Arbitrary Rectilinear Block Packing Based on Sequence Pair

Maggie Zhiwei Kang
maggiek@cadence.com
Cadence Design Systems, Inc.

Wayne Wei-Ming Dai
dai@cse.ucsc.edu
University of California at Santa Cruz

Abstract

Based on the sequence pair structure [1], this paper proposes a novel method to represent arbitrary shaped rectilinear blocks. The necessary and sufficient conditions are derived such that non-overlapped packing of arbitrary rectilinear blocks can always be guaranteed regardless of the dimensions of the blocks. A stochastic search is applied, three sequence pair operations are defined to search the feasible solution space both continuously and exhaustively. Theoretical results derived in this paper show that an optimal solution can always be reachable through finite steps of the stochastic search. As such, the algorithm becomes a significant breakthrough in the general packing problem both theoretically and practically.

1 Introduction

Due to the layout complexity in deep submicron technology, routing layers have increased up to five or six, most of the channel routing is being replaced by area routing and the block planning becomes more and more like a block packing problem: a set of macro blocks are packed together without overlaps such that the overall area as well as other objectives is minimized under some physical constraints.

Rectangle packing (RP) is the simplest formulation of the block packing problem. Let M be a set of m rectangular blocks whose height and width are given in real numbers with fixed orientations. A packing of M is a non-overlapping placement of the blocks, the minimum bounding box of the packing is referred to as the *base rectangle*, which corresponds to the chip image in practical application. RP can be shown to be NP-hard by reducing it into an NP-complete problem, which is a rectangle packing problem with fixed width of the base rectangle [2].

Since the height and width of blocks can be continuous real numbers, RP is not simply a combinatorial optimization problem. Several numerical approaches have been proposed [3, 4]. An alternative approach is stochastic search, in which the solution space is defined by a data representation. The packing area can be quickly and accurately calculated based on the data structure. The incremental change of the packing topology can be easily carried out on the data structure, through which the stochastic heuristic searches a data representation which yields the best packing. From the description of Section 2, we will see that the SP is an ideal structure for representing the rectangle packing [1].

ICCAD98, San Jose, CA, USA

1.1 Data Representation of Rectangle Packing

To represent the placement of blocks, researchers have introduced the concept of a *meta-grid* to describe the topological grid composed of orthogonal lines on the plane without containing physical dimensions. There are two kinds of meta-grid structures: slicing and non-slicing. A meta-grid has a *slicing* structure if it can be obtained by recursively cutting a rectangle into two parts by either a vertical or a horizontal line [5], as shown in Fig. 1 (a). Otherwise the meta-grid has a *non-slicing* structure, as shown in Fig. 1(b).

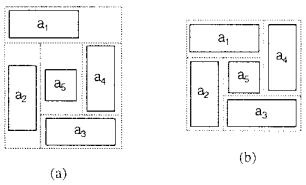


Figure 1: (a) slicing floorplan and (b) non-slicing floorplan.

Slicing structure provides a simple way for optimizing the block orientations, defining reasonable channels in global routing and appropriately ordering the channels during detail routing. Wong and Liu [6] proposed a normalized Polish expression to represent a slicing structure which enables the efficient neighborhood search.

Typically the slicing structure is very limited since most of the dissections are non-slicing. To cover this intrinsic disadvantage, many efforts [7, 8, 9] have been tried but are not satisfactory. On the other hand, with the increase of routing layers, channel routing is being replaced by area routing; blocks are packed together to minimize the area. As such, the wasted area due to slicing structures is more evident, and the non-slicing structure becomes more attractive.

Recently, Murata et al. [1] introduced the sequence pair (SP) and Nakatake et al. [10] proposed the bounded slicing grid (BSG) structures to represent the general floorplan

including non-slicing structures. Both SP and BSG provide a finite solution space in which an optimal solution is included. In both structures, the floorplan topology is represented such that the binary relationship "left of" or "below" between any two blocks is uniquely defined. The packing in the x and y dimensions are carried out independently. Section 2 will describe the sequence pair structure in detail.

1.2 Rectilinear Block Packing Problem

Few previous works have studied rectilinear shaped blocks. One of the most noticeable work is the *bounded 2D contour searching algorithm* proposed by [11]. Arbitrarily shaped rectilinear blocks are represented by a set of four linear profiles, each one of them specifies the profile viewed from one side of the block. When blocks are compacted along a certain direction, 2D contour searching is carried out on the profiles of the compacted design. This compaction method cannot be applied to the block packing problem due to the complicated calculations of the profiles.

On the other hand, a BSG-based method was proposed by [12] to solve the packing of rectangular, L-shaped, T-shaped, and soft blocks. Most recently, [13] proposed an algorithm based on both BSG and SP structures, in which the topology constrained block packing can be optimized given blocks belong to a specific class of convex rectilinear shapes. An SP-based algorithm was proposed by [14] to pack the mountain-shaped blocks. Based on BSG structures, an algorithm was proposed by [15] where each block is sliced into a set of rectangular sub-blocks, one of the sub-blocks is selected to be a *master* and the others *slaves*. Only the master sub-block is assigned into BSG domain. After the compaction, the slaves are attached with the master. A post-process eliminates overlaps by pushing the neighboring blocks away. Obviously the optimal solution is not guaranteed to be included.

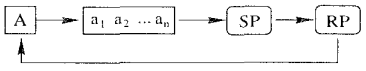


Figure 2: Each rectilinear shaped macro block, A, is partitioned into a set of rectangular sub-blocks, each of them is individually handled in the sequence pair (SP) as a unit block. After the unit blocks are compacted in horizontal and vertical dimensions, a post process aligns both x and y coordinates of the unit blocks such that the original macro shapes are recovered.

1.3 Major Contribution of This Paper

Based on a sequence pair structure, this paper solves the optimization of the arbitrary block packing problem. The methodology is shown in Fig. 2 where every rectilinear macro block is partitioned into a set of rectangular sub-blocks, each of them is individually handled in the sequence pair as a unit block. After the unit blocks are compacted in horizontal and vertical dimensions, a post process aligns both x and y coordinates of the unit blocks such as to yield an optimal packing of the arbitrary rectilinear blocks for the given sequence pair.

A sequence pair is *feasible* if the shapes of rectilinear macro blocks can be recoveried regardless of the dimensions of the blocks. This paper derives three conditions on the sequence pair, each of them is shown to be *necessary*; a sequence pair is infeasible if the sequence pair does not satisfy this condition. On the other hand, they are also shown to be *sufficient*; a sequence pair is feasible if the sequence pair satisfies the three conditions.

