Area Minimization for Hierarchical Floorplans *

Peichen Pan†, Weiping Shi‡, and C. L. Liu†
†Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
‡Department of Computer Science
University of North Texas
Denton, TX 76203

Abstract

Two results are presented in this paper. First we settle the open problem on the complexity of the area minimization problem for hierarchical floorplans by showing it to be NP-complete. We then present a pseudo-polynomial area minimization algorithm for hierarchical floorplans of order-5. The algorithm is based on a new algorithm for determining the set of nonredundant realizations of a wheel. The new algorithm for wheels has time cost $O(k^2 \log k)$ and space cost $O(k^2)$ if each of the (five) blocks in a wheel has at most k realizations — a reduction by a factor of k in both costs in comparison with previous algorithms. The area minimization algorithm was implemented. Our experimental results show that the algorithm is indeed very fast.

1 Introduction

Area minimization (also called floorplan sizing) is a subtask in the floorplanning phase of VLSI chip design. It is the problem of selecting a layout alternative for each subcircuit on a chip so as to minimize the total chip area after the floorplan (which specifies the relative positions of the subcircuits on the chip) has been determined. This problem has been studied extensively [2, 7, 8, 9, 10, 12, 13, 14, 15, 16].

In practice, many floorplans are constructed by top-down partitioning (or by bottom-up clustering). In each step, a group of subcircuits is selected and partitioned into at most p subgroups (or the other way around for clustering). A floorplan obtained this way is called a hierarchical floorplan (of order-p). For practical reasons, p is usually confined to be a small integer, in most cases $p \leq 5$ [3, 6]. A hierarchical floorplan of order-p can be naturally described by a p-ary tree which corresponds to the partitioning process. (Note that p is a constant. Although a hierarchy can always be extracted from any floorplan [2], the corresponding tree will have unbound degrees in general.)

In this paper we study the area minimization problem for hierarchical floorplans. Of particular interest is hierarchical floorplans of order-5 since higher order hierarchical floorplans are rarely used in practice. The complexity of the area minimization problem for hierarchical floorplans was a long-standing open problem. In this paper we settle this problem by showing it to be NP-complete. Next, we present a new pseudo-polynomial area minimization algorithm for hierarchical floorplans of order-5.

The remainder of this paper is organized as follows: Section 2 introduces some definitions. Section 3 shows the NP-completeness of the problem. Section 4 presents the new area minimization algorithm. Section 5 lists our experimental results. Finally, Section 6 concludes the paper.

2 Preliminaries

A *floorplan* is a dissection of an enveloping rectangle by horizontal and vertical line segments into rectangular (*basic*) *blocks*. Given a set of partitioning patterns P, we can construct a corresponding set of hierarchical floorplans by recursively partitioning a block according to one of the patterns in P, starting with a single block. A *slice* is a partitioning pattern with two blocks. There are only two different slices as shown in Figure 1. When P consists of only the two slices, the corresponding hierarchical floorplans are called *slicing floorplans*. A *wheel* is a non-slicing partitioning pattern with five blocks. There are two different wheels as shown in Figure 1. When P consists of the two slices as well as the two wheels, the corresponding hierarchical floorplans are called *hierarchical floorplans of order-5* [1]. Figure 2 shows an example illustrating the construction of a hierarchical floorplan of order-5.

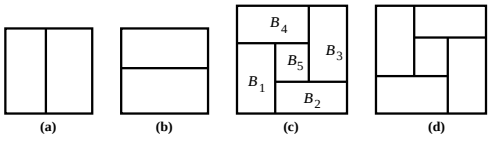

Figure 1: (a) Vertical slice, (b) Horizontal slice, (c) Left wheel, (d) Right wheel.

A layout alternative of a subcircuit is called a *realization* of the corresponding block in the floorplan. A *realization* of a floorplan is obtained by selecting a realization for each block and arrange them according to the floorplan. A realization (of a block or a floorplan) has two dimensions, the

*Research of Peichen Pan and C.L. Liu was partially supported by the National Science Foundation under grant MIP-9222408. Research of Weiping Shi was partially supported by the National Science Foundation under grant MIP-9309120.

