On Handling Arbitrary Rectilinear Shape Constraint [*]

Xiaoping Tang
Cadence Design Systems
San Jose, CA 95134 USA
xtang@cadence.com

Martin D.F. Wong
University of Illinois
Urbana, IL 61801 USA
mdfwong@uiuc.edu

Abstract

Non-rectangular (rectilinear) shape occurs very often in deep submicron floorplanning. Most previous algorithms are designed to handle only convex rectilinear blocks. However, handling concave rectilinear shape is necessary since a simple "U" shape is concave. A few works could address concave rectilinear block explicitly. In [2], a necessary and sufficient condition of feasible sequence pair is proposed for arbitrary rectilinear shape in terms of constraint graph. However, no constraint is imposed on sequence pair representation itself. The search for feasible sequence pair mainly depends on the simulated annealing, which implies unnecessary inefficiency. In many cases, it takes very long time or even is unable to find the feasible placement. Furthermore, it takes $O(n^3)$ runtime to evaluate each sequence pair, which leaves much space for improvement. In this paper, we propose a new method to handle arbitrary rectilinear shape constraint based on sequence pair representation. We explore the topological property of feasible sequence pair, and use it to eliminate lots of infeasible sequence pairs, which implies speeding up the convergence of simulated annealing process. The evaluation of a sequence pair is based on longest common subsequence computation, and achieves significantly faster runtime ($O(mn \log\log n)$ time where m is the number of rectilinear-shape constraints, n is the number of rectangular blocks/subblocks). The algorithm can handle fixed-frame floorplanning and min-area floorplanning as well.

1. Introduction

Floorplanning is the early stage of physical design and determines overall chip performance. The hierarchical approach to handle continuously increasing complexity of VLSI circuits and the wide use of IP block make floorplanning even more important. The past years is the age for inventing floorplanning representation. For slicing structure, there are binary tree[13] and normalized Polish expression[19]. For non-slicing structure, many representations are invented recently, such as topology representation (BSG[11], sequence pair[10], TCG[7]), packing representation(O-tree[3], B*-tree[1]), and mosaic representation (CBL[4], Q-sequence[16], twin binary tree[22], twin binary sequence[24]).

In a real-world floorplanning problem, additional constraints are imposed on subsets of blocks for users' purpose. Different designs have different requirements. Thus it is important and useful to allow users to specify placement constraints during floorplanning.

With the advent of deep submicron technology, integrated circuit blocks are often not rectangular. Non-rectangular (rectilinear) shaped blocks are introduced to facilitate the usage of chip area and improve the blocks' connectivity.

Rectilinear shape has been studied extensively in the literature. Rectilinear blocks can be categorized into two types of blocks: convex rectilinear blocks and concave rectilinear blocks. A rectilinear block is defined as convex if any two points in the block have a shortest Manhattan path inside the block. Otherwise, the block is concave. L/T-shape is a special case of convex rectilinear shape. Most floorplan representations have been extended to handle rectilinear shape, such as slicing structure[23], BSG[5, 15], sequence pair[6, 2], O-tree[14], CBL[9], B*-tree[20], and TCG[8]. Each rectilinear block is partitioned into a set of rectangular subblocks. The relationship between the subblocks in a rectilinear block is represented in an encoding scheme or in additional constraint. Most of the algorithms are designed to handle only convex rectilinear blocks. However, handling concave rectilinear is necessary. For example, a very simple "U" shape is concave rectilinear. Only a few works could address concave rectilinear explicitly. In [2], a necessary and sufficient condition of feasible sequence pair is proposed for arbitrary rectilinear shape in terms of constraint graph. However, no constraint is imposed on sequence pair representation itself. The search for feasible sequence pair mainly depends on the simulated annealing, which implies unnecessary inefficiency. In many cases, it takes very long time or even is unable to find the feasible placement. Furthermore, it takes $O(n^3)$ runtime to evaluate each sequence pair, which leaves much space for improvement. Thus handling concave rectilinear shape constraint remains to be a problem.

