A Revisit to Floorplan Optimization by Lagrangian Relaxation*

Chuan Lin
Magma Design Automation
Santa Clara, CA 95054
clin@magma-da.com

Hai Zhou
EECS Department
Northwestern University
Evanston, IL 60208
haizhou@eecs.northwestern.edu

Chris Chu
ECE Department
Iowa State University
Ames, IA 50011
cnchu@iastate.edu

ABSTRACT

With the advent of deep sub-micron (DSM) era, floorplanning has become increasingly important in physical design process. In this paper we clarify a misunderstanding in using Lagrangian relaxation for the minimum area floorplanning problem. We show that the problem is not convex and its optimal solution cannot be obtained by solving its Lagrangian dual problem. We then propose a modified convex formulation and solve it using min-cost flow technique and trust region method. Experimental results under module aspect ratio bound [0.5,2.0] show that the running time of our floorplanner scales well with the problem size in MCNC benchmark. Compared with the floorplanner in [27], our floorplanner is 9.5X faster for the largest case "ami49". It also generates a floorplan with smaller deadspace for almost all test cases. In addition, since the generated floorplan has an aspect ratio closer to 1, it is more friendly to packaging. Our floorplanner is also amicable to including interconnect cost and other physical design metrics.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*;
J.6 [**Computer-Aided Engineering**]: Computer-Aided Design

General Terms

Algorithms, Design

Keywords

Floorplan, Lagrangian relaxation

1. INTRODUCTION

Floorplanning is an early stage of physical design that determines the positions, shapes and orientations of circuit modules. In contrast to placement that assumes hard modules, more flexibilities are available at floorplanning stage since detailed physical information is not fixed yet. With the advent of deep sub-micron (DSM) era, hierarchical design and IP (module reuse) based methodology have been widely adopted. They both make floorplanning more and more important in physical design process.

There are two categories of floorplan: slicing and non-slicing. Slicing floorplan can be obtained by recursively cutting rectangles horizontally or vertically into small rectangles. Non-slicing floorplan is not restricted to be slicing and thus more general. Figure 1 illustrates an example of each. Among many representations of slicing floorplan, there are binary tree [19] and normalized Polish expression [25]. For non-slicing structure, there are topology representation (BSG [17], sequence pair [15], TCG [12], ACG [29]), packing representation (O-tree [9], B*-tree [4]), and mosaic representation (CBL [10], Q-sequence [21], twin binary tree [26], twin binary sequence [28]), etc.

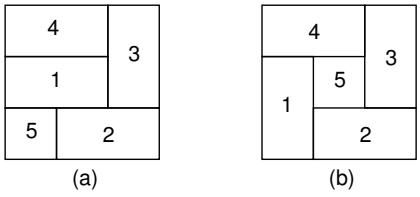

Figure 1: (a) A slicing floorplan. (b) A non-slicing floorplan.

Various ways have been proposed to construct a floorplan from a given representation. For example, a constraint graph can be generated from a sequence pair [15] to characterize the geometrical relations (left, right, up, and down) between every two modules. One commonly used metric that measures the quality of a floorplan is its area. If all modules are hard, the min-area floorplan can be obtained by a longest path computation on the constraint graph. However, when soft modules with flexible shapes are involved, as is the common situation at floorplanning stage, determining the shape for each module such that the resulting floorplan has the minimum area becomes difficult.

Extensive researches have been conducted on floorplan area minimization. For slicing floorplan, Stockmeyer [22] used shape curve to represent all possible shapes of a module and showed that the minimal area can be found efficiently. There were also numerical optimization methods proposed by Wang *et al.* [23] and Moh *et al.* [14]. In [14], the problem was formulated as a geometric programming and solved using some standard convex optimization techniques. For non-slicing floorplan, the problem is more complicated. Pan *et al.* [20] and Wang *et al.* [24] attempted to

*This work was done at Northwestern University and supported by the NSF under CCR-0238484.

ICCAD 2006, November 5–9, 2006, San Jose, California, USA.

generalize Stockmeyer's algorithm [22]. Kang *et al.* [11] extended BSG [17] to handle soft modules using heuristics. But their methods were either sub-optimal or not applicable to all general non-slicing structures. On the other hand, Murata *et al.* [16] followed the framework of [14] and focused on reducing the number of variables and functions to improve the efficiency. But the running time of their method was still very long. Young *et al.* [27] applied the Lagrangian relaxation technique [13] and showed that the problem can be simplified based on the Kuhn-Tucker conditions [13].

Our contribution in this paper is twofold. Firstly, we show that the minimum area floorplanning problem is not convex and its optimal solution cannot be obtained by solving its Lagrangian dual problem. Therefore, the claim in [27] that an optimal solution to the Lagrangian dual problem will also minimize the area is not correct. Nor can [27] use the Kuhn-Tucker conditions of the minimum area floorplanning problem to simplify its dual. This clarifies a misunderstanding in using "Strong Duality Theorem" (Theorem 6.2.4 of [2]) by pointing out that the theorem is not applicable to a problem that is not convex but can be transformed to a convex formulation.