Furthermore, this paper proves that there always exists a feasible sequence pair for a packing of convex rectilinear blocks, and vice versa. Due to this fact, an optimal packing of convex blocks can be found by exhausting the finite number of feasible sequence pairs. In the second part of this paper, a stochastic search is applied to the optimization of convex block packing. Three operations are defined on the sequence pair, each of them incrementally changes a feasible sequence pair and the resultant sequence pair remains feasible. The results of this paper show that the optimal solution can always be reachable through a finite steps of the operations, and the stochastic search based on the three operations can cover the feasible solution space both continuously and exhaustively.

In the following, Section 2 first introduces the sequence pair structure, and Section 3 describes the representation method for arbitrary rectilinear blocks in a sequence pair. The concept of feasible sequence pairs is defined, and three conditions on a sequence pair are presented. Section 4 shows that the three conditions are necessary and sufficient for a sequence pair to be feasible. In Section 5, it is proven that there always exists a feasible sequence pair for a packing of convex rectilinear blocks. Based on this fact, Section 6 applies a stochastic search on the convex block packing. The experimental results are reported in Section 7 followed by the conclusion of this paper.

2 Sequence Pair (SP) Structure

To clarify the notation, we call the rectangular blocks represented in sequence pair *unit blocks*. A *sequence pair* for a set of n unit blocks is a pair of sequences of n symbols which represent the unit blocks : (Γ_1, Γ_2). Given a sequence pair ($a\ b\ d\ e\ c\ f$, $c\ b\ f\ a\ d\ e$) , an oblique grid can be constructed as shown in Fig. 3 (a) : 45^0 slope lines are named from left to right by the symbols in the first sequence Γ_1, and -45^0 slope lines are similarly named by the symbols in the second sequence Γ_2. Each unit block is placed at the cross of the two slope lines which are named by the same symbol corresponding to the unit block. As shown in Fig. 3 (b), the plane can be divided by the two crossing slope lines into four cones for any unit block b. Unit a is in the upper cone of b, then a is above b. Similarly, unit d, e and f is in the right cone of b, then they are right to b. In general, the sequence pair imposes the relationship between each pair of unit blocks as follows :

$$(\cdots a \cdots b \cdots, \quad \cdots a \cdots b \cdots) \quad \Rightarrow \quad a \textit{ is left to } b,$$
$$(\cdots b \cdots a \cdots, \quad \cdots a \cdots b \cdots) \quad \Rightarrow \quad a \textit{ is below } b.$$

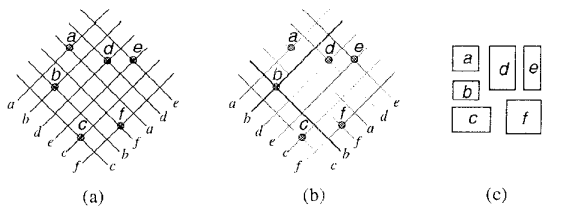


Figure 3: (a) Oblique grid of ($a\ b\ d\ e\ c\ f$, $c\ b\ f\ a\ d\ e$) , (b) the four cones of block b, and (c) the corresponding packing of the six blocks.

Given the sequence pair, a horizontal directed graph G_h is derived as shown in Fig. 4 (a). Each vertex corresponds to a unit block, there is an arc from unit a to d if and only if a is left to d. In particular, there is a source s_h connected to each leftmost unit and a sink t_h connected from each rightmost unit. Each vertex has a weight which equals to the width of the corresponding unit block. The vertical graph G_v can be similarly derived as shown in Fig. 4 (b).

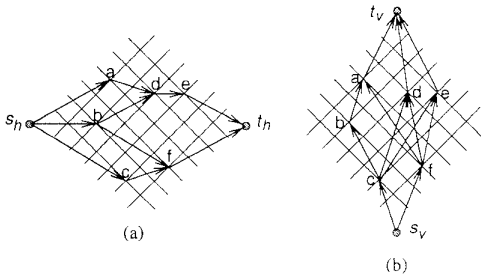


Figure 4: The two vertex weighted directed acyclic graphs derived from the sequence pair ($a\ b\ d\ e\ c\ f$, $c\ b\ f\ a\ d\ e$) , in which the transitive arcs are deleted for the simplification.

Both G_h and G_v are vertex weighted directed acyclic graphs, the packing of unit blocks can be obtained by simply applying the well-known *longest path algorithm* on both graphs. The x and y coordinates of each unit block are determined by the longest path from source to the vertex of the unit in G_h and G_v, respectively. Similarly, the width and height of the overall packing can be determined by the source-to-sink longest path of G_h and G_v.

For any two unit blocks, there is always an arc in either G_h or G_v, but not both. Due to this fact, the x and y coordinates can be independently determined, and the resultant packing is guaranteed not to contain any overlap. Since the width and the height are independently minimum, the resultant packing is optimal for the given sequence pair. The longest path calculation can be done in the time $O(n^2)$, where n is the number of blocks. The following Theorem has been proven by [1] :

Theorem 1 *There always exists a sequence pair corresponding to a rectangle packing, and vice versa.*

Therefore the optimal packing can always be found by exhausting the finite number $(n!)^2$ of sequence pairs.

3 Arbitrarily Rectilinear Blocks in Sequence Pair

3.1 H, V-Partition

Let A denote an arbitrary shaped rectilinear macro block. A can be partitioned into a set of rectangular sub-blocks by slicing A from the left to right along every vertical boundary of A. As shown in Fig. 5 (a), the partition is referred to as a *horizontal partition* or *H-partition*. Similarly A can be *vertically partitioned* or *V-partitioned* as shown in Fig. 5 (b). Given that A is H-partitioned or V-partitioned into a_1, a_2, $\cdots$ a_n, each sub-block $a_i \in A$ is individually represented in SP as a unit block. We call the pair of permutations on $a_i \in A$ in the sequence pair a *permutation pair of* A.

For any two unit blocks a and b placed in a plane without overlaps, let u_a and l_a denote the upper and lower boundary of block a, respectively. Block a is left of b if the right boundary of a is left of the left boundary of b. If $[l_a, u_a] \cap [l_b, u_b] \neq \phi$ as shown in Fig. 6 (a), a is *strictly* left of b. The permutation pair of a, b is (a b, a b). Otherwise $[l_a, u_a] \cap [l_b, u_b] = \phi$ as shown in Fig. 6 (b), a is both left of and below b, and two permutation pairs of a, b are possible.

Given that macro block A is either H-partitioned or V-partitioned, the topology between the sub-blocks of A is strictly

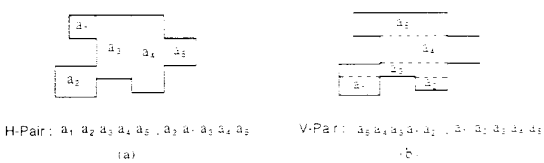


Figure 5: (a) H-partition slices block A on every vertical boundary from the left to right. (b) V-partition slices block A on every horizontal boundary from the bottom to top.

