Corner Block List: An Effective and Efficient Topological Representation of Non-Slicing Floorplan

Xianlong Hong [1], Gang Huang [1], Yici Cai[1], Jiangchun Gu [1], Sheqin Dong[1], Chung-Kuan Cheng [2], Jun Gu[3]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, 100084 P. R. China

[2] Department of Computer Science and Engineering, University of California, San Diego La Jolla, CA 92093-0114, USA, Email: kuan@cs.ucsd.edu

[3]Department of Computer Science, Science & Technology University of Hong Kong

Abstract—In this paper, a corner block list — a new efficient topological representation for non-slicing floorplan is proposed with applications to VLSI floorplan and building block placement. Given a corner block list, it takes only linear time to construct the floorplan. Unlike the O-tree structure, which determines the exact floorplan based on given block sizes, corner block list defines the floorplan independent of the block sizes. Thus, the structure is better suited for floorplan optimization with various size configurations of each block. Based on this new structure and the simulated annealing technique, an efficient floorplan algorithm is given. Soft blocks and the aspect ratio of the chip are taken into account in the simulated annealing process. The experimental results demonstrate the algorithm is quite promising.

1. Introduction

Floorplaning is a critical phase in physical design of VLSI circuits and has received much more attention recently due to the increased importance of hierarchical design and IP blocks. To optimize the floorplan, topological representation is one of the most important and fundamental issues. The representation structure affects the effectiveness and the efficiency of the optimization algorithm directly.

VLSI floorplans are often grouped into two categories: slicing structure and non-slicing structure. A slicing floorplan can be obtained by recursively cutting a rectangle into two parts by either a vertical or a horizontal line. The slicing floorplan covers a limited design space, but can be expressed by an efficient binary tree or Polish expression. There are $O(n!\ 2^{3n-3}/\ n^{1.5})$ combinations of the slicing tree. It takes only O(n) time to derive a floorplan from a representation[1].

For the non-slicing structure, the problem is much harder. Onodera *et al.*[2] enumerated floorplans by four relationships between two blocks. The number of combinations of their representation model is $O(2^{n(n+2)})$.

Sequence pair (SP) [3] and Bounded-Slice line Grid (BSG) [5] are two elegant models for non-slicing floorplans. In SP, two permutations are used to present the topological relations of blocks. The number of combinations of the SP is $O((n!)^2)$. From SP to its corresponding placement, it takes $O(n^2)$ time[3] [4]. In BSG, a special n×n grid is applied for the placement of n blocks. BSG has $O(n^2)$ time complexity and $n!C(n^2, n)$ combinations, which has much redundancy.

Recently, Guo *at el* [6] presented an ordered tree to represent non-slicing floorplans. O-tree has very small solution space $O(n!2^{2n-2}/n^{1.5})$ and O(n) time complexity. However, it can only represent a special placement—LB-compact placement, which is obtained by compressing blocks toward the left and bottom boundary of the chip. Most importantly, O-tree structure is not a topological representation.

In this paper, we present a new topological representation — corner block list, which can represent general non-slicing floorplans. It defines a floorplan independent of the block sizes. Using corner block list, we propose a simulated annealing algorithm for VLSI floorplan.

The advantages of corner block list are as follows:

1. The time complexity to transform a corner block list to a placement configuration is O(n).

2. The number of combinations of corner block list is $O(n!\ 2^{3n-3}/\ n^{1.5})$. Figure 1 shows the comparison of combination numbers of BSG, SP, CB and O-tree in log scale. The functions are normalized by n! which is the complexity of placement of cells with a standard width and height.

