Selected Sequence-Pair: An Efficient Decodable Packing Representation in Linear Time using Sequence-Pair

Chikaaki KODAMA and Kunihiro FUJIYOSHI
Department of Electrical and Electronic Engineering, Tokyo University of Agriculture & Technology,
Koganei Tokyo, 184-8588, Japan e-mail: kodamada@fjlab.ei.tuat.ac.jp, fujiyosi@cc.tuat.ac.jp

Abstract— In this paper, we propose "selected sequence-pair" (SSP), a sequence-pair (seq-pair) with the limited number of subsequences called adjacent crosses. Its features are: (1) The smallest packing based on a given SSP can be obtained in $O(n)$ time, where n is the number of rectangles. (2) An arbitrary packing can be represented by SSP. (3) The total representation number of SSP of size n is not more than that of rectangular dissection of the same size with $n-\lfloor\sqrt{4n-1}\rfloor$ empty rooms (the necessary number of empty rooms to represent an arbitrary packing). To realize these features of SSP, we propose an algorithm to enumerate all adjacent crosses on a seq-pair in linear time of $n+k$ (k is the number of adjacent crosses). Also we apply a conventional method to convert a seq-pair without adjacent crosses to an equivalent Q-sequence, representation of rectangular dissection, in $O(n+k)$ time. A move operation to obtain an adjacent solution efficiently is proposed to perturb SSP for Simulated Annealing. From experimental results, we confirmed the proposed method was carried out in linear time and was more efficient than the conventional method when SSP size got bigger.

I. Introduction

One of the major problems in VLSI layout design is in placing the rectangular modules as densely as possible. The merits of sequence-pair (seq-pair) [1], a representation used in its solution, are in that it can represent an arbitrary rectangle packing and each seq-pair always has its corresponding packing(s). Since a seq-pair defines the relative positions of all pairs of rectangles, it has wide applications. For instance, in pre-placed modules [2], rectilinear polygon packing [3], boundary constraints [6] and conversion to an equivalent rectangular dissection from a packing [7]. In [1], a method to get the bottom left corner packing (there is no block that can shift left and down from its original position with other components fixed) from a given seq-pair of n rectangles in $O(n^2)$ time was proposed using horizontal/vertical constraint graphs. Later, another method to get packing in $O(n \log n)$ time [4] was proposed. Recently a study [5] showed that a packing can be obtained in $O(n \log\log n)$ time. However, since it takes more than linear time, further improvement is required for the use in practice.

When a given seq-pair of n rectangles includes k subsequences called adjacent crosses, Murata et al. proposed a method to convert such seq-pair to a rectangular dissection of the same relative positions in $O((n+k)^2)$ time. They showed the number of adjacent crosses (k) is corresponding to the number of empty rooms in the rectangular dissection since an adjacent cross corresponds to an empty room on one to one basis [7]. Recently our study [15, 16] showed a method to convert a seq-pair to an equivalent rectangular dissection in $O(n+k)$ time giving the position of adjacent crosses and the inserting order of dummy modules into such adjacent crosses.

Also, in [11], we proved that the necessary number of adjacent crosses for representing a packing of n rectangles was not more than $n-3$, and we conjectured that the tight upper bound of the number of such adjacent crosses is $\mathcal{K}(n)=n-\lfloor\sqrt{4n-1}\rfloor$ [11]. Takashima et al. insisted on the justice of this conjecture in [13].

Therefore Takashima et al. proposed a method to search an optimal solution for a packing of n rectangles using rectangular dissection with $\mathcal{K}(n)$ empty rooms [13, 14]. In this method, however, it is seldom that all of the given empty rooms are necessary in the process of searching. Therefore it is considered that there is still room for improvement to obtain an optimal solution efficiently.

In this paper, we propose a new packing representation named "selected sequence-pair" (SSP), a seq-pair where k is not more than $\mathcal{K}(n)$. In order to decode SSP fastly, we propose an algorithm to enumerate all adjacent crosses on a given seq-pair in $O(n+k)$ time and to obtain inserting order of dummy modules into such adjacent crosses. We also apply the methods [15, 16] proposed by us which obtain a packing from a seq-pair without adjacent crosses via a Q-sequence [10], representation of a rectangular dissection, in $O(n+k)$ time. By combining the algorithm proposed here with the conventional methods, we can obtain the bottom left corner packing based on SSP in $O(n)$ time.

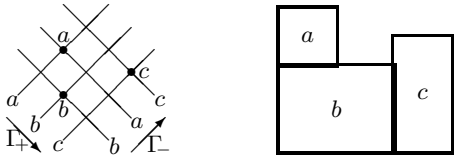
The outstanding merits of SSP are as follows:
(1) The bottom left corner packing based on a given SSP can be obtained in $O(n)$ time. (Note that obtaining a packing of n rectangles in the smaller time complexity than $O(n)$ is theoretically impossible.)
(2) An arbitrary packing can be represented by SSP.
(3) The total representation number of SSP of size n is not more than that of rectangular dissection of the same size with $\mathcal{K}(n)$ empty rooms (the necessary number of empty rooms for representing an arbitrary packing).

Also, a move operation for obtaining adjacent solution efficiently is proposed to perturb SSP for Simulated Annealing. From experimental results, we confirmed the proposed method was carried out in linear time and was more efficient than the conventional method when SSP size got bigger.