[1] Since wheels are the only non-slicing patterns with five or fewer blocks, this definition is the same as if P consists of all patterns with five or fewer blocks.

width and height of the smallest rectangle that can accommodate it. To simplify the presentation, in this paper when we say a realization we always mean the corresponding rectangle. Adding the information on how each realization is composed to our discussion is straightforward but it will unnecessarily complicate the presentation.

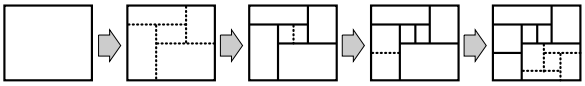

Figure 2: Construction of an order-5 floorplan.

The area minimization problem can be formally stated as follows: *Given a floorplan F and a set of realizations for each of its (basic) blocks, determine a realization of F which has the minimum area.* In this paper we are interested in the case in which F is a hierarchical floorplan. We also assume that the width and height of any realization of a block are non-negative integers.

Let r be a realization, we use $w(r)$ and $h(r)$ to denote the width and height of r, respectively. For two realizations r_1 and r_2, r_1 is said to *dominate* r_2 if the conditions $w(r_1) \leq w(r_2)$ and $h(r_1) \leq h(r_2)$ are satisfied. For a set of realizations, a realization in the set is said to be *redundant* if it is dominated by another realization in the set. Otherwise, the realization is said to be *nonredundant*. A set of nonredundant realizations obviously has the following properties: (i) no two realizations in the set have the same width or height, and (ii) if the realizations are sorted in increasing order according to one dimension, they will be arranged in decreasing order according to the other dimension. These properties suggest the following simple algorithm for computing the set of nonredundant realizations from a given set of realizations: Sort all realizations in the set in increasing order according to the width (if two realizations have the same width, delete the one with larger height), then inspect the realizations in this order one by one and retain only those which keep the height in decreasing order. We shall use $L(F)$ to denote the set of nonredundant realizations of F (either a block or a floorplan).

Obviously, $L(F)$ contains all the minimum area realizations of F. We also have the following simple fact.

Fact 1 *Suppose F_1 is a sub-floorplan[2] of F. Let F' be the floorplan obtained from F by replacing F_1 by a basic block B and $L(B) = L(F_1)$, then $L(F) = L(F')$.*

3 An NP-completeness result

In this section, we shall show that the area minimization problem for hierarchical floorplans is NP-complete. This result is built upon the following theorem. Its proof is omitted due to space limitation.

Theorem 1 *The area minimization problem for hierarchical floorplans of order-5 is NP-complete.* □

[2] A sub-floorplan is a set of blocks which form a rectangular superblock.

With the above result, we can now show the NP-completeness of the area minimization problem for hierarchical floorplans. It was shown in [11] that any nonslicing partitioning pattern contains a wheel-type structure. (See Figure 3(a).) In Figure 3(a), if we assume all unshaded blocks have only one realization $(0, 0)$, the resultant pattern is equivalent to a wheel formed by the five shaded blocks as far as the area is concerned. Theorem 1, therefore, implies that the area minimization problem for hierarchical floorplans constructed by any set of partitioning patterns with at least a nonslicing pattern is NP-complete. In other words, slicing floorplans are the only type of floorplans for which area minimization can be accomplished in polynomial time, assuming $P \neq NP$.

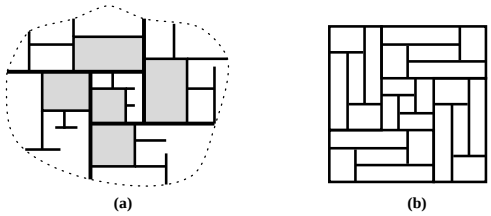

Figure 3: (a) A partially drawn nonslicing partitioning pattern, (b) The floorplan used in **EX1** to **EX5**.

4 An area minimization algorithm

In this section we present an area minimization algorithm for hierarchical floorplans of order-5. The algorithm employs a hierarchical approach [2, 12], in which a fast algorithm for determining the set of nonredundant realizations of a wheel plays an important role.

The area minimization algorithm determines $L(F)$, the set of nonredundant realizations of a floorplan F. Since $L(F)$ contains all the minimum area realizations of F, determining $L(F)$ also gives the designer the freedom to choose among all the minimum area realizations of F the one with a preferred aspect ratio. The area minimization algorithm can be summarized as follows:

AreaMin(F)
  if F has only one block B, **return** $L(B)$;
  $F_1 \leftarrow$ a slice or a wheel in F;
  Determine $L(F_1)$;
  $F' \leftarrow$ the floorplan obtained from F by replacing F_1 by a basic block B;
  $L(B) \leftarrow L(F_1)$;
  return AreaMin(F').

