Floorplan Area Minimization using Lagrangian Relaxation

F.Y. Young[1], Chris C.N. Chu[2], W.S. Luk[3] and Y.C. Wong[3]

[1]Department of Computer Science and Engineering
The Chinese University of Hong Kong
New Territories, Hong Kong

[2]Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011

[3]Synopsys, Inc.
700 East Middlefield Road
Mountain View, CA 94043-4033

ABSTRACT

Floorplan area minimization is an important problem because many modules have shape flexibilities during the floorplanning stage. Area minimization in general non-slicing floorplan is a complicated problem. Many previous works have attempted to tackle this problem [9; 6; 5; 1] using heuristics or numerical methods but none of them can solve it optimally and efficiently. In this paper, we show how this problem can be solved optimally by a geometric programming using Lagrangian relaxation. The resulting Lagrangian relaxation subproblem is so simple that the size of each module can be found in constant time. We implemented our idea in a simulated annealing framework based on the sequence pair representation. The area minimization procedure is invoked in every iteration of the annealing process but the total execution time is still very much faster than that of the most updated previous work [4]. For a benchmark data with 49 modules, we take 19.5 hours using a 270 MHz Sun Ultra 5 while the convex programming approach in [4] takes seven days using a 250 MHz DEC Alpha. This area minimization method will be applicable to any other floorplanning algorithm which uses constraint graphs to find module positions in the final packing.

1. INTRODUCTION

Floorplanning has become increasingly important in physical design of VLSI circuits because of the advance in the deep sub-micron technology. Many floorplanning algorithms were proposed in recent years and many of them use constraint graphs to find module positions in the final packing. Unfortunately, it is not known how shape flexibility of soft modules can be handled in constraint graphs efficiently. This is an important problem because soft modules are common in the floorplanning stage when many designs are not done in details yet. Some previous works [9; 6; 5; 1] have attempted to tackle this problem but none of them succeeded in obtaining the optimal solution efficiently.

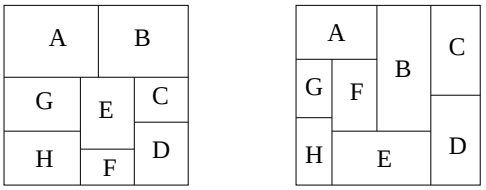
There are two types of floorplan: slicing and non-slicing. Slicing floorplan is a floorplan which can be obtained by cutting rectangles recursively. Non-slicing floorplan is one which is not restricted to be slicing. Figure 1 shows an example of each. Non-slicing floorplan is a more general representation and it can describe all kinds of packings. However slicing floorplan has an important advantage over non-slicing floorplan, which is, there are efficient algorithms to handle shape flexibility in slicing floorplan optimally. A well known approach by Wong et. at [10] uses shape curve to represent all possible shapes of a module. Wang et. al [8] and Moh et. al [2] use numerical optimization methods. Moh et. al [2] formulate the problem as a geometric programming and find its global minimum using some standard convex optimization techniques. However their formulations are all limited to placement topology of rectangular dissection, i.e. slicing.

This area minimization problem becomes much more complicated in non-slicing floorplan. Both Pan et. al [6] and Wang et. al [9] try to generalize Stockmeyer's algorithm [7] to non-slicing floorplan. Kang et. al [1] extend the BSG method [5] to handle soft modules using heuristics. These methods are either sub-optimal or not applicable to all general non-slicing structures. Murata et. al [4] follow the framework of [2] and try to reduce the number of variables and functions when formulating the problem so as to improve the efficiency. However, the execution time of their method to find an exact solution is still very long. It takes seven days to solve a problem with 49 modules.

We will present an efficient method to handle shape flexibilities of soft modules in general floorplans optimally. The problem is formulated as a geometric program but we use

the Lagrangian relaxation technique [3], a general technique for constrained non-linear optimization, to solve the problem efficiently. This technique transforms the problem into a sequence of subproblems called Lagrangian relaxation subproblems. Each subproblem can be significantly simplified by the Kuhn-Tucker conditions. The resulting subproblem is so simple that the size of each module can be found in constant time.

The rest of this paper is organized as follow. We will formulate the problem in the next section. Section three describes briefly the sequence pair floorplanning algorithm. We formulate the geometric program in section four. In section five, we will explain in details the Lagrangian relaxation. Experimental results will be given in the last section.



