Floorplan Sizing by Linear Programming Approximation

Pinhong Chen
EECS Dept.
Univ. of California at Berkeley
Berkeley, CA94720, USA
pinhong@eecs.berkeley.edu

Ernest S. Kuh
EECS Dept.
Univ. of California at Berkeley
Berkeley, CA94720, USA
kuh@eecs.berkeley.edu

ABSTRACT

In this paper, we present an approximation algorithm by linear programming(*LP*) for floorplan sizing problem. Given any topological constraints between blocks, we can formulate it as an LP problem with a cost function for the minimum bounding box area. Unlike slicing structures, this approach can handle any topological constraints as well as soft/hard/preplaced blocks, and timing constraints. Empirically, our method needs few iterations to find the optimum solution and shows one order of improvement over previous methods both in run time and capability to handle a larger problem size even on a very limited computing resource PC.

1. INTRODUCTION

Floorplanning is essential for a chip performance, which is in terms of the chip size and timing. It is more and more important for a hierarchical layout style used in very deep sub-micron designs. The issues have been studied([1] for an overview) for more than a decade. In a floorplan, besides the topological relationship between blocks, it is important to determine the aspect ratio of each module such that the total area is minimized. In very deep sub-micron designs, the issue of timing brings another dimension of complexity, which is to meet the timing constraints of a design.

For slicing structures[1], it is easy to determine the aspect ratio for each block. However, the topology it inherits is very restricted; moreover, preplaced blocks and timing constraints are not so easy to handle in a single framework. Here we focus on general topological constraints so that any topology can be handled properly.

In [2; 3], the authors propose a $ln(\cdot)$ function transformation to formulate it as a convex programming, and they show the convexity of this floorplan sizing problem. However, the size of the problem it can handle is limited due to its computation-intensive convex programming. In [4], the authors use the analogy between the resistive network and a floorplan. The algorithm they use is an unconstrained optimization with a penalty function. In [5], the authors propose an effective way to deal with soft/hard/pre-placed blocks using a constrained optimization algorithm for convex programming.

All of the previous works suffer from the high complexity to solve a convex programming problem. Also, none of them proposes any approach to handling timing constraints. In this paper, we propose linearization of area constraints and introduce new variables to handle timing constraints. Soft/hard/pre-placed blocks are also considered in one framework. The underlying computation formulation is just linear programming, and we prove how to obtain the global optimum by solving several linear programs.

This paper is organized as follows. In Section 2, we describe the definitions used in this paper, and define the floorplan sizing problem. Then, it is approximated as a linear programming problem in Section 3. We discuss how to search for a local and the global minimum in Section 4. Experimental results are shown in Section 5.

2. DEFINITION

A *block* is defined as a rectangle. A *soft block* is defined as a rectangle with area constraint but its width, w_i, and height, h_i, can be flexible or under some aspect ratio constraints. A *hard block* is defined as a rectangle with fixed width and height. A *preplaced block* is defined as a rectangle with fixed X and Y coordinates. A *packing* is referred to as a non-overlapping placement of a set of blocks $B = \{B_1, B_2, \ldots, B_n\}$. Let (x_i, y_i) be the center coordinate of block B_i. A *horizontal constraint* for two blocks is that the difference of the X coordinates of the two blocks must be greater than or equal to some distance so that the two blocks are not overlapping, such as shown in Figure 1, where x_i and x_j are the center X coordinates

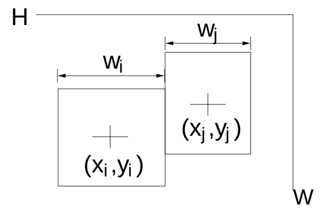

Figure 1: Horizontal Constraint

of block B_i and B_j respectively, and w_i and w_j are the width of block B_i and B_j, respectively. The horizontal constraint imposes the inequality $x_j - x_i \geq (w_i + w_j)/2$. A *vertical constraint* is similarly defined. These horizontal/vertical(H/V) constraints form a *H/V constraint graph*, in which each block is represented by a vertex, and the weighted edges represent the X(Y) distance that the two blocks must keep from each other horizontally(vertically). We refer to the H/V constraint graphs as the topological constraints.