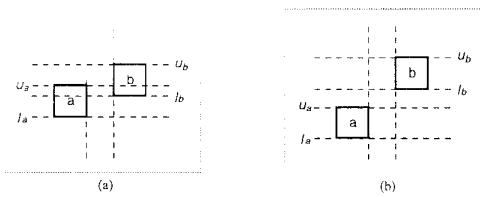


Figure 6: For any two unit blocks a and b placed in a plane without overlaps, a is left of b if the right boundary of a is left of the left boundary of b. In (a), $[l_a, u_a] \cap [l_b, u_b] \neq \phi$, a is *strictly* left of b. The permutation pair of a, b is (a b, a b). In (b), $[l_a, u_a] \cap [l_b, u_b] = \phi$, a is both left of and below b, and two permutation pairs are possible.

defined. Accordingly, there is exactly one permutation pair of A corresponding to the partition. The following Lemma is true :

Lemma 1 *Given an arbitrary rectilinear macro block A, there exists only one permutation pair of A corresponding to the H-partition of A, which is referred to as an* H-pair *of A. Similarly there exists only one permutation pair of A corresponding to the V-partition of A, which is referred to as a* V-pair.

In the example shown in Fig. 5, the H-pair of A is (a_1 a_2 a_3 a_4 a_5, a_2 a_1 a_3 a_4 a_5) and V-pair is (a_5 a_4 a_3 a_1 a_2, a_1 a_2 a_3 a_4 a_5).

3.2 X, Y-Alignment

Given a sequence pair representing topological relationships among unit blocks of rectilinear macro blocks, the unit blocks are compacted left and downward using the longest path algorithm in the x and y directions, respectively. If any unit block is further moved left or downward after the compaction, overlaps will occur. In other words, X, Y-alignment can only move unit blocks right and upward. Given a macro block A is H-partitioned as shown in Fig. 7 (a), and the sequence pair is (c a_1 a_2 e a_3 a_4 d a_5 f, e a_2 a_1 a_3 f c a_4 a_5 d), the unit blocks are compacted as shown in Fig. 7 (b). In the x direction, a_5 is the right-most sub-block of A. X-Alignment(A) freezes a_5 and moves a_4 to the right until it hits a_5, then moves a_3 to the right until it hits a_4, and so on. After the right move of a_1 and a_2, the sub-blocks of A are aligned together in the x direction as shown in Fig. 7 (c). In the y direction, the highest sub-block of A is a_4. Y-Alignment(A) freezes a_4, and moves the other sub-blocks of A upward to align with a_4 as shown in Fig. 7 (d). When macro block A is V-partitioned, X, Y-alignment can be symmetrically carried out. Obviously the following Theorem is true :

Theorem 2 *Given a sequence pair representing topological relationships among unit blocks of rectilinear macro blocks, X, Y-Alignment yields an optimal packing for the sequence pair.*

Given a sequence pair, the shapes of macro blocks may not be recovered by X, Y-Alignment.

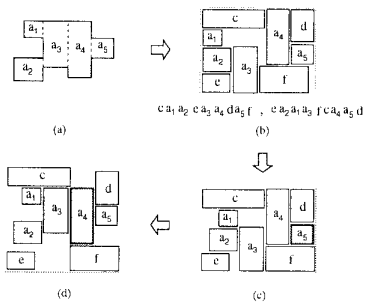


Figure 7: (a) Macro block A is H-partitioned into a_1 a_2 a_3 a_4 a_5. (b) Given a sequence pair (c a_1 a_2 e a_3 a_4 d a_5 f, e a_2 a_1 a_3 f c a_4 a_5 d), the unit blocks are compacted left and downward using the longest path algorithm in the x and y directions, respectively. (c) In the x direction, a_5 is the right-most sub-block of A. X-Alignment(A) freezes a_5 and moves a_4 to the right until it hits a_5, then moves a_3 to the right until it hits a_4, and so on. After the right move of a_1 and a_2, the sub-blocks of A are aligned together in the x direction. (d) In the y direction, the highest sub-block of A is a_4. Y-Alignment(A) freezes a_4, and moves the other sub-blocks of A upward to align with a_4.

Definition 1 *A sequence pair is* feasible *if after X, Y-Alignment, the shapes of rectilinear blocks are guaranteed to be recovered regardless of the dimensions of the blocks.*

Definition 2 *A sequence pair is* infeasible *if it is not feasible, e.g. the shapes of rectilinear blocks may not be recovered after X, Y-Alignment.*

Accordingly, we define the necessary and sufficient conditions on sequence pair as follows:

Definition 3 *A condition is* necessary **iff** *when the condition is not satisfied, the sequence pair is infeasible.*

Definition 4 *A set of conditions are* sufficient **iff** *when the set of conditions are satisfied, the sequence pair is feasible.*

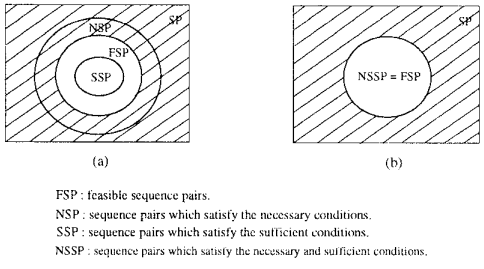


Figure 8: (a) Let FSP denote a set of feasible sequence pairs, NSP the sequence pairs which satisfy the necessary conditions, and SSP the sequence pairs which satisfy the sufficient conditions, then $SSP \subseteq FSP \subseteq NSP$. (b) If we can derive a set of conditions which are necessary and sufficient, the solution space for feasible sequence pairs can be completely characterized by the necessary and sufficient conditions: a sequence pair is feasible if the conditions are satisfied, otherwise infeasible.

Let FSP denote a set of feasible sequence pairs, NSP the sequence pairs which satisfy the necessary conditions, and SSP the sequence pairs which satisfy the sufficient conditions, then $SSP \subseteq FSP \subseteq NSP$ as shown in Fig. 8 (a). If the necessary conditions are not met, the sequence pair is infeasible. If the necessary conditions are met, the sequence pair may or may not be feasible. On the other hand, if the sufficient conditions are met, the sequence pair is feasible. If the sufficient conditions are not met, the sequence pair may or may not be feasible. As such, necessary conditions can not completely characterize the solution space for feasible sequence pairs, neither can sufficient conditions.

If we can derive a set of conditions which are necessary and sufficient, as shown in Fig. 8 (b), the solution space for feasible sequence pairs can be completely characterized by the necessary and sufficient conditions: a sequence pair is feasible if the conditions are satisfied, otherwise infeasible. In the following, we first present three conditions on sequence pair, and later we will prove that the three conditions are both necessary and sufficient.