II. Previous Works

II.1. Sequence-Pair

A **sequence-pair** (seq-pair) is an ordered pair of Γ_+ and Γ_-, where each of Γ_+ and Γ_- is a permutation of names of given n



(a) Oblique grid (b) Packing

Fig. 1.: Example of oblique-grid and packing of seq-pair $(abc; bac)$

modules. For example, $(\Gamma_+;\Gamma_-) = (a\,b\,c\,d\,;\, b\,d\,a\,c)$ is a seq-pair of module set $\{a, b, c, d\}$. If module x is the i'th in Γ_+, we denote $\Gamma_+(i) = x$, as well as $\Gamma_+^{-1}(x) = i$. Similar notation is also used for Γ_-. To help intuitive understanding, we use a notation such as

$$(\Gamma_+;\Gamma_-) = (\cdot\cdot\, a \cdot\cdot b \cdot\cdot; \cdot\cdot\, a \cdot\cdot b \cdot\cdot)$$

by which we mean

$$\Gamma_+^{-1}(a) < \Gamma_+^{1}(b) \quad \text{and} \quad \Gamma_-^{-1}(a) < \Gamma_-^{-1}(b).$$

For every module pair $\{a, b\}$, a is in the left of b (equivalently, b is in the right of a) if

$$(\Gamma_+;\Gamma_-) = (\cdot\cdot\, a \cdot\cdot b \cdot\cdot; \cdot\cdot\, a \cdot\cdot b \cdot\cdot).$$

Similarly, a is below b (equivalently, b is above a) if

$$(\Gamma_+;\Gamma_-) = (\cdot\cdot\, b \cdot\cdot a \cdot\cdot; \cdot\cdot\, a \cdot\cdot b \cdot\cdot).$$

An **HV-relation-set (HVRS)** for a set of modules is a set of horizontal (in the left / right of) or vertical (above / below) relations for all module pairs. For example, seq-pair $(a\,b\,c\,;\, c\,a\,b)$ corresponds to HVRS $\{a$ is in the left of b, c is below a, c is below $b\}$. A seq-pair always corresponds to a realizable HVRS, and there is a seq-pair whose HVRS can lead to an minimum area placement.

The H/V constraints of a seq-pair can be intuitively understood using the **oblique-grid** notation. It is an $n \times n$ grid obliquely drawn on the plane so that the Γ_+ is observed along the sequence of the positive slopes from left to right and the Γ_- is observed similarly along the negative slopes. For example, Fig.1(a) shows the oblique grid of seq-pair $(a\,b\,c\,;\, b\,a\,c)$. It shows the H/V constraints: block c is in the right quarter view range (between $-45°$ and $+45°$) of block a on the oblique grid, then c should be placed in the right of a.

From one seq-pair, a bottom left corner packing can be obtained in $O(n \log\log n)$ time [5].

II.2. Rectangular Dissection

A **rectangular dissection** is a dissection of a rectangle into a set of rectangles called **rooms** with exclusive assignments of modules to rooms (no two modules share a single room.) Examples are shown in Fig.2. Only T-intersections are used to form the dissection except for the four corners of the outermost bounding rectangle [18]. (Two T-intersections on a point may look like a cross.) Each of the line segments and edges of the bounding rectangle is called a **seg**. A room is **occupied** if a module is assigned to the room, otherwise **empty**.

In this paper, as in the conventional floorplan or the topological placement, we only focus on the topology between rooms and segs in a rectangular dissection. Consequently, if the number of rooms are given, the variety of rectangular dissection becomes finite [19].

HVRS of rectangular dissection is defined as follows. If a vertical (horizontal) seg s adjoins the right of (below) the room a and the left of (above) room b, we say "room a is left of (above) room b via seg s."

II.3. Adjacent Cross

In seq-pair $\mathcal{S}$, when rectangles a, b, c, d are put in the order of

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

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

we say "$\mathcal{S}$ has an **adjacent cross**" or "a, b, c, d make an adjacent cross" [7]. In $\mathcal{S}$, b and c are adjacent in Γ_+. Also, a and d are adjacent in Γ_-. We denote the adjacent cross as bc/ad and call "ad" ("bc") a Γ_- (Γ_+) **adjacent pair**. A rectangular dissection whose HVRS is the same as that of a seq-pair of n rectangles with adjacent crosses cannot be realized with n rooms. We need to introduce as many empty rooms as adjacent crosses.

For example, the rectangular dissection satisfying HVRS of a seq-pair (1 2 3 4 ; 2 4 1 3) is shown in Fig2.(a). It confirms the necessity to introduce an empty room in the middle of four rooms.

II.3.1 Inserting Operation of a Dummy Module into an Adjacent Cross

To convert a seq-pair with adjacent crosses to a rectangular dissection of the same HVRS, the first thing to do is inserting dummy modules to a seq-pair to remove adjacent crosses [7].

Theorem 1 When a seq-pair has n rectangles and k adjacent crosses, the minimum number of rooms in the rectangular dissection of the same HVRS as the seq-pair is $n+k$. ■

The insertion of dummy modules into seq-pair with adjacent crosses can be realized as follows. Assume rectangles a, b, c, d in seq-pair $\mathcal{S}$ make an adjacent cross ab/cd. Insert dummy module x between a and b on Γ_+ and between c and d on Γ_-. Then, a, b, c, d don't make an adjacent cross.