Each block is also a *timing node* if there are some nets routed to it. Each timing node is associated with a timing variable, t_i, which represents the latest signal arrival time. If there is some signal coming from timing node j to timing node i, we have

$$t_j + d_j + k|x_i - x_j| + k|y_i - y_j| \le t_i \qquad (1)$$

where d_j is the inertial output delay for block B_j, and k is a constant factor for transforming from distance to time domain. A latest arrival time requirement therefore imposes on some timing nodes as $t_i \le TC_i$ where TC_i is a given required latest arrival time. We call this inequality as well as Eq. 1 as *timing constraints*.

2.1 Floorplan Sizing Problem

Floorplan Sizing Problem(FSP):
Given a set of soft block, a set of hard block, and a set of preplaced block, optimize the bounding rectangle area such that every block is inside it without any overlap. For ease of reference, we call FSP a *TFSP* if it is required to meet timing constraints.

We assume here the topological constraints have been given, i.e., H/V constraint graphs have been given. Thus, we focus on the sizing problem and honor the given topological constraints. Also, we assume the timing constraints can be formulated just related to linear distance only. The floorplan sizing is also subject to these timing constraints.

3. LINEAR PROGRAMMING APPROXIMATION

This problem can be simply formulated as a non-convex quadratic program exactly. However, it is not easy to solve a non-convex quadratic programming problem. It is known that a general quadratic program is an NP-complete problem[6]. However, this sizing problem formulation can be further rewritten as a convex programming problem proposed by [2] with a simple transformation. Solving convex programming is also no easy task, since it takes lots of computation time, and the size of the convex program can not handle too many blocks efficiently.

3.1 Area Constraint Approximation

First, observe that the block area constraint, $w_i h_i = a_i$,can be changed as an inequality constraints, that is, $w_i h_i \ge a_i$. If the space is allocated for more than the required amount, it is still feasible for layout implementation, so this transformation is valid. Second, we linearize it by Tayor expansion as

$$h_i \ge h_i^{(0)} + \frac{h_i^{(0)}}{w_i^{(0)}}(w_i - w_i^{(0)}) \qquad (2)$$

This approximation can be done at several $w^{(j)}$ values, and the figure looks like Figure 2. The area bounded by these inequality constraints forms the approximation of $w_i h_i \ge a_i$. Also, given an error bound ε, we can approximate the area constraint by a set of linear inequalities, and the resulting **area error** can be bounded by ε for some ranges of w_i and h_i, since the maximum area error always occurs at the cross point of two lines.

LEMMA 1. *The area constraint of each module can be approximated by linear inequalities.*

After linear approximation of the area constraints, we refer to FSP as a *Relaxed Floorplanning Sizing Problem(RFSP)*. RFSP is therefore defined on a linear constrained set, which is a convex set[7].

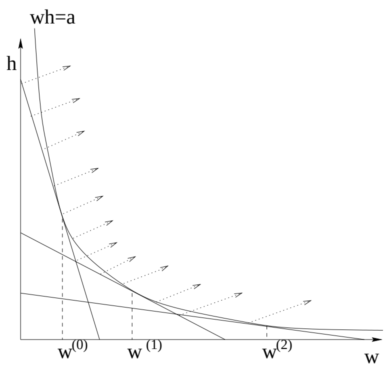

Figure 2: Block Area Constraint Approximation

3.2 Transformation of Timing Constraints

It is generally not easy to eliminate an absolute value function, since two conditions have to be considered, and it becomes an exponential number of cases to check. However, the timing constraint (1) can be transformed into a regular linear inequality by removing absolute function as $x_i - x_j \le M_{ij}$ and $x_j - x_i \le M_{ij}$ Then, we can replace $|x_i - x_j|$ by M_{ij}. The value of M_{ij} can be greater than $|x_i - x_j|$, but it is not less. It does not matter since the timing constraint bounds the maximum value of t_i and recursively constrains M_{ij}. Actually, $k(M_{ij} - |x_i - x_j|)$ can be regarded as a timing slack. By modifying the objective function, it is also possible to optimize for timing slacks instead of timing constraints.