We observe that handling concave rectilinear shape is inherently dimension dependent, i.e., exploring topology constraint on floorplan representation alone can not guarantee a feasible placement. However, topology constraint can be imposed on representation for eliminating lots of infeasible representations, reducing solution space, and speeding up the convergence of search process. Thus we use the topology representation, sequence pair, to study the problem. We first investigate the topology property of feasible sequence pair satisfying rectilinear shape constraints. Then only the sequence pairs which satisfy the topology property are generated and evaluated. Since the feasibility of a sequence pair depends on the actual dimension of blocks, it is impossible to search in solution space including only feasible sequence pairs. We use simulated annealing to search for optimal floorplan satisfying given constraints. Another observation we made is that perturbation in simulated annealing should distribute the probability evenly across all solutions in the solution space, and the feasible solutions should reach each other through a sequence of "moves". The property of even distribution and reachability is very important for simulated annealing to perform well. Otherwise, it may be entrapped in a local optimal. To evaluate infeasible solution, a common used technique is to add a penalty term in cost function. However, the factor of penalty is hard to decide: too large may affect the smoothness of simulated annealing; and too small may be useless in guiding simulated annealing to converge to feasible solution. In contrast to traditional approach, we use a novel cost function which unifies the evaluation of feasible and infeasible sequence pairs.

Tang and Wong[17] presented an $O(n \log\log n)$ algorithm to evaluate a sequence pair based on computing the longest common subsequence (LCS) of two weighted sequences, and showed excellent experimental results. The LCS method does not need to construct constraint graph. Since constraint graph is more general than LCS, people may think there exists limitation for LCS, e.g., to handle rectilinear shape constraints. In the paper, we employ LCS computation and demonstrate its capability in dealing with constraints. Further, the algorithm evaluates each sequence pair to obtain a floorplan in $O(mn \log\log n)$ where m is the number of rectilinear-shape constraints and n is the number of rectangular blocks/subblocks, which is significantly faster than the original $O(n^3)$ method operating on constraint graph.[1] The method is suitable to both fixed-frame and min-area floorplanning optimization. Our experimental results on MCNC benchmark for block placement show the promise of the method.

2. Preliminary

A sequence pair is a pair of sequences of n elements representing a list of n blocks. The two sequences specify the geometric relations (such as left-of, right-of, below, above) between each pair of blocks as follows:

$$(\ldots b_i \ldots b_j \ldots, \ldots b_i \ldots b_j \ldots) \quad \Rightarrow \quad b_i \text{ is to the left of } b_j \tag{1}$$

$$(\ldots b_j \ldots b_i \ldots, \ldots b_i \ldots b_j \ldots) \quad \Rightarrow \quad b_i \text{ is below } b_j \tag{2}$$

[*]This work was partially supported by the National Science Foundation under grants CCR-0244236 and CCR-0306244.

[1]Some recent results show that this can be done in $O(mn^2)$ time.

It is shown in [17] that the coordinates of blocks and the total width and height of floorplan can be obtained by computing longest common subsequence in terms of the two sequences. Given a sequence pair (X,Y), the total width of floorplan equals the length of the longest common subsequence of X and Y where weights are blocks' widths. Analogously, the total height of floorplan is determined by dealing with the longest common subsequence of X^R and Y where X^R is the reverse of X and weights are blocks' heights. The coordinates of a block are calculated as follows. Let (X,Y)=(X_1bX_2, Y_1bY_2) and $lcs(X,Y)$ denote the length of the longest common subsequence of X and Y. Then $(X^R,Y) = (X_2^R b X_1^R,\ Y_1bY_2)$. The x-coordinate of block b equals $lcs(X_1,Y_1)$ with blocks' widths as weights. The y-coordinate of block b is $lcs(X_2^R,Y_1)$ with blocks' heights as weights. In addition, all the computations of blocks' x/y coordinates can be integrated into a single longest common subsequence computation for a sequence pair.