Slicing Floorplan Non-slicing Floorplan

Figure 1: Slicing and Non-slicing Floorplan

2. PROBLEM FORMULATION

We consider two kinds of modules: hard modules and soft modules. A hard module is a module whose dimension is fixed. A soft module is one whose area is fixed but its dimension can be changed as long as its aspect ratio, i.e. the ratio of height to width, is within a given range. In this problem, we are given n modules of areas A_1, A_2, ..., A_n and their aspect ratio ranges $[r_{1,min}, r_{1,max}]$, $[r_{2,min}, r_{2,max}]$, ..., $[r_{n,min}, r_{n,max}]$. In case of a hard module, the maximum and minimum aspect ratio will be the same.

A *packing* of a set of modules is a non-overlap placement of the modules. A *feasible packing* is a packing such that the widths and heights of the soft modules are consistent with their aspect ratio constraints and area constraints. We measure the area of a packing as the area of the smallest rectangle enclosing all the modules.

We are also given the netlist information: $net_1, net_2, ..., net_m$ and the relative positions of the I/O pins $p_1, p_2, ..., p_q$ along the boundary of the chip. For each net net_i where $1 \leq i \leq m$, we are given its weight and the I/O pin and the set of modules it is connected to. Our objective is to obtain a feasible packing minimizing a linear combination of the total packing area and the interconnect cost.

We use the simulated annealing technique to search the solution space. For each intermediate solution in the annealing process, there can be many different realizations of the packing due to the shape flexibility of the soft modules. The most important contribution of our work is an efficient method to minimize the total area optimally given a certain packing topology described by the vertical and horizontal constraint graphs.

3. SEQUENCE PAIR AND CONSTRAINT GRAPH

We use sequence-pair to represent a general floorplan in the annealing process. A sequence-pair of a set of module is a pair of combinations of the module names. For example, $s = (abcd, bacd)$ is a sequence-pair of the module set $\{a, b, c, d\}$. We can derive the relative positions between the modules from a sequence-pair s by the following rules:

H-constraint: If $s = (\ ..\ a\ ..\ b\ ..,\ ..\ a\ ..\ b\ ..\ )$, then module b is on the right of module a.

V-constraint: If $s = (\ ..\ a\ ..\ b\ ..,\ ..\ b\ ..\ a\ ..\ )$, then module b is below module a.

We can use constraint graphs to represent these horizontal and vertical placement constraints. A horizontal (vertical) constraint graph G_h (G_v) for a set of n modules is a graph of n vertices with the vertices representing the modules and the edges representing the horizontal (vertical) placement constraints. For example, if module b is on the right of module a, we add an edge from a to b in the horizontal constraint graph. Similarly, if module b is above module a, we add an edge from a to b in the vertical constraint graph. We can build these graphs directly from the sequence-pair representation s:

- Add an edge from a to b in the horizontal constraint graph G_h if $s = (\ ..\ a\ ..\ b\ ..,\ ..\ a\ ..\ b\ ..\ )$.
- Add an edge from b to a in the vertical constraint graph G_v if $s = (\ ..\ a\ ..\ b\ ..,\ ..\ b\ ..\ a\ ..\ )$

4. FORMULATION OF THE GEOMETRIC PROGRAM

We are given n modules M_1, M_2, ..., M_n of areas A_1, A_2, ..., A_n. For each module M_i where $1 \leq i \leq n$, its minimum and maximum aspect ratios are $r_{i,min}$ and $r_{i,max}$ respectively. The minimum and maximum width are thus $L_i = \sqrt{A_i / r_{i,max}}$ and $U_i = \sqrt{A_i / r_{i,min}}$ respectively. Let x_i denote the smallest x position of the lower left corner of module i satisfying all the horizontal constraints in the horizontal constraint graph G_h. Similarly, y_i denotes the smallest y position of the lower left corner of module i satisfying all the vertical constraints in the vertical constraint graph G_v. Then for each edge $e(i, j)$ from module i to module j in G_h, we have the following constraint:

$$x_i + w_i \leq x_j$$

where w_i is the width of module i. Similarly, for each edge $e(i, j)$ from module i to module j in G_v, we have the following constraint:

$$y_i + \frac{A_i}{w_i} \leq y_j$$

In the horizontal constraint graph G_h, we denote the set of sources and sinks by s_h and t_h respectively where a source is a vertex without in-coming edge and a sink is a vertex without out-going edge. Similarly, we use s_v and t_v to denote the set of sources and sinks in G_v respectively. Then for each module i in s_h:

$$x_i = 0;$$

and for each module i in s_v:

$$y_i = 0;$$

For simplicity, we add one dummy vertex labeled $n+1$ to each G_h and G_v. Edges $e(i, n+1)$ are added to G_h for all $i \in t_h$ and, similarly, $e(i, n+1)$ are added to G_v for all $i \in t_v$. From now onwards, we assume that the constraint graphs G_h and G_v contain these additional vertices and edges. The problem can be formulated as the following geometric programming PP (Primal Problem):

$$\begin{array}{lll} \text{Minimize} & x_{n+1}y_{n+1} & \\ \text{Subject to} & x_i + w_i \le x_j & \forall e(i,j) \in G_h \quad \text{(A)} \\ & y_i + \frac{A_i}{w_i} \le y_j & \forall e(i,j) \in G_v \quad \text{(B)} \\ & L_i \le w_i \le U_i & \forall 1 \le i \le n \quad \text{(C)} \end{array}$$

5. LAGRANGIAN RELAXATION

According to the Lagrangian relaxation procedure, we can introduce non-negative multipliers, called Lagrange multipliers, to the constraints in order to get rid of those difficult constraints and incorporate them into the objective function. Let $\lambda_{i,j}$ denotes the multiplier for the constraint $x_i + w_i \le x_j$ in (A) and $\mu_{i,j}$ denotes the multiplier for the constraint $y_i + \frac{A_i}{w_i} \le y_j$ in (B). Let $\vec{\lambda}$ and $\vec{\mu}$ be vectors of all the Lagrange multipliers introduced to the constraints in (A) and (B) respectively. Then the Lagrangian relaxation subproblem associated with the multiplier $\vec{\lambda}$ and $\vec{\mu}$, denoted by $LRS/(\vec{\lambda}, \vec{\mu})$, becomes:

$$\begin{array}{ll} \text{Minimize} & x_{n+1}y_{n+1} + \\ & \sum_{e(i,j)\in G_h} \lambda_{i,j}(x_i + w_i - x_j) + \\ & \sum_{e(i,j)\in G_v} \mu_{i,j}(y_i + \frac{A_i}{w_i} - y_j) \\ \text{Subject to} & L_i \le w_i \le U_i \qquad \forall 1 \le i \le n \end{array}$$

Let $Q(\vec{\lambda}, \vec{\mu})$ denotes the optimal value of the problem $LRS/(\vec{\lambda}, \vec{\mu})$. We define the Lagrangian dual problem LDP of PP as follows:

$$\begin{array}{ll} \text{Maximize} & Q(\vec{\lambda}, \vec{\mu}) \\ \text{Subject to} & \vec{\lambda} \ge 0 \text{ and } \vec{\mu} \ge 0 \end{array}$$

Since PP can be transformed into a convex problem, we can apply Theorem 6.2.4 of [3] and imply that if $(\vec{\lambda}, \vec{\mu})$ is the optimal solution to LDP, the optimal solution of $LRS/(\vec{\lambda}, \vec{\mu})$ will also optimize PP.

5.1 Simplification of the Lagrangian Relaxation Subproblem

The Lagrangian relaxation subprogram $LRS/(\vec{\lambda}, \vec{\mu})$ can be greatly simplified using the Kuhn-Tucker conditions. Consider the Lagrangian ζ of PP [3]:

$$\begin{aligned} \zeta &= x_{n+1}y_{n+1} + \sum_{e(i,j)\in G_h} \lambda_{i,j}(x_i + w_i - x_j) + \\ &\quad \sum_{e(i,j)\in G_v} \mu_{i,j}(y_i + \frac{A_i}{w_i} - y_j) + \\ &\quad \sum_{1\le i\le n} u_i(L_i - w_i) + \sum_{1\le i\le n} v_i(w_i - U_i) \\ &= x_{n+1}y_{n+1} - \sum_{e(i,n+1)\in G_h} \lambda_{i,n+1}x_{n+1} - \\ &\quad \sum_{e(i,n+1)\in G_v} \mu_{i,n+1}y_{n+1} + \\ &\quad \sum_{1\le i\le n} (\sum_{e(i,j)\in G_h} \lambda_{i,j} - \sum_{e(j,i)\in G_h} \lambda_{j,i})x_i + \\ &\quad \sum_{1\le i\le n} (\sum_{e(i,j)\in G_v} \mu_{i,j} - \sum_{e(j,i)\in G_v} \mu_{j,i})y_i + \\ &\quad \sum_{1\le i\le n} ((\sum_{e(i,j)\in G_h} \lambda_{i,j})w_i + (\sum_{e(i,j)\in G_v} \mu_{i,j})\frac{A_i}{w_i}) + \\ &\quad \sum_{1\le i\le n} u_i(L_i - w_i) + \sum_{1\le i\le n} v_i(w_i - U_i) \end{aligned}$$