3.3 Three Conditions on Sequence Pairs

Before presenting the detailed conditions, we first define four relations for the unit blocks in a sequence pair.

- Given three unit blocks a_i, $a_j \in A$ and $c \notin A$, if c is between a_i and a_j in both sequences, for example (a_i c a_j, a_i c a_j). we call c *interrupts* a_i and a_j.
- Given two pairs of unit blocks a_i, $a_j \in A$ and b_i, $b_j \in B$, $A \neq B$. In the first sequence:

  $a_i \cdots a_j \cdots b_i \cdots b_j$, we call ($a_i$, a_j) and (b_i, b_j) as *separates* of each other.

  $a_i \cdots b_i \cdots a_j \cdots b_j$, we call ($a_i$, a_j) and (b_i, b_j) as *interleaves* of each other.

  $a_i \cdots b_i \cdots b_j \cdots a_j$, we call ($a_i$, a_j) as *covering* (b_i, b_j).
- Similarly, the *separate*, *interleave* and *cover* relations can be defined in the second sequence.

The first condition is referred to as *condition-1*:

> For any H-partitioned macro A, the permutation pair of A equals the H-Pair of A, similarly for any V-partitioned A, the permutation pair of A equals the V-pair of A.

The second condition is referred to as *condition-2*:

> Any two units a_i, $a_j \in A$ are not interrupted by a unit $c \notin A$.

Finally the third condition is referred to as *condition-3*:

> Any two pairs of unit blocks a_i, $a_j \in A$ and $b_{i'}$, $b_{j'} \in B$, $A \neq B$. (a_i, a_j) separates ($b_{i'}$, $b_{j'}$) in the first or second sequence.

Overall the three conditions are called *3-conditions*.

4 3-Conditions Are Necessary and Sufficient

4.1 3-Conditions Are Necessary

Given the sequence pair which does not satisfy condition-1, without loss of generality, we assume the permutation pair of an H-partitioned macro block A does not equal the H-pair of A. There must exist two sub-blocks a_i, $a_j \in A$, whose permutations in SP are different from those in the H-pair of A. For example, sub-block a_1 is left of a_2 in the H-partitioned A, the H-pair of A is (a_1 a_2 , a_1 a_2). On the other hand, the permutations of a_1, a_2 in the sequence pair is (a_1 a_2 , a_2 a_1). X, Y-Alignment preserve the topological relationships defined in the sequence pair, block a_1 will be above a_2 after the alignment and the shape of macro block A cannot be recovered. Therefore condition-1 is a necessary condition.

Given the sequence pair does not satisfy condition-2, there must exist two unit blocks a_i, $a_j \in A$ interrupted by a unit block $c \notin A$. The sequence pair will be either ($a_i \cdots c \cdots a_j$, $a_i \cdots c \cdots a_j$) or ($a_i \cdots c \cdots a_j$, $a_j \cdots c \cdots a_i$). In the first case, c is right of a_i while left of a_j, a_i and a_j may not be aligned in the x direction as shown in Fig. 9 (a). While in the second case, c is below a_i while above a_j, a_i and a_j may not be aligned in the y direction

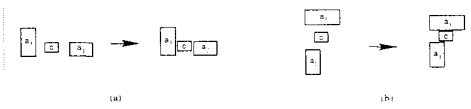


Figure 9: (a) Unit block c is right of a_i while left of a_j, a_i and a_j may not be aligned in the x direction. (b) Block c is below a_i while above a_j, a_i and a_j may not be aligned in the y direction.

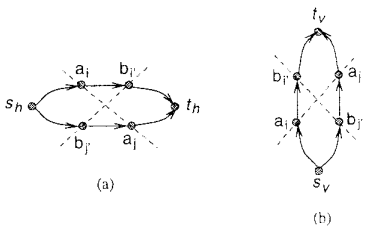


Figure 10: (a) In the horizontal graph G_h, if unit block $a_i \in A$ is left of unit $b_{i'} \in B$, $A \neq B$, and unit block $a_j \in A$ is right of unit $b_{j'} \in B$, a_i, a_j and $b_{i'}$, $b_{j'}$ are *H-crossed* to each other. (b) In the vertical graph G_v, if a_i is below $b_{i'}$, and a_j is above $b_{j'}$, a_i, a_j and $b_{i'}$, $b_{j'}$ are *V-crossed* to each other.

as shown in Fig. 9 (b). Therefore condition-2 is also a necessary condition.

In the following, we refer two unit blocks a_i, $a_j \in A$ as *a-pair*. Similarly two unit blocks $b_{i'}$, $b_{j'} \in B$ as *b-pair*, where $A \neq B$. If a_i is left of $b_{i'}$ while a_j is right of $b_{j'}$ as shown in Fig. 10 (a), we call *a*-pair and *b*-pair *H-crossing* each other. In X-Alignment(A), the right move of a_i may push unit $b_{i'}$ to the right. Then $b_{j'}$ has to be moved to the right along with $b_{i'}$, which may push unit a_j to the right. Again a_i is moved to the right along with a_j and so on ... , the x alignment might continue infinitely. When the similar situation happens in the vertical dimension as shown in Fig. 10 (b), we call *a*-pair and *b*-pair *V-crossing* each other, and Y-Alignment might continue infinitely. We can conclude the following fact:

Lemma 2 *If two pairs of unit blocks are H-crossed or V-crossed to each other, the shapes of the corresponding macro blocks may not be recovered after X, Y-Alignment.*

Given the sequence pair does not satisfy condition-3, there must exist *a*-pair and *b*-pair which do not separate each other in either sequence. Without loss of generality, we assume a_i is before a_j, and $b_{i'}$ is before $b_{j'}$ in the first sequence Γ_1. Then Γ_1 will be either $a_i\ b_{i'}\ a_j\ b_{j'}$, or $a_i\ b_{i'}\ b_{j'}\ a_j$. When the first sequence is $a_i\ b_{i'}\ a_j\ b_{j'}$, the second sequence Γ_2 is enumerated in Fig. 11. Since *a*-pair does not separate *b*-pair in Γ_2, the second sequence can not be the case (1.1.1), (1.1.2), (1.3.3), (1.3.4), (2.1.3), (2.1.4), (2.3.1) or (2.3.2). On the other hand, if the second sequence is the case (1.1.4), (1.3.1), (1.3.2), (2.1.1), (2.3.3) or (2.3.4), condition-2 will be violated. Therefore Γ_2 can only be case (1.1.3) or (2.1.2). The two corresponding sequence pairs are:

1. $(a_i\ b_{i'}\ a_j\ b_{j'}\ ,\ b_{i'}\ a_i\ b_{j'}\ a_j)$
2. $(a_i\ b_{i'}\ a_j\ b_{j'}\ ,\ a_j\ b_{j'}\ a_i\ b_{i'})$