Secondly, we propose a modified convex formulation. Since it is convex, "Strong Duality Theorem" implies that we can solve it by solving its Lagrangian dual problem, which can be simplified by the Kuhn-Tucker conditions of the modified formulation. We then present an algorithm based on min-cost flow technique and trust region method. Experimental results under module aspect ratio bound [0.5,2.0] show that the running time of our floorplanner scales well with the problem size in MCNC benchmark. Compared with the floorplanner in [27], our floorplanner is 9.5X faster for the largest case "ami49". It also generates a floorplan with smaller deadspace for almost all test cases. In addition, since the generated floorplan has an aspect ratio closer to 1, it is more friendly to packaging. Our floorplanner is also amicable to including interconnect cost and other physical design metrics.

The rest of this paper is organized as follows. Section 2 presents the problem formulation. Section 3 describes the transformation to the Lagrangian dual problem and show that it does not give the minimum area solution. Section 4 discusses other approaches and proposes a modified formulation that uses perimeter as the objective function. Section 5 presents an algorithm to solve the modified formulation based on min-cost flow technique and trust region method. Experimental results are given in Section 6, followed by conclusions in Section 7.

2. PROBLEM FORMULATION

Suppose that we are given n modules of areas $A_1, A_2, ..., A_n$ and module aspect ratio (height/width) ranges $[r_1^{min}, r_1^{max}], [r_2^{min}, r_2^{max}]$, ..., $[r_n^{min}, r_n^{max}]$. Hard modules are those with $r^{min} = r^{max}$, i.e., their shapes are fixed. Let w_i be the width of module i, $\forall 1 \le i \le n$. According to the aspect ratio range, the minimum width L_i and the maximum width U_i of a module i can be computed as

$$L_i = \sqrt{A_i / r_i^{max}} \quad \text{and} \quad U_i = \sqrt{A_i / r_i^{min}}$$

Therefore, $L_i \le w_i \le U_i$, $\forall 1 \le i \le n$. We use $x_i \in \Re$ and $y_i \in \Re$ to denote the x and y coordinates of the left-bottom corner of module i respectively.

A horizontal (vertical) constraint graph G_h (G_v) is a graph of n vertices, where the vertices represent the modules and the edges represent the horizontal (vertical) non-overlapping constraints. For example, if module j is to the right of module i, we add an edge from i to j in the horizontal constraint graph. Similarly, if module j is above module i, we add an edge from i to j in the vertical constraint graph. In general, we have $O(n^2)$ number of edges, some of which can be removed since they are implied by transitivity of other edges. We denote the set of sources and sinks in G_h by s_h and t_h respectively, where a source has no incoming edges and a sink has no outgoing edges. Likewise, the set of sources and the set of sinks in G_v are denoted as s_v and t_v respectively. To ease the presentation, we introduce a left bottommost dummy module 0 and a right topmost dummy module $n+1$. They occupy no area. We then add to G_h edges from 0 to each vertex in s_h, and edges from each vertex in t_h to $n+1$. Similar edges are added to G_v. In particular, the horizontal and vertical constraint graphs of the non-slicing floorplan in Figure 1(b) are shown in Figure 2, where module 0 and 6 are dummy modules.

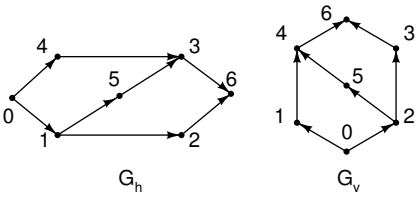

Figure 2: Constraint graphs of the floorplan in Figure 1(b).

A constraint graph can be generated from many representations such as sequence pair [15]. Given a constraint graph, the problem of minimum area floorplanning, denoted as P_{area}, can be formulated as follows.

PROBLEM 1 (MINIMUM AREA FLOORPLANNING).

$$\begin{array}{lll} P_{area}: \text{Minimize} & (x_{n+1}-x_0)(y_{n+1}-y_0) & \\ \text{subject to} & x_j \ge x_i + w_i & ,\forall (i,j) \in G_h \\ & y_j \ge y_i + A_i/w_i & ,\forall (i,j) \in G_v \\ & L_i \le w_i \le U_i & ,\forall 1 \le i \le n \end{array}$$

Note that although we focus on area minimization, it is possible to include interconnect cost and other physical design metrics when we measure the quality of a min-area floorplan. For example, a weighted sum of all metrics is commonly used.

Problem P_{area} is difficult to solve since it is not convex, as stated in the following lemma.

LEMMA 1. *Problem P_{area} is not convex.*

PROOF. A convex optimization problem is one of the form [3]

$$\begin{array}{lll} \text{Minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \le 0 & ,\forall i = 1,...,m \\ & a_i^T x = b_i & ,\forall i = 1,...,p \end{array}$$

where $f_0, ..., f_m$ are convex functions, $a_1, ..., a_p$ and $b_1, ..., b_p$ are constant vectors. We will show that the objective function of P_{area}, denoted as f, is not convex.

According to the definition of convexity, f is convex only if for any two points u and u' in its domain, we have

$$f(\gamma u + (1-\gamma)u') \quad \le \quad \gamma f(u) + (1-\gamma) f(u'), \quad \forall \gamma \in [0,1]$$

However, if we choose $x_{n+1} - x_0 = 0$, $y_{n+1} - y_0 = 1$, $x'_{n+1} - x'_0 = 1$, $y'_{n+1} - y'_0 = 0$, and $\gamma = 0.5$, then the left-hand-side of the above inequality is 0.25 and the right-hand-side is 0. Therefore, the inequality is not true, and thus P_{area} is not convex. □

In the next section we will apply Lagrangian relaxation to transform P_{area} to its Lagrangian dual.