3. Sequential Rectilinear Shape

For better presentation and easy understanding, we classify rectilinear shape into two categories: **sequential** and **non-sequential**. we first discuss how to handle sequential rectilinear shape in the section, and then generalize to non-sequential in the next section.

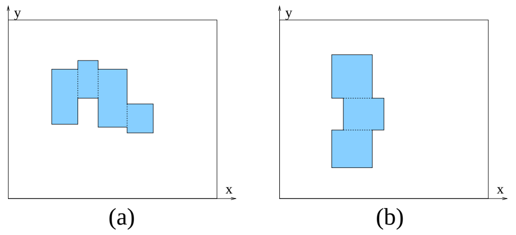

Figure 1: (a) H-sequential rectilinear block; (b) V-sequential rectilinear block.

A rectilinear block is said to be **H-sequential** if no vertical line cuts the block into more than two parts. Analogously, we define a rectilinear block to be **V-sequential** if no horizontal line cuts the block into more than two parts. **Sequential** rectilinear consists of H-sequential and V-sequential. As we can see in Figure 1, H(V)-sequential rectilinear can be divided into a sequence of subblocks using vertical(horizontal) cuts. It is obvious that convex rectilinear is both H-sequential and V-sequential. Some (but not all) concave rectilinear blocks are H/V-sequential.

In the following, we mainly discuss how to handle H-sequential rectilinear. Then we show V-sequential rectilinear can be handled similarly.

Rectilinear block is partitioned into a set of rectangular subblocks. Each rectangular subblock is treated as a block in floorplan representation. The rectilinear shape is recovered during placement.

Suppose that a rectilinear block b is H-sequentialized into a set of rectangular subblocks b_i, $i = 1,2,...,k$. Thus the subblocks b_i must follow the adjacency-order condition as stated in Theorem 1.

Theorem 1. *If a rectilinear block b is H-sequentialized into k subblocks, b_i, $i = 1,2,...,k$, then the corresponding sequence pair representation must satisfy that $(X,Y) = (...b_1...b_2\ ......b_i......b_k...,\ ...b_1...b_2......b_i......b_k...)$ and there is no common elements between b_i and b_{i+1} in (X,Y), $1 \le i \le k-1$.*

In addition, if there exist multiple H-sequential rectilinear blocks, the multiple sets of subblocks must satisfy the noncrossing condition as stated in Theorem 2.

Theorem 2. *Given any two rectilinear blocks, a (H-sequentialized into a_i, $i = 1,...,k$) and b (H-sequentialized into b_j, $j = 1,...,k'$), the corresponding sequence pair representation must satisfy that no relative position like $(...a_i...b_j...b_{j'}...a_{i'}...,...b_j...a_i...a_{i'}...b_{j'}...)$ appears in sequence pair (X,Y).*

If a sequence pair satisfies the condition given in Theorem 1, the subblocks belonging to a rectilinear block can possibly be abutted one by one in x direction and then be aligned in y direction to recover the original shape. An example is illustrated in Figure 2. Note that the alignment operation does not change the topological relation specified by sequence pair. Let us determine a reference subblock

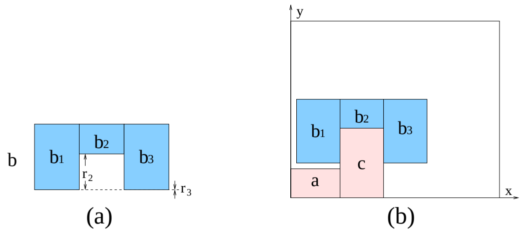

Figure 2: (a) The original shape of rectilinear block b. (b) Sequence pair (X,Y)=$(b_1\ a\ b_2\ c\ b_3,\ a\ b_1\ c\ b_2\ b_3)$ satisfies the condition in Theorem 1. Subblocks b_1, b_2 and b_3, can be aligned to recover the original shape of b. Note that there is no common element between b_1 and b_2 (or between b_2 and b_3) in (X,Y).

