Available online at www.sciencedirect.com
ScienceDirect
INTEGRATION, the VLSI journal 40 (2007) 274–284
INTEGRATION the VLSI journal
www.elsevier.com/locate/vlsi

A fast algorithm for rectilinear block packing based on selected sequence-pair

Kunihiro Fujiyoshi, Chikaaki Kodama*, Akira Ikeda

Department of Electrical and Electronic Engineering, Tokyo University of Agriculture and Technology, Tokyo 184-8588, Japan

Received 3 December 2004; received in revised form 10 January 2006; accepted 11 January 2006

Abstract

In this paper, we present a method of rectilinear block packing using selected sequence-pair (SSP), a rectangle packing representation. We also propose a fast algorithm to obtain a rectilinear block packing in $O((p+1)n)$ time keeping all the constraints imposed by a given SSP. Here, p is the number of rectilinear blocks excluding rectangles and n is that of rectangle sub-blocks obtained by partitioning each rectilinear block. So far, the fastest method based on a sequence-pair required $O(n^2+\ell^3)$ time, where ℓ is the number of rectilinear blocks. If p is constant, the proposed algorithm requires $O(n)$ time, which is equal to the trivial lower bound of the time complexity for decoding. The effectiveness of the proposed method was confirmed by the experimental comparisons, especially when p is constant.

Keywords: Selected sequence-pair; Sequence-pair; Rectilinear block; Placement; Packing; Floorplan

1. Introduction

The first stage of VLSI physical design is the floorplanning, that is, a rough placement of major circuit components such as modules and wires. To accomplish a design of high-performance VLSI, several strict constraints (such as timing, delay, area, power and others) should be considered, and macro-library, namely, macro-cells (blocks) such as IP should be effectively utilized. In such a case, some modules are not rectangles but rectilinear blocks such as modules with alignment constraints, reused macro-cells, a thermal countermeasure module placement (adjoining heat up and non-heat up modules) and so on. Therefore, a method of packing blocks including rectilinear shape is desired.

Several ideas for a convex rectilinear block packing [1–3] and L-shaped and T-shaped block packing [4–6] were proposed but these methods are impossible to represent arbitrary rectilinear block packing. Several methods of representing arbitrary rectilinear block packing were proposed [7–10]. However, some blocks may overlap [8] and a certain kind of packing is impossible to represent [7,9,10].

In [11], a method of representing a rectilinear block packing based on sequence-pair [12] was proposed. Each rectilinear block is partitioned into rectangle sub-blocks by horizontal and vertical lines since a sequence-pair can handle only rectangle blocks. Note that if a rectilinear block is in the shape of rectangle, it is regarded as a sub-block itself. Also in [11], an algorithm to obtain a rectilinear block packing in $O(n^3)$ time was proposed (n is the number of rectangle sub-blocks) keeping all the constraints imposed by a given sequence-pair. However, the time complexity is very big for practical use. Recently, a method to obtain a rectilinear block packing in $O(n^2+\ell^3)$ time was proposed [13] (ℓ is the number of rectilinear blocks). However, it takes a long time to carry out the algorithm according to the increase of the number of rectangles.

Another method to obtain a rectilinear block packing in $O(mn \log\log n)$ time was proposed in [14],[1] where m is not the number of rectilinear blocks but the number of sets of H/V-sequential sub-blocks. In [14], "a rectilinear block is

*Corresponding author. Tel.: +81 423887437; fax: +81 423887250.
E-mail address: kodamada@fjlab.ei.tuat.ac.jp (C. Kodama).

[1]Note that we presented the preliminary version of this paper [15] earlier than [14].

0167-9260/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2006.01.003

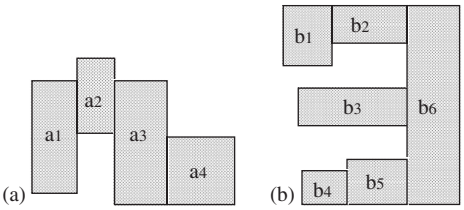


Fig. 1. Example of rectilinear blocks consisting of H-sequential sub-blocks. (a) A rectilinear block consisting of a set of H-sequential sub-blocks $\{a_1, a_2, a_3, a_4\}$ and (b) a rectilinear block consisting of three sets of H-sequential sub-blocks $\{b_1, b_2, b_6\}$, $\{b_3, b_6\}$ and $\{b_4, b_5, b_6\}$.

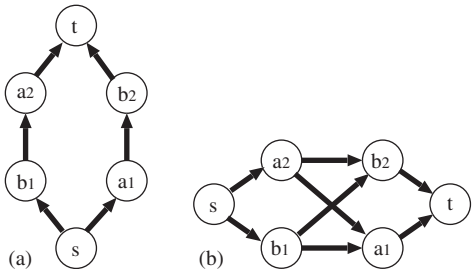


Fig. 2. Example of a feasible TCG T with no corresponding rectilinear block packing. (a) C_v of T and (b) C_h of T.

said to be H-sequential (V-sequential) if no vertical (horizontal) line cuts the block into more than two parts''. The blocks partitioned from a H-sequential (V-sequential) rectilinear block is said to be H-sequential (V-sequential) sub-blocks. An example of a rectilinear block partitioned into four H-sequential sub-blocks is shown in Fig. 1(a). When a rectilinear block is a non-sequential rectilinear shape, it can be partitioned into multiple sequences of H/V-sequential sub-blocks. Therefore, m may not be less than the number of rectilinear blocks since there are rectilinear blocks consisting of two or more sets of H/V-sequential sub-blocks. An example of such rectilinear block is shown in Fig. 1(b). It is easily understood that this block consists of three sets of H-sequential sub-blocks. Therefore, when complex rectilinear blocks are packed, it takes a long time to decode since the value of m becomes big.

Other methods to represent arbitrary shaped rectilinear block packing using O-tree [16] and TCG [17] were proposed. Since a method called ''expansion'' used in [9] is allowed there, rectilinear blocks may be deformed. Additionally in [17], there is no packing corresponding to a feasible TCG shown in Fig. 2. Also, no proofs are shown in the most of theorems and lemmas, so this is unreliable.

In this paper, we present a method of rectilinear block packing using a rectangle packing representation called ''*selected sequence-pair*'' (SSP) [18]. SSP can represent an arbitrary rectangle packing and is decodable in linear time of its size. Also, we propose an efficient algorithm to obtain a rectilinear block packing in $O((p+1)n)$ time[2] keeping all the constraints imposed by a given SSP. Time complexity of our method is close to that of [14] but p is the number of rectilinear blocks excluding rectangles.[3] So, obviously $p \leqslant m$ because m is more than the number of rectilinear blocks as mentioned before. Hence, when complex rectilinear blocks are packed, our method is faster than the method of [14]. The proposed method requires only $O(n)$ time if p is constant. Note that obtaining a rectilinear block packing of n sub-blocks in smaller time complexity than $O(n)$ is theoretically impossible. In the layout design using macro-cells, most of the modules are rectangles and the others are non-rectangular blocks. The effectiveness of the proposed method is shown in such cases and confirmed by experiments.