3. APPLYING LAGRANGIAN RELAXATION TO P_{AREA}

3.1 Lagrangian relaxation

Lagrangian relaxation [13] is a general technique for solving constrained optimization problems. In Lagrangian relaxation, "troublesome" constraints are "relaxed" and incorporated into the objective after multiplying them by constants called Lagrangian multipliers, one multiplier for each constraint. For given multipliers, the relaxed problem is called *Lagrangian subproblem*. Finding the optimal multipliers under which the Lagrangian subproblem attains the best objective value is called *Lagrangian dual problem*.

When applying Lagrangian relaxation to P_{area}, we relax the non-overlapping constraints since they are difficult to handle. We introduce $\lambda_{i,j}$ for the constraint $x_j \geq x_i + w_i$ for all $(i,j) \in G_h$, and $\mu_{i,j}$ for the constraint $y_j \geq y_i + A_i/w_i$ for all $(i,j) \in G_v$. The Lagrangian subproblem of P_{area} associated with λ and μ is formulated as follows.

PROBLEM 2 (LAGRANGIAN SUBPROBLEM OF P_{area}).

$$\begin{aligned} LS_{area}(\lambda,\mu): \text{ Minimize } & (x_{n+1}-x_0)(y_{n+1}-y_0) \\ & - \sum_{(i,j)\in G_h} \lambda_{i,j}(x_j - x_i - w_i) \\ & - \sum_{(i,j)\in G_v} \mu_{i,j}(y_j - y_i - A_i/w_i) \\ \text{subject to } & L_i \leq w_i \leq U_i \quad ,\forall 1 \leq i \leq n \end{aligned}$$

Let F_{area} denote the objective function of $LS_{area}(\lambda,\mu)$. Note that

$$\sum_{(i,j)\in G_h} \lambda_{i,j}(x_j - x_i) = \sum_{0\leq i\leq n+1} (\sum_{(j,i)\in G_h} \lambda_{j,i} - \sum_{(i,j)\in G_h} \lambda_{i,j})x_i$$

$$\sum_{(i,j)\in G_v} \mu_{i,j}(y_j - y_i) = \sum_{0\leq i\leq n+1} (\sum_{(j,i)\in G_v} \mu_{j,i} - \sum_{(i,j)\in G_v} \mu_{i,j})y_i$$

Substituting them into F_{area} yields

$$\begin{aligned} F_{area} = & (x_{n+1}-x_0)(y_{n+1}-y_0) \\ & + \sum_{(0,i)\in G_h} \lambda_{0,i}x_0 - \sum_{(i,n+1)\in G_h} \lambda_{i,n+1}x_{n+1} \\ & + \sum_{(0,i)\in G_v} \mu_{0,i}y_0 - \sum_{(i,n+1)\in G_v} \mu_{i,n+1}y_{n+1} \\ & + \sum_{1\leq i\leq n} (\sum_{(i,j)\in G_h} \lambda_{i,j} - \sum_{(j,i)\in G_h} \lambda_{j,i})x_i \\ & + \sum_{1\leq i\leq n} (\sum_{(i,j)\in G_v} \mu_{i,j} - \sum_{(j,i)\in G_v} \mu_{j,i})y_i \\ & + \sum_{1\leq i\leq n} (w_i \sum_{(i,j)\in G_h} \lambda_{i,j} + \frac{A_i}{w_i} \sum_{(i,j)\in G_v} \mu_{i,j}) \end{aligned}$$

Let $Q_{area}(\lambda,\mu)$ denote the optimal objective value of $LS_{area}(\lambda,\mu)$ for given λ and μ. We say that $\lambda \geq 0$ when $\lambda_{i,j} \geq 0$ for all $(i,j) \in G_h$. Similar for $\mu \geq 0$. The Lagrangian dual problem LD_{area} of P_{area} is as follows.

PROBLEM 3 (LAGRANGIAN DUAL PROBLEM OF P_{area}).

$$\begin{aligned} LD_{area}: \text{ Maximize } & Q_{area}(\lambda,\mu) \\ \text{subject to } & \lambda \geq 0,\ \mu \geq 0 \end{aligned}$$

3.2 Kuhn-Tucker conditions

Kuhn-Tucker conditions [13] are necessary conditions that characterize the optimal solution of a general constrained optimization problem. To specify the Kuhn-Tucker conditions of P_{area}, we define the Lagrangian ζ of P_{area} as follows, which requires all the constraints to be relaxed into the objective.

$$\zeta = F_{area} - \sum_{1\leq i\leq n} \alpha_i(w_i - L_i) - \sum_{1\leq i\leq n} \beta_i(U_i - w_i)$$

where α_i and β_i are the Lagrangian multipliers for the constraint $L_i \leq w_i$ and $w_i \leq U_i$, $\forall 1 \leq i \leq n$, respectively.

According to the definition in [13], the Kuhn-Tucker conditions of P_{area} can be specified in the following theorem.