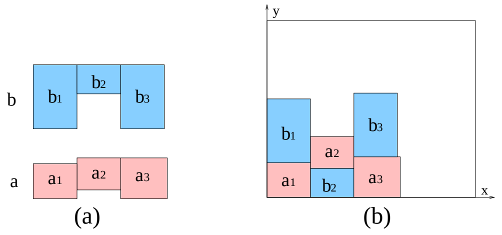

Figure 3: (a) The original shapes of a and b. (b) The two sets of subblocks (a_i and b_i, $i = 1,2,3$) can not be aligned to recover the original shape although they satisfy the adjacency-order condition in sequence pair representation. The sequence pair is $(X,Y) = (b_1\ a_1\ a_2\ b_2\ b_3\ a_3,\ a_1\ b_1\ b_2\ a_2\ a_3\ b_3)$.

(say b_1 for convenience), and then the relative position of each subblock in y direction is r_i, $i = 1,...,k$ (surely $r_1 = 0$). Let $x(b_i)(y(b_i))$ be the $x(y)$ coordinate of b_i, and $w(b_i)(h(b_i))$ be the width(height) of b_i, $i = 1,...,k$ respectively. Thus the alignment operation is performed as follows:

$$y_{max} \leftarrow max\{y(b_i) - r_i | i = 1,2,...,k\}$$
$$y(b_i) \leftarrow max(y(b_i), y_{max} + r_i) \quad \forall i \in \{1,2,...,k\}$$

Theorem 2 prevents the situation as shown in Figure 3. The two sets of rectangular subblocks (a_i and b_i, $i = 1,2,3$) can not be assembled to original rectilinear shape because a_i and b_i are crossing, although both a_i and b_i satisfy the adjacency-order condition as stated in Theorem 1.

The two necessary conditions (Theorem 1 and Theorem 2) guarantee the topological relation of subblocks for rectilinear blocks. It should be noted that we may need to adjust the x coordinate of the first subblock of an H-sequential rectilinear block to make them tightly abutted, referring to Figure 2. The adjustment is performed as follows:

$$x(b_1) \leftarrow x(b_k) - \sum_{i=1}^{k-1} w(b_i)$$

However, the two conditions are only necessary, but not sufficient. The feasibility of constrained sequence pair is actual dimension dependent. For example, referring to Figure 4(a), the sequence pair $(b_1\ a\ b_2\ c\ b_3, a\ b_1\ c\ b_2\ b_3)$ satisfies the conditions, but the constrained subblocks b_1,b_2 and b_3 can not be assembled to recover the original shape because c is wider than b_2 and subblocks b_i can not be tightly abutted. We address this issue by adding a "dummy" block at the right as shown in Figure 4(a) and re-evaluate the sequence pair. If the rectilinear shape can not be recovered, then the dummy block will be pushed out of the frame as shown in Figure 4(b). The out-of-frame value would measure the violation. Formally, the dummy block is determined as follows. Given a problem that k subblocks $\{b_i | i = 1,2,...,k\}$ of a rectilinear block b be in a frame $W \times H$, where b_i has dimension $w(b_i) \times h(b_i)$ respectively. Let δ denote the dummy block and $x(b_i)$ be the x coordinate of subblock b_i. The width of δ is

$$w(\delta) = W - x(b_k) - w(b_k)$$

and the height of δ is $h(\delta) = 0$.

Note that the alignment operation may introduce overlap among blocks/subblocks. We will re-evaluate the sequence pair to resolve overlap. The detail will be discussed in Section 5. Figure 4(b)

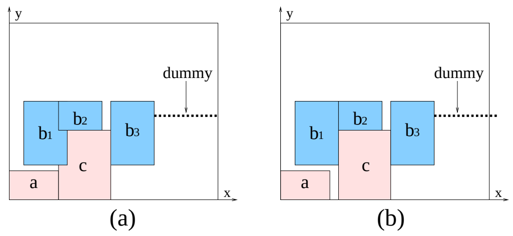

Figure 4: (a) Alignment operation is performed and Dummy block is introduced at the right of a rectilinear block. (b) The result after re-evaluating the sequence pair. In violation of rectilinear shape constraint, the dummy block is pushed out of frame. The out-of-frame value would measure the violation.

actually shows the result after resolving overlap. For V-sequential rectilinear shape, we will analogously deal with the two sequences (X^R,Y) and align the subblocks in x direction. It can be handled similar to H-sequential rectilinear.