The organization of this paper is as follows. Section 2 introduces a sequence-pair and an SSP. Section 3 proposes a new decoding method for a rectilinear block packing. Section 4 presents experimental results. Section 5 gives conclusion.

2. Sequence-pair and selected sequence-pair

2.1. Sequence-pair

A *sequence-pair* (seq-pair) [12] is an ordered pair of Γ_+ and Γ_-, where each of Γ_+ and Γ_- is a permutation of names of given n blocks. For example, $(\Gamma_+;\Gamma_-) = (a\,b\,c\,d\,;\,b\,d\,a\,c)$ is a seq-pair of block set $\{a,b,c,d\}$. If block x is the i'th in Γ_+, we denote $\Gamma_+^{-1}(x) = i$. Similar notation is also used for Γ_-. To help intuitive understanding, we use a notation such as $(\Gamma_+;\Gamma_-) = (\cdots a \cdots b \cdots;\cdots a \cdots b \cdots)$ by which we mean $\Gamma_+^{-1}(a) < \Gamma_+^{-1}(b)$ and $\Gamma_-^{-1}(a) < \Gamma_-^{-1}(b)$.

A seq-pair imposes a ''*horizontal/vertical* (H/V) *constraint*'' for every pair of blocks as follows. For every block pair $\{a,b\}$, a is in the left of b (equivalently, b is in the right of a) if $(\Gamma_+;\Gamma_-) = (\cdots a \cdots b \cdots;\cdots a \cdots b \cdots)$. Similarly, a is below b (equivalently, b is above a) if $(\Gamma_+;\Gamma_-) = (\cdots b \cdots a \cdots;\cdots a \cdots b \cdots)$. For example, seq-pair (1 2 3 4; 2 4 1 3) has relative position like Fig. 3.

One of the optimal packings under the H/V constraint can be obtained by applying the well-known longest path algorithm for vertex weighted directed acyclic graphs.

H/V constraint graph: Based on the horizontal (left of) constraint imposed by a seq-pair, a directed and vertex-weighted graph $G_H(V,E)$ (V: vertex set, E: directed edge set) called *horizontal constraint graph* is constructed as follows [12]:

(1) V: source s, sink t and vertices labeled with module names.
(2) E: (s,x) and (x,t) for each module x, and (x,x') if and only if $x' \in \{x''|x''$ is constrained to ''right of x'' in the seq-pair$\}$.

[2]When $p = 0$, time complexity of our method is reduced to $O(n)$, the time complexity required to decode SSP.

[3]Rectangular shape blocks before partitioning, not rectangle sub-blocks after partitioning.

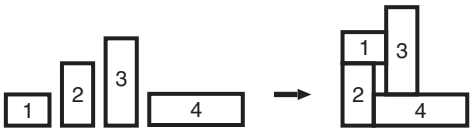


Fig. 3. A packing represented by seq-pair (1 2 3 4; 2 4 1 3).

(3) Vertex-weight: zero for s and t, width of module x for the other vertices.

Similarly, the *vertical constraint graph* is constructed using vertical (below) constraints and the height of each module.

In this paper, in order to distinguish H/V constraint graph of seq-pair from that of selected sequence-pair (mentioned later), we call the former "H/V constraint graph (SP)". Note that the number of edges in H/V constraint graph (SP) is $O(n^2)$ when the number of modules is n.

Adjacent cross [19]: In seq-pair $\mathscr{S}$, when rectangles a, b, c, d are put in the order of

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

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

we say "$\mathscr{S}$ has an *adjacent cross*" or "a, b, c, d make an adjacent cross" [19].

In the former $\mathscr{S}$, b and c are adjacent in this order in Γ_+. Also, a and d are adjacent in this order in Γ_-. So, we denote the adjacent cross as bc/ad. Also, in the latter $\mathscr{S}$, we denote it as bc/da. Seq-pair (1 2 3 4; 2 4 1 3) of Fig. 3 has an adjacent cross 2 3/4 1. The number of adjacent crosses included in a seq-pair with n elements is proved to be at most $\lceil (n-2)/2 \rceil \lfloor (n-2)/2 \rfloor$ [19]. Takashima et al. proved that the necessary number of adjacent crosses for representing an arbitrary packing of n rectangles is at most $n - \lfloor \sqrt{4n-1} \rfloor$ [21].

2.2. Selected sequence-pair (SSP)

A "*selected sequence-pair*" (SSP) [18] is defined as a seq-pair with adjacent crosses of $n - \lfloor \sqrt{4n-1} \rfloor$ or less, where n is the number of rectangles. The outstanding merits of SSP are as follows: (1) The smallest packing based on a given SSP can be obtained in $O(n)$ time, which is faster than based on seq-pair ($O(n \log \log n)$ time [22]). (2) An arbitrary packing can be represented by SSP. (3) The total number of SSP representation of size n is less than that of the same-sized seq-pair when the number of modules is more than four. An efficient MOVE operation for SSP was recently proposed in [18,23]. Therefore, we can search a good solution more efficiently than seq-pair, using Simulated Annealing with the MOVE operation.

From a given SSP with k adjacent crosses and n rectangles, we can obtain one of the optimal packing under the H/V constraint of the SSP in $O(n+k)$ time by the following procedure. Note that a packing can be obtained from SSP in $O(n)$ time as a consequence since $k \leqslant n - \lfloor \sqrt{4n-1} \rfloor$.

Procedure SSP Decoder [18].

Step 1: Enumerate all adjacent crosses on a given SSP $\mathscr{S}$.

Step 2: Based on the position of adjacent crosses and the order of inserting dummy rectangles obtained by Step 1, insert k dummy rectangles in $\mathscr{S}$ to remove all adjacent crosses. Then $\mathscr{S}'$ without adjacent crosses is obtained. The size of $\mathscr{S}'$ is $n+k$.

Step 3: Convert $\mathscr{S}'$ to a corresponding Q-sequence (rectangular dissection representation [24]).

Step 4: Based on a rectangular dissection obtained by decoding the Q-sequence, make H/V constraint graph [25] and obtain a rectangle packing.

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

The time complexity in each step is $O(n+k)$.

If an SSP $\mathscr{S} = (1\,2\,3\,4\,5\,6; 4\,2\,6\,3\,1\,5)$ is inputted in SSP Decoder, two adjacent crosses 4 5/6 3 and 3 4/2 6 are found by Step 1. In Step 2, based on the obtained position of adjacent crosses and the order of dummy rectangle insertion by Step 1, dummy rectangles D_1 and D_2 are inserted to remove all adjacent crosses. Then, an SSP without adjacent cross $\mathscr{S}' = (1\,2\,3\,D_2\,4\,D_1\,5\,6; 4\,2\,D_2\,6\,D_1\,3\,1\,5)$ is obtained.