When the first sequence Γ_1 is $a_i\ b_{i'}\ b_{j'}\ a_j$, the second sequence is enumerated in Fig. 12. Since *a*-pair does not separate *b*-pair in Γ_2, the second sequence can not be the case (1.1.1), (1.1.2), (1.3.3), (1.3.4), (2.1.3), (2.1.4), (2.3.1) or (2.3.2). On the other hand, if the second sequence is the case (1.1.3), (1.3.2), (2.1.2) or (2.3.3), condition-2 will be violated in the sequence pair. Therefore Γ_2 can only be case (1.1.4), (1.3.1), (2.1.1) or (2.3.4). The four corresponding sequence pairs are:

3. $(a_i\ b_{i'}\ b_{j'}\ a_j\ ,\ b_{i'}\ a_i\ a_j\ b_{j'})$
4. $(a_i\ b_{i'}\ b_{j'}\ a_j\ ,\ b_{j'}\ a_i\ a_j\ b_{i'})$
5. $(a_i\ b_{i'}\ b_{j'}\ a_j\ ,\ b_{i'}\ a_j\ a_i\ b_{j'})$
6. $(a_i\ b_{i'}\ b_{j'}\ a_j\ ,\ b_{j'}\ a_j\ a_i\ b_{i'})$

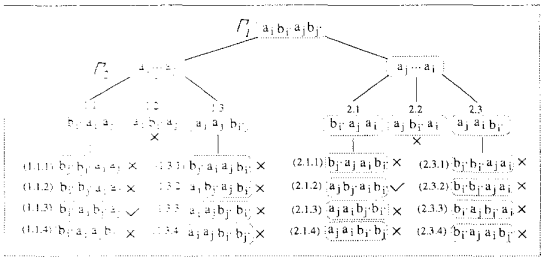


Figure 11: Given *a*-pair does not separate *b*-pair in either sequence, and the first sequence is $a_i\ b_{i'}\ a_j\ b_{j'}$. The second sequence Γ_2 is enumerated in this Figure. Since *a*-pair does not separate *b*-pair in Γ_2, the second sequence can not be the case (1.1.1), (1.1.2), (1.3.3), (1.3.4), (2.1.3), (2.1.4), (2.3.1) or (2.3.2). On the other hand, if the second sequence is the case (1.1.4), (1.3.1), (1.3.2), (2.1.1), (2.3.3) or (2.3.4), condition-2 will be violated. Therefore Γ_2 can only be case (1.1.3) or (2.1.2).

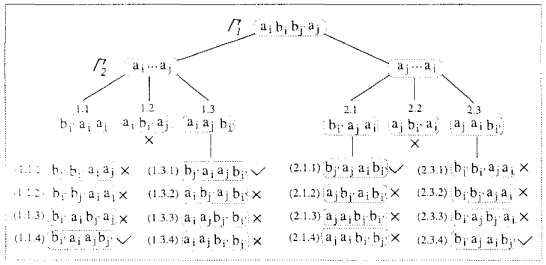


Figure 12: Given *a*-pair does not separate *b*-pair in either sequence, and the first sequence is $a_i\ b_{i'}\ b_{j'}\ a_j$. The second sequence Γ_2 is enumerated in this Figure. Since *a*-pair does not separate *b*-pair in Γ_2, the second sequence can not be the case (1.1.1), (1.1.2), (1.3.3), (1.3.4), (2.1.3), (2.1.4), (2.3.1) or (2.3.2). On the other hand, if the second sequence is the case (1.1.3), (1.3.2). (2.1.2) or (2.3.3), condition-2 will be violated. Therefore Γ_2 can only be case (1.1.4), (1.3.1), (2.1.1) or (2.3.4).

In each of the above six sequence pairs, *a*-pair and *b*-pair are either H-crossed or V-crossed to each other, the corresponding sequence pair is infeasible. Therefore condition-3 is a necessary condition.

Lemma 3 *The 3-conditions are necessary for a sequence pair to be feasible.*

Based on the similar analysis, we can derive the following property:

Lemma 4 *Given condition-3 is satisfied in the sequence pair, any two pairs of unit blocks a_i, $a_j \in A$ and $b_{i'}$, $b_{j'} \in B$, $A \neq B$, are neither H-crossed nor V-crossed to each other.*

4.2 3-Conditions Are Sufficient

Given 3-conditions are satisfied in the sequence pair, the following Lemma is true:

Lemma 5 *If a unit block $a_i \in A$ is left of a unit $b_{i'} \in B$, $A \neq B$, then no unit of A is right of a unit of B. Similarly, if a_i is below $b_{i'}$, no unit of A is above a unit of B.*

Lemma 5 can be proven by the contradiction. Let's assume that a_i is left of $b_{i'}$, and a unit block $a_j \in A$ is right of a unit $b_{j'} \in B$. If $a_i = a_j$ or $b_{i'} = b_{j'}$, we can easily derive that a_i interrupts $(b_{i'}, b_{j'})$ or $b_{i'}$ interrupts (a_i, a_j) in the sequence pair, which contradicts the assumption. Thus $a_i \neq a_j$, $b_{i'} \neq b_{j'}$. As such a-pair and b-pair are H-crossed to each other, then a-pair does not separate b-pair in either sequence. Again the assumption that sequence pair satisfies the 3-conditions is contradicted. Therefore Lemma 5 is true. Based on this property, we can sort the macro blocks in x and y direction, respectively:

x-order: macro block A is before macro block B if a unit of A is left of a unit of B;

y-order: macro block A is before macro block B if a unit of A is below a unit of B.

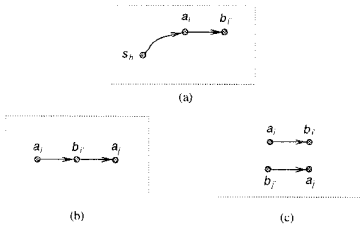


Figure 13: (a) In X-Alignment(A), unit block $a_i \in A$ is moved to the right, a unit $b_{i'} \in B$ is pushed to the right by a_i only when $b_{i'}$ is right of a_i and $B \neq A$. The right move of $b_{i'}$ may further affect the unit blocks of A if and only if: (b) a unit block $a_j \in A$ is right to $b_{i'}$, or (c) a unit block $a_j \in A$ is right of a unit block $b_{j'} \in B$.

In the following, we show that the shapes of macro blocks can be exactly recovered by applying X, Y-Alignment on each macro block in x and y-order, respectively. Based on the horizontal graph G_h, X-Alignment(A) is carried out, a unit $a_i \in A$ is moved right, the unit $b_{i'} \in B$ is pushed to the right by a_i only when $b_{i'}$ is right of a_i and $B \neq A$, as shown in Fig. 13 (a). The right move of $b_{i'}$ may further affect the unit blocks of A if and only if:

1. a unit block $a_j \in A$ is right of $b_{i'}$, or
2. a unit block $a_j \in A$ is right of another unit block $b_{j'} \in B$.