II.4. Q-sequence

Q-sequence (Q-seq) proposed by Sakanushi et al. in [10] is a method to represent a rectangular dissection of general structure with n rooms by letters of total length of $3n$. Each room

(a) Rectangular dissection corresponding to seq-pair (1234;2413)

(b) Rectangular dissection represented by Q-seq $\mathcal{R}_2\mathcal{R}_1\mathcal{B}_5\mathcal{B}_1 1\mathcal{B}_3\mathcal{B}_2 2$ $\mathcal{R}_6\mathcal{R}_4\mathcal{R}_3 3\mathcal{B}_4 4\mathcal{R}_5 5\mathcal{B}_6 6$

Fig. 2.: Examples of rectangular dissection

x in a Q-seq has necessarily its corresponding $\mathcal{R}_x$ and $\mathcal{B}_x$. For example, rectangular dissection of Fig. 2(b) is represented by a Q-seq

"$\mathcal{Q} = \mathcal{R}_2\mathcal{R}_1\mathcal{B}_5\mathcal{B}_1 1\mathcal{B}_3\mathcal{B}_2 2\mathcal{R}_6\mathcal{R}_4\mathcal{R}_3 3\mathcal{B}_4 4\mathcal{R}_5 5\mathcal{B}_6 6$."

Q-seq is made from **Q-state** defined as follows [12].

Definition 1 (Q-state) In a rectangular dissection, when the lower (right) end of a vertical (horizontal) seg s touches a horizontal (vertical) seg at the bottom right corner of room x in the shape of '$\perp$' ('$\dashv$'), Q-state of room x has a room name followed by $\mathcal{R}_y$ ($\mathcal{B}_y$) corresponding to room y which is adjacent to the right (bottom) of seg s in the order from the bottom (right). ■

For example, a rectangular dissection of Fig.2(b) has a vertical and a horizontal seg touching in the shape of '$\perp$' at the bottom right corner of room 2 and, adjacent to the right of this vertical line, room 6, 4 and 3 stand in a column from the bottom. The Q-state of room 2 is "$2\mathcal{R}_6\mathcal{R}_4\mathcal{R}_3$."

In Q-seq, "$\mathcal{R}_i$" ("$\mathcal{B}_i$") before the leftmost room name shows that room i is adjacent to the left seg (upper seg) of the bounding rectangle. For example, the top "$\mathcal{R}_2\mathcal{R}_1$" in $\mathcal{Q}$ means room 2 and room 1 are adjacent to the left seg of the bounding rectangle. Similarly "$\mathcal{B}_5\mathcal{B}_1$" means room 5 and room 1 are adjacent to the upper seg of the bounding rectangle.

II.5. Algorithm for Converting Seq-pair without Adjacent Crosses to Q-seq

For easy understanding, we rename each rectangle as $\Gamma_+ = (1, 2, 3 \cdots n)$ ($\Gamma_- = (1, 2, 3 \cdots n)$) without loss of generality. In the following, we call this renamed seq-pair a Γ_- (Γ_+) **normalized seq-pair**. For example, a Γ_- (Γ_+) normalized seq-pair of $(a\,b\,c\,d\,e\,f; b\,f\,d\,c\,a\,e)$ is $(1\,2\,3\,4\,5\,6; 2\,6\,4\,3\,1\,5)$ $((5\,1\,4\,3\,6\,2; 1\,2\,3\,4\,5\,6))$.

In [15, 16], We proposed an algorithm for converting $\mathcal{S}'$, a Γ_- normalized seq-pair without adjacent crosses, to Q-seq. This algorithm and decoding of Q-seq make it possible to convert $\mathcal{S}'$ to a rectangular dissection in $O(m)$ time when the size of $\mathcal{S}'$ is m (conventionally it took $O(m^2)$ time [7]) and a packing can be obtained in $O(m)$ time. In this algorithm, we obtain the following four links (rs, rb, ls, lb) on $\Gamma_-(i)$ which is the ith element from the left in the given seq-pair.

Definition 2 Among the rectangles whose numbers are smaller than $\Gamma_-(i)$, $rs(i)$ denotes the closest position to $\Gamma_-(i)$ on the right of $\Gamma_-(i)$ and $ls(i)$ denotes the closest position to $\Gamma_-(i)$ on the left of $\Gamma_-(i)$.
Among the rectangles whose numbers are bigger than $\Gamma_-(i)$, $rb(i)$ denotes the closest position to $\Gamma_-(i)$ on the right of $\Gamma_-(i)$ and $lb(i)$ denotes the closest position to $\Gamma_-(i)$ on the left of $\Gamma_-(i)$. ■

$rs(i), rb(i), ls(i), lb(i)$ can be obtained for all rectangles in $O(m^2)$ time easily when the number of rectangles is m. But, Obtaining them in $O(m)$ time is possible by the following sophisticated algorithm.

Algorithm Link rs

Input: Γ_- of $\mathcal{S}'$ with m rectangles
Output: rs link of Γ_- of $\mathcal{S}'$

```
Initialize the stack;
push 0 to the stack;
for (i = 1, 2, ··· , m){
  while (Γ_-(i) < Topmost element of the stack)
    Pop the topmost element from the stack, and
    set up an rs link from the topmost element toward Γ_-(i)
    (that is rs(Γ_-^{-1}(pop())) = i) ;
  push Γ_-(i) to the stack;
}
```

(**Algorithm** Link rs **End**)

We consider the time complexity of Link rs. A double loop exists in Link rs and the inner loop always pops an element each time from the stack. Besides, at the bottom of the stack, zero exists which is never popped.