After $\mathscr{S}'$ is converted to Q-sequence $\mathscr{Q}$ in Step 3, a rectangular dissection (mentioned later in Section 2.2.1) is obtained from $\mathscr{Q}$ and a rectangle packing is obtained from the rectangular dissection [18]. As a result, a rectangle packing based on $\mathscr{S}'$ is obtained in Step 4. Please confirm the correctness in detail (why it can be realized in $O(n+k)$ time) by referring to [18].

2.2.1. Rectangular dissection

A *rectangular dissection* is a dissection of a rectangle into a set of rectangular regions called *rooms* with exclusive assignment of modules to rooms (no two modules share a single room) [25]. An example of rectangular dissection is shown in Fig. 4(a). Each dissection line and edge of the bounding rectangle is called a *seg*.

Based on a rectangular dissection obtained by a decoding method of Q-sequence in Step 4 of SSP Decoder, we can obtain horizontal (vertical) constraint graph

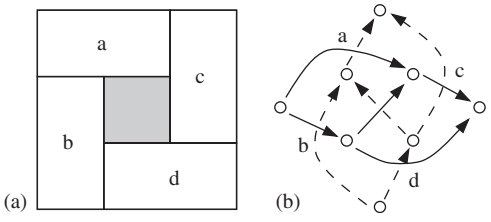


Fig. 4. Examples of a rectangular dissection and its H/V constraint graph (RD). (a) Rectangular dissection with five rooms. No module is assigned to the middle room (gray colored room) and (b) H/V constraint graph (RD). Solid lines show the horizontal constraint graph and broken lines show the vertical constraint graph.

[25] as follows. Each vertical (horizontal) seg corresponds to a vertex in the horizontal (vertical) constraint graph and each room corresponds to a weighted edge (u, v), where u is the vertex corresponding to the left (bottom) side of the room and v is the vertex corresponding to the right (top) side of the room. The weight of an edge corresponding to a room with an assigned module is the width (height) of the module, and that of an edge corresponding to a room with no module assigned is zero. An example of H/V constraint graph corresponding to Fig. 4(a) is shown in Fig. 4(b).

x (y) coordinate of the lower left corner of each room, actually of each module, is set to the longest path value from the source in the horizontal (vertical) constraint graph.

In this paper, H/V constraint graph obtained from a rectangular dissection is called "H/V constraint graph (RD)". The number of edges in H/V constraint graph (RD) is $O(n)$ when the number of rooms is n since H/V constraint graph (RD) is a planer graph [25].

2.3. Conventional method of rectilinear block packing

A polygonal block whose outside frame consists of horizontal and vertical lines only is called a "*rectilinear block*". A rectangular shape block is regarded as a kind of a rectilinear block. In [11], a seq-pair-based method of representing rectilinear block packing was proposed. Basically, only rectangle blocks can be handled by a seq-pair, so each rectilinear block is partitioned into rectangles by horizontal and/or vertical lines. These rectangle blocks are called "*sub-blocks*". If a rectilinear block is in the shape of rectangle, it is regarded as a sub-block itself. An example is shown in Fig. 6(a).

A procedure to obtain a bottom left corner packing based on H/V constraints imposed by a given seq-pair is as follows. First, H/V constraint graphs (SP) are made from a seq-pair as mentioned in Section 2.1. The weight of each vertex moves to all of its output edges. Then, edges called "*relative position edge pair*" are added to the graphs in order to pack and align simultaneously.

Let $a_1, a_2, \ldots, a_\ell$ be sub-blocks of a rectilinear block a. When a_i and a_{i+1} are adjacent parts of a common rectilinear block, a relative position edge pair consisting of directed weighted edges (a_i, a_{i+1}) and (a_{i+1}, a_i) is introduced into both H/V constraint graphs (SP). The weight of a directed edge (a_i, a_{i+1}) on horizontal constraint graph is defined as $X(a_{i+1}) - X(a_i)$, where $X(a_i)$ is the difference of x coordinate from the leftmost vertical edge of rectilinear block a to that of a_i. The weight of a directed edge (a_i, a_{i+1}) on vertical constraint graph is defined as $Y(a_{i+1}) - Y(a_i)$, where $Y(a_i)$ is the difference of y coordinate from the lowest horizontal edge of rectilinear block a to that of a_i. The weight of directed edges (a_{i+1}, a_i) is defined similarly. Clearly, the total edge weight of relative position edge pair is zero. Fig. 5(a) is an example of horizontal relative position edge pair, and Fig. 5(b) is an example of vertical relative position edge pair. We cannot use the longest path algorithm for DAG since a relative position edge pair makes a graph cyclic. Ford's shortest path algorithm is applied to compute the longest path length and obtain the coordinates of rectangles.

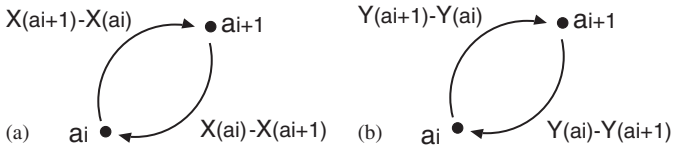


Fig. 5. Example of relative position edge pair. (a) Horizontal relative position edge pair and (b) vertical relative position edge pair.

If all blocks are rectangles, a packing corresponding to an arbitrary seq-pair always exists. If any non-rectangle blocks are contained, some seq-pair which has no corresponding packing may exist. If there is a rectilinear block packing which keeps all H/V constraints imposed by a seq-pair, then the seq-pair is said to be "*feasible*". And a seq-pair which is not feasible is said to be "*infeasible*". A seq-pair is feasible if and only if a positive cycle does not exist on both H/V constraint graphs.

Let "$Dist(v)$" denote the current estimated longest distance from source to vertex v by Ford's shortest path algorithm. Also, let "$Length(u, v)$" denote the weight of a directed edge (u, v). If we check an edge (u, v) whose weight satisfies $Dist(v) < Dist(u) + Length(u, v)$, we make the value of $Dist(v)$ the sum of $Dist(u)$ and $Length(u, v)$. We call this operation "*relaxation*". The longest path length for each vertex can be calculated or a positive cycle can be found in $O(n^3)$ time by iterating relaxations for all edges at most n times. Here, n is the number of sub-blocks.

3. Fast algorithm for rectilinear block packing

The conventional decoding method of rectilinear block packing [11] based on seq-pair consists of iteration of relaxation for all edges of H/V constraint graphs (SP) since Ford's algorithm is applied to it.

Here, edges can be classified into H/V edges and edges of relative position edge pairs. The number of H/V edges is ${}_nC_2$ or less and the number of edges of relative position edge pairs is less than $2n$ (n is the number of sub-blocks). Since time taken for relaxation per one edge is constant, relaxations for all edges take $O(n^2)$ time. Also, since the relaxations must be iterated n times maximally, the whole time taken is $O(n^3)$. Hence, if n is big, the execution speed gets very slow.