In the first case, $b_{i'}$ is right of a_i while left of a_j as shown in Fig. 13 (b). Then b_i interrupts a_i, a_j in the sequence pair, condition-2 is violated. In the second case, a_i, a_j and $b_{i'}$, $b_{j'}$ are H-crossed to each other as shown in Fig. 13 (c). Then a-pair does not separate b-pair in either sequence and condition-3 is violated. Due to the assumption that sequence pair satisfies the 3-conditions, neither case 1 nor case 2 will happen. In other words, the right move of a_i in x-align(A) will not affect any other unit of A. X-Alignment(A) exactly aligns the x coordinates of A.

On the other hand, when X-Alignment is carried out on macro B after X-Alignment(A), no unit of B can be left of a unit of A due to the x-order. As such, the right move of unit blocks in X-Alignment(B) will not affect macro block A. It implies that once X-Alignment(A) is carried out, the x coordinates of A will not be affected later. We can conclude the following Lemma:

Lemma 6 *The 3-conditions are sufficient for a sequence pair to be feasible.*

Followed by Lemma 3 and Lemma 6, we can conclude:

Theorem 3 *The 3-conditions are necessary and sufficient for a sequence pair to be feasible.*

5 Convex Rectilinear Block Packing

A rectilinear block is called *convex* if any two points in the block have a shortest Manhattan path inside the block, as shown in Fig. 14 (a). Otherwise the block is called *concave*, as shown in Fig. 14 (b). It can be observed that some packing of concave blocks can not be represented by the feasible sequence pair. For example, in the packing of Fig. 14 (c), block c is right of block a_1 while left of a_3, then c must interrupt a_1, a_3 in the corresponding sequence pair.

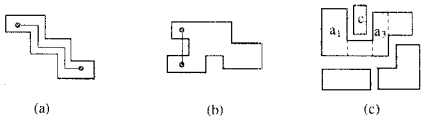


Figure 14: (a) Any two points of the convex rectilinear block have a shortest Manhattan path inside the block. (b) At least two points in the concave block have no shortest Manhattan path located inside the block. (c) The packing of concave blocks can not be representable by feasible sequence pairs.

Let Π denote an arbitrary convex block packing. By H- or V-partitioning the rectilinear macro blocks, Π becomes a rectangle packing. The corresponding sequence pair $SP(\Pi)$ can be easily derived, in which the sub-blocks are treated as individual unit blocks. Obviously condition-1 is satisfied in $SP(\Pi)$.

If condition-2 is not satisfied in the sequence pair, there exist two unit blocks a_i, $a_j \in A$ interrupted by a unit $c \notin A$. It can be easily derived that A has concave shape, which contradicts the convex assumption. Therefore condition-2 is also satisfied in $SP(\Pi)$.

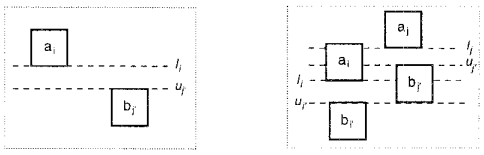


Figure 15: Let l_i and u_i denote the lower and upper boundary of unit a_i, respectively. Given sequence pair $SP(\Pi) = (a_i\ b_{i'}\ a_j\ b_{j'},\ b_{i'}\ a_i\ b_{j'}\ a_j)$, a_i is above $b_{i'}$ and a_j is above $b_{j'}$. Then in the packing Π, $l_i \geq u_{i'}$ and $l_j \geq u_{j'}$. (a) $l_i \geq u_{j'}$: block a_i is both left of and above $b_{j'}$. Then $SP(\Pi)$ can be transformed to $(a_i\ b_{i'}\ a_j\ b_{j'},\ b_{i'}\ b_{j'}\ a_i\ a_j)$. (b) $l_i < u_{j'}$: $l_j \geq u_{j'} > l_i \geq u_{i'}$, then $l_j > u_{i'}$. Block a_j is both right of and above $b_{i'}$ in the packing Π. $SP(\Pi)$ can be transformed to $(a_i\ a_j\ b_{i'}\ b_{j'},\ b_{i'}\ b_{j'}\ a_i\ a_j)$.

If the condition-3 is not satisfied in the sequence pair $SP(\Pi)$, there exist a_i, $a_j \in A$ and $b_{i'}$, $b_{j'} \in B$ such that a-pair does not separate b-pair in either sequence. According to the analysis of Fig. 11 and Fig. 12, a total of six non-symmetrical sequence pairs are possible. When the sequence pair is the first case, $SP(\Pi) = (a_i\ b_{i'}\ a_j\ b_{j'},\ b_{i'}\ a_i\ b_{j'}\ a_j)$, a_i is above $b_{i'}$ and a_j is above $b_{j'}$. Let l_i and u_i denote the lower and upper y coordinate of a_i, respectively. Then $l_i \geq u_{i'}$ and $l_j \geq u_{j'}$. If $l_i \geq u_{j'}$ as shown in Fig. 15 (a), a_i is both left of and above $b_{j'}$, $SP(\Pi)$ can be transformed as follows:

$$(a_i\ b_{i'}\ a_j\ b_{j'},\ b_{i'}\ a_i\ b_{j'}\ a_j) \Rightarrow (a_i\ b_{i'}\ a_j\ b_{j'},\ b_{i'}\ b_{j'}\ a_i\ a_j) \quad (1)$$

Otherwise $l_i < u_{j'}$ as shown in Fig. 15 (b), $l_j \geq u_{j'} > l_i \geq u_{i'}$, then $l_j > u_{i'}$. Block a_j is both right of and above $b_{i'}$ in the packing Π. Thus $SP(\Pi)$ can be transformed as follows:

$$(a_i\ b_{i'}\ a_j\ b_{j'},\ \ b_{i'}\ a_i\ b_{j'}\ a_j) \Rightarrow (a_i\ a_j\ b_{i'}\ b_{j'},\ \ b_{i'}\ b_{j'}\ a_i\ a_j). \quad (2)$$

Obviously the transformation will not cause any violation of 3-conditions, a-pair will separate b-pair in at least one sequence of $SP(\Pi)$, meanwhile the topology defined by the sequence pair is consistent with the packing Π. For case 2 through case 6. $SP(\Pi)$ can be similarly transformed. In such a way, a feasible sequence pair can be eventually achieved, that is

Lemma 7 *There always exists a feasible sequence pair corresponding to a packing of convex rectilinear blocks.*

For M convex macro blocks, each of them includes at most n rectangular sub-blocks, the optimal solution for convex rectilinear block packing can be found by exhausting the finite number $O((nM)!^2)$ of feasible sequence pairs.