4. FINDING THE OPTIMUM SOLUTION

4.1 Convex Solution Space

THEOREM 1. *The feasible solution of RFSP is defined on a convex set. And, the projection of the solution space on W-H plane is also convex.*

(Proof:) Since the constraints are all linear, the solution space of RFSP is a convex set. The projection of solution space on W-H plane does not change the convexity. □
Let β be the projection of solution space on W-H plane, and it looks like Figure 3.

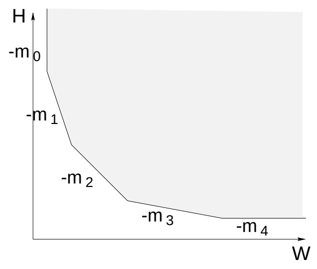

Figure 3: Projection of Solution Space

COROLLARY 1. *The piecewise linear boundary on W-H plane forms an ordered sequence,*

$$-m_0 \le -m_1 \le \ldots \le -m_n$$

where $-m_i$ is the slope for line segment i and $m_i \ge 0$.

LEMMA 2. *The optimal solution (W^*, H^*) of RFSP must be on the boundary of β.*

(Proof:)Suppose (W^*, H^*) is not on the boundary of β. We always can find either (W', H^*) or (W^*, H') with smaller costs, where $W' < W^*$ and $H' < H^*$. □

Given a point (W_0, H_0) along a line with slope $-m$, where $m > 0$, we compute the cost function change when there is a ΔW. Specifically, we have

$$\begin{aligned} f(W,H) &= W \cdot H, \qquad \Delta H = -m\Delta W \\ \Delta f &= f(W_0 + \Delta W, H_0 + \Delta H) - f(W_0, H_0) \\ &= W_0 \Delta H + H_0 \Delta W + \Delta H \Delta W \\ &= \Delta W(-mW_0 + H_0 - m\Delta W) \qquad (3) \end{aligned}$$

Therefore, $\Delta f > 0$ if

$$0 < \Delta W < \frac{H_0}{m} - W_0, \text{ and } \frac{H_0}{W_0} > m \qquad (4)$$

or

$$0 > \Delta W > \frac{H_0}{m} - W_0, \text{ and } \frac{H_0}{W_0} < m \qquad (5)$$

THEOREM 2. *The optimal solution of RFSP must be on the cross points of the piecewise-linear boundary of β.*

(Proof:) Suppose the minimum point is on a line segment with slope $-m$. Δf is required to be greater than 0 for $\Delta W > 0$ and $\Delta W < 0$. According to (4) and (5), it can not happen at the same m. That is to say, one of the two sides must be decreasing. With Lemma 2, the theorem follows. □

COROLLARY 2. *A sufficient condition for local minimum is $-m_i \le -\frac{H^*}{W^*} \le -m_{i+1}$, where $-m_i$ and $-m_{i+1}$ are the slopes of two consecutive line segments on the boundary of β.*

4.2 Cost Function Approximation

Here we approximate the cost function $W \times H$ by linear approximation(Taylor expansion):

$$\begin{aligned} f(W,H) &= W_0H_0 + H_0(W - W_0) \\ &+ W_0(H - H_0) + (W - W_0)(H - H_0) \qquad (6) \\ &\approx W_0H_0 + H_0(W - W_0) + W_0(H - H_0) \qquad (7) \end{aligned}$$

Since it is a cost function, we can delete all the constant terms:

$$\min f(W,H) \approx \min \hat{f} = \min H_0W + W_0H$$

If $(W^* - W_0)(H^* - H_0)$ is small enough, this is a good approximation.

4.3 Finding a Local Minimum

The cost function approximation above thus implies a simple algorithm for minimization:

ALGORITHM 1. *Starting from some initial value (W_0, H_0), solve a linear programming problem with cost function $H_0W + W_0H$, replace (W_0, H_0) by the optimal solution of LP, (W^*, H^*), and repeat solving the LP problem until it converges.*

LEMMA 3. *The search algorithm in Algorithm 1 finds a local minimum.*

(Proof:)The optimal solution of LP for the solution space like β is always on the cross point of 2 line segments of the boundary of β. The slopes form a relation $-m_i \le -\frac{H^*}{W^*} \le -m_{i+1}$, where $-m_i$ and $-m_{i+1}$ are the slopes of two consecutive line segments. This condition matches the sufficient condition in Corollary 2. Also, it can be shown Algorithm 1 finds the solution monotonically decreasing at each step. Hence, the algorithm finds a local minimum. □

4.4 Finding the Global Minimum

A local minimum may not be satisfactory. It is desired to know how far we are from the global minimum given some local minima.