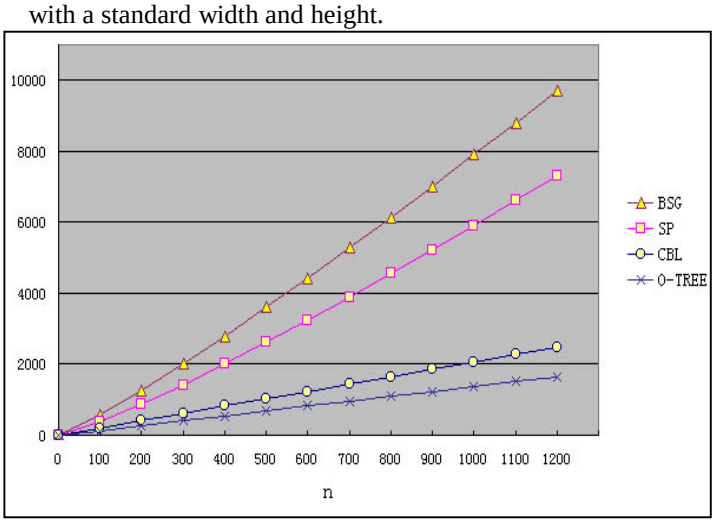


Figure 1 Comparison of combination number of 4 modes normalized by n!

3. Corner block list takes only n(3+[lg n]) bits to describe, where [lg n]denotes the minimum integral number which is not less than lg n. A corner block list needs fewer bits than SP, BSG and the same as binary tree.

4. Corner block list represents the floorplan independent of the blocks sizes, so we can use corner block list to optimize the blocks with multiple configurations of widths and heights. This

is its advantage over O-tree structure.

The organization of this paper is as follows: In section 2, a class of floorplan and some concepts are introduced. Then in section 3, a corner block structure is presented. In section 4, based on simulated annealing algorithm and corner block list, an efficient floorplan and block placement algorithm is presented. In section 5, experiment results for some MCNC benchmarks are presented. Finally, conclusion and acknowledgments are given.

2. Preliminaries

In this section, we define a class of floorplans that is a simplified version of general floorplan.

2.1 Mosaic Floorplan

A floorplan divides the chip into rectangular rooms with horizontal and vertical segments. Each room is assigned to no more than one block. We define a class of floorplan, mosaic floorplan. A floorplan belongs to the class of mosaic floorplan if and only if it observes the following three properties.

1. Floorplan of No Empty Space: There is no empty space in the floorplan, i.e. each room is assigned one and only one block. The class of floorplan of this property covers the set of the slicing floorplan (figure 2.1) and contains the floorplan that cannot be sliced (figure 2.2). In a no empty space floorplan, the internal segments intersect and form T-junctions. A T-junction is composed of two segments: a non-crossing segment and a crossing segment. The non-crossing segment has one end touching point in the interval of the crossing segment.

2. Topological Equivalence on Segment Sliding: The topology

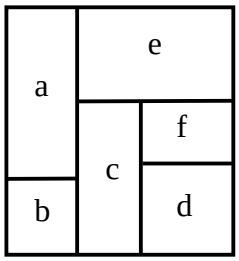


Figure 2.1

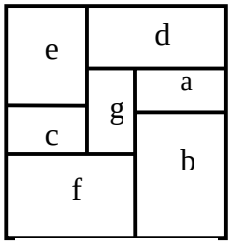


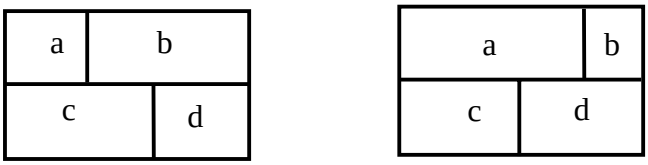
Figure 2.2

is defined to be equivalent before and after the non-crossing segment of the T-junction slides (Figure 2.3 (a) and (b)). The segment positions are to be determined by the exact widths and heights of the blocks.

3. Non-Degenerate Topology: There is no degenerate case where two distinct T-junctions meet at the same point. If a degenerate situation happens, we separate two T-junctions by sliding a non-crossing segment of a T-junction by a small distance.