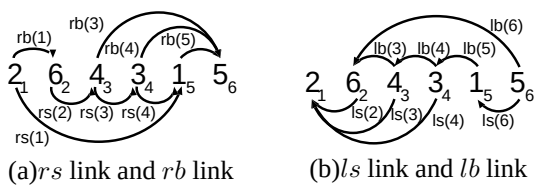
Obviously the number of times for the elements to be pushed into the stack is exactly $m+1$, so this algorithm is carried out in $O(m)$ time. We can obtain rb, ls and lb links similarly.

$\mathcal{S}'$ can be converted to Q-seq by the following algorithm keeping HVRS of $\mathcal{S}'$.

Algorithm SeqPair–Qseq

```
Input: m rectangles, Γ_- of S'
Output: Q-sequence
/* Make Link */
  Make rs,rb,ls and lb link of Γ_-;
/* rs chain */
  Output R with subindex of the traced element name
  by tracing rs link from the leftest element of Γ_-
  until no destination of rs link remains;
/* ls chain */
  Output B with subindex of the traced element name
  by tracing ls link from the rightest element of Γ_-
  until no destination of ls link remains;
/* Check adjacent relation between each room */
  for (room = 1, 2, ··· , m − 1){
    output room;  /*output room name*/
    if (Γ_-^{-1}(room) < Γ_-^{-1}(room+1)){
    /* rb-rs chain */
      Output R with subindex of the traced element name
      by tracing rb link once and rs link
      from Γ_-^{-1}(room) to Γ_-^{-1}(room+1) }
    else{
    /* lb-ls chain */
      Output B with subindex of the traced element name
      by tracing lb link once and ls link
      from Γ_-^{-1}(room) to Γ_-^{-1}(room+1) }
  }
  output m; /*output last room name*/
```

(**Algorithm** SeqPair–Qseq **End**)



(a)rs link and rb link (b)ls link and lb link

Fig. 3.: Four links of $\mathcal{S}$

III. SELECTED SEQUENCE-PAIR

When a seq-pair of n rectangles has adjacent crosses of $\mathcal{K}(n) = n - \lfloor\sqrt{4n-1}\rfloor$ or less, we call it a **selected sequence-pair** (SSP). Since an empty room in rectangular dissection corresponds to an adjacent cross in seq-pair on one to one basis [7] and it was proved that an arbitrary packing was represented by a rectangular dissection with $\mathcal{K}(n)$ empty rooms or less [13], the following theorem is obtained.

Theorem 2 An arbitrary packing of n rectangles can be derived from a seq-pair (SSP) using not more than $\mathcal{K}(n)$ adjacent crosses. ■

III.1. Algorithm to obtain packing from SSP

From an SSP which has k adjacent crosses and represents a placement of n rectangles, we can obtain a packing in $O(n)$ time by the following procedure.

Procedure SSP Decoder

Step 1: Enumerate all adjacent crosses on a given SSP $\mathcal{S}$ by algorithm Adjcross List ($O(n+k)$ time).

Step 2: Based on the obtained position of adjacent crosses and the order to insert dummy rectangles by Step 1, insert k dummy rectangles to remove all adjacent crosses ($O(n+k)$ time) [7]. Then the size of SSP $\mathcal{S}'$ without adjacent crosses is $m=n+k$.

Step 3: Convert $\mathcal{S}'$ to Q-seq $\mathcal{Q}$ by algorithm SeqPair–Qseq ($O(m)$ time) [15, 16].

Step 4: From $\mathcal{Q}$, restore a rectangular dissection [9] and assign rectangles to obtain the packing ($O(m)$time).

(**Procedure** SSP Decoder **End**)

Step 2, **Step** 3 and **Step** 4 are conventional methods and later we will explain **Step** 1 and **Step** 2. **Step** 3 is mentioned in section II.5. As for **Step** 4, refer to [9].

III.2. Step 1: Enumeration of All Adjacent Crosses

III.2.1 A Method to Enumerate All Adjacent Crosses

When a $\Gamma_{\perp}$ normalized seq-pair of n rectangles has k adjacent crosses, it is possible to enumerate all adjacent crosses in $O(n+k)$ time by algorithm Adjcross List .

Algorithm Adjcross List

Prepare an empty double linked list where nodes must be sorted in an ascending order of the number from the head of the list and do the following steps for each rectangle i on Γ_{-} of a $\Gamma_{\perp}$ normalized seq-pair from the left. After completing the search from the left on Γ_{-}, do the same procedure from the right.

Focus on rectangle i and

Step 1: Move the pointer to the position where i is to be inserted on a double linked list. When the pointer is in a position and we call a rectangle to be inserted there m, if the pointer moves toward smaller rectangle name than m and jumps over another rectangle which we call p on the list, output p, m, i, $p+1$ which make an adjacent cross $p\,p+1/m\,i$.

Step 2: If $i+1$ has not been focused yet but will be focused later, insert i on the double linked list. If $i-1$ is found on the double linked list, delete it.