THEOREM 3. *Given two local minima (W_a, H_a) and (W_b, H_b), such as shown in Figure 4, the lower bound of the cost function inside the triangle a, b and t is $W_t \cdot H_t$, where (W_t, H_t) is the cross point of line L_a and L_b with slopes $-H_a/W_a$ and $-H_b/W_b$, respectively.*

(Proof:)Suppose we have a point $(W_a + \Delta W, H_a + \Delta H)$ moving along L_a. Since L_a has a slope $-H_a/W_a$, we have $\Delta H = -(H_a/W_a)\Delta W$, so

$$\Delta f = (W_a + \Delta W)(H_a + \Delta H) - W_aH_a = -\frac{H_a}{W_a}\Delta W^2$$

That is to say, point t has a minimum value along line segment from point a to point t. It is similar for the line segment from point b to point t. Following the argument similar to Lemma 2, we have the minimum value over the triangle a, b and t. □

This theorem implies a search algorithm for finding the global minimum.

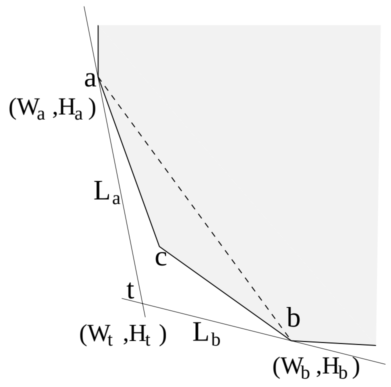

Figure 4: The lower bound of the cost function between two minimum points

ALGORITHM 2. *Use two initial conditions $W_0 \ll H_0$ and $W_1 \gg H_1$ to search for two local minima $a = (W_a, H_a)$ and $b = (W_b, H_b)$ by Algorithm 1. Compute the lower bound using Theorem 3. If the lower bound is close enough to either one of two local minima, output the optimum solution. Otherwise, use Algorithm 1 with initial cost function $(H_a - H_b)W + (W_b - W_a)H$ to find a local minimum $c = (W_c, H_c)$. Repeat the previous steps and recursively search the regions between a and c and the region between c and b, using most updated lower bound to terminate redundant computations. See Figure 4 for graphical interpretation.*

At the first step, this algorithm reaches two local minima, if the two local minimum points are close enough or the lower bound computed by these two minimum points is close to one of the two minimum values, then it is the solution, or we can recursively search the two regions separated by c in Figure 4, and bound the computation by Theorem 3 until the accuracy is acceptable.

5. EXPERIMENTAL RESULTS

We test our algorithm on several artificial problems and MCNC benchmark circuits. The platform is based on a Linux OS system on a 233Mhz Pentium PC with 64M bytes memory. An LP solver, LPAKO 4.3f(`http://ORLAB.snu.ac.kr/`), is used to solve the linear programming problem, and a simple script (Perl code) is used to generate the constraints and interface with the LP solver.

We do some passes of 1-D compaction for each case before running Algorithm 2. Each module has an aspect ratio constraint, $0.3333 \leq h_i/w_i \leq 3$. The total area of the blocks in a circuit is normalized to 1. All blocks are soft for ease of comparison, since soft blocks dominate the run time. The results are shown in Table 1, where the first column is the name of the circuit, the second column is the number of soft blocks, the third column is the number of linear inequality constraints, the 4th column is the total number of LP runs, the 5th column is the total run time, the 6th column is the total area error due to the approximation of area constraints, and the last column shows the input normalized area versus the output normalized area. The area errors arise from the linear constrained set to approximate a block area Eq. 2. It is possible to bound it , or reduce it by more approximating lines. An alternative approach could be used to conservatively approximate the block area by designing the cross points of lines falling on the hyperbola. Atpe, C20, C100 and C200 have an output area less than 1.00 due to the area constraint approximation. All the area constraints are approximated by 5 linear inequalities. All the results shown are at the stopping criteria that the final result can not be greater than the lower bound by 3%. More than 500 blocks could be done on a more powerful platform without any problem. The LP solver capability is the major factor for the size of the problems it can handle.