2.2 Constraint Graph

A constraint graph for a floorplan is a graph G = (V, E), where

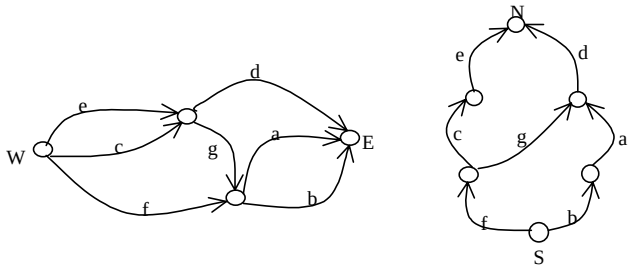


(a) (b)

Figure 2.3

the nodes in V are segments which slice the space and form the rooms of the floorplan with additional nodes used for boundaries of the placement, and the edges in E are the rooms of placement blocks. The edges in E are directed. There are two kinds of edges: one with the direction from a left node to a right node, another with the direction from a bottom node to a top node. A node is a source node of an edge if the edge is an outgoing edge of the node. A node is a destination node of an edge if the edge is an incoming edge of the node. There are two kinds of constraint graphs: the horizontal constraint graph (HCG) and vertical constraint graph (VCG). In HCG, we use a west pole "W" and an east pole "E" to represent the left and right boundary of the chip. The edges represent the horizontal relation directed from left to right. In VCG, we use a south pole "S" and a north pole "N" to represent the bottom and top boundary of the chip. The edges represent the vertical relation directed from bottom to top. In the sequel, for simplicity we mix the usage of the edge and the block.

Figure 2.4 illustrates the corresponding horizontal constraint graph and vertical constraint graph of the floorplan in figure 2.2.



(a) Horizontal (b) Vertical

Figure 2.4 Constraint Graphs

2.3 Corner Edge and Corner Block

For an edge that points to east or north pole, we define it to be a corner edge. The block that its two constraint edges are corner edges in both HCG and VCG is defined to be a corner block. For example, in figure 2.4, block "d" is a corner block, while "a" "b" and "e" are not.

Lemma 2.1: Given a mosaic floorplan of one or more blocks, there exists a unique corner block.

2.4 Orientation of Corner Block

We define the orientation of a corner block according to the joint of its left and bottom segment and the T-junction containing the joint, e.g. the bottom left corner of block d in figure 2.2. At this point, the T-junction has only two kinds of orientations: T rotated by 90 degrees (figure 2.2) and by 180 degrees counterclockwise. If T is rotated by 90 degrees, we define the corner block to be vertical oriented, and it is denoted by a "0". Otherwise, we define the corner block to be horizontal oriented, and it is denoted by a "1". For example, in Figure 2.2, the orientation of corner block d is vertical and is denoted by "0".

2.5 Corner Block Deletion and Insertion

1) Corner Block Deletion

If the corner block is horizontal oriented, to delete the corner block we shift its left segment to the right boundary of the chip, and pull the attached T-junctions along with the segment. If the corner block is vertical oriented, we shift its bottom segment to the top boundary of the chip, and pull the attached T-junctions along with the segment.

For example, for the floorplan in Figure 2.2, the corner block

d is vertical oriented, thus we shift its bottom segment to the top boundary of the chip, pull up the attached T-junctions and delete block d (figure 2.5).

We can operate the deletion on the corresponding constraint graphs directly. For example, after we delete constraint edges "d" in both constraint graph HCG and VCG, the new constraint

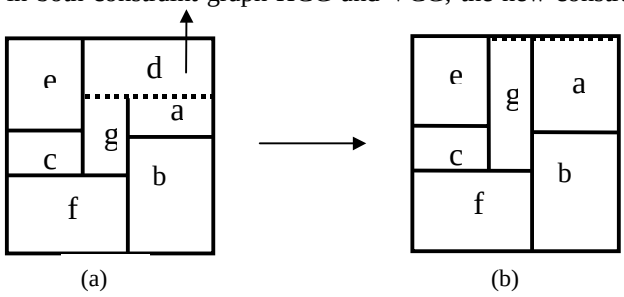


Figure 2.5

graphs are modified as shown in figure 2.6. Because the corner edge "d" is vertical, its source node is pulled to the north pole and merged into the north pole, so the edges "a" and "g" become the incoming edges of the north pole.