THEOREM 1. *Suppose that (x^*,y^*,w^*) is an optimal solution to P_{area}. Then there must exist Lagrangian multipliers $(\lambda^*,\mu^*,\alpha^*,\beta^*)$ such that*

$$\left.\frac{\partial\zeta}{\partial x_0}\right|_{\lambda^*,y^*} = -(y^*_{n+1} - y^*_0) + \sum_{(0,i)\in G_h} \lambda^*_{0,i} = 0 \quad (1)$$

$$\left.\frac{\partial\zeta}{\partial y_0}\right|_{\mu^*,x^*} = -(x^*_{n+1} - x^*_0) + \sum_{(0,i)\in G_v} \mu^*_{0,i} = 0 \quad (2)$$

$$\left.\frac{\partial\zeta}{\partial x_{n+1}}\right|_{\lambda^*,y^*} = (y^*_{n+1} - y^*_0) - \sum_{(i,n+1)\in G_h} \lambda^*_{i,n+1} = 0 \quad (3)$$

$$\left.\frac{\partial\zeta}{\partial y_{n+1}}\right|_{\mu^*,x^*} = (x^*_{n+1} - x^*_0) - \sum_{(i,n+1)\in G_v} \mu^*_{i,n+1} = 0 \quad (4)$$

and for all $1 \leq i \leq n$,

$$\left.\frac{\partial\zeta}{\partial x_i}\right|_{\lambda^*} = \sum_{(i,j)\in G_h} \lambda^*_{i,j} - \sum_{(j,i)\in G_h} \lambda^*_{j,i} = 0 \quad (5)$$

$$\left.\frac{\partial\zeta}{\partial y_i}\right|_{\mu^*} = \sum_{(i,j)\in G_h} \mu^*_{i,j} - \sum_{(j,i)\in G_h} \mu^*_{j,i} = 0 \quad (6)$$

$$\left.\frac{\partial\zeta}{\partial w_i}\right|_{\lambda^*,\mu^*,\alpha^*,\beta^*,w^*} = -\alpha^*_i + \beta^*_i + \sum_{(i,j)\in G_h} \lambda^*_{i,j} - \frac{A_i}{(w^*_i)^2} \sum_{(i,j)\in G_v} u^*_{i,j} = 0 \quad (7)$$

$$\alpha^*_i(w^*_i - L_i) = \beta^*(U_i - w^*_i) = 0 \quad (8)$$

However, we cannot use the above Kuhn-Tucker conditions of P_{area} to simplify LD_{area} since $(x^*,y^*,\lambda^*,\mu^*)$ is not an optimal solution to LD_{area}. In fact, we show in the next section that LD_{area} is unbounded below. In [3], a Lagrangian dual problem that is unbounded below is also called *infeasible*.

3.3 Infeasibility of LD_{area}

The infeasibility of LD_{area} is stated in the following lemma.

LEMMA 2. *LD_{area} is infeasible (unbounded below).*

PROOF. The infeasibility of LD_{area} can be established if we can show that for any given $\lambda \geq 0$ and $\mu \geq 0$, $LS_{area}(\lambda,\mu)$ is unbounded below. In order to make $F_{area} \to -\infty$, we can choose $x_0 = 0$, $x_{n+1} \to \infty$, $y_{n+1} - y_0 < \sum_{(i,n+1)\in G_h} \lambda_{i,n+1}$, and $x_i = y_i = 0$ for all $i \in [1,n]$. □

The importance of Lemma 2 is that it clarifies a misunderstanding in using "Strong Duality Theorem" (Theorem 6.2.4 of [2]) in Lagrangian relaxation, which we quote below for the ease of reference.

THEOREM 2 (STRONG DUALITY THEOREM).
Let S be a nonempty convex set in $\Re^n$, let $f: \Re^n \to \Re$ and $g: \Re^n \to$

$\Re^m$ *be convex. Suppose that the following constraint qualification holds true, that is, there exists an* $\hat{x} \in \mathcal{S}$ *such that* $g(\hat{x}) < 0$. *Then*

$$\inf\{f(x) : x \in \mathcal{S}, g(x) \leq 0\} \quad = \quad \sup\{\theta(\lambda) : \lambda \geq 0\}$$

where $\theta(\lambda) = \inf\{f(x) + \lambda^T g(x) : x \in \mathcal{S}\}$. *The left-hand-side refers to the objective value of minimizing* $f(x)$ *subject to* $x \in \mathcal{S}$ *and* $g(x) \geq 0$, *and the right-hand-side is the objective value of its Lagrangian dual problem with* λ *being the Lagrangian multiplier.*

Furthermore, if $\inf\{f(x) : x \in \mathcal{S}, g(x) \leq 0\}$ *is finite, then* $\sup\{\theta(\lambda) : \lambda \geq 0\}$ *is achieved at* $\tilde{\lambda}$ *with* $\tilde{\lambda} \geq 0$. *If* $\inf\{f(x) : x \in \mathcal{S}, g(x) \leq 0\}$ *is achieved at* $\tilde{x}$, *then* $\tilde{\lambda}^T g(\tilde{x}) = 0$.

COROLLARY 2.1. $\tilde{x}$ *is a solution to* $\inf\{f(x) + \tilde{\lambda}^T g(x) : x \in \mathcal{S}\}$. *On the other hand, if* x^* *is a solution to* $\inf\{f(x) + \tilde{\lambda}^T g(x) : x \in \mathcal{S}\}$ *satisfying* $g(x^*) \geq 0$, *then* x^* *is one such* $\tilde{x}$.