4. Arbitrary Rectilinear Shape

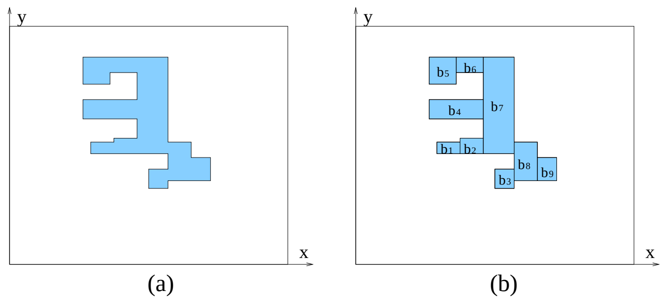

Figure 5: (a) A non-sequential rectilinear block; (b) it is H-partitioned into four sequences of H-sequential subblocks, (b_3,b_8,b_9), (b_1,b_2,b_7,b_8), (b_4,b_7), and (b_5,b_6,b_7).

A non-sequential rectilinear shape can not be partitioned into a sequence of H-sequential or V-sequential subblocks. However, it can be partitioned into multiple sequences of H-sequential or V-sequential subblocks. A rectilinear block is said to be H-partitioned if it is partitioned into multiple sequences of H-sequential subblocks. Analogously, a rectilinear block is said to be V-partitioned if it is partitioned into multiple sequences of V-sequential subblocks. Let us consider an example shown in Figure 5. The rectilinear block is H-partitioned into four sequences of H-sequential subblocks, $s_1 = (b_3,b_8,b_9)$, $s_2 = (b_1,b_2,b_7,b_8)$, $s_3 = (b_4,b_7)$, and $s_4 = (b_5,b_6,b_7)$ in bottom-up order. Any two adjacent sequences have at least one subblock in common, for example, s_1 and s_2 have a common subblock b_8, s_3 and s_4 have a common subblock b_7. Each sequence of subblocks is constrained similar to an H-sequential rectilinear shape (the alignment operation in y direction will be different). Thus the sequences must satisfy the adjacency-order condition and non-crossing condition stated in Theorem 1 and Theorem 2 respectively. In addition, the sequences, s_i, $1 \le i \le 4$, need to follow the bottom-up order. In other words, the sequence (b_1,b_4,b_5) should be treated as constrained by a V-alignment constraint. The only difference is that the subblocks $\{b_1,b_4,b_5\}$ are not abutted with each other. Thus the sequence of subblocks must appear in sequence pair like $(X^R,Y) = (...b_1...b_4....b_5...,...b_1...b_4...b_5...)$. So do the sequences (b_1,b_4,b_6) and (b_3,b_7). All these conditions together guarantee the topological relation between subblocks to recover the original rectilinear shape. We add dummy blocks to keep the relative positions.

For better manipulation of data, we use an orthogonal link list (called **O-list**) to represent the partition of the rectilinear block. An O-list consists of a set of H-lists and a set of V-lists as shown in Figure 6. Each H-list contains the sequence of subblocks to be aligned horizontally, and each V-list contains the sequence of subblocks to be aligned vertically. For H-partitioned subblocks, each H-list is treated as an H-sequential constraint, requiring to satisfy the adjacency-order condition. Any two adjacent H-lists are required to satisfy the non-crossing condition. Each V-list is treated as a V-sequential constraint, requiring to satisfy the order condition.