AreaMin follows the reverse process in which F was constructed. At each stage, a slice or a wheel is replaced by a basic block with the set of realizations of the basic block being that of the slice or wheel. By Fact 1 this process does not change the set of nonredundant realizations. Thus, when a floorplan with only one block is reached, it means the set of nonredundant realizations of F has been

determined. **AreaMin** is actually a general area minimization algorithm. It can be used for any hierarchical floorplan. In general, the sets of nonredundant realizations of all partitioning patterns in P should be determined.

In **AreaMin** the area minimization for hierarchical floorplans of order-5 is reduced to that of determining the sets of nonredundant realizations of slices and wheels (determining $L(F_1)$ in **AreaMin**). An efficient algorithm for determining the set of nonredundant realizations of a slice was presented in [8, 10]. What remains is an efficient algorithm for determining the set of nonredundant realizations of a wheel. The algorithm we shall present in Section 4.1 has time cost $O(k^2\log^2 k)$ and space cost $O(k^2)$ if each block in the wheel has at most k realizations. The time cost is further improved to $O(k^2 \log k)$ in Section 4.2 [3].

We now determine the time complexity of **AreaMin**. Suppose F has n blocks and the dimensions of all realizations of the blocks are upper-bounded by a positive integer M. It is easy to see that the number of nonredundant realizations of any sub-floorplan of F is at most nM. Thus, $|L(F_1)| \leq nM$. The number of calls to determining $L(F_1)$ is obviously less than n, so the time cost of **AreaMin** is $O(n(nM)^2\log(nM)) = O(n^3M^2\log(nM))$, a polynomial in n and M. Thus, **AreaMin** is pseudo-polynomial.

4.1 An algorithm for determining the set of nonredundant realizations of a wheel

Because of the obvious symmetry between the two wheels, we only consider the left wheel. The problem we shall focus on is: *For the left wheel W in Figure 1(c), determine $L(W)$ for given $L(B_i)$, $1 \leq i \leq 5$.*

If r is a realization of W, we use $r(B_i)$ to denote the realization selected for B_i in r. We also assume the nonredundant realizations of a block are ordered in increasing width and decreasing height. The t-th realization of block B_i is denoted (w_i^t, h_i^t). Let k_i denote the size of $L(B_i)$ and k be the largest among k_i, $1 \leq i \leq 5$. Previous algorithms for determining $L(W)$ have time cost $O(k^3 \log k)$ and space cost $O(k^3)$ [9, 12]. Here, we shall present an algorithm with time cost $O(k^2\log^2 k)$ and space cost $O(k^2)$.

We first define a procedure $\mathbf{BS}(S, b, flag)$, where $S = \{(w_1, h_1), (w_2, h_2), \ldots, (w_s, h_s)\}$ and $w_1 < w_2 < \cdots < w_s$, $h_1 > h_2 > \cdots > h_s$, b is a nonnegative integer, and $flag$ is either $width$ or $height$. When $flag = width$, $\mathbf{BS}(S, b, flag)$ outputs the maximum i such that $w_i \leq b$ (or nil if $w_l > b$ for $1 \leq l \leq s$); when $flag = height$, $\mathbf{BS}(S, b, flag)$ outputs the minimum i such that $h_i \leq b$ (or nil if $h_l > b$ for $1 \leq l \leq s$). Obviously, **BS** can be implemented in $O(\log s)$ time and $O(1)$ space by using binary search.

Our approach is to generate a superset of $L(W)$ first. Then, redundant realizations are removed from the superset to obtain $L(W)$. To generate the superset, there are two cases according to how the widths and heights of the nonredundant realizations are determined.

Case 1. The nonredundant realizations the widths or heights of which are determined by two of the blocks.