The Kuhn-Tucker conditions imply that $\partial\zeta/\partial x_i = 0$ and $\partial\zeta/\partial y_i = 0$ for all $1 \le i \le n+1$ at the optimal solution of PP. Therefore, in searching for the $\vec{\lambda}$ and $\vec{\mu}$ to optimize LDP, we only need to consider those multipliers such that these conditions are satisfied. Therefore for all $1 \le i \le n$:

$$\begin{aligned} \partial\zeta/\partial x_i &= \sum_{e(i,j)\in G_h} \lambda_{i,j} - \sum_{e(j,i)\in G_h} \lambda_{j,i} = 0 \\ \partial\zeta/\partial y_i &= \sum_{e(i,j)\in G_v} \mu_{i,j} - \sum_{e(j,i)\in G_v} \mu_{j,i} = 0 \end{aligned}$$

and

$$\begin{aligned} \partial\zeta/\partial x_{n+1} &= y_{n+1} - \sum_{e(i,n+1)\in G_h} \lambda_{i,n+1} = 0 \\ \partial\zeta/\partial y_{n+1} &= x_{n+1} - \sum_{e(i,n+1)\in G_v} \mu_{i,n+1} = 0 \end{aligned}$$

Rearrange:

$$\sum_{e(j,i)\in G_h} \lambda_{j,i} = \sum_{e(i,j)\in G_h} \lambda_{i,j} \quad (1)$$

$$\sum_{e(j,i)\in G_v} \mu_{j,i} = \sum_{e(i,j)\in G_v} \mu_{i,j} \quad (2)$$

and

$$y_{n+1} = \sum_{e(i,n+1)\in G_h} \lambda_{i,n+1}$$
$$x_{n+1} = \sum_{e(i,n+1)\in G_v} \mu_{i,n+1}$$

We use Ω to denote the set of $(\vec{\lambda},\vec{\mu})$ satisfying the above relationship ((1) and (2)) for a given pair of horizontal and vertical constraint graphs. If $(\vec{\lambda},\vec{\mu}) \in \Omega$, the objective function F of $LRS/(\vec{\lambda},\vec{\mu})$ becomes:

$$F = \sum_{1\le i\le n} ((\sum_{e(i,j)\in G_h} \lambda_{i,j})w_i + (\sum_{e(i,j)\in G_v} \mu_{i,j})\frac{A_i}{w_i}) - \sum_{e(i,n+1)\in G_h} \lambda_{i,n+1}x_{n+1} - \sum_{e(i,n+1)\in G_v} \mu_{i,n+1}y_{n+1} + (\sum_{e(i,n+1)\in G_h} \lambda_{i,n+1})(\sum_{e(i,n+1)\in G_v} \mu_{i,n+1})$$

where $(\sum_{e(i,n+1)\in G_h} \lambda_{i,n+1})(\sum_{e(i,n+1)\in G_v} \mu_{i,n+1})$ is a constant for a fixed $(\vec{\lambda},\vec{\mu})$.