(**Algorithm** Adjcross List **End**)

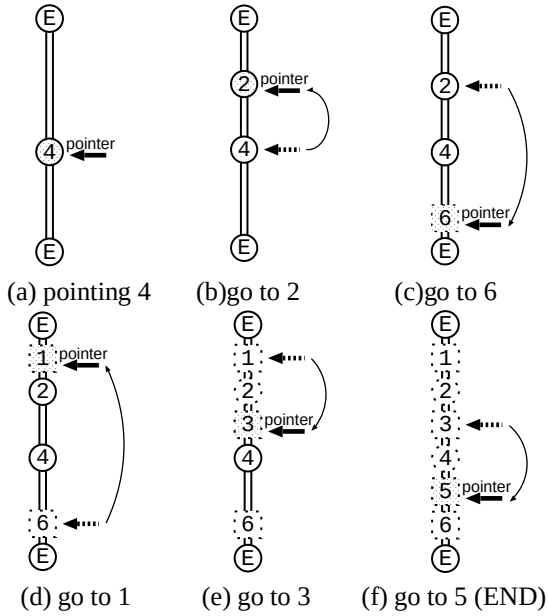


Fig. 4.: Example of Adjcross List realization on seq-pair (1 2 3 4 5 6;4 2 6 1 3 5). **(a)** Insert "4" on $\Gamma_{\perp}$ into the list as the search of "5" is unfocused. **(b)** Insert "2" as the search of "3" is unfocused. **(c)** Don't insert "6" as "7" doesn't exist. **(d)** To focus on "1," "4" and "2" were jumped over upward, so "4 5/6 1" and "2 3/6 1" make adjacent crosses. As the search of "2" was done, don't insert "1." **(e)** Don't insert "3" as the search of "4" was done. Also delete "2." **(f)** Finally don't insert "5" as the search of "6" was done. Also delete "4."

Adjcross List is obtained by altering SeqPair–Qseq mentioned in Sec.II.5 like sweeping a plane in computational geometry to enumerate efficiently all adjacent crosses in parallel.

If a $\Gamma_{\perp}$ normalized seq-pair $(1\,2\,3\,4\,5\,6;4\,2\,6\,1\,3\,5)$ is inputted, two adjacent crosses "$4\,5/6\,1$" and "$2\,3/6\,1$" are found (as shown in Fig.5(a)) by the operation shown in Fig.4. Another adjacent cross "$3\,4/2\,6$" is found by repeating the same operation from the right to the left on $\Gamma_{\perp}$ of inputted seq-pair.

The following theorem comprises on Adjcross List.

Theorem 3 For an arbitrary seq-pair of size n with k adjacent crosses, algorithm Adjcross List can enumerate all adjacent crosses in $O(n+k)$ time.

(Proof) Algorithm Adjcross List sweeps $\Gamma_{\perp}$ starting from the left and then from the right and can find each of adjacent crosses, if they exist, while sweeping the $\Gamma_{\perp}$ adjacent pair of the adjacent cross.

When our focus moves from one element to the next one on Γ_{-}, the number of times that the pointer moves on the double linked list exceeds by one the number of elements passed by the pointer. When the pointer moves toward a smaller element name, the total number of times that the pointer passes one element is k in the whole Adjcross List since an adjacent cross is found every time the pointer passes each element. The difference between moving toward larger element names and toward smaller element names is one. Therefore the time complexity is $O(n+k)$. ■

III.2.2 Inserting Order of Dummy Module to Remove All Adjacent Crosses

In case a plural number of adjacent crosses exist, in order to remove them, we have to consider the order of dummy module insertion. In the following, this order is explained using the order of finding adjacent crosses by Adjcross List.

Assume that adjacent cross ab/cd is found before ab/ef in the search of a Γ_- normalized seq-pair from left to right on Γ_- by Adjcross List.

In this case, between adjacent elements a and b on Γ_+, a dummy module p to remove ab/cd must be inserted left to a dummy module q to remove ab/ef. Accordingly, the order of insertion becomes "a, p, q, b" on Γ_+.

The order of insertion is similar when the search was carried out on Γ_- of a Γ_- normalized seq-pair from right to left by algorithm Adjcross List.

Next, assume that adjacent cross cd/st is found before ef/st in the search on Γ_- of a Γ_- normalized seq-pair from left to right by algorithm Adjcross List.

In this case, between adjacent elements s and t on Γ_-, a dummy module u to remove cd/st must be inserted left to a dummy module v to remove ef/st. Accordingly, the order of insertion becomes "s, u, v, t" on Γ_-.

Also assume that adjacent cross cd/st is found before ef/st in the search on Γ_- of a Γ_- normalized seq-pair from right to left by algorithm Adjcross List.

In this case, between adjacent elements s and t on Γ_-, a dummy module u to remove cd/st must be inserted right to a dummy module v to remove ef/st. Accordingly, the order of insertion becomes "s, v, u, t" on Γ_-.

In algorithm Adjcross List, adjacent crosses found by searching Γ_- from left to right and those found by searching from right to left never hold adjacent elements (a, b and s, t mentioned above) in common. For this reason, the inserting operation can be carried out independently.