When a convex macro block A is H-partitioned as shown in Fig. 16 (a), a_i denotes the i^{th} leftmost sub-block, the H-pair of A is $(a_1 \cdots a_i\ a_{i+1} \cdots a_m,\ \ a_1 \cdots a_i\ a_{i+1} \cdots a_m)$. When A is V-partitioned as shown in Fig. 16 (b), a_i denotes the i^{th} lowest sub-block, the V-pair of A is $(a_m \cdots a_{i+1}\ a_i \cdots a_1,\ \ a_1 \cdots a_i\ a_{i+1} \cdots a_m)$.

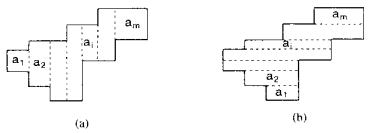


Figure 16: (a) Convex block A is H-partitioned where a_i is the i^{th} leftmost sub-block of A. The H-pair of A equals to $(a_1 \cdots a_i\ a_{i+1} \cdots a_m,\ \ a_1 \cdots a_i\ a_{i+1} \cdots a_m)$. (b) When A is V-partitioned, a_i is the i^{th} lowest sub-block of A. The V-pair A equals to $(a_m \cdots a_{i+1}\ a_i \cdots a_1,\ \ a_1 \cdots a_i\ a_{i+1} \cdots a_m)$.

6 Stochastic Search on Convex Block Packing

In the following, a stochastic search is applied to the optimization of convex block packing. Three sequence pair operations are defined to incrementally change the feasible sequence pair: *rotation*, Γ_1-*mutation*, and Γ_2-*mutation*.

The proofs of the Lemmas and Theorems stated in this Section are very comprehensive, and will not be addressed in this paper due to the limitation of paper length. All of the detail proofs can be found in [16].

6.1 Rotation

Rotation rotates a macro block by 90^0 in the clockwise direction as shown in Fig. 17. Given macro block $A = \{a_1, a_2, \cdots, a_n\}$, *rotate*$(A)$ switches the height with the width for each unit block of A. The sequence pair is accordingly changed by switching unit a_i with a_{m+1-i}, $i \in [1,\ n]$, in the first sequence (90^0 to 180^0, 270^0 to 0^0) or the second sequence (0^0 to 90^0, 180^0 to 270^0). Rotation takes $O(n)$ time, which is close to constant time, and has the following property:

Lemma 8 *When a rotation is carried out on a macro block represented in a feasible sequence pair, the resultant sequence pair remains feasible.*

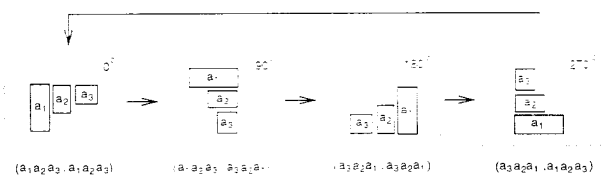


Figure 17: A macro block $A = \{a_1, a_2, a_3\}$ is rotated from 0^0 through 90^0 and 180^0 to 270^0 in the clockwise direction. *rotate*(A) switches the height and width for each unit block of A, and changes the sequence pair by switching a_i with a_{m+1-i} in the first or second sequence.

6.2 Γ_1-Mutation And Γ_2-Mutation

Γ_1-Mutation switches two adjacent unit blocks in the first sequence: $\cdots\ \underline{a\ b}\ \cdots \Rightarrow \cdots\ \underline{b\ a}\ \cdots$, in which $a \in A$, $b \in B$, $A \neq B$. Γ_2-Mutation is similarly carried out on the second sequence. Both operations take constant time. Since two mutations are symmetrical, we will only discuss Γ_1-mutation.

Given Γ_1-mutation is carried out on a feasible sequence pair, the permutation pair of every macro block will be preserved in the resultant sequence pair, as such condition-1 is satisfied after the operation. In the following, we assume $A = \{a_1\ a_2 \cdots a_m\}$ and $B = \{b_1\ b_2 \cdots b_n\}$. By relabeling the sub-blocks of A and B, a_i and b_i are the i^{th} sub-block of A and B in the first sequence Γ_1, respectively.

Lemma 9 *When Γ_1-mutation is carried out on a feasible sequence pair: $\cdots\ \underline{a_i\ b_j}\ \cdots \Rightarrow \cdots\ \underline{b_j\ a_i}\ \cdots$, the resultant sequence pair may violate condition-2 or condition-3 only when $a_i = a_m$, $b_j = b_1$. And the violation only happens on a_m and b_1.*

As we can see, an infeasible solution can be generated by a mutation. During the stochastic search, the infeasible solutions can be avoided by simply cancelling the operation. However the continuity of the local search may be destroyed, the optimal solution may not be reachable. Based on above analysis, the infeasible solution can only be generated when a mutation is carried out on $\cdots a_1\ a_2 \cdots \underline{a_m\ b_1}\ b_2 \cdots b_n \cdots$, where switching a_m with b_1 may cause the violation of condition-2 or condition-3. Due to this fact, we can develop a very simple procedure called *adaptation* to adapt the infeasible solution into a new feasible solution. In such a way, the continuous search of the feasible solution space can be guaranteed.

6.3 Adaptation

Given that Γ_1-mutation generates an infeasible solution, the original first sequence Γ_1 must be $\cdots a_1 \cdots a_2 \cdots \underline{a_m\ b_1} \cdots b_2 \cdots b_n \cdots$. For example,

$$\Gamma_1 = a_1\ c\ a_2\ d\ e\ a_3\ f\ \underline{a_4\ b_1}\ g\ b_2\ h\ i\ j\ b_3 \quad (3)$$

We define *squeeze-to-right* (Γ_1, A) as follows: applying Γ_1-mutation on a_3 and its right neighbor until a_3 is left adjacent to a_4, then applying Γ_1-mutation on a_2 and its right neighbor until a_2 is left adjacent to a_3, and so on ...:

$$\Gamma_1 = c\ d\ e\ f\ \underline{a_1\ a_2\ a_3\ a_4}\ b_1\ g\ b_2\ h\ i\ j\ b_3 \quad (4)$$

Based on the analysis of the mutations, it can be guaranteed that the intermediate sequence pairs generated during squeeze-to-right (Γ_1, A) are always feasible. Similarly we define *squeeze-to-left* (Γ_1, B) as follows: applying Γ_1-mutation on b_2 and its left neighbor until b_2 is right adjacent to b_1, then applying Γ_1-mutation on b_3 and its left neighbor until b_3 is right adjacent to b_2:

$$\Gamma_1 = c\ d\ e\ f\ a_1\ a_2\ a_3\ a_4\ \underline{b_1\ b_2\ b_3}\ g\ h\ i\ j \quad (5)$$

The intermediate sequence pairs generated during squeeze-to-left (Γ_1, B) are always feasible.