5.2 Solving $LRS/(\vec{\lambda},\vec{\mu})$

In this section, we consider solving the Lagrangian relaxation subproblem $LRS/(\vec{\lambda},\vec{\mu})$ when $(\vec{\lambda},\vec{\mu}) \in \Omega$, i.e. computing w_i for $1 \le i \le n$. F can be written as:

$$F = k + \sum_{1\le i\le n} ((\sum_{e(i,j)\in G_h} \lambda_{i,j})w_i + (\sum_{e(i,j)\in G_v} \mu_{i,j})\frac{A_i}{w_i})$$

where $k = (\sum_{e(i,n+1)\in G_h} \lambda_{i,n+1})(\sum_{e(i,n+1)\in G_v} \mu_{i,n+1})$ is a constant. Differentiate F with respect to w_i in order to get the optimal value of w_i to minimize F:

$$\frac{\partial F}{\partial w_i} = 0$$
$$\sum_{e(i,j)\in G_h} \lambda_{i,j} - \frac{A_i}{w_i^2} \sum_{e(i,j)\in G_v} \mu_{i,j} = 0$$
$$w_i = \sqrt{\frac{A_i \times \sum_{e(i,j)\in G_v} \mu_{i,j}}{\sum_{e(i,j)\in G_h} \lambda_{i,j}}}$$

Recall that w_i must lie within the range $[L_i, U_i]$. Let w_i^* denote $\sqrt{\frac{A_i \times \sum_{e(i,j)\in G_v} \mu_{i,j}}{\sum_{e(i,j)\in G_h} \lambda_{i,j}}}$. Since $\partial F/\partial w_i$ is positive for $w_i < w_i^*$ and negative for $w_i > w_i^*$, the optimal w_i can be computed as:

$$w_i = \min\{U_i, \max\{L_i, w_i^*\}\}$$

The algorithm *Find-Width* outlines the steps to solve $LRS/(\vec{\lambda},\vec{\mu})$:

Algorithm *Find-width*
/* This algorithm solves $PP((\vec{\lambda},\vec{\mu})$ optimally given $(\vec{\lambda},\vec{\mu}) \in \Omega$ */
Input: Areas A_1, A_2, ..., A_n
Lower bounds of widths L_1, L_2, ..., L_n
Upper bounds of widths U_1, U_2, ..., U_n
Constraint graphs G_v and G_h
Lagrange multipliers $(\vec{\lambda},\vec{\mu}) \in \Omega$
Output: Widths w_1, w_2, ..., w_n
1. For $i = 1$ *to* n
2. $sum_1 = sum_2 = 0$
3. For all $e(i,j) \in G_h$
4. Compute $sum_1 = sum_1 + \lambda_{i,j}$
5. For all $e(i,j) \in G_v$
6. Compute $sum_2 = sum_2 + \mu_{i,j}$
7. If $(sum_1 \neq 0)$ and $(sum_2/sum_1 \geq 0)$
7. Compute $w^* = \sqrt{A_i * sum_2/sum_1}$
8. $w_i = \min\{U_i, \max\{L_i, w^*\}\}$

5.3 Solving LDP

As explained above, we only need to consider those $(\vec{\lambda},\vec{\mu}) \in \Omega$ in order to maximize $Q(\vec{\lambda},\vec{\mu})$ in the LDP problem. We used a subgradient optimization method to search for the optimal $(\vec{\lambda},\vec{\mu})$. Starting from an arbitrary $(\vec{\lambda},\vec{\mu}) \in \Omega$ in step k, we will move to a new pair $(\vec{\lambda'},\vec{\mu'})$ by following the subgradient direction:

$$\lambda'_{i,j} = [\lambda_{i,j} + \rho_k(x_i + w_i - x_j)]^+$$
$$\mu'_{i,j} = [\mu_{i,j} + \rho_k(y_i + \frac{A_i}{w_i} - y_j)]^+$$

where

$$[x]^+ = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{if } x \le 0. \end{cases}$$

and ρ_k is a step size such that $lim_{k\to\infty}\rho_k = 0$ and $\sum_{k=1}^{\infty}\rho_k = \infty$. After updating $\vec{\lambda}$ and $\vec{\mu}$, we will *project* $(\vec{\lambda'},\vec{\mu'})$ back to the nearest point $(\vec{\lambda^*},\vec{\mu^*})$ in Ω and solve the Lagrangian relaxation subproblem $LRS/(\vec{\lambda^*},\vec{\mu^*})$ using the method described in section 5.2. This procedure is repeated until the solution converges. The following algorithm summarizes the steps to solve LDP:

Algorithm *Solve-LDP*
/* This algorithm solves the LDP problem optimally. Given the placement topology described by the constraint graphs, it computes the optimal values for the widths of the modules to minimize the total packing area. */
Input: Areas A_1, A_2, ..., A_n
Lower bounds of widths L_1, L_2, ..., L_n
Upper bounds of widths U_1, U_2, ..., U_n
Constraint graphs G_v and G_h
Output: Widths w_1, w_2, ..., w_n
1. Initialize $(\vec{\lambda},\vec{\mu})$ and ρ_1