III.3. Step 2: Removal of all adjacent crosses by dummy rectangle insertion

Based on the obtained position of adjacent crosses and the order to insert dummy rectangles by Step 1, we must insert k dummy rectangles to remove all adjacent crosses. It can be carried out in $O(n+k)$ time by converting seq-pair representation from array to list. Then the size of SSP $\mathcal{S}'$ which has no adjacent crosses is $m=n+k$. For example, by inserting operation, a seq-pair $(1\,2\,x\,3\,z\,4\,y\,5\,6; 4\,2\,z\,6\,y\,x\,1\,3\,5)$ is obtained from a seq-pair $(1\,2\,3\,4\,5\,6; 4\,2\,6\,1\,3\,5)$ (Fig.5(b)).

IV. Solution Space of SSP for Simulated Annealing Search

Obtaining a packing by combining the search methods such as Simulated Annealing and SSP, it is necessary to define a **move** operation and to construct solution space. It is desirable for the move operation to guarantee reachability from an arbitrary SSP to another arbitrary SSP and to obtain random adjacent feasible solution in linear time of the number of rectangles. Therefore, in this paper we use the same operation as [8]: For a seq-pair $(\Gamma_+; \Gamma_-)$, one module (**move-module**) is moved in either Γ_+ or Γ_-. This reachability was proved in [8] by expanding the proof

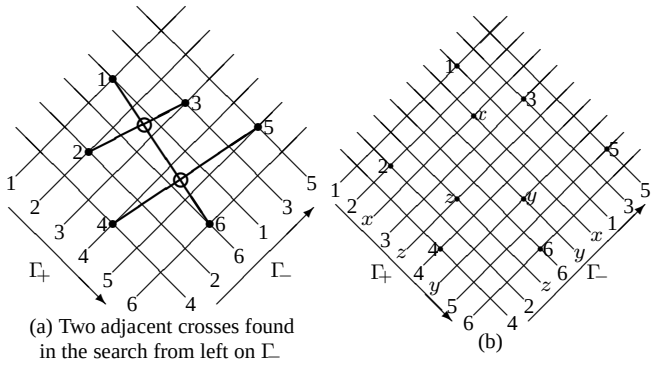


Fig. 5.: The insertion order of dummy modules to remove adjacent crosses

that the solution space constructed only by a seq-pair without adjacent crosses is reachable. The detailed proof is omitted here.

When adjacent solutions contain infeasible solutions, techniques to search the solution space by Simulated Annealing may be specified as follows:

(1) Obtain an adjacent solution randomly until it becomes feasible.

(2) Impose very large cost to infeasible solutions.

A defect of (1) is that it needs a lot of time to obtain one feasible adjacent solution if the proportion of feasible solutions in adjacent solutions is small. When (2) is used, there is no guarantee that Simulated Annealing finds a feasible solution.

An alternative method was proposed in [8]:

(3) Enumerate all feasible solutions, and then select a feasible solution randomly.

We adopt the way to enumerate feasible adjacent solutions and pick up one of them at random. The procedure in detail is shown below.

Step 1: In permutation Γ of either Γ_+ or Γ_-, determine one move destination at random.

Step 2: In Γ, regard all elements except those at both sides of move destination as the candidate of a move element. Then measure the fluctuation of the number of adjacent crosses which occurs when every move element is inserted into the move destination.

Step 3: In checking the fluctuation at **Step 2**, if there is no move element resulting in SSP, go back to **Step 1**.

Step 4: Among move elements resulting in SSP, choose one element at random and insert it into the move destination.

In this procedure, there may be a case to go from **Step 3** back to **Step 1** and a frequent occurrence of this makes efficient realization in linear time impossible. However, it is confirmed by computational experiment that this hardly occurs.

The fluctuation of the number of adjacent crosses for all move element candidates in step 2 can be obtained in $O(n)$ time by checking the following three cases. The following description is for the move operation on Γ_- and the move operation for Γ_+ is shown in parentheses.

(1) Adjacent crosses which disappear after the move operation.

(2) Adjacent crosses generated in the form of Γ_- (Γ_+) adjacent pair one of which is a move element.

(3) Adjacent crosses generated in the form of Γ_+ (Γ_-) adjacent pair one of which is a move element.

(1) is a case where one of the elements consisting of an adjacent cross is a move element and by moving it, the adjacent cross disappears. When one of the Γ_- (Γ_+) adjacent pair in an adjacent cross is a move element and a substitute for the move element exists, we don't count such an adjacent cross as "decrease."

(2) is carried out by the following algorithm where the input is a Γ_- (Γ_+) normalized SSP. The move destination is between ℓ and r, and p shows a move element. Then the increase of adjacent crosses is calculated.

```
for (c=0, p=max(ℓ,r); p≤n−2; p++){
   if(the move destination is between p and p+1.)
      ++c;
   record the value of c as the increase of move element p+2
}
for (c=0, p=min(ℓ,r); p>2; p−−){
   if(the move destination is between p and p−1.)
      ++c;
   record the value of c as the increase of move element p−2
}
```

Since **(3)** is too complicated, we don't explain here.

As for the calculation of the fluctuation of the number of adjacent crosses in three cases, refer to [17] which has detailed explanation with some examples.

V. Experiments and Results

V.1. The Number of SSP Representations

We compared the number of SSP representations with that of Q-seq representations. In SSP, by the exhaustive search of a Γ_- normalized seq-pair of n rectangles, we obtain the number of a Γ_- normalized SSP and get the multiplied value of it by the combinational number of n rectangle names ($n!$).