After the squeeze operations, the unit blocks of both A and B are consecutive in the first sequence. Then adaptation swaps A and B : $\cdots$ $\underbrace{a_1\ a_2\ \cdots a_m}$ $\underbrace{b_1\ b_2\ \cdots b_n}$ $\cdots$ $\Rightarrow$ $\cdots$ $\underbrace{b_1\ b_2\ \cdots b_n}$ $\underbrace{a_1\ a_2\ \cdots a_m}$ $\cdots$. Obviously no violation of 3-conditions can occur during the swapping, and the resultant sequence pair is feasible. In the above example, the first sequence is adapted to:

$$\Gamma_1 = c\ d\ e\ f\ b_1\ b_2\ b_3\ a_1\ a_2\ a_3\ a_4\ g\ h\ i\ j \qquad (6)$$

In $O(m+n)$ time, adaptation generates a new feasible sequence pair which is different from the original one.

Lemma 10 *The resultant infeasible sequence pair after a mutation can always be transformed to a feasible sequence pair by the adaptation.*

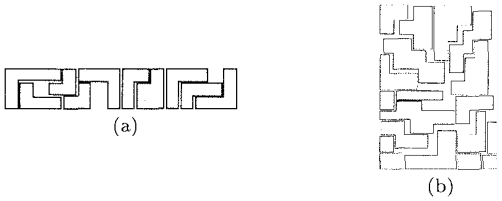
As such, the local moves of the stochastic search are completed. Each local move (a rotation or a mutation followed by an adaptation) takes the time proportional to the number of unit blocks in a macro block, which is close to constant time. In addition, each local move generates a feasible sequence pair. In such a way, the feasible solution space can be continuously searched by the local moves. On the other hand, the following Theorem guarantees that *Rotation + Mutations + Adaptation = Exhaustive Search.*

Theorem 4 *The optimal solution can always be reachable through a finite steps of the rotation and mutations followed by the adaptation.*

7 Experimental Results and Conclusion

7.1 Packing Results

We have implemented the algorithm proposed in this paper using C language and tested it on SUN SPARC 20 workstation. The data are generated randomly. Figure 18 (a) gives the packing of ten L-shaped blocks, which takes about two minutes, the total packing area is 1.09 times of the total block area. On the other hand, Figure 18 (b) reports the packing result of 19 convex rectilinear blocks, which takes about 35 minutes, the total packing area is 1.13 times of the total block area.


(a)

(b)

Figure 18: (a) The packing of ten L-shaped blocks, which takes about two minutes, the total packing area is 1.09 times of total block area. (b) The packing result of 19 convex rectilinear blocks, which takes about 45 minutes, the total packing area is 1.13 times of total block area.

7.2 Concluding Remarks

In this paper, for the first time, we solve the arbitrary shaped rectilinear block packing problem. Rectilinear macro blocks are partitioned into a set of rectangular sub-blocks, each of them is individually represented as a unit block in the sequence pair. The feasible solution space is defined. Three conditions on the sequence pair are derived, which are necessary and sufficient for a sequence pair to be feasible. Furthermore it is proven that there always exists a feasible sequence pair corresponding to a packing of convex rectilinear blocks. Based on this fact, a stochastic search is applied on the optimization of convex block packing. Local moves are defined to search the feasible solution space both continuously and exhaustively.

References

[1] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-Packing-Based Module Placement," in *IEEE/ACM International Conf. on Computer Aided Design*, (San Jose, CA), pp. 472–479, November 1995.

[2] B. S. Baker, E. G. Coffman, and R. L. Rivest, "Orthogonal Packings in Two Dimensions," *SIAM J. Comput.*, no. 4, pp. 846–855, 1980.

[3] L. Sha and R. W. Dutton, "An Analytical Algorithm for Placement of Arbitrarily Sized Rectangular Blocks," in *Proc. 22th ACM/IEEE Design Automation Conf.*, pp. 602–608, 1985.

[4] A. Alon and U. Ascher, "Model and Solution Strategy for Placement of Rectangular Blocks in the Euclidean Plane," *IEEE Trans. on CAD*, no. 3, pp. 378–386, 1988.

[5] R. Otten, "Automatic Floorplan Design," in *Proc. of 19th ACM/IEEE Design Automation Conf.*, pp. 261–267, 1982.

[6] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan Design," in *Proc. of 23rd ACM/IEEE Design Automation Conf.*, pp. 101–107, June 1986.

[7] W. W.-M. Dai, B. Eschermann, E. S. Kuh, and M. Pedram, "Hierarchical Placement and Floorplanning in BEAR," *IEEE Trans. Computer-Aided Design*, vol. CAD-8, no. 12, pp. 1335–1349, 1989.

[8] T. C. Wang and D. F. Wong, "An Optimal Algorithm for Floorplan Area Optimization," in *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 180–186, 1990.

[9] T. C. Wang and D. F. Wong, "A Graph Theoretic Technique to Speed up Floorplan Area Optimization," in *Proc. 29th ACM/IEEE Design Automation Conf.*, pp. 62–68, 1992.

[10] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module Placement on BSG-Structure and IC Layout Applications," in *IEEE/ACM International Cof. on Computer Aided Design*, (San Jose, CA), pp. 484–491, November 1996.

[11] T. Chang Lee, "A Bounded 2D Contour Searching Algorithm for Floorplan Design with Arbitrarily Shaped Rectilinear and Soft Modules," in *Proc. 30th ACM/IEEE Design Automation Conf.*, (Dallas, TX), pp. 525–530, June 1993.

[12] M. Kang and W. W.-M. Dai, "General Floorplanning with L-shaped, T-shaped and Soft Blocks Based on Bounded Slicing Grid Structure," in *Proc. of Asia and South Pacific Design Automation Conf. 1997*, (Chiba, Japan), pp. 265–270, Feburary 1997.

[13] M. Kang and W. W.-M. Dai, "Topology Constrained Rectilinear Block Packing for Layout Reuse," in *International Symposium of Physical Design*, (Monterey, CA), pp. 179–186, April 1998.

[14] J. Xu and C. Kuan Cheng, "Rectilinear Block Placement Using Permutation-pair," in *International Symposium of Physical Design*, (Monterey, CA), pp. 173–178, April 1998.

[15] S. Nakatake, M. Furuya, and Y. Kanitani, "Module Placement on BSG-Structure with Pre-Placed Modules and Rectilinear Modules," in *Proc. 1998 Asia and South Pacific Design Automation Conf.*, (Yokohama, Japan), pp. 571–576, January 1998.

[16] M. Z. Kang, *Floorplanning for Deep Submicron VLSI Design*. PhD thesis, Univ. of California at Santa Cruz, Santa Cruz, CA, June 1998.