A series of relaxations of all H/V edges are equivalent to the bottom left corner packing method, keeping the constraints of a seq-pair. Therefore, a series of relaxation is replaced by a decoding method of SSP.

The maximal necessary number of iteration of relaxation of all edges can be reduced from n to $p + 1$ by using a procedure mentioned later (p is the number of rectilinear blocks without rectangular shape blocks). Hence, the

proposed method obtains a rectilinear block packing in $O((p+1)n)$ time, keeping all the constraints imposed by a given SSP or it judges a given SSP as infeasible in $O((p+1)n)$ time.

3.1. Restoring method of rectilinear block

In the proposed method, we use a restoring algorithm, which can be carried out in linear time of the number of sub-blocks. To use this algorithm, the following pair of graphs must be obtained for the given input data in advance.

H/V restoring graphs: H/V restoring graphs consist of plural connected components where each connected component corresponds to one rectilinear block. One of sub-blocks of each rectilinear block is defined as a root block and each sub-block corresponds to a vertex. In a connected component, vertices of a root block and of non-root blocks are connected by pairs of directed weighted edges. The weight of edges is defined the same as a relative position edge pair.

Examples of restoring graphs of a rectilinear block are shown in Fig. 6(b), where the root block of rectilinear block a is a_3. The restoring method of rectilinear blocks in this paper decides the position of each sub-block for x and y direction separately. In the following, only the procedure for x direction is shown. For y direction, each coordinate of sub-block can be decided similarly.

Procedure Restoring Rectilinear Block (RRB) for x Direction.

Step 1: For all edges directed from non-root blocks to root blocks on H-restoring graph, carry out relaxation.

Step 2: For all edges directed from root blocks to non-root blocks on H-restoring graph, carry out relaxation.

(**Procedure End**)

The object of this procedure is only to restore each rectilinear block, so the output may not satisfy the constraints imposed by SSP. Fig. 6(c)–(e) are examples of RRB for x direction restoration of a rectilinear block shown in Fig. 6(a). For easy understanding, y coordinates are set to the final value.

3.2. Decoding SSP to rectilinear block packing

In the proposed method of a rectilinear block packing, the bottom left corner packing based on the given SSP is obtained by computing the coordinates of the lower left corner of all sub-blocks for x and y direction separately. In the following, we propose the algorithm PackingX for calculation of x direction. PackingY is defined for y direction similarly. PackingX returns x coordinates of the lower left corner of all sub-blocks and those of vertical dissection lines, or the input SSP is judged to be infeasible when x coordinates do not converge by PackingX.

```
PackingX (SSP S, width of each rectangle, horizontal
restoring graph R_x,# rectilinear block p){
 variable: A vector representing x coordinate of each sub-
block and vertical dissection line: x[ ];
 Obtain a horizontal constraint graph (RD) G_x from S
(refer to Sect. 2.2 for details);
 Obtain a horizontal constraint graph G'_x from G_x by
  • inserting a vertex of corresponding sub-block name v
    to each edge of G_x,
  • giving 0 as the weight of incoming edge of v and
  • giving the width of sub-block v as the weight outgoing
    edge of v;
 Set each element of vector x[ ] to 0;
 for (i = 1, ..., p + 1){
   Relative Constraint Imposer(G'_x, x[ ]);
   if (Restoring Rectilinear Block(R_x, x[ ])
                              = = NOT_MOVE)
      return x[ ]; /* S is feasible */
 }
 return "S is infeasible";
}                          (AlgorithmPackingX END)
```

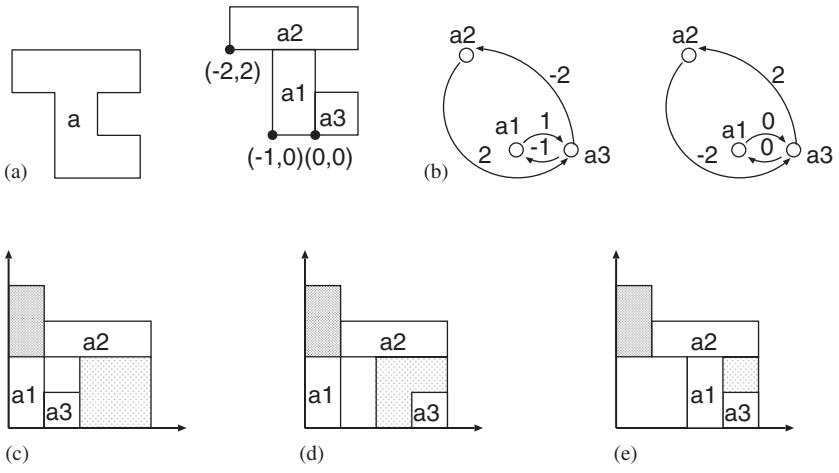
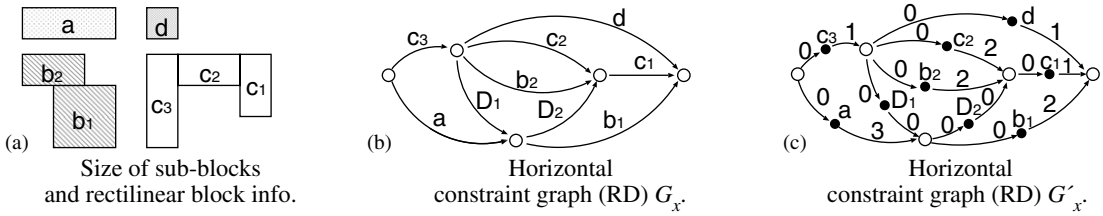
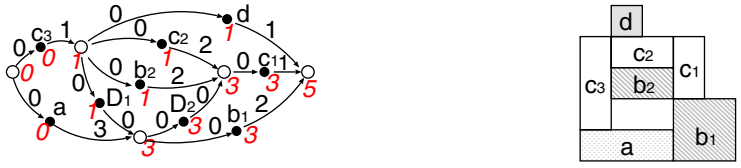
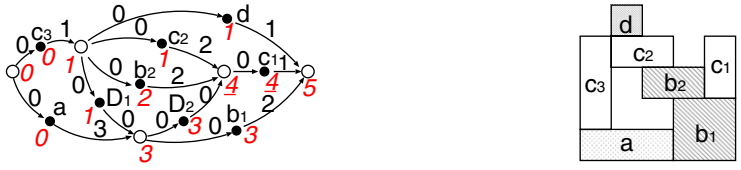


Fig. 6. Examples of restoring graphs and restoring a rectilinear block for x direction (root block a_3). (a) Rectilinear block a and sub-blocks a_1, a_2, a_3, (b) horizontal/vertical restoring graphs, (c) initial placement, (d) after moving a_3 in Step 1 and (e) after moving non-root blocks in Step 2.