[3] An algorithm with the same performance was claimed in [1]. However, the algorithm is incorrect

The nonredundant realizations in this case can be further divided into classes according to the blocks which determine the widths and heights of the realizations. The algorithm will generate a superclass for each class. There are eight classes all together, but we only need to consider two of them: Class 1: those the widths of which are determined by B_4 and B_3 and the heights of which are determined by B_3 and B_2, and Class 2: those the widths of which are determined by B_4 and B_3 and the heights of which are determined by B_4, B_5, and B_2. A superclass of any of the other classes can be generated symmetrically since they can be turned into either Class 1 or Class 2 by rotating W by 90°, or 180°, or 270° clockwise. We consider Class 1 and Class 2 separately.

Subcase 1.1: Class 1.

For (w_4, h_4) in $L(B_4)$ and (w_3, h_3) in $L(B_3)$, a realization of W is called the *I-support* for (w_4, h_4) and (w_3, h_3) if it is the one with minimum height among those realizations r's such that $r(B_3) = (w_3, h_3)$, $r(B_4) = (w_4, h_4)$, $w(r) = w_4 + w_3$, and $h(r) = h_3 + h(r(B_2))$. Obviously, a nonredundant realizations in Class 1 must be the I-support for some (w_4, h_4) in $L(B_4)$ and (w_3, h_3) in $L(B_3)$. To obtain a superclass of Class 1, we simply generate the set consisting of all I-supports (there are at most k_4k_3 of them). We now describe a procedure for determining the I-support r for given (w_4, h_4) and (w_3, h_3) if it exists. Noticing that if $h_3 < h_4$, there is no corresponding I-support, we assume $h_3 \geq h_4$. Since $h(r) = h_3 + h(r(B_2))$, we have $h(r(B_5)) \leq h_3 - h_4$. Let $t_5 = \mathbf{BS}(L(B_5), h_3 - h_4, height)$. If $t_5 = nil$, again there is no I-support for (w_4, h_4) and (w_3, h_3). Otherwise, we simply let $r(B_5) = (w_5^{t_5}, h_5^{t_5})$, the t_5-th realization in $L(B_5)$ because it is the one with minimum width among those realizations in $L(B_5)$ whose heights are less than or equal to $h_3 - h_4$. We now search $L(B_2)$ to determine $r(B_2)$. Suppose the t-th realization (w_2^t, h_2^t) in $L(B_2)$ is currently being considered. We check whether there is a realization in $L(B_1)$ that can be fit into the slot with width $w_4 + w_3 - \mathbf{max}\{w_2^t, w_5^{t_5} + w_3\}$ and height $h_2^t + h_3 - h_4$ (this is the space left for B_1). This can be done by considering the t_1-th realization $(w_1^{t_1}, h_1^{t_1})$ in $L(B_1)$ where $t_1 = \mathbf{BS}(L(B_1), h_2^t + h_3 - h_4, height)$. If $w_1^{t_1} \leq w_4 + w_3 - \mathbf{max}\{w_2^t, w_5^{t_5} + w_3\}$, obviously $(w_1^{t_1}, h_1^{t_1})$ can be placed in the slot, otherwise no realization in $L(B_1)$ can. In the former situation, all realizations before (w_2^t, h_2^t) in $L(B_2)$ can be ignored because $r(B_2)$ is the one with minimum height among all realizations in $L(B_2)$ that has this property. In the latter situation, all realizations after (w_2^t, h_2^t) (itself including) in $L(B_2)$ can be ignored simply because any of them can not leave enough space for B_1. Notice that **BS** is called once for determining t_5 at the beginning and once for testing each (w_2^t, h_2^t). If we use binary search to determine $r(B_2)$, the I-support can be found in $O(\log^2 k)$ time and $O(1)$ space. (Note that $r(B_1)$ is determined as a by-product of determining $r(B_2)$.) By calling the procedure k_4k_3 times, all the I-supports can be determined in $O(k^2\log^2 k)$ time and $O(k^2)$ space.

Subcase 1.2: Class 2

Let r' be a nonredundant realization in Class 2 with

$r'(B_i) = (w_i, h_i)$ for $1 \le i \le 5$. Obviously, $h_5 + h_4 \ge h_3$. We can further assume

$$h_3 = \mathbf{max}\{h \mid (w,h) \in L(B_3) \text{ and } h \le h_4 + h_5\} \qquad (1)$$

For otherwise, either r' is redundant (when $w_4 + w_3 > w_1 + w_2$), or there is a realization that has the same width and height as r' but does not belong to this class (when $w_4 + w_3 = w_1 + w_2$). We therefore consider the pair (w_4, h_4) and (w_3, h_3) here only when there is a (w_5, h_5) such that (1) is satisfied.