2. $k = 1$
3. Repeat
4. Call *Find-width()*
5. Compute $x_i, y_i \ \forall 1 \leq i \leq n+1$ using the longest path algorithm
6. Compute $\lambda'_{i,j} = [\lambda_{i,j} + \rho_k(x_i + w_i - x_j)]^+$ $\forall e(i,j) \in G_h$
7. Compute $\mu'_{i,j} = [\mu_{i,j} + \rho_k(y_i + \frac{A_i}{w_i} - y_j)]^+$ $\forall e(i,j) \in G_v$
8. Project $(\vec{\lambda'}, \vec{\mu'})$ to $(\vec{\lambda^*}, \vec{\mu^*})$ such that $(\vec{\lambda^*}, \vec{\mu^*}) \in \Omega$
9. $k = k + 1$
10. $(\vec{\lambda}, \vec{\mu}) = (\vec{\lambda^*}, \vec{\mu^*})$
11. Until w_i's converge

5.4 Projection

As described above, we used a subgradient optimization method to search for the optimal $(\vec{\lambda}, \vec{\mu})$. Starting from an arbitrary $(\vec{\lambda}, \vec{\mu}) \in \Omega$, we will move to a new pair $(\vec{\lambda'}, \vec{\mu'})$ by following the subgradient direction. $(\vec{\lambda'}, \vec{\mu'})$ will then be projected back to the nearest point $(\vec{\lambda^*}, \vec{\mu^*})$ in Ω based on the 2-norm measure. This projection step is done by finding an orthonormal bases $\vec{\lambda_1}, \ldots, \vec{\lambda_p}, \vec{\mu_1}, \ldots, \vec{\mu_q}$ of Ω. Then

$$\vec{\lambda^*} = \sum_{i=1}^{p} (\vec{\lambda'} \cdot \vec{\lambda_i}) \vec{\lambda_i}$$

$$\vec{\mu^*} = \sum_{i=1}^{q} (\vec{\mu'} \cdot \vec{\mu_i}) \vec{\mu_i}$$

To find the orthonormal bases spanning Ω, we first find a set I of independent vectors spanning Ω using the Gaussian Elimination. Then we apply the Gram-Schmidt process to obtain the orthonormal bases from I. Notice that in equation (1) and (2), each variable will appear at most twice and their coefficients are either 1 or -1, so the Gaussian Elimination step takes $O(n^2)$ time, instead of $O(n^3)$, where n is the total number of modules and there is no floating point division throughout the whole process. In addition, the structure of the constraint graphs remains unchanged if we exchange two modules in a move, so we do not need to re-compute the orthonormal bases in those cases.

6. EXPERIMENTAL RESULTS

We tested our floorplanner on a set of MCNC benchmark data. For each experiment, the initial temperature is decided such that an acceptance ratio is 95% at the beginning. The temperature is lowered at a constant rate (0.9) and the number of iterations at one temperature step is a constant. All the experiments were carried out on a 270 MHz Sun Ultra 5.

We did two sets of experiments. In the first set, we allow the aspect ratio of each module to range from 0.1 to 10.0 in order to compare the results with [4]. Table 1 shows the comparison. For ami33, it takes us about 3 hours to obtain a packing with 1.2% deadspace while the result given by [4] is about 21 hours. For ami49, our approach needs about 19.5 hours while [4] records about 7 days. We can see that our method is much more useful practically in terms of running time and the packing quality is also good.

In the second set of experiments, we allow the aspect ratio of each module to range from 0.5 to 2.0. $[0.5, 2]$ is a more reasonable range and this can better demonstrate the speed and quality of the floorplanning algorithm. Table 2 display the result. Figure 2 and 3 show two result packings.