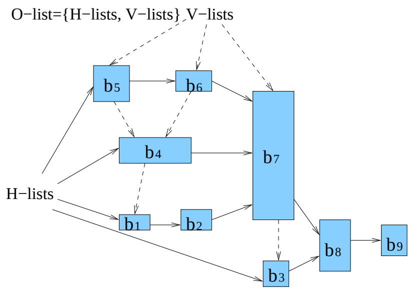

Figure 6: The orthogonal link list, O-list, is {H-lists, V-lists}.

5. Algorithm

Note that the calculations of x and y coordinates of all the blocks are done independently by evaluating (X,Y) and (X^R,Y) respectively. In the following we mainly describe the algorithm to evaluate (X,Y) in the presence of constraints since the evaluation of (X^R,Y) can be done similarly.

The underline engine of our algorithm is based on computing longest common subsequence presented in [17] with alignment operation plugged in. Since the alignment operation may introduce overlap among blocks/subblocks, we need to re-evaluate the sequence pair to resolve overlap. One iteration of LCS computation may not align all subblocks imposed by rectilinear-shape constraints, because alignment operations for different rectilinear shapes are affected by each other. However, since there is no crossing between rectilinear-shape constraints as stated in Theorem 2, rectilinear shapes can be kept in bottom-up order in y direction. First iteration of LCS computation aligns the bottommost rectilinear shape. Another iteration will align at least another set of subblocks related to a rectilinear-shape constraint, and so on. Thus at most $m+1$ iterations are needed, where m is the total number of rectilinear-shape constraints. On the other hand, if after $m+1$ iterations, still not all subblocks are aligned for all rectilinear shapes, then there must exist crossing between rectilinear shapes (violating the condition in Theorem 2). In this way, our algorithm can detect whether crossing exists. [2]

Let $lcs'(X,Y)$ denote the return value of the algorithm, i.e. the length of the longest common subsequence with alignment operation and the dummy blocks introduced by constraints. Consequently, $lcs'(X,Y) \ge W$, the width of the frame.

Analogously we can evaluate (X^R,Y) to compute the y coordinates of blocks/subblocks. Similarly $lcs'(X^R,Y) \ge H$, the height of the frame. The time complexity of the algorithm is stated in the following theorem.

Theorem 3. *The algorithm evaluates a sequence pair in $O(mn \log\log n)$ time where m is the total number of rectilinear shapes and n is the number of block/subblocks.*

If a sequence pair is feasible, then the placement produced by the sequence pair must be contained within the given frame. Thus, we have the following necessary and sufficient condition.

Theorem 4. *A sequence pair (X,Y) is feasible if and only if $lcs'(X,Y) = W$ and $lcs'(X^R,Y) = H$.*

In real-world floorplanning on fixed die-size, we not only place all blocks/subblocks within the given frame, but also consider some other objectives, such as minimizing wire length. Floorplanning is an NP-hard problem. We use simulated annealing to search for an optimal floorplan.

The following operations are used to generate a neighbor sequence pair in simulated annealing.

Move1 is to move a block/subblock in the first sequence X from the original position to a destination position. Move1 still maintains the relative ordering of other blocks/subblocks. Note that Move1 does not change the second sequence Y. The destination of a moved unconstrained block or rectilinear-constrained subblock is within an interval. It is easy to compute the intervals and pick a random destination within the intervals. Obviously, Move1 can be done in linear time.

Move2 is similar to Move1 except that it operates on the second sequence Y.

[2]Recall that Bellman-Ford algorithm, which computes shortest path, uses $|V|$ iterations to detect negative cycle.

Rotation is to rotate an unconstrained block (e.g. exchange the width and height of the block). Rotation does not change the sequence pair representation. This operation can be done in constant time.

Flip is used to change the orientation of a rectilinear shape. If a rectilinear shape is H-partitioned to a set of subblocks, then Flip operation will cause the shape to be V-partitioned, and vice versa. The operation on sequence pair is to reverse the order of the set of subblocks in the first sequence, and we also need to update the O-list of the rectilinear shape. Flip can be done in linear time.