PROOF. Suppose otherwise that $\tilde{x}$ does not minimize $f(x) + \tilde{\lambda}^T g(x)$, then $\theta(\tilde{\lambda}) < f(\tilde{x}) + \tilde{\lambda}^T g(\tilde{x})$, i.e., $\theta(\tilde{\lambda}) < f(\tilde{x})$ since $\tilde{\lambda}^T g(\tilde{x}) = 0$. However, $\theta(\tilde{\lambda}) = f(\tilde{x})$ by Theorem 2. Therefore, $\tilde{x}$ minimizes $f(x) + \tilde{\lambda}^T g(x)$.

On the other hand, if there is a solution x^* to $\inf\{f(x) + \tilde{\lambda}^T g(x) : x \in \mathcal{S}\}$ satisfying $g(x^*) \geq 0$, then we have $f(\tilde{x}) = f(x^*) + \tilde{\lambda}^T g(x^*) \geq f(x^*)$. In addition, $f(\tilde{x}) \leq f(x^*)$ since $\tilde{x}$ is a solution to $\inf\{f(x) : x \in \mathcal{S}, g(x) \leq 0\}$. Therefore, $f(\tilde{x}) = f(x^*)$, i.e., x^* is also a solution to $\inf\{f(x) : x \in \mathcal{S}, g(x) \leq 0\}$. □

More specifically, if a constrained optimization problem is not convex, Theorem 2 does not guarantee that the optimal objective value of the problem is equal to that of its dual problem even though the problem can be transformed to a convex formulation. Problem P_{area} is such a witness by Lemma 1, 2, and the fact that P_{area} can be transformed to a convex formulation by geometric programming [14].

A few more explanations may be helpful in understanding the infeasibility of a Lagrangian dual problem. Let p^* and d^* denote the optimal objective values of a given problem and its Lagrangian dual problem respectively. We refer to the difference, $p^* - d^*$, as the *optimal duality gap*, which is always non-negative [3]. If the given problem is convex, Theorem 2 ensures that the gap is zero. For a non-convex problem, there is, in general, no guarantee whether the gap is zero or not. The infeasibility of the Lagrangian dual problem is an extreme case of non-zero gap. More interestingly, the size of the gap may vary with the constraints. For example, if we add into P_{area} additional constraints $x_i, y_i \geq 0$ for all $i \in [0, n+1]$, its Lagrangian dual problem is still infeasible, which can be proved by the same proof of Lemma 2. If we continue to add constraint $x_0 = y_0 = 0$, the Lagrangian dual problem becomes feasible since the Lagrangian subproblem has an objective value of 0 when all λ and μ are zero. In fact, we can show, by a proof similar to that of Lemma 2, that this is the only case when F_{area} is not unbounded below. However, such a case is less interesting since it leads to $x^*_{n+1} = y^*_{n+1} = 0$ by (1)-(4). Note that no matter which set of constraints is used, the optimal duality gap is non-zero for the minimum area floorplanning problem.

Therefore, the claim in [27] that an optimal solution to the Lagrangian dual problem will also minimize the area is not correct. Nor can [27] use the Kuhn-Tucker conditions of the minimum area floorplanning problem to simplify its dual.

4. OTHER APPROACHES

In this section, we discuss a few other approaches that deal with the infeasibility of LD_{area} from different aspects of view.

4.1 Geometric programming formulation

Since P_{area} can be transformed to a convex formulation, to which Theorem 2 is applicable, a natural idea is to see if the Lagrangian dual of the convex formulation can be solved efficiently.

First of all, by rearranging the variables and assigning $x_0 = y_0 = 0$, P_{area} can be equivalently stated as follows.

$$\begin{array}{lll} P_{area}: \text{ Minimize} & x_{n+1}y_{n+1} & \\ \text{subject to} & \frac{w_i}{x_j} \leq 1 & ,\forall i \in s_h, (i,j) \in G_h \\ & \frac{A_i}{w_i y_j} \leq 1 & ,\forall i \in s_v, (i,j) \in G_v \\ & \frac{x_i}{x_j} + \frac{w_i}{x_j} \leq 1 & ,\forall i \neq 0, (i,j) \in G_h \\ & \frac{y_i}{y_j} + \frac{A_i}{w_i y_j} \leq 1 & ,\forall i \neq 0, (i,j) \in G_v \\ & \frac{L_i}{w_i} \leq 1 & ,\forall 1 \leq i \leq n \\ & \frac{w_i}{U_i} \leq 1 & ,\forall 1 \leq i \leq n \end{array}$$

We define X_i, Y_i, and W_i of module i for all $1 \leq i \leq n$ as the logarithms of x_i, y_i, and w_i respectively, i.e.,

$$e^{X_i} = x_i, \quad e^{Y_i} = y_i, \quad e^{W_i} = w_i$$

The geometric programming formulation of P_{area} is given as follows.