Lemma 2.2: Given a mosaic floorplan, the revised floorplan after corner block deletion remains to be mosaic.

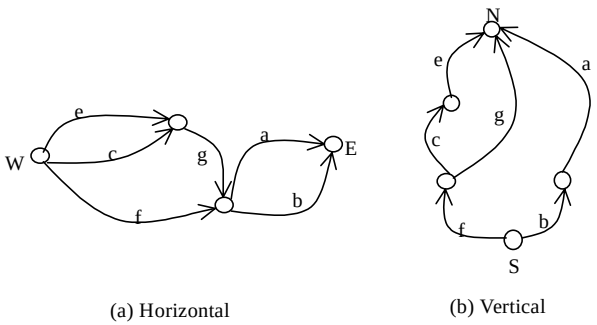


Figure 2.6 Constraint Graphs after corner edge "d" has been deleted

2) Corner Block Insertion

Corner block insertion is the inverse of deletion. If the inserting corner block is vertical oriented, we push down the horizontal segment at the top side of the chip covering a designated set of T-junctions, and then get a room for the inserting corner block (figure 2.7). If the corner block is

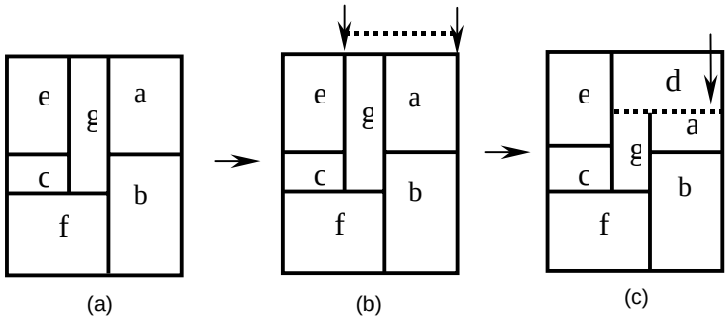


Figure 2.7

horizontal oriented, we push left the vertical segment at the right of the chip. The process to insert the corner block in the constrained graph also can be derived similarly.

Lemma 2.3: Given a mosaic floorplan, the revised floor -plan after corner block insertion remains to be mosaic.

3. Corner Block List

The corner block list is constructed from the record of a recursive corner block deletion. For each block deletion, we keep a record of block name, corner block orientation, and number of T-junctions uncovered. At the end of deletion iterations, we concatenate the data of these three items in a reversed order. Thus, we have a sequence S of block names, a list L of orientations, and a list T of T-junction information. Note that at the nth deletion, there is only one block in the floorplan. The orientation and T-junction information of the nth block do not matter. Thus, the orientation and number of T-junction of the nth block is not included in lists L and T.

T-junctions of the deleted corner block. The number of 1s corresponds to the number of attached T-junctions. Each string of 1s is ended with a 0 to separate from the record of the next corner block deletion.

Definition 3.1: The three tuple (S, L, T) is called a corner block list.

Example 3.1: We use the floorplan of figure 2.2 as an example. First, block d is deleted. d is vertical oriented and there is one T- junction attached at the bottom edge of block d. Thus, we keep a record (d, 0, 10) (Fig. 2.5(b) and Fig. 2.6). Block a, b, g, e, c, f are deleted successively. We concatenate these record in a reverse order of deletion and derive a corner block list (S, L, T), where S=(fcegbad), L=(001100), and T=(001010010).