Ckt	#Blks	#Cntr	#LP Runs	Run Time	Area Error	In./Out. Area
xerox	10	148	12	4s	1.07%	1.24/1.00
atpe	10	147	10	4s	1.08%	1.11/0.96
hp	11	165	11	4s	1.13%	1.25/1.02
c11	11	167	15	6s	1.79%	1.19/1.07
c20	20	343	11	8s	2.91%	1.23/0.98
ami33	33	656	11	18s	1.53%	1.31/1.03
ami49	49	1072	10	34s	0.87%	1.21/1.00
c55	55	1312	13	57s	1.75%	1.24/1.09
c100	100	2806	11	185s	1.81%	1.31/0.99
c200	200	6504	8	636s	1.75%	1.83/0.99
c400	400	19458	10	5345s	1.47%	1.22/1.00
c500	500	23011	10	7684s	1.78%	2.49/1.17

Table 1: The results of Algorithm 2

Compared with the results shown in [2], although our platform and memory space is much more limited, our results still shows 3X to 5X improvement and no difficulty for 500 blocks, while the largest size of problem they can handle is just 106 blocks and it also suffers from no feasible solution found sometimes due the nonlinear programming initialization. Compared with the results shown in [5], their run time is almost one order greater than our results, even if they use much more powerful platform and larger memory space.

One result of Algorithm 2 on the experimental circuits is shown in Figure 5.

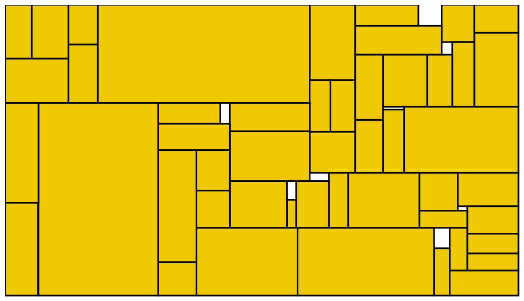

Figure 5: The result for MCNC benchmark circuit ami49.

6. FUTURE WORK

More aggressive approach may be integration of the LP solver and Algorithm 1 and 2, since the LP runs are always on the same constrained space and only the cost function is changed in Algorithm 1. It is also possible to use some LP techniques like parameterized LP, in which the width of a chip is taken as a parameter, and the min function of the height can be derived as a piecewise linear and convex function of the width[7]. For that many constraints, it is possible to add constraints when it is necessary to reduce the LP solving work. The dual simplex method is ideal for this[7].

7. SUMMARY AND CONCLUSION

We propose an efficient and powerful algorithm to solve the floorplan sizing problem, and only linear programming is required to find the global optimum. Also, our method can handle not just soft/hard/pre-placed blocks but also consider timing constraints without any problem. The problem size we can handle and run time is shown favorably compared with any existing approaches.

8. REFERENCES

[1] N. Sherwani. *"Algorithms for VLSI Physical Design Automation"*. Kluwer Academic Publishers, 1995.

[2] T. S. Moh, T. S. Chang, and S.L.Hakimi. "Globally optimal floorplanning for a layout problem". *IEEE Trans. on Circuit and Systems - I: Fundamental Theory and Applications*, Vol.43:pp.713–720, Sep. 1996.

[3] T. Chen and M. K. H. Fan. "On Convex Formulation of the Floorplan Area Minimization Problem". In *Proc. of International Symposium of Physical Design*, pages 124–128, Apr. 1998.

[4] K. Wang and W.K. Chen. "Floorplan Area Optimization using Network Analogous Approach". In *Proc. of IEEE International Symposium on Circuits and Systems*, pages 167–170, 1995.

[5] H. Murata and E. S. Kuh. "Sequence-Pair Based Placement Method for Hard/Soft/Pre-placed Modules". In *Proc. of International Symposium of Physical Design*, pages 167–172, Apr. 1998.

[6] M. R. Garey and D. S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*. W. H. Freeman, 1979.

[7] C. H. Papadimitriou and K. Steiglitz. *"Combinatorial Optimization-Algorithms and Complexity"*. Prentice-Hall, Inc., 1982.