Note that if a sequence pair (X,Y) is infeasible then $lcs'(X,Y) > W$ or $lcs'(X^R,Y) > H$. Therefore $lcs'(X,Y) - W$ and $lcs'(X^R,Y) - H$ measure the violation in x and y direction respectively. Naturally, we use the following cost function

$$C = lcs'(X,Y) \cdot lcs'(X^R,Y) + \lambda L$$

where L is wire length, and λ is coefficient for balancing the two factors. The cost function unifies the evaluation of both feasible and infeasible sequence pairs.

6. Reachability

The solution space we explore in simulated annealing includes only sequence pairs that satisfy the ordering condition imposed by Theorem 1. We relax the condition imposed by Theorem 2, and let the algorithm to evaluate it as mentioned above. We can prove within the solution space any sequence pair can reach any other sequence pair through a sequence of move operations. Given any two different sequence pairs in the solution space, one can be changed to the other as follows. First, we use flip operation to make the ordering of constrained subblocks be the same in the two sequences. Then, one sequence can be “sorted” into the other via move1/move2 operations. Note that move1/move2 does not change the relative ordering of constrained subblocks. We need at most m flip operations. Sorting one sequence requires at most n move1/move2 operations. In total, at most $m+n$ operations are needed to transform a sequence pair to another in the solution space. Therefore, the diameter of solution space is at most $m+n$ (diameter is the maximum distance between any two solutions where the distance between two solutions is the minimum number of moves transforming one to the other).

7. Experimental Results

Table 1: Min-area results of floorplanning with rectilinear shapes.

circuit	block	constraints recti.	constraints subblock	time (s)	area (mm^2)	dead space
apte	9	2	6	3	47.08	1.1%
xerox	10	2	6	6	20.72	6.6%
hp	11	2	6	8	9.342	5.5%
ami33-1	33	3	9	23	1.282	9.8%
ami33-2	33	4	12	25	1.268	8.8%
ami49-1	49	3	9	30	38.28	7.4%
ami49-2	49	4	12	32	38.82	8.7%
ami49-3	49	5	14	32	38.82	8.7%

We have implemented the algorithm and tested on problems with various rectilinear shape constraints. Many existing floorplanners minimize chip area or use minimizing area as an objective. Our algorithm can also be adapted for min-area optimization by shrinking the frame when a feasible sequence pair is met in simulated annealing.

Our goal is mainly to test the capability of the approach. Although rectilinear shape is studied extensively in the literature, most of the previous works handle convex rectilinear shape only. There is no common benchmark for rectilinear block placement. The test problems are derived from MCNC benchmarks for block placement. We choose a number of blocks and make them constrained in various rectilinear shapes. The algorithm performs very well on all test problems. Table 1 lists the experimental results for minimizing area, where all blocks are hard blocks. The experiments were carried out on a Pentium III Mobile(1.1Ghz). As an illustration, Figure 7 displays the final packing result for ami49-3.

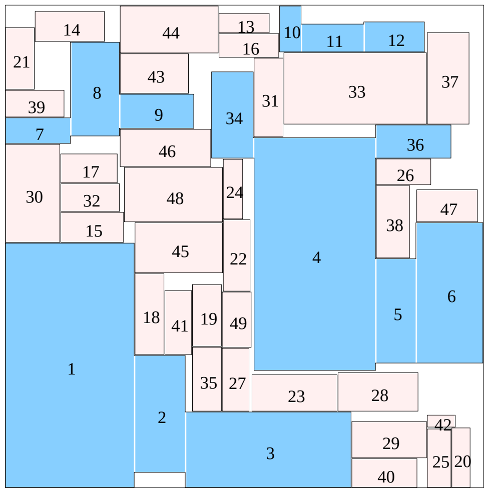

Figure 7: The result packing of ami49-3. There are four rectilinear shapes: block (1, 2, 3), (4, 5, 6, 34, 36), (7, 8, 9) and (10, 11, 12).

8. References

[1] Y.C. Chang, Y.W. Chang, G.M. Wu, and S.W. Wu. “ B*-trees: a new representation for non-slicing floorplans”, DAC-2000, pp. 458-463, 2000.

