	平均失敗回数 ($\times 10^{-3}$)	0.662	2.34	6.90	10.4	13.3	14.8	15.7	16.0	15.9	14.5	14.8	15.8
	試行回数	$4*10^9$	10^8	10^8	10^8	10^7	10^7	10^6	10^6	10^6	10^5	10^5	10^5

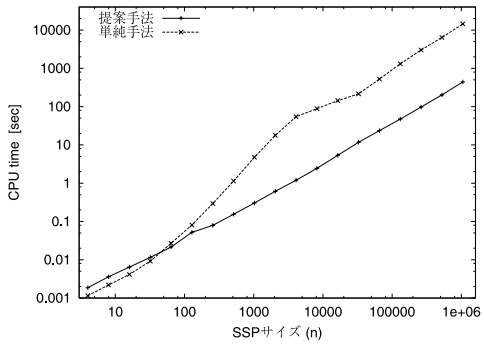


図 11 隣接解生成時間の比較（Pentium IV 2.4 GHz で測定）
Fig. 11 Comparison of time to make an adjacent solution.

頻繁に失敗し，しかも SSP サイズが大きくなると容易には隣接 SSP が得られないことが分かる．

提案手法と単純手法における，1 回の隣接解生成での所要時間を比較したものを**図 11** に示す．提案手法はほとんど直線になっており，その傾きを最小自乗法で求めると $n^{0.993}$ に比例していてほぼ線形時間で実行可能なことが分かる．また，図から SSP サイズが 50 以上では提案手法の方が高速であることが分かる．

5. ま と め

近年提案された，SSP（selected sequence-pair）と SA 法を組み合わせてパッキング探索をするため，到達可能性が保証できる隣接 SSP と，効率良く隣接 SSP を得ることができる手法を提案した．そして計算機実験により，提案手法はほぼ線形な時間で隣接 SSP を得ることができることを確かめた．

今後の課題としては，SA 法によるパッキング実験を多量に繰り返して CPU 時間に対する面積の分布を調べることにより，提案手法で隣接 SSP ごとに選択される確率が異なってしまうことの影響を調べることがあげられるであろう．

参 考 文 献

1) Murata, H., Fujiyoshi, K., Nakatake, S. and Kajitani, Y.: VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair, *IEEE Trans. CAD*, Vol.15, No.12, pp.1518–1524 (1996).
2) Murata, H., Fujiyoshi, K. and Kaneko, M.: VLSI/PCB Placement with Obstacles Based on Sequence Pair, *IEEE Trans. CAD*, Vol.17, No.1, pp.60–68 (1998).
3) Fujiyoshi, K. and Murata, H.: Arbitrary Convex and Concave Rectilinear Block Packing using Sequence-Pair, *IEEE Trans. CAD*, Vol.19, No.2, pp.224–233 (2000).
4) Lai, J., Lin, M.-S., Wang, T.-C. and Wang, Li-C.: Module Placement with Boundary Constraints Using the Sequence-Pair Representation, *ASP-DAC*, pp.515–520 (2001).
5) Murata, H., Fujiyoshi, K., Watanabe, T. and Kajitani, Y.: A Mapping from Sequence-Pair to Rectangular Dissection, *ASP-DAC*, pp.625–633 (1997).
6) Kodama, C. and Fujiyoshi, K.: Selected Sequence-Pair: An efficient decodable packing representation in linear time using sequence-pair, *ASP-DAC*, pp.331–337 (2003).
7) Guo, P., Cheng, C.-K. and Yoshimura, T.: An O-Tree Representation of Non-Slicing Floorplan, *IEEE DAC*, pp.268–273 (1999).
8) Takahashi, T.: A New Encoding Scheme for Rectangle Packing Problem, *ASP-DAC*, pp.175–178 (2000).
9) 池田，児玉，中込，藤吉：Selected Sequence-Pair を用いたレクトリニア多角形パッキングの高速化，信学技報，VLD2003-103, pp.199–204 (2003).
10) Tang, X. and Wong, D.F.: FAST-SP: A FAST algorithm for Block Placement based on Sequence-Pair, *ASP-DAC*, pp.521–526 (2001).
11) Stockmeyer, L.: Optimal Orientations of Cells in Slicing Floorplan Designs, *Info. and Control*, Vol.57, pp.91–101 (1983).
12) Takashima, Y. and Murata, H., The tight upper bound of the empty rooms in floorplan, *SASIMI2001*, pp.264–271 (2001).
13) 藤吉，大村，井尻：Simulated Annealing 法探索に適した Sequence-Pair によるパッキング解空間，信学技報，VLD99-118, pp.9–16 (2000).
14) 清田，藤吉：Sequence-Pair 表記された一般構造フロアプランの Simulated Annealing 法探索，信学論，Vol.J84-A, No.7, pp.929–938 (2001).

(平成 16 年 7 月 12 日受付)
(平成 17 年 5 月 9 日採録)

藤吉 邦洋（正会員）

平成 4 年東京工業大学大学院博士後期課程満期退学．平成 9 年東京農工大学工学部電子情報工学科講師．現在，同大学工学部助教授．博士（工学）．VLSI 設計と組合せアルゴリズムの研究に従事．IEEE，電子情報通信学会各会員．

児玉　親亮

平成 13 年東京農工大学大学院工学研究科博士前期課程修了．平成 15 年より同大学院工学教育部博士後期課程在学．IEEE，電子情報通信学会各学生会員．

清田　紘司

平成 13 年東京農工大学大学院工学研究科博士前期課程修了．在学中，VLSI レイアウト設計に関する研究に従事．