For example, if $n=5$, the number of Γ_- normalized seq-pair is $5!=120$. Since $\mathcal{K}(5)=5-\lfloor\sqrt{4\cdot5-1}\rfloor=1$, the number of adjacent crosses in SSP of size 5 is at most one. Among 120 Γ_- normalized seq-pairs, four have adjacent crosses of more than one (these are not SSP). Therefore the number of representation of SSP of size 5 is $(5!-4)\times5!=13920$.

As for Q-seq, we obtain the combinational number for the total of n rooms and given $\mathcal{K}(n)$ empty rooms from the recurrence formula [9] and get the multiplied value of it by ${}_{n+\mathcal{K}(n)}\mathrm{P}_n$ which is the variety of labeling n room names.

If $n=5$, the combinational number of Q-seq of size 6 including $\mathcal{K}(5)=1$ empty room is 422. This is multiplied by ${}_6\mathrm{P}_5=6!/(6-5)!=720$ which is the variety of labeling five rooms among six, and the resultant 303840 is the number of representation of Q-seq with the number of modules 5.

The results are shown in Fig.6. It shows that the number of SSP representation is smaller and more suitable for the optimal solution search.

V.2. Experimental Comparison between the Proposed Method and the Conventional method

As mentioned above, it was theoretically proved that the bottom left corner packing based on an SSP (size n) can be obtained in $O(n)$ time by the proposed method. Since this method is carried out via SSP of size $n+k$ (k is the number of adjacent crosses), the lower bound of time complexity is also $O(n)$. Here, we confirm by experiment that the time necessary to carry out the proposed method is linear to n. The program was implemented in C language and the experiment was made on Pentium IV 1.6GHz.

We prepare an SSP which has adjacent crosses of $\mathcal{K}(n)-n/100$ or more as an initial SSP and, by increasing its size from 2^2 to 2^{15}, generate SSP of the number of 10^5 to 10^7 by the move operation mentioned in Section IV. Then we measure the time taken for decoding and obtain an average time for each SSP size.

To compare the difference of the time complexity, the results obtained by the conventional original decoding method of seq-pair [1] and by Fast-SP method [5] in addition to the experimental results obtained by the proposed method are shown in Fig.7. The time complexity of the original decoding method is $O(n^2)$ and that of Fast-SP method is $O(n\log\log n)$. Since the results of Fast-SP method were obtained on Sun Ultra Enterprise 3000 (200MHz) [5], we cannot compare simply with the results of the proposed method. However, we can read the difference of performance from the slope of the line in the graph and see that the proposed method, the original method and Fast-SP method each is in proportion to $n^{0.998}$, $n^{1.69}$ and $n^{1.13}$ which are calculated using the least square method. From the results, we can confirm the proposed method is carried out in linear time. The proposed method needs more time than the conventional methods for decoding when n is small but needs less time when n gets bigger.

We also implemented a packing method using the SSP based on Simulated Annealing in C language. This experiment was made on Pentium IV 1.6GHz without rotating rectangles using ami33 and it took 106 sec. The experimental result is shown in Fig.8.

VI. Conclusions

We proposed a "selected sequence pair" (SSP), a sequence-pair where the number of sub-sequence called "adjacent cross" is $\mathcal{K}(n)=n-\lfloor\sqrt{4n-1}\rfloor$ or less.

The outstanding merits of SSP are shown below. **(1)** The bottom left corner packing based on an SSP can be obtained

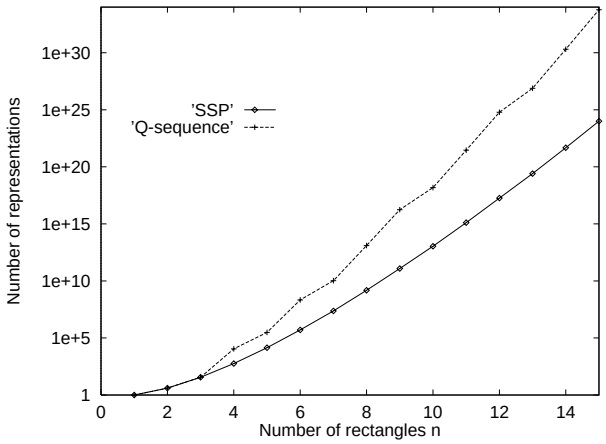


Fig. 6.: Comparison of the number of representations

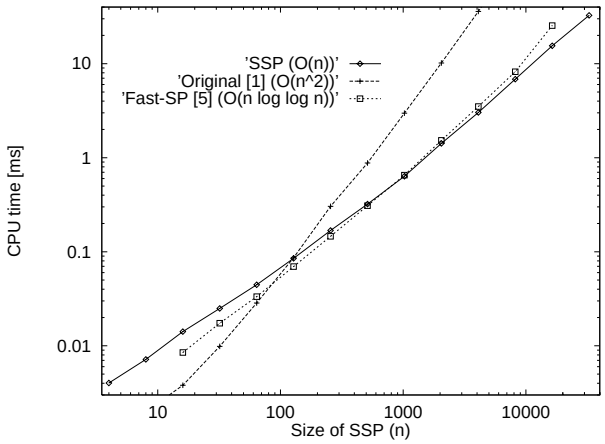


Fig. 7.: Comparison of experimental results

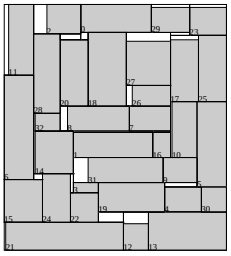


Fig. 8.: The resultant packing of ami33 (the resultant area: 1.235633mm^2, the total area of ami33 modules: 1.156449mm^2)

in $O(n)$ time. **(2)** An arbitrary packing can be represented by SSP . **(3)** The total representation number of SSP of size n is not more than that of rectangular dissection of the same size with $\mathcal{K}(n)$ empty rooms

Also on SSP, in order to use a search method like Simulated Annealing, we have proposed a move operation to obtain an adjacent solution efficiently.