3.1 Algorithms to Transform Between Corner Block List and Floorplan

Algorithm 3.1 Transformation from floorplan to corner block list

While there is a corner block available, repeat

(1). Delete the corner block.

(2). If the corner block is not the last one, record (block name, orientation, T-subsequence).

Add the last block to the block name list and concatenate all records in a reverse order of deletion to construct the list (S, L, T).

From lemmas 2.1 and 2.2, the sequence of corner block deletion is unique. Thus, we can derive the uniqueness of the corner block list.

Theorem 3.1: Given a mosaic floorplan, there exists a unique corner block list corresponding to the topology of the *floorplan*.

Algorithm 3.2 Transformation from corner block list to floorplan

(1). Initialize the floorplan with block S[1];

(2). For i = 2 to n do

Insert block S[i] with an orientation L[i] and covering the T-junctions according to the record in list T. If the number of covering is more than the T-junctions available, exit and report error.

A corner block list may not correspond to a floorplan, this is because of the constraints of list T. The number of erased T-junctions should not be more than the T-junctions available on the side of insertion.

Theorem 3.2: The resultant floorplan of algorithm 3.2 is mosaic if the solution exists.

3.2 The Complexity

Summing up all the contributions of corner block list (S, L, T), we have block sequence S taking n[lg n] bits and orientation

list L taking n-1 bits, while list T taking no more than 2n-3 bits, we derive the storage size of the representation.

Theorem 3.3: The storage size of corner block is no more than n(3+[lg n]) bits.

Theorem 3.4: The number of combinations of corner block list can be expressed as $O(n! \, 2^{3n-3}/ \, n^{1.5})$.

Theorem 3.5: The computational complexity of algorithm 3.1 is O(n).

Theorem 3.6: The computational complexity of algorithm 3.2 is O(n).

Remarks: The number of combinations of corner block list is equal to the number of slicing tree structure, $O(n! \, 2^{3n-3}/ \, n^{1.5})$, while the set of slicing floorplan is a subset of mosaic floorplan. The gain on a larger coverage is obtained from the elimination of redundancy.

Example: Figure 3.1 illustrates the redundancy of slicing tree structure. The floorplan can be represented by five Polish expressions: abHcHdHeV, abcHHdHeV, abHcdHHeV, abcHdHHeV, abcdHHHeV, while it takes a unique corner block list representation: abcde, 1110, 0001110. Note that in Polish representation, only the first expression is considered in searching process[1].

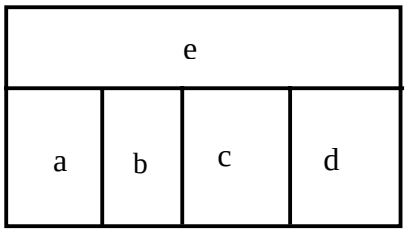


Figure 3.1 Slicing Structure with abHcHdHeV

4. Floorplan and Block Placement Algorithm

Our algorithm is based on simulated annealing algorithm[7]. The creation of a neighboring solution can be based on:

1) Randomly exchange the order of the blocks

2) Randomly choose a position in L, change 1 to 0, or 0 to 1;

3) Randomly choose a position in T, change 1 to 0 or 0 to 1

4) Rotate the module for 90^0,$180^o$and 270^0.

5) Reflect the modules in both horizontal and vertical orientations.

The 4th and 5th operations are used in the process of optimization of wire length.

For the floorplan problem with soft blocks, we prepare some alternate blocks to substitute the original one. In the simulated annealing process, we exchange not only the geometric positions of the blocks but also the alternate blocks with different aspect ratio. In our floorplan algorithm, we choose 16 alternate blocks, and apply the following operation:

6) Randomly choose a block, than randomly choose an alternate block to substitute the original one.

5. Experimental

The placement algorithm has been implemented in the C programming language, and all experiments are performed on a SUN spark20 workstation.