[2] K. Fujiyoshi, and H. Murata. “Arbitrary convex and concave rectilinear block packing using sequence pair”, ISPD-99, pp. 103-110, 1999.

[3] P.N. Guo, C.K. Cheng, and T. Yoshimura, “An O-tree representation of non-slicing floorplans and its applications”, DAC-99, pp. 268-273, 1999.

[4] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.K. Cheng, and J. Gu. “Corner block list: an effective and efficient topological representation of non-slicing floorplan”, ICCAD-00, pp. 8-12, 2000.

[5] M. Kang and W.W.M. Dai. “General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure”, ASP-DAC’97, pp. 265-270, 1997.

[6] M. Kang and W.W.M. Dai. “Arbitrary rectilinear block packing based on sequence pair”, ICCAD-98, pp. 259-266, 1998.

[7] J.M. Lin and Y.W. Chang. “TCG: a transitive closure graph-based representation for non-slicing floorplans”, DAC-01, pp. 764-769, 2001.

[8] J.M. Lin, H.L. Chen, and Y.W. Chang. “Arbitrary convex and concave rectilinear module packing using TCG”, DATE-02, pp. 69-75, 2002.

[9] Y. Ma, X. Hong, S. Dong, Y. Cai, C.K. Cheng, and J. Gu, “Floorplanning with abutment constraints and L-shaped/T-shaped blocks based on corner block list”, DAC’01, pp. 770-775, 2001.

[10] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. “VLSI module placement based on rectangle-packing by the sequence pair”, *IEEE TCAD*, vol. 15:12, pp. 1518-1524, 1996.

[11] S. Nakatake, H. Murata, K. Fujiyoshi, and Y. Kajitani. “Module placement on BSG-structure and IC layout applications”, ICCAD-96, pp. 484-491, 1996.

[12] T. Ohtsuki, N. Sugiyama, and H. Kawanishi. “An optimization technique for integrated circuit layout design”, ICCST Kyoto, Japan, pp. 67-68, 1970.

[13] R.H.J.M. Otten. “Automatic floorplan design”, DAC-82, pp. 261-267, 1982.

[14] Y. Pang, C.K. Cheng, K. Lampaert, and W. Xie. “Rectilinear block packing using O-tree representation”, ISPD-2001, pp. 156-161, 2001.

[15] K. Sakanushi, M. Furuya and Y. Kajitani. “The multi-BSG: stochastic approach to an optimum packing of convex-rectilinear blocks”, ICCAD-98, pp. 267-274, 1998.

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

[17] X. Tang and D.F. Wong. “FAST-SP: A fast algorithm for block placement based sequence pair”, ASPDAC-01, pp. 521-526, 2001.

[18] X. Tang and D.F. Wong. “Floorplanning with alignment and performance constraints”, DAC-02, pp. 848-853, 2002.

[19] D.F. Wong and C.L. Liu. “A new algorithm for floorplan design”, DAC-86, pp. 101-107, 1986.

[20] G.M. Wu, Y.C. Chang, and Y.W. Chang. “Rectilinear block placement using B*-trees”, IEEE ICCD, pp. 351 -356, 2000.

[21] J. Xu, P.N. Guo, and C.K. Cheng. “Rectilinear block placement using sequence pair”, ISPD-98, pp. 173-178, 1998.

[22] B. Yao, H. Chen, C.K. Cheng, and R. Graham. “Revisiting floorplan representation”, ISPD-01, pp. 138-143, 2001.

[23] F.Y. Young, D.F. Wong, and H.H. Yang. “On extending slicing floorplan to handle L/T-shaped modules and abutment constraints”, IEEE TCAD, Vol 20:6 , pp. 800-807, 2001.

[24] F.Y. Young, C.N. Chu, and Z.C. Shen. “Twin binary sequences: a non-redundant representation for general non-slicing floorplan”, ISPD-02, pp. 196-201, 2002.