For (w_4, h_4) in $L(B_4)$ and (w_5, h_5) in $L(B_5)$, Let $t_3 = \mathbf{BS}(L(B_3), h_4 + h_5, height) \neq nil$. A realization of W is called the *II-support* for (w_4, h_4) and (w_5, h_5) if it is the one with minimum height among those realizations r's such that $r(B_4) = (w_4, h_4)$, $r(B_5) = (w_5, h_5)$, $w(r) = w_4 + w_3^{t_3}$, and $h(r) = h_4 + h_5 + h(r(B_2))$. To obtain a superclass of Class 2, we simply generate the set consisting of all the II-supports (there are at most $k_4 k_5$ of them). Let r be the II-support for (w_4, h_4) and (w_5, h_5), if it exists. By our assumption, $r(B_3) = (w_3^{t_3}, h_3^{t_3})$. We now search $L(B_2)$ to determine $r(B_2)$. The situation is very similar to that in determining an I-support. A similar procedure can be designed to determine r in $O(\log^2 k)$ time and $O(1)$ space. Thus, all the II-supports can be determined in $O(k^2 \log^2 k)$ time and $O(k^2)$ space.

By putting all the superclasses together, we obtain a set of realizations of W which contains all nonredundant realizations in Case 1 in $O(k^2 \log^2 k)$ time and $O(k^2)$ space.

Case 2. The nonredundant realizations not in Case 1.

For a nonredundant realization in this case, neither its width nor its height is determined by two blocks in the wheel. For any (w_1, h_1) in $L(B_1)$ and (w_5, h_5) in $L(B_5)$, a realization r such that

$$\begin{aligned}
&r(B_1) = (w_1, h_1) \\
&r(B_5) = (w_5, h_5) \\
&r(B_4) = (w_4^{t_4}, h_4^{t_4}) \quad t_4 = \mathbf{BS}(L(B_4), w_1 + w_5, width) \\
&r(B_3) = (w_3^{t_3}, h_3^{t_3}) \quad t_3 = \mathbf{BS}(L(B_3), h_5 + h_4^{t_4}, height) \\
&r(B_2) = (w_2^{t_2}, h_2^{t_2}) \quad t_2 = \mathbf{BS}(L(B_2), w_5 + w_3^{t_3}, width)
\end{aligned}$$

is called the *III-support* for (w_1, h_1) and (w_5, h_5). It can be shown that a nonredundant realization in this case must be an III-support. For given (w_1, h_1) and (w_5, h_5), the III-support obviously can be determined by calling **BS** three times (according to the definition of III-supports). Thus, all the III-supports (there are at most $k_1 k_5$ of them) can be determined in $O(k^2 \log k)$ time and $O(k^2)$ space.

Let R denote the set of realizations of W generated in both Case 1 and Case 2. By construction we have $L(W) \subseteq R$ and $|R| = O(k^2)$. The last step of the algorithm is to remove the redundant realizations in R. This can be accomplished by sorting the realization in R according to their widths and removing all redundant ones as mentioned in Section 2, which takes $O(k^2 \log k^2) = O(k^2 \log k)$ time and $O(k^2)$ space. Thus, we have the following result.

Theorem 2 *If each block has at most k realizations, the set of nonredundant realizations of a wheel can be determined in $O(k^2 \log^2 k)$ time and $O(k^2)$ space.* □

4.2 Improvements

The basic algorithm presented in Section 4.1 can be further improved. We now show how to improve the time cost of the basic algorithm to $O(k^2 \log k)$.

To achieve this, we only need to speed up the procedures for determining the I-supports and II-supports. We consider the I-supports first. For fixed (w_3, h_3) in $L(B_3)$, let r_i denote the I-support for (w_4^i, h_4^i) and (w_3, h_3). (If it does not exist, let $r_i = nil$.) It suffices to show that $r_1, r_2, \ldots, r_{k_4}$ can be determined in $O(k \log k)$ time. The following lemma is the key observation.