PROBLEM 4 (GEOMETRIC PROGRAMMING).

$$\begin{array}{lll} GP: \text{ Minimize} & X_{n+1} + Y_{n+1} & \\ \text{subject to} & W_i - X_j \leq 0 & ,\forall i \in s_h, (i,j) \in G_h \\ & -W_i - Y_j + \log A_i \leq 0 & ,\forall i \in s_v, (i,j) \in G_v \\ & \log(e^{X_i - X_j} + e^{W_i - X_j}) \leq 0 & ,\forall i \neq 0, (i,j) \in G_h \\ & \log(e^{Y_i - Y_j} + A_i e^{-W_i - Y_j}) \leq 0 & ,\forall i \neq 0, (i,j) \in G_v \\ & -W_i + \log L_i \leq 0 & ,\forall 1 \leq i \leq n \\ & W_i - \log U_i \leq 0 & ,\forall 1 \leq i \leq n \end{array}$$

Since *GP* is convex [3] and the constraint qualification in Theorem 2 is true, Corollary 2.1 ensures that an optimal solution to *GP* is also an optimal solution to the Lagrangian dual of *GP*. Therefore, we can use the Kuhn-Tucker conditions of *GP* to simply its dual.

However, we find that the Kuhn-Tucker conditions of *GP* are much more complicated than those of P_{area}. As a result, its dual problem is as complicated as *GP* itself, which was shown in [16] to be expensive to solve.

4.2 Subgradient method for finding λ^* and μ^*

In order to take advantage of the Kuhn-Tucker conditions of P_{area}, Young *et al.* [27] focused on (5)-(8). A key observation they made is that if $L_i < w^*_i < U_i$, then (8) implies $\alpha^*_i = \beta^*_i = 0$, which, together with (7), leads to

$$w^*_i = \sqrt{\frac{A_i \sum_{(i,j) \in G_v} u^*_{i,j}}{\sum_{(i,j) \in G_h} \lambda^*_{i,j}}}$$

Therefore, w^* can be specified in terms of λ^* and μ^* as

$$w^*_i = \min\left(U_i, \max\left(L_i, \sqrt{\frac{A_i \sum_{(i,j) \in G_v} \mu^*_{i,j}}{\sum_{(i,j) \in G_h} \lambda^*_{i,j}}}\right)\right) \tag{9}$$

What remains is to find the optimal λ^* and μ^*, for which they used the subgradient method. It works as follows. Starting from an arbitrary λ and μ satisfying (5)-(6), the module width w_i for each

module i is computed by (9). Based on the obtained module widths and heights, the x and y positions of each module are computed by a longest path computation. After that, λ and μ are updated by following the subgradient direction:

$$\lambda'_{i,j} = \max\left(0,\ \lambda_{i,j}+\rho_k(x_i+w_i-x_j)\right)$$
$$\mu'_{i,j} = \max\left(0,\ \mu_{i,j}+\rho_k(y_i+\frac{A_i}{w_i}-y_j)\right)$$

where ρ_k is a step size parameter such that $\lim_{k\to\infty}\rho_k = 0$ and $\sum_{k=1}^{\infty}\rho_k = \infty$. The updated λ and μ are then projected to the nearest point satisfying (5)-(6). This is done through the orthonormal bases of the solution space of (5)-(6), which can be obtained by Gram-Schmidt process [8].

However, since they obtain x and y from the non-overlapping constraints instead of from solving LS_{area}, there is no guarantee that the subgradient method will converge. The convergence becomes even more questionable with the involvement of projection. Even if it converges, there is no guarantee that the converged solution is an optimal solution to P_{area}, or to LD_{area}. In other words, their approach is a heuristic.

4.3 Minimum perimeter formulation

Having seen the difficulty of solving P_{area}, we focus on LD_{area} again. By Lemma 2, LD_{area} is infeasible (unbounded below). The infeasibility comes from the non-convexity of the objective function of P_{area}.

Intuitively, if we can modify the objective function to make it convex, then the dual problem will become feasible. On the other hand, we want the modified objective function to be related to area. Therefore, we propose to use perimeter as follows.

PROBLEM 5 (MINIMUM PERIMETER FLOORPLANNING).

$$\begin{aligned} P_{peri}: \text{ Minimize } & (x_{n+1}-x_0)+(y_{n+1}-y_0) \\ \text{subject to } & x_j \ge x_i+w_i && ,\forall (i,j)\in G_h \\ & y_j \ge y_i+A_i/w_i && ,\forall (i,j)\in G_v \\ & L_i \le w_i \le U_i && ,\forall 1\le i\le n \end{aligned}$$

We show that P_{peri} is convex.

LEMMA 3. *Problem P_{peri} is convex.*

PROOF. By the definition of convexity, A_i/w_i is convex since $A_i/(\gamma w_i+(1-\gamma)w'_i) \le \gamma A_i/w_i+(1-\gamma)A_i/w'_i$ for all $w_i, w'_i > 0$ and $\gamma\in[0,1]$. In addition, all linear functions are convex and a non-negative weighted sum of convex functions is also convex. Therefore, both the objective and the constraints of P_{peri} are convex. So is P_{peri}. □

What remains is to justify P_{peri} by relating it to P_{area}. We make the following two observations.

OBSERVATION 1. *The experimental results in [27] show that the finally generated floorplan usually has very small deadspace, i.e., the area of the floorplan is close to the sum of the module areas, which is a constant.*

OBSERVATION 2. *Since minimizing the perimeter of a rectangle subject to a constant area will result in a perfect square, minimizing the perimeter of a floorplan subject to very small deadspace will lead to a floorplan close to a square. When a floorplan is close to a square, i.e., its width W is approximately equal to its height H, its area WH is close to W^2 or H^2, and its perimeter is close to $4W$ or $4H$.*

From the preceding two observations, we conclude that minimizing the perimeter of a floorplan will result in a small area. In addition, the aspect ratio of the floorplan will be close to 1. They are confirmed by the experimental results in Section 6.