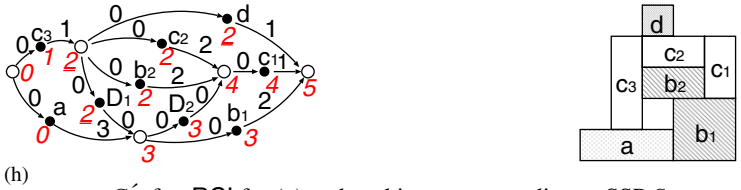


Fig. 7. Example of obtaining rectilinear block packing based on SSP $S = (c_3\, d\, c_2\, b_2\, a\, c_1\, b_1; a\, c_3\, b_1\, b_2\, c_2\, c_1\, d)$ for x direction. First, since S has adjacent crosses, dummy rectangles D_1, D_2 are inserted to obtain an SSP without adjacent crosses. Then $S' = (c_3\, d\, c_2\, b_2\, D_1\, a\, D_2\, c_1\, b_1; a\, c_3\, D_1\, b_1\, D_2\, b_2\, c_2\, c_1\, d)$ is obtained. (b) A horizontal constraint graph (RD) G_x based on S' is obtained. (c) By inserting vertices corresponding to sub-blocks, a horizontal constraint graph G'_x is obtained. Here, black vertices with attached names correspond to sub-blocks of the same name and white vertices correspond to vertical dissection lines. (d)–(h) The process of determining x coordinate of the lower left corner of each sub-block.

In PackingX, "Restoring Rectilinear Block" (RRB) which restores all rectilinear blocks returns NOT_MOVE when all sub-blocks do not move.

As mentioned in the beginning of this section, if relaxation of H/V edges is replaced by a decoding method of SSP, the input SSP is converted in vain to the same H/V constraint graph (RD) every time the operation is iterated. Hence, the conversion from an SSP to a corresponding H/V constraint graph (RD) is put outside for-loop of PackingX to speed it up. Inside for-loop, if RRB does not return NOT_MOVE, sub-blocks may not satisfy the horizontal constraints imposed by the SSP. So, we

must move the sub-blocks to coordinates satisfying them. Coordinates of the lower left corner of sub-blocks and dissection lines are determined by computing the longest path to each vertex of the constraint graph. We call this operation "Relative Constraint Imposer" (RCI).

RCI and RRB take $O(n)$ time, respectively. The maximum number of required iteration of these operations is $p+1$. So, the total time complexity is $O((p+1)n)$ to obtain a rectilinear block packing based on an SSP or to judge an SSP as infeasible.

In Fig. 7, an example of packing process of two rectilinear blocks b, c ($p=2$) and rectangles a, d are shown. For easy understanding, each y coordinate is already set to the final value. After RRB, overlaps of sub-blocks may be found as shown in Fig. 7(e). It will be understood that there are rectilinear blocks not restored yet after RCI as shown in Fig. 7(d) and (f).

3.3. Validity of the proposed PackingX

Correctness of the proposed method is shown in the following theorem.

Theorem 1. *From a given SSP and restoring graphs, the proposed decoding algorithm can compute the coordinates of each sub-block in the bottom left corner packing based on it if it is feasible, and can find it infeasible otherwise.*

Proof. If the given SSP is feasible, there exists a bottom left corner packing P according to the definition of the feasible seq-pair. We call the coordinates of each sub-block on P "*final coordinates*," which is equal to the longest path length from the source to the corresponding vertex on the horizontal and vertical constraint graphs with relative position edge pairs, as shown in [11].

As denoted in Section 2.3, each relative position edge pair connects a pair of vertices corresponding to adjacent sub-blocks in a rectilinear block. Change every relative position edge pair to connect vertices of a root block and of non-root blocks in the rectilinear block.

Here, the weight of edges is set by the same way as restoring graphs defined in Section 3.1. Then, it is easily understood that the longest path length to the vertex corresponding to each sub-block on the horizontal and vertical constraint graphs obtained by the above operation is equal to the converged coordinates of the sub-block.

Call a vertex corresponds to a rectangle "*rectangle vertex*," and that corresponds to a root block other than rectangles "*root vertex*". We will show only for the x coordinate in the following, but the similar proof is possible for the y coordinate.

To each vertex on the constraint graph, there may be several number of longest paths from the source. Focus on a longest path which passes the minimum number of root vertices among them and call the number "*minimum number of passed root vertices on a longest path*," in short "MNRVLP". Note that for a vertex corresponds to a non-root block, the root vertex for the rectilinear block of it should not be counted. Because a path cannot go through any vertex more than once, MNRVLP is not bigger than p, the number of rectilinear blocks excluding rectangles, and MNRVLP for a vertex other than rectangle vertex is less than p.

After the first invocation of "relative constraint imposer," the coordinate of every rectangle vertex whose MNRVLP is zero has converged to the final value. After the first invocation of "restoring rectilinear block," the coordinate of every vertex whose MNRVLP is zero has converged to the final value.

If the coordinates of all the vertices whose MNRVLP are less than k converge to the final value, an invocation of

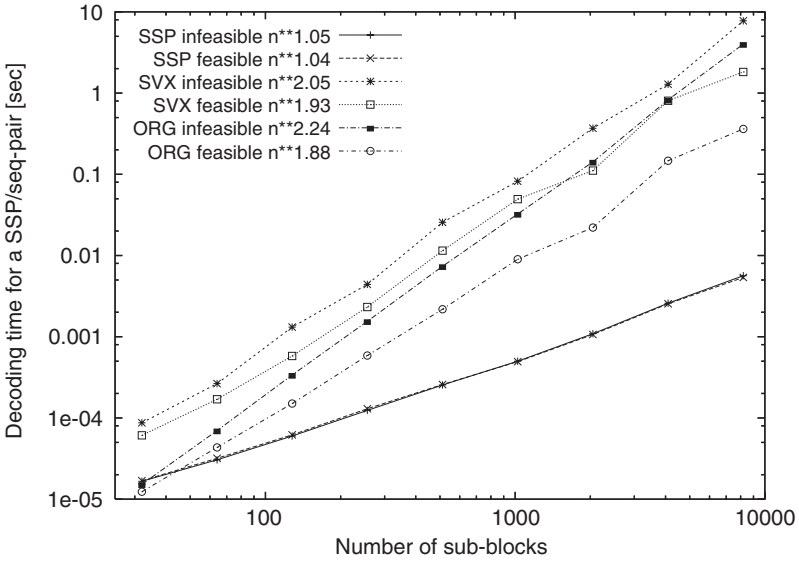


Fig. 8. Comparison of decoding speed ($p=1, \ell=3n/4+1$) of SSP (proposed method) with SVX (conventional method in [13]), and ORG (conventional method in [11]). Each value in explanatory note is slope obtained by the least-square method.