		Our Method		[4]
Data	n	Deadspace %	Time (sec)	Time (sec)
xerox	10	0.0	54	789
apte	9	0.0	41	1198
hp	11	0.2	71	1346
ami33	33	1.2	10628	75684
ami49	49	3.1	70234	612103

Table 1: Comparing with the results in [4]

Data	n	% Deadspace (%)	Time (sec)
xerox	10	0.0	63
apte	9	0.1	43
hp	11	1.3	108
ami33	33	2.0	13899
ami49	49	6.1	96815

Table 2: Results of testing with the benchmark data using aspect ratio bound $[0.5, 2.0]$

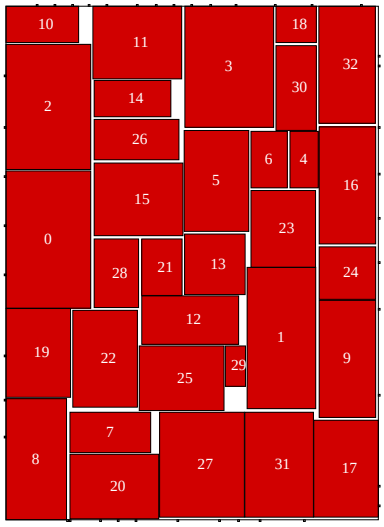


Figure 2: A result packing of ami33 with aspect ratio bound $[0.5, 2.0]$**. It has 2% deadspace.**

7. REMARKS

In this paper, we presents a method to minimize area in general non-slicing floorplan. We formulate the problem as a convex program and simplify it using Lagrangian Relaxation. Experimental results show that our approach works much faster than previous methods and gives high quality packings. The step size ρ for the subgradient optimization

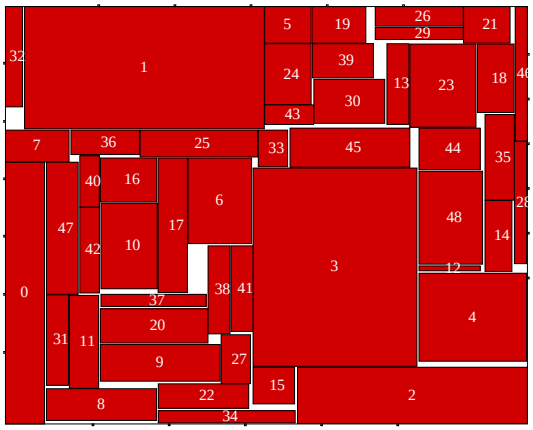


Figure 3: A result packing of ami49 with aspect ratio bound $[0.1, 10.0]$. It has 3% deadspace.

of $\vec{\lambda}$ and $\vec{\mu}$ is an important factor affecting the packing quality. We are interested in finding appropriate value for this parameter automatically. Besides, the implementation can be further improved to increase the speed. We will continue to work on these issues in the future.

8. ACKNOWLEDGEMENT

We would like to thank Professor Martin D.F. Wong for his helpful advices.

9. REFERENCES

[1] M. Kang and W. W.M. Dai. General Floorplanning with L-shaped, T-shaped and Soft Blocks Based on Bounded Slicing Grid Structure. *IEEE Asia and South Pacific Design Automation Conference*, pages 265–270, 1997.

[2] T.-S. Moh, T.-S. Chang, and S. L. Hakimi. Globally Optimal Floorplanning for a Layout Problem. *IEEE Transaction on Circuit and Systems - I: Fundamental Theory and Applications*, 43(9):713–720, 1996.

[3] M.S. Bazaraa and H.D. Sherali and C.M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, Inc., second edition, 1997.

[4] H. Murata and Ernest S. Kuh. Sequence-Pair Based Placement Method for Hard/Soft/Pre-placed Modules. *International Symposium on Physical Design*, pages 167–172, 1998.

[5] S. Nakatake, K. Fujiyoushi, H. Murata, and Y. Kajitani. Module Placement on BSG-Structure and IC Layout Applications. *Proceedings IEEE International Conference on Computer-Aided Design*, pages 484–491, 1996.

[6] Peichen Pan and C.L. Liu. Area Minimization for General Floorplans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 606–609, 1992.

[7] L. Stockmeyer. Optimal Orientations of Cells in Slicing Floorplan Designs. *Information and Control*, 59:91–101, 1983.

[8] Ting-Chi Wang and D.F. Wong. An Optimal Algorithm for Floorplan Area Optimization. *Proceedings of the ACM/IEEE Design Automation Conference*, pages 180–186, 1990.

[9] Ting-Chi Wang and D.F. Wong. Optimal Floorplan Area Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2(8):992–1001, 1992.

[10] D.F. Wong and C.L. Liu. A New Algorithm for Floorplan Design. *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pages 101–107, 1986.