Lemma 1 *If $r_i \neq nil$, then $r_{i+1} \neq nil$ and $h(r_{i+1}(B_2)) \le h(r_i(B_2))$.*

For convenience, we imagine there is a fictitious 0-th realization in $L(B_2)$ and a fictitious realization of W r_0 such that $r_0(B_2)$ is the 0-th realization in $L(B_2)$. We will determine $r_1, r_2, \ldots, r_{k_4}$ one by one in this order. For each i, the realizations in $L(B_2)$ are examined consecutively in the order of decreasing height to search for $r_i(B_2)$, started with $r_{i-1}(B_2)$. To examine (w_2^t, h_2^t), the t-th realization in $L(B_2)$, we check whether or not there is enough room left to place a realization of B_1 if the $(t+1)$-th realization in $L(B_2)$ is selected for B_2 (which can be accomplished by one call to **BS**). If there is enough room, we continue to examine the $(t+1)$-th realization in $L(B_2)$. Otherwise, we have $r_i(B_2) = (w_2^t, h_2^t)$ and go on to determine r_{i+1}. As can be seen, in each step we consider either the next realization in $L(B_4)$ or the next realization in $L(B_2)$. Hence, the total number of calls to **BS** to check whether or not there is enough room is upper-bounded by $k_4 + k_2$. As a result, $r_1, r_2, \ldots, r_{k_4}$ can be determined in $O(k \log k)$ time.

For II-supports, if we let p_i denote the II-support for (w_4^i, h_4^i) and (w_5, h_5) for a fixed realization (w_5, h_5) in $L(B_5)$, we have the following observation:

Lemma 2 *If $p_i \neq nil$ and $\mathbf{BS}(L(B_3), h_4^{i+1} + h_5, height) \neq nil$, then $p_{i+1} \neq nil$ and $h(p_{i+1}(B_2)) \le h(p_i(B_2))$.*

As in the case of I-supports, we determine $p_1, p_2, \ldots, p_{k_4}$ one by one in this order. By a method similar to that for I-supports, we can determine $p_1, p_2, \ldots, p_{k_4}$ altogether in $O(k \log k)$ time.

By the above analysis, we have the following result:

Theorem 3 *If each block has at most k realizations, the set of nonredundant realizations of a wheel can be determined in $O(k^2 \log k)$ time and $O(k^2)$ space.* □

Finally, we would like to point out whether the above result can be further improved relates to a long-standing open problem in Algorithm Theory if the realizations in $L(W)$ are required to be in sorted order. (To remove redundant realizations, it seems that the realizations in $L(W)$ must be sorted.) The problem is called **sorting** $X + Y$ which asks whether the sorting of $\{x_i + y_j \mid i, j = 1, \ldots, k\}$ can be done in $o(k^2 \log k)$ time [4]. We can reduce **sorting** $X + Y$ to our problem in $O(k)$ time.

5 Experimental Results

AreaMin was implemented in C and executed on a SPARCstation 10. The improvements presented in Section 4.2 have not been implemented. We ran the program on a set of examples, most of them are from the literature. **EX1** to **EX5** are taken from [12]. They all use the floorplan shown in Figure 3(b). For **EX1** each block has three realizations: $(4,1)$, $(2,2)$, and $(1,4)$. For **EX2** each block has four realizations: $(6,1)$, $(3,2)$, $(2,3)$, and $(1,6)$. For **EX3** each block has five realizations: $(16,1)$, $(8,2)$, $(4,4)$, $(2,8)$, and $(1,16)$. For **EX4** each block has six realizations: $(12,1)$, $(6,2)$, $(4,3)$, $(3,4)$, $(2,6)$, and $(1,12)$. For **EX5** each block has eight realizations: $(24,1)$, $(12,2)$, $(8,3)$, $(6,4)$, $(4,6)$, $(3,8)$, $(2,12)$, and $(1,24)$. **EX6** is the 24-block floorplan together with the sets of realizations[4] in [13] (also see [12]). **EX7** to **EX12** are derived from **EX1** to **EX6**, respectively. **EX7** is constructed by substituting each of the five blocks of a wheel with a copy of **EX1**. **EX8** to **EX12** are constructed in the same way from **EX2** to **EX6**, respectively.