To realize these merits, we proposed the algorithm which can enumerate all adjacent crosses on a sequence-pair in linear time of the sum of the number of rectangles and the number of adjacent crosses. Also we applied a theorem that an arbitrary packing of n rectangular modules can be represented by a rectangular dissection with $\mathcal{K}(n)$ empty rooms or less and a conventional method to convert a seq-pair without adjacent crosses to an equivalent Q-sequence, representation of rectangular dissection in $O(n+k)$ time. A move operation to obtain an adjacent solution efficiently was proposed to perturb SSP for Simulated Annealing.

From experimental results, we confirmed the proposed method was carried out in linear time and was more efficient than the conventional method when SSP size got bigger.

ACKNOWLEDGMENTS

The authors are grateful to Associate Professor Atsushi Takahashi of Tokyo Institute of Technology, Research Associate Keishi Sakanushi of Osaka University and Dr. Kengo R. Azegami of Fujitsu Labs. for their advice. This study is a part of the CAD21 project.

REFERENCES

[1] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani: "Rectangle-packing-based module placement," in IEEE ICCAD, pp.472–479, 1995.

[2] H. Murata, K. Fujiyoshi, and M. Kaneko: "VLSI/PCB placement with obstacles based on sequence pair," IEEE Trans. CAD, vol.17, no.1, pp.60–68, 1998.

[3] K. Fujiyoshi, and H. Murata: "Arbitrary convex and concave rectilinear block packing using sequence-pair," IEEE Trans. CAD, vol.19, no.2, pp.224–233, 2000.

[4] T. Takahashi: "An algorithm for finding a maximum weight decreasing sequence in a permutation, motivated by rectangle packing problem," Technical Report IEICE, VLD96-30, pp.31–35, 1996 (in Japanese).

[5] X. Tang, R. Tian and D.F. Wong: "Fast evaluation of sequence pair in block placement by longest common subsequence computation," IEEE Trans. CAD, vol.20, no.12, pp.1406–1413, 2001.

[6] J. Lai, M.-S. Lin, T.-C. Wang, and Li-C. Wang: "Module placement with boundary constraints using the sequence-pair representation," in IEEE ASP-DAC, pp.515–520, 2001.

[7] H. Murata, K. Fujiyoshi, T. Watanabe, and Y. Kajitani: "A mapping from sequence-pair to rectangular dissection," in IEEE ASP-DAC, pp.625–633, 1997.

[8] K. Kiyota, K. Fujiyoshi: "Simulated annealing search though general structure floorplans using sequence-pair," in IEEE ISCAS, pp.III-77–III-80, 2000.

[9] K. Sakanushi, and Y. Kajitani, "Counting of the topological dissections by reduct-seq representation," Technical Report IEICE, COMP2000-17, pp.25–32, 2000.

[10] K. Sakanushi and Y. Kajitani, "The quarter-state sequence (Q-sequence) to represent the floorplan and applications to layout optimization," in IEEE APCCAS, pp.829–832, 2000.

[11] C. Kodama and K. Fujiyoshi: "Selected sequence-pair," Technical Report IEICE, VLD2001-17, pp.65–72, May, 2001 (in Japanese).

[12] M. Tsuboi, C. Kodama, K. Sakanushi, K. Fujiyoshi and A. Takahashi "Linear time decodable rectangular dissection to represent arbitrary packing using Q-sequence," in SASIMI, pp.272–278, Oct., 2001.

[13] Y. Takashima and H. Murata: "The tight upper bound of the empty rooms in floorplan," in SASIMI, pp.264–271, Oct., 2001.

[14] C. Zhuang, K. Sakanushi, L. Jin and Y. Kajitani: "An enhanced Q-sequence augmented with essential empty room insertions and parenthesis trees," in DATE, pp.61–68, Mar., 2002.

[15] C. Kodama and K. Fujiyoshi: "An efficient decoding method of sequence-pair," in IEEE APCCAS, vol.2, pp.131–136, Oct., 2002.

[16] C. Kodama and K. Fujiyoshi: "An efficient decoding method of sequence-pair with reduced redundancy," IEICE Trans. Fundamentals, vol.E85-A, no.12, pp.2785–2794, Dec., 2002.

[17] K. Fujiyoshi, C. Kodama and K. Kiyota: "An effective MOVE operation for selected sequence-pair," Technical Report IEICE, VLD2002-6, pp.31–36, May, 2002 (in Japanese).

[18] L. Stockmeyer: "Optimal orientations of cells in slicing floorplan designs," Info.and Control, vol.57, pp.91–101, 1983.

[19] T. Ohtsuki, N. Suzigama and H. Hawanishi: "An optimization technique for integrated circuit layout design," Proc. of ICCST, pp.67–68, 1970.