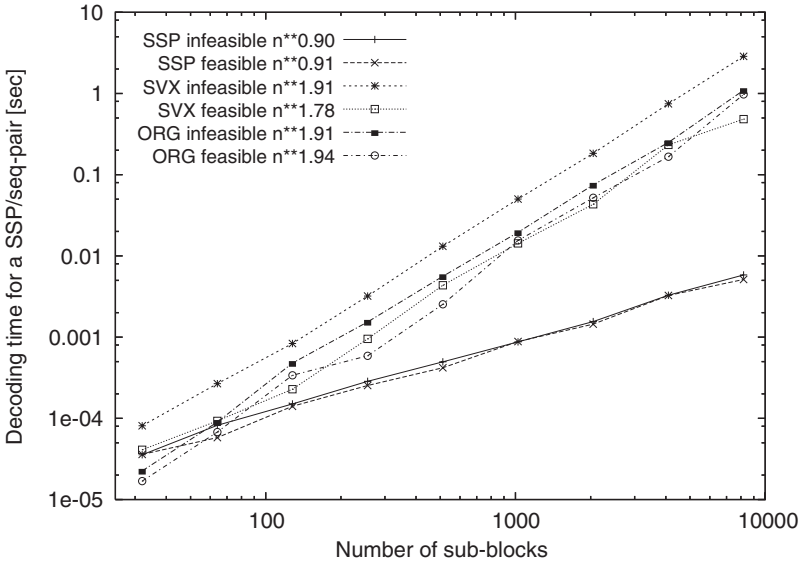


Fig. 9. Comparison of decoding speed when $m = p = 16$.

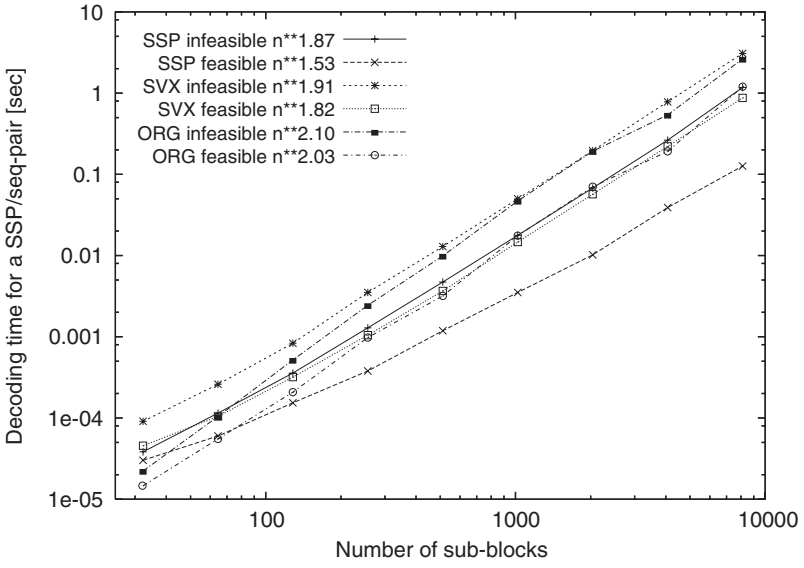


Fig. 10. Comparison of decoding speed when $m = p = n/2$.

"relative constraint imposer" makes the coordinates of all the rectangle vertices whose MNRVLP are k converge to the final value. In addition to this, an invocation of "restoring rectilinear block" makes the coordinates of all vertices whose MNRVLP are k converge to the final value.

Hence, by induction, after the pth invocation of "restoring rectilinear block," the coordinates of all vertices whose MNRVLP are less than p have converged to the final value. Also, after the $(p+1)$th invocation of "relative constraint imposer," the coordinates of all rectangle vertices whose MNRVLP are not more than p have converged to the final value. Therefore, if the given SSP is feasible, the coordinates of all sub-blocks have converged after the $(p+1)$th invocation of "relative constraint imposer".

If no relaxation occurs in "restoring rectilinear block," it is clear that all the coordinates of sub-blocks converge and the procedure ends. It is obvious that when some relaxation occurs in the $(p+1)$th invocation of "restoring rectilinear block," the given SSP can be judged as infeasible. □

4. Experimental results

4.1. Comparison of decoding speed

To compare decoding speed with the conventional method, the proposed method requiring $O((p+1)n)$ time (p and n are the number of rectilinear blocks excluding rectangles and rectangle sub-blocks, respectively) is implemented in Simulated Annealing with the MOVE

operation for SSP [18,23] in C++. The conventional methods [11,13] are implemented in C and C++, respectively (the methods proposed in [11,13] require $O(n^3)$ and $O(n^2+\ell^3)$ time, respectively, where ℓ is the number of rectilinear blocks). We carry out experiments to compare the decoding time by Pentium 4 2.4 GHz using Simulated Annealing program with ultra-high temperature which hardly rejects adjacent solutions (times for perturbation are excluded).

In the experiments, three kinds of data sets generated randomly are inputted. (1) a rectilinear block consisting of $n/4$ sub-blocks and $3n/4$ rectangles, that is, the number of rectilinear blocks $\ell=(3/4)n+1$ and $p=1$. (2) sixteen rectilinear blocks consisting of $n/16$ sub-blocks, that is, $\ell=p=16$. (3) $n/2$ rectilinear blocks consisting of two sub-blocks, that is, $\ell=p=n/2$.

The initial solution of Simulated Annealing in these methods are packings where all rectilinear blocks are placed in a row. When an obtained adjacent solution is infeasible, it is abandoned and another one is obtained from the present solution. This operation is repeated until more than 10 000 solutions are obtained separately for both feasible and infeasible cases and average CPU time for one solution is calculated in both cases respectively. Figs. 8, 9 and 10 show the experimental results of the above three cases when n is $32, 64, \ldots, 8192$. The slopes of the above results calculated by the least-square method are also shown in Figs. 8, 9 and 10. The abbreviation SSP is for the proposed method, and SVX and ORG each is for the conventional method in [11] and [13], respectively. In every respect, the proposed method is faster than the conventional method except when $n=32$ as shown in Figs. 8, 9 and 10.

In (1), since $p=1$, $\ell=(3/4)n+1$, the conventional methods took $O(n^3)$ time and the proposed method took $O(n)$ time. In (2), since $\ell=p=16$, SVX and ORG took $O(n^2)$ time and $O(n^3)$ time respectively. The proposed method took $O(n)$ time. In (3), since $\ell=p=n/2$, the conventional methods took $O(n^3)$ time and the proposed method took $O(n^2)$ time.

In the experimental results of the methods requiring $O(n^2)$ time or $O(n^3)$ time, the slopes of the graph in Figs. 8, 9 and 10 are gentler than the time complexity. The reason is that O-notation states the upper bounds on running time and our experiments converge on an average in a short time except for those requiring $O(n)$ time. Besides, O-notation is available when n and ℓ are big enough, but in this case, they are not.