Table 1 lists our experimental results. Under the column *total* we list the total number of possible realizations for each example. Column *examined* lists the total number of realizations of the sub-floorplans ever generated by **AreaMin** for each example as this number reflects the number of realizations examined by **AreaMin**. From the table it is clear that **AreaMin** ignores a large number of redundant realizations. For example, although **EX11** has 8^{125} possible realizations, our algorithm examined only 12785 realizations and took only 3.42 seconds. The running times for the other 11 test examples are all under two seconds as shown in Table 1. The experimental results clearly indicate that our algorithm is very efficient.

test example	# blocks	# total	# examined	time (sec)
EX1	25	3^{25}	168	0.02
EX2	25	4^{25}	365	0.03
EX3	25	5^{25}	776	0.13
EX4	25	6^{25}	994	0.18
EX5	25	8^{25}	2031	0.46
EX6	24	2.03×10^{16}	441	0.08
EX7	125	3^{125}	865	0.12
EX8	125	4^{125}	1930	0.27
EX9	125	5^{125}	4730	1.05
EX10	125	6^{125}	5723	1.23
EX11	125	8^{125}	12785	3.42
EX12	120	3.45×10^{81}	2382	1.78

Table 1: Experimental results

[4] In this example, not all the widths and heights of the realizations are integers, but our algorithm can handle this situation without any modification. The integral requirement is mainly for complexity analysis.

6 Conclusions

In this paper, we showed that the area minimization problem for hierarchical floorplans is NP-complete. This settled the open problem on the complexity of the area minimization problem for hierarchical floorplans. We then presented a fast pseudo-polynomial area minimization algorithm for hierarchical floorplans of order-5. The algorithm is based on a new algorithm for determining the set of nonredundant realizations of a wheel. The new algorithm for wheels runs faster and uses less memory than previous algorithms.

References

[1] C.-H. Chen and I.G. Tollis, "Area optimization of spiral floorplans," *Technical Report, The University of Texas at Dallas,* 1992.

[2] K. Chong and S. Sahni, "Optimal realizations of floorplans," in *IEEE Trans. on Computer-Aided Design*, vol. CAD-12, no. 6, pp. 793-801, 1993.

[3] W.-M. Dai and E.S. Kuh, "Simultaneous floor planning and global routing for hierarchical building block layout," in *IEEE Trans. on Computer-Aided Design*, vol. CAD-6, no. 5, pp. 828-837, 1987.

[4] M. L. Fredman, "How good is the information theory bound in sorting?," in *Theoretical Computer Science* 1, pp. 355-361, 1976.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness.* Freeman, San Francisco, 1979.

[6] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout.* John Wiley & Sons, New York, 1990.

[7] T. Lengauer and R. Muller, "Robust and accurate hierarchical floorplanning with integrated global wiring," in *IEEE Trans. on Computer-Aided Design*, vol. CAD-12, no. 6, pp. 802-809, 1993.

[8] R.H.J.M. Otten, "Automatic floorplan design," in *Proc. 19th ACM/IEEE Design Automation Conf.,* 1982, pp. 261-267.

[9] P. Pan and C. L. Liu, "Area minimization for general floorplans," in *Digest Int'l. Conf. on Computer-Aided Design,* 1992, pp. 606-609.

[10] L. Stockmeyer, "Optimal orientations of cells in slicing floorplan designs," in *Info. and Control,* vol. 59, pp. 91-101, 1983.

[11] K. J. Supowit and E. A. Slutz, "Placement algorithms for custom VLSI," in *Computer Aided Design,* vol. 16, no. 1, pp. 45-50, 1984.

[12] Ting-Chi Wang and D.F. Wong, "Optimal floorplan area optimization," in *IEEE Trans. on Computer-Aided Design*, vol. CAD-11, no. 8, pp. 992-1002, 1992.

[13] S. Wimer, I. Koren, and I. Cederbaum, "Optimal aspect ratios of building blocks in VLSI," in *IEEE Trans. on Computer-Aided Design*, vol. 8, no 2, pp. 139-145, 1989.

[14] D.F. Wong and P.S. Sakhamuri, "Efficient floorplan area optimization," in *Proc. 26th ACM/IEEE Design Automation Conf.,* 1989, pp. 586-589.

[15] K. H. Yeap and M. Sarrafzadeh, "An integrated algorithm for optimal floorplan sizing and enumeration," in *European Design Automation Conf.,* 1993, pp. 29-33.

[16] G. Zimmermann, "A new area and shape function estimation technique for VLSI layouts," in *Proc. 25th ACM/IEEE Design Automation Conf.,* 1988, pp. 60-65.