5.1 Test Examples

MCNC benchmarks are used for the experiments. The largest example is ami49, which has 49 blocks and 408 nets. In order to test the effective of the algorithms, especially the ability for large circuits, we create several examples based on ami33. The example test66, test99 and test198 have 66, 99, 198 blocks respectively.

5.2 Experimental Results

Table 1 is block placement results without soft blocks. Table 2 presents the floorplan results, where each block is soft. Due to CB has lower time complexity than sequence pair and BSG, the running time decrease a lot in the same temperature and inner loop conditions. The linear time complexity of CB let us obtain a placement configuration with several hundreds even thousands blocks in twenty minutes. Table 3 shows placement results with minimizing area and total wirelength. It is comparable with O-tree mode. Table 4 shows the floorplanning results using soft blocks after getting topologies corresponding to placement results with hard blocks. This means we can use corner block list to refine the blocks with multiple configurations of widths and heights under the fixed topology of blocks, which can be got by placement with hard blocks. Although the usage of placement is very low, but after refinement of blocks' shape, we can get very good floorplan result with very high usage.

6. Conclusion

CB - a new effective representation for non-slicing floorplan, has the same computing complexity as that of binary tree of slicing structure. However, it can not only represent all floorplans with slicing structure, but also represent non-slicing floorplans. The time complexity of CB is much lower than the other non-slicing structures such as SP and BSG. CB has almost the same time and space complexity as O-tree, however, it is better suited for floorplan optimization with various size configurations of each block. Based on CB and simulated annealing technique, an efficient algorithm for floorplan and block placement is given. The experiment results demonstrate that our algorithm is quite promising.

7. Acknowledgements

This work was supported by Chinese National Nature Science Funds under the grants 69776027 and Chinese 973 National Fundamental Funds under the grants G1998030411. We thank John Lillis for technical discussions.

References

[1] D.F.Wong & C.L.Liu, " A new algorithm for floorplan design". In: Proc. of 23rd ACM/IEEE Design Automation Conference, 101-107, 1986.

[2] H. Onodera, Y.Taniguychi & K.Tamaru, " Branch and bound placement for building block layout" in: Proc. 28th ACM/IEEE Design Automation Conference, 423-439, 1991

[3] H. Murata, K. Fujiyoshi, S.Nakatake, & Y. Kajitani, " Rectangle-packing-based module placement", in: Proc. of International Conference on Computer Aided Design, 472 -479. 1995.

[4] T.Tahahashi, " An algorithm for finding a maximum-weight decreasing sequence in a permutation, motivated by rectangle packing problem", Technical report of IEICE, vol. VLD 96, no.201, 31-35, 1996.

[5] S.Nakatake, H. Murata, K. Fujiyoshi, Y. Kajitani, "Module placement on BSG-structure and IC layout application" in: Proc. of International Conference on Computer Aided Design, 484-490, 1996.

[6] P.N.Guo, C.K.Cheng, " An O-tree representation of non-slicing floorplan and its applications", in: ACM/IEEE Design Automation Conference, 1999.

[7] S.Kirkpatrick *et al.* "Optimization by simulated annealing," Science. vol. 220, 671-680, 1983

Table 1. BBL Placement Results Based on CB

Example	Area (mm^2)	Area usage	Runtime (sec.)	Other
Xerox	20.96	0.922	30	
Hp	66.14	0.932	32	
Ami33	1.201	0.963	36	figure 5.1
Ami49	38.58	0.918	65	
Test66	2.383	0.947	86	
Test99	3.672	0.946	110	
Test198	7.506	0.926	165	figure 5.2