Following the same procedure as in Section 3, we obtain the Lagrangian subproblem of P_{peri} as follows.

PROBLEM 6 (LAGRANGIAN SUBPROBLEM OF P_{peri}).

$$\begin{aligned} LS_{peri}(\lambda,\mu): \text{ Minimize } & (x_{n+1}-x_0)+(y_{n+1}-y_0) \\ & -\sum_{(i,j)\in G_h}\lambda_{i,j}(x_j-x_i-w_i) \\ & -\sum_{(i,j)\in G_v}\mu_{i,j}(y_j-y_i-A_i/w_i) \\ \text{subject to } & L_i \le w_i \le U_i && ,\forall 1\le i\le n \end{aligned}$$

Let F_{peri} denote the objective function of $LS_{peri}(\lambda,\mu)$. Similar to F_{area}, we can simplify F_{peri} to

$$\begin{aligned} F_{peri} = & (1-\sum_{(i,n+1)\in G_h}\lambda_{i,n+1})x_{n+1}-(1-\sum_{(0,i)\in G_h}\lambda_{0,i})x_0 \\ & +(1-\sum_{(i,n+1)\in G_v}\mu_{i,n+1})y_{n+1}-(1-\sum_{(0,i)\in G_v}\mu_{0,i})y_0 \\ & +\sum_{1\le i\le n}(\sum_{(i,j)\in G_h}\lambda_{i,j}-\sum_{(j,i)\in G_h}\lambda_{j,i})x_i \\ & +\sum_{1\le i\le n}(\sum_{(i,j)\in G_v}\mu_{i,j}-\sum_{(j,i)\in G_v}\mu_{j,i})y_i \\ & +\sum_{1\le i\le n}(w_i\sum_{(i,j)\in G_h}\lambda_{i,j}+\frac{A_i}{w_i}\sum_{(i,j)\in G_v}\mu_{i,j}) \end{aligned}$$

Let $Q_{peri}(\lambda,\mu)$ denote the optimal objective value of $LS_{peri}(\lambda,\mu)$ for given λ and μ. The Lagrangian dual problem of P_{peri} is as follows.

PROBLEM 7 (LAGRANGIAN DUAL PROBLEM OF P_{peri}).

$$\begin{aligned} LD_{peri}: \text{ Maximize } & Q_{peri}(\lambda,\mu) \\ \text{subject to } & \lambda\ge 0,\ \mu\ge 0 \end{aligned}$$

Since P_{peri} is convex by Lemma 3 and the constraint qualification in Theorem 2 is true, Corollary 2.1 ensures that an optimal solution to P_{peri} is also an optimal solution to LD_{peri}. Therefore, the Kuhn-Tucker conditions of P_{peri} can be used to simplify F_{peri}, and thus LD_{peri}. We obtained the simplified LD_{peri} as follows, where (9)-(13) are from the Kuhn-Tucker conditions of P_{peri}.

$$\text{Maximize} \quad F_{peri} = \sum_{1\le i\le n}(w_i\sum_{(i,j)\in G_h}\lambda_{i,j}+\frac{A_i}{w_i}\sum_{(i,j)\in G_v}\mu_{i,j})$$

$$\text{where} \quad w_i = \min\left(U_i,\ \max\left(L_i,\ \sqrt{\frac{A_i\sum_{(i,j)\in G_v}\mu_{i,j}}{\sum_{(i,j)\in G_h}\lambda_{i,j}}}\right)\right) \tag{9}$$

$$\text{subject to} \quad \sum_{(i,j)\in G_h}\lambda_{i,j} = \sum_{(j,i)\in G_h}\lambda_{j,i} \quad ,\forall 1\le i\le n \tag{10}$$

$$\sum_{(i,j)\in G_v}\mu_{i,j} = \sum_{(j,i)\in G_v}\mu_{j,i} \quad ,\forall 1\le i\le n \tag{11}$$

$$\sum_{(0,i)\in G_h}\lambda_{0,i} = \sum_{(i,n+1)\in G_h}\lambda_{i,n+1} = 1 \tag{12}$$

$$\sum_{(0,i)\in G_v}\mu_{0,i} = \sum_{(i,n+1)\in G_v}\mu_{i,n+1} = 1 \tag{13}$$

$$\lambda\ge 0, \quad \mu\ge 0 \tag{14}$$

In fact, (10)-(13) have an intuitive explanation. Suppose that some of (10)-(13) is not satisfied under given λ and μ, then the

coefficient of some x_i or y_i, $0 \le i \le n+1$, in F_{peri} is not zero. As a result, $Q_{peri}(\lambda,\mu) = -\infty$. Since LD_{peri} asks for optimal λ and μ that maximize $Q_{peri}(\lambda,\mu)$, the cases resulting in $Q_{peri}(\lambda,\mu) = -\infty$ can be ignored. Therefore, we can include (10)-(13) as constraints in LD_{peri}.

We say that λ and μ are *feasible* if they satisfy (10)-(14). We present an algorithm in the next section that solves LD_{peri} by min-cost flow technique and trust region method.