4.2. Experimental results using Simulated Annealing

The solution search for the proposed method was carried out by using Simulated Annealing. Fig. 11 is a packing result obtained on AthlonXP 1700+ for the same data as Fig. 10 in [26], where a rotation or reflection is not taken into consideration. The data consists of 10 L-shape blocks and 30 rectangles. "Area [%]" is obtained by dividing a bounding box area by the total area of all blocks.

In Table 1, the experimental results of the proposed and the conventional methods are compared. It is confirmed that the result of Simulated Annealing search using the proposed method is better than that using the conventional methods. The result of the proposed method is better than that of the method only for L-shaped block packing [11] by 0.4%. The packing result obtained by the method [11] is fairly dense.

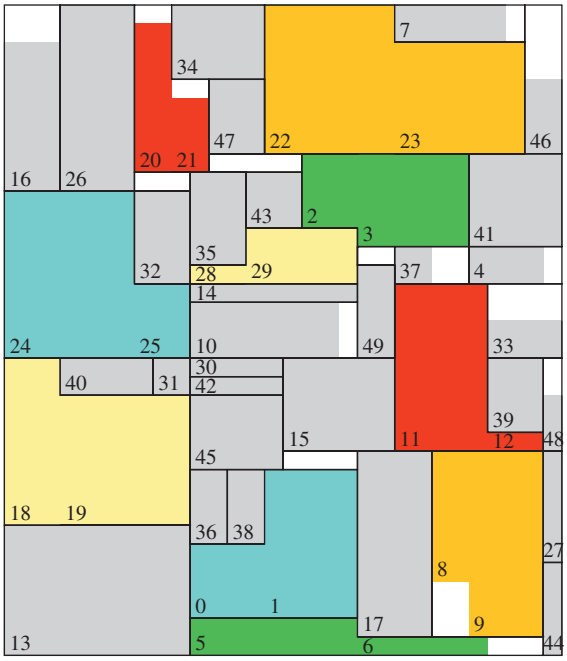


Fig. 11. Example of L-shaped block packing.

Table 1
Comparison of Simulated Annealing search

Method	Target	CPU	Time (s)	Area (%)
SSP (proposed)	Rectilinear	Pentium 4 2.4 GHz	14.92	106.3
seq-pair [11]	L-shape	Ultra SPARC 200 MHz (Pentium 4 2.4 GHz)	147 (16.88)	106.7
MP-BSG [2]	Core convex	Pentium III 910 MHz	374	108.6
seq-pair [5]	L-shape	Celeron 400 MHz	11	109.3
seq-pair [4]	L-shape	Ultra SPARC 200 MHz	297	109.8
BSG [26]	L-shape	SS5 80 MHz	300	111.5

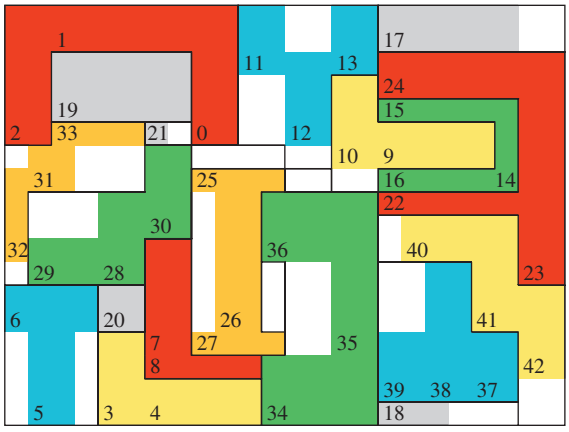


Fig. 12. Nineteen blocks (14 rectilinear blocks and five rectangles) consisting of 43 sub-blocks.

Table 2
Comparison of Simulated Annealing search for 86 blocks

Method	Area (%)	Average time (s)
	Average (min, max)	
Proposed	106.8 (105.0, 109.1)	17 943
Conventional [11]	107.0 (106.0, 109.3)	37 160
Conventional [13]	107.1 (105.3, 108.6)	92 322

A packing result for the same data as Fig. 12(c) in [11] is shown in Fig. 12. The data consists of 14 rectilinear blocks and five rectangles, and the total area of all blocks is 353. The packing took 1251 seconds by AthlonXP 1700+ and its bounding-box area is 441. On the contrary, Fig. 12(c) in [11] took 13 446 seconds by Ultra SPARC 200 MHz and its bounding-box area is 468.

In order to confirm the solution search speed including time required to generate adjacent solutions for Simulated Annealing, we carried out experimental comparison with the conventional methods [11,13] under the same annealing schedule (initial temperature, final temperature, ratio of falling temperature and the number of times evaluating every temperature). The input data consists of two sets of ami33 of MCNC Benchmark and 20 rectilinear blocks of GSRC Benchmark-Level IV n100_20%. Here, to balance the size, the height and the width, each of the rectilinear blocks is enlarged eight times. The scale of data is bigger than the input data used in Table 1. Average, minimum and maximum area of the results of five times experiments for three methods are shown in Table 2 and the best result, which is obtained by the proposed method, is shown in Fig. 13. According to Table 2, the proposed method is faster than the conventional methods in the total time of obtaining one feasible solution and of decoding the solution. Compared with the conventional methods using sequence-pair, the result of the proposed method (using SSP) is by no means inferior, so the solution space does not deteriorate by using SSP.

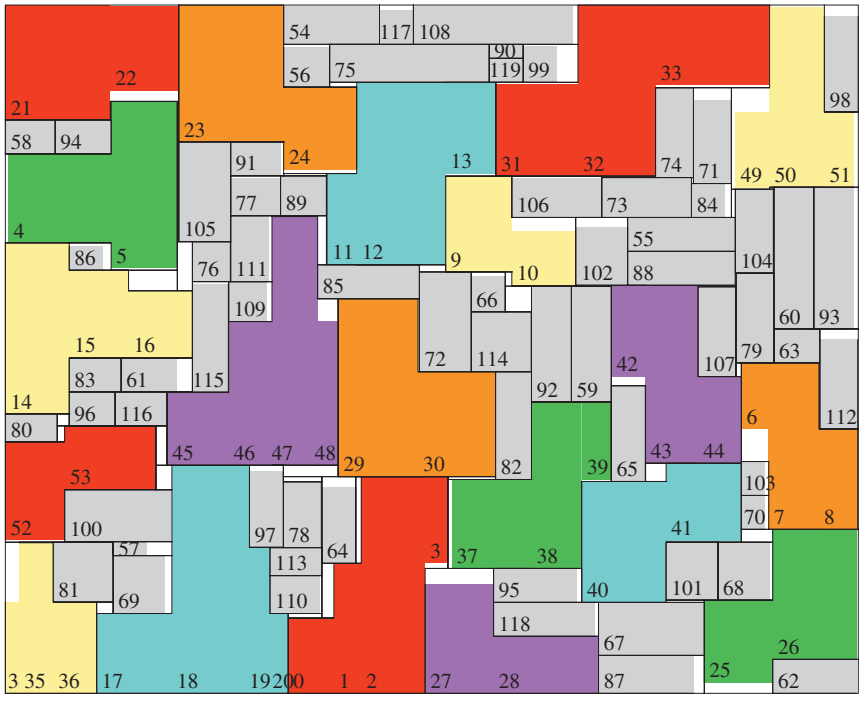


Fig. 13. Eighty-six blocks (20 rectilinear blocks and 66 rectangles) consisting of 120 sub-blocks.