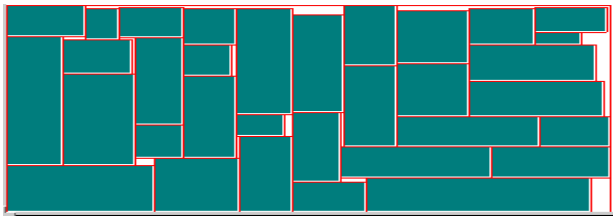


Figure 5.1 placement of ami33

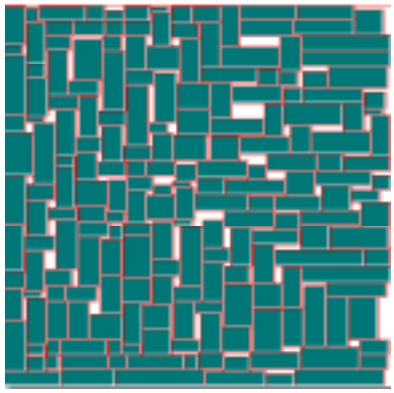


Figure 5.2 Placement Result of Test198

Table 2. Floorplan Results Based on CB

Examples	Area(mm^2)	usage	Run time (sec.)
Ami33	1.176	0.983	87
Xerox	19.75	0.979	76
Hp	62.93	0.980	75
Apte	46.63	0.966	78
Ami49	36.09	0.982	179

Table 3. Placement Results with Total Wirelengths

examples	area+0.1*wirelength (minimum/average)		area + wirelength (minimum/average)		0.1*area+wirelength (minimum/ average)		Improve over O-tree area/ wire (average) (w_1=w_2=0.5)
	area	Wire	area	Wire	area	Wire	
apte	46.079 /46.731	405.845 /429.241	47.429 /47.916	194.950 /202.830	47.230 /48.202	103.832 /177.608	10%/45%
xerox	20.124 /20.347	604.751 /667.713	20.233 /20.378	403.466 /504.108	20.686 /20.953	403.466 /533.916	9%/-12%
ami33	1.2149 /1.2186	54.075 /61.649	1.2255 /1.2343	51.674 /56.726	1.2450 /1.2510	39.431 /48.947	9.2%/5.2%
ami49	38.337 /38.477	973.845 /1164.694	38.378 /39.410	732.844 /870.432	41.749 /45.720	710.220 /775.551	6%/-12%

Table 4. Floorplanning Results with Fixed Topologies of Blocks

examples	Placement area(mm^2)/usage(%)	Floorplan area(mm^2)/usage(%)	Improve over placement (%)	Average improvement(%)
ami33	1.390 / 83.184	1.216 / 95.099	11.915	5.552
ami49	41.400 / 85.616(Fig. 5.3)	36.980 / 95.851(Fig. 5.4)	10.235	8.526
apte	54.554 / 82.612	47.171 / 95.541	12.929	12.185
hp	69.783 / 88.374	62.849 / 98.143	9.769	6.672
xerox	22.755 / 85.035	19.696 / 98.240	13.205	10.272

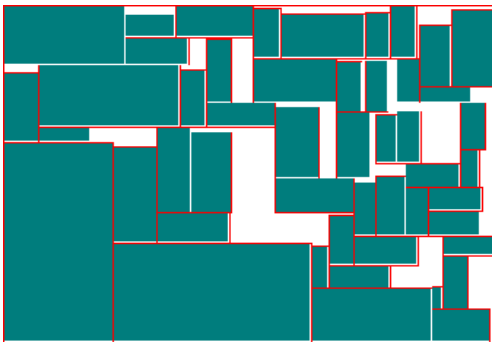


Fig. 5.3 Placement result of ami49

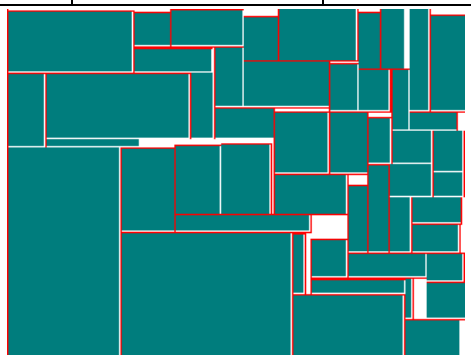


Fig. 5.4 Floorplanning result of ami49