5. SOLVING LD_{PERI} BY MIN-COST FLOW AND TRUST REGION METHOD

If we treat λ and μ as flow on G_h and G_v respectively, then the constraints (10)-(11) are also known as *flow conservation*, i.e., incoming flow equals outgoing flow. The constraint (12) states that the total amount of flow going out of module 0 and going into module $n+1$ is 1 in G_h. This is also true in G_v by (13). The constraint (14) ensures that the flow on each edge is non-negative.

Although F_{peri} is greatly simplified by (10)-(14), it is still a complicated function of λ and μ. However, given feasible λ and μ, we can apply Taylor-series expansion to approximate F_{peri} in a small region around λ and μ. Specifically, we have

$$F_{peri}(\lambda+\delta,\mu+\sigma) \approx F_{peri}(\lambda,\mu) + \left.\frac{\partial F}{\partial \lambda}\right|_{\lambda,\mu} \cdot \delta + \left.\frac{\partial F}{\partial \mu}\right|_{\lambda,\mu} \cdot \sigma$$

where $|\delta_{i,j}| \le \Delta$ for all $(i,j) \in G_h$, $|\sigma_{i,j}| \le \Delta$ for all $(i,j) \in G_v$, and Δ is a non-negative number called the *radius* of the region.

Let $T(\lambda+\delta,\mu+\sigma)$ denote the right-hand-side of the above approximation. The approximation is especially accurate when Δ is chosen small enough.

To solve LD_{peri}, we start with a feasible flow and solve a sequence of *local problems*. Each local problem, denoted as TP, finds the optimal flow within a small region around the current feasible λ and μ, formulated as follows.

PROBLEM 8 (LOCAL PROBLEM BY TAYLOR EXPANSION).

$$\begin{aligned}
TP: \text{Maximize} \quad & T(\lambda+\delta,\mu+\sigma) \\
\text{subject to} \quad & \sum_{(i,j)\in G_h} \delta_{i,j} = \sum_{(j,i)\in G_h} \delta_{j,i} \quad ,\forall 1 \le i \le n \\
& \sum_{(i,j)\in G_v} \sigma_{i,j} = \sum_{(j,i)\in G_v} \sigma_{j,i} \quad ,\forall 1 \le i \le n \\
& \sum_{(0,i)\in G_h} \delta_{0,i} = \sum_{(i,n+1)\in G_h} \delta_{i,n+1} = 0 \\
& \sum_{(0,i)\in G_v} \sigma_{0,i} = \sum_{(i,n+1)\in G_v} \sigma_{i,n+1} = 0 \\
& \max(-\lambda_{i,j}, -\Delta) \le \delta_{i,j} \le \Delta \ ,\forall (i,j) \in G_h \\
& \max(-\mu_{i,j}, -\Delta) \le \sigma_{i,j} \le \Delta \ ,\forall (i,j) \in G_v
\end{aligned}$$

The first four constraints in TP come from the requirement that $\lambda+\delta$ and $\mu+\sigma$ need to satisfy (10)-(14) to be feasible. The remaining two constraints ensure nonnegative flow under $\lambda+\delta$ and $\mu+\sigma$.

If we treat $\frac{\partial F}{\partial \lambda}\big|_{\lambda,\mu}$ as the costs of δ and $\frac{\partial F}{\partial \mu}\big|_{\lambda,\mu}$ as the costs of σ, and note that maximizing $T(\lambda+\delta,\mu+\sigma)$ is equivalent to minimizing $-T(\lambda+\delta,\mu+\sigma)$, then TP becomes a min-cost flow problem [1]. In addition, since w_i is determined by λ and μ by (9), $\forall 1 \le i \le n$, we treat them as constants in TP for simplicity[1]. Let $c_{i,j}$ denote the cost on edge $(i,j) \in G_h \cup G_v$. Then $T(\lambda+\delta,\mu+\sigma)$ can be written as

$$F_{peri}(\lambda,\mu) + \sum_{1\le i\le n} \Big(\sum_{(i,j)\in G_h} c_{i,j}\delta_{i,j} + \sum_{(i,j)\in G_v} c_{i,j}\sigma_{i,j} \Big)$$

where

$$c_{i,j} = w_i \quad ,\forall (i,j) \in G_h \tag{15}$$

$$c_{i,j} = \frac{A_i}{w_i} \quad ,\forall (i,j) \in G_v \tag{16}$$

Let δ^* and σ^* be the optimal solution to TP found by any min-cost flow algorithm [1]. We have $F_{peri}(\lambda+\delta^*,\mu+\sigma^*) > F_{peri}(\lambda,\mu)$ for a small enough Δ if λ and μ are not the optimal flow. We then update λ and μ by $\lambda+\delta^*$ and $\mu+\sigma^*$ respectively, and iterate the process until convergence.

What remains is to determine the value of Δ for each iteration. We adopt the concept of *trust region* [18] and define the ratio

$$\rho = \frac{F_{peri}(\lambda+\delta^*,\mu+\sigma^*) - F_{peri}(\lambda,\mu)}{T(\lambda+\delta^*,\mu+\sigma^*) - F_{peri}(\lambda,\mu)} \tag{17}$$

The numerator is called the *actual improvement*, and the denominator is the *predicted improvement*. Since (δ^*,σ^*) is the optimal solution to TP, the predicted improvement will always be nonnegative. In fact, ρ has an intuitive interpretation. If ρ is close to 1