5. Conclusion

We proposed a method to represent an arbitrary rectangle packing by using selected sequence-pair (SSP). We also proposed an efficient decoding method to obtain a rectilinear block packing in $O((p+1)n)$ time (n is the number of sub-blocks and p is the number of rectilinear blocks excluding rectangles) keeping all the constraints imposed by a given selected sequence-pair. We confirmed the proposed method was faster than the conventional methods by experimental comparisons.

References

[1] M.Z. Kang, W.W.-M. Dai, Arbitrary rectilinear block packing based on sequence pair, in: Proceedings of the ACM/IEEE ICCAD, 1998, pp. 259–266.

[2] K. Sakanushi, S. Nakatake, Y. Kajitani, Placement algorithm for rectilinear core cells by multi-layered parametric BSG, IEICE Trans. Fundamentals J85-A (9) (2002) 938–949.

[3] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, S.-W. Wu, B*-Trees: a new representation for non-slicing floorplans, in: Proceedings of the IEEE DAC, 2000, pp. 458–463.

[4] K. Fujiyoshi, H. Murata, A method to pack L-shaped and rectangular modules using sequence-pair, in: 11th IEICE Circuit and Systems Karuizawa Workshop, 1998, pp. 113–118 (in Japanese).

[5] H. Saito, K. Fujiyoshi, The improved method of L-shaped block packing, in: 13th IEICE Circuit and Systems Karuizawa Workshop, 2000, pp. 245–250 (in Japanese).

[6] Y. Ma, X. Hong, S. Dong, Y. Cai, C.-K. Cheng, J. Gu, Floor-planning with abutment constraints and L-shaped/T-shaped blocks based on corner block list, in: Proceedings of the ACM/IEEE DAC, 2001, pp. 770–775.

[7] M. Kang, W.W.-M. Dai, General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure, in: Proceedings of the IEEE ASP-DAC, 1997, pp. 265–270.

[8] J. Dufour, R. McBride, P. Zhang, C.-K. Cheng, A building block placement tool, in: Proceedings of the IEEE ASP-DAC, 1997, pp. 271–276.

[9] J. Xu, P.-N. Guo, C.-K. Cheng, Rectilinear block placement using sequence-pair, in: Proceedings of the ISPD, 1998, pp. 173–178.

[10] M.Z. Kang, W.W.-M. Dai, Topology constrained rectilinear block packing for layout reuse, in: Proceedings of the ISPD, 1998, pp. 179–186.

[11] K. Fujiyoshi, H. Murata, Arbitrary convex and concave rectilinear block packing using sequence-pair, IEEE Trans. CAD 19 (2) (2000) 224–233.

[12] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, VLSI module placement based on rectangle-packing by the sequence-pair, IEEE Trans. CAD 15 (12) (1996) 1518–1524.

[13] K. Machida, K. Fujiyoshi, The improvement of rectilinear block packing using sequence-par, IEICE Technical Report, VLD2000-134, 2001, pp. 1–6 (in Japanese).

[14] X. Tang, M.D.F. Wong, On handling arbitrary rectilinear shape constraint, in: IEEE ASP-DAC, 2004, pp. 38–41.

[15] A. Ikeda, C. Kodama, A. Nakagomi, K. Fujiyoshi, A fast algorithm for rectilinear block packing using SSP, IEICE Technical Report, VLD2003-103, 2003, pp. 199–204 (in Japanese).

[16] Y. Pang, C.-K. Cheng, K. Lampaert, W. Xie, Rectilinear block packing using O-tree representation, in: Proceedings of the ISPD, 2001, pp. 156–161.

[17] J.-M. Lin, H.-L. Chen, Y.-W. Chang, Arbitrary convex and concave rectilinear module packing using TCG, in: Proceedings of the IEEE DATE, 2002, pp. 69–75.

[18] C. Kodama, K. Fujiyoshi, Selected Sequence-Pair: an efficient decodable packing representation in linear time using sequence-pair, in: Proceedings of the IEEE ASP-DAC, 2003, pp. 331–337.

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

[21] Y. Takashima, H. Murata, The tight upper bound of the empty rooms in floorplan, in: SASIMI2001, 2001, pp. 264–271.

[22] X. Tang, R. Tian, D.F. Wong, Fast evaluation of sequence pair in block placement by longest common subsequence computation, IEEE Trans. CAD 20 (12) (2001) 1406–1413.

[23] K. Fujiyoshi, C. Kodama, K. Kiyota, An efficient MOVE operation for selected sequence-pair, IPSJ J 46 (7) (2005) 1711–1722 (in Japanese).

[24] K. Sakanushi, Y. Kajitani, D.P. Mehta, The quarter-state sequence floorplan representation, IEEE Trans. CAS-I 50 (3) (2003) 376–386.

[25] Sadiq M. Sait, Habib Youssef, VLSI physical design automation, IEEE Press, New York, 1995.

[26] S. Nakatake, K. Fujiyoshi, H. Murata, Y. Kajitani, Module placement on BSG-structure and IC layout applications, IEEE Trans. CAD 17 (6) (1998) 519–530.

Kunihiro Fujiyoshi received his B.E., M.E. and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1987, 1989 and 1994, respectively. From 1992 to 1996, he was a research associate of School of Information Science at Japan Advanced Institute of Science and Technology, Ishikawa. He was with Tokyo University of Agriculture & Technology as a lecturer from 1997 and has been an associate professor since 2000 of Department of Electronic and Information Engineering. His research interests are in combinatorial algorithms and VLSI layout design. He is a member of IEEE, IEICE and Information Processing Society of Japan.

Chikaaki Kodama received his B.E. and M.E. degrees in electronic and information engineering from Tokyo University of Agriculture & Technology, Tokyo, Japan, in 1999 and 2001, respectively. He was with Fujitsu Ltd. from 2001 to 2003. Currently, he is studying toward the D.E. degree in the Department of Electronic and Information Engineering from Tokyo University of Agriculture & Technology. His research interests are VLSI layout design, especially floor-planning and packing, and apparel CAD system. He is a student member of IEEE and IEICE.

Akira Ikeda received his B.E. and M.E. degrees in electrical and electronic engineering from Tokyo University of Agriculture & Technology, Tokyo, Japan, in 2003 and 2005, respectively. Currently, he is with JEDAT Inc. His research interests were VLSI layout design, especially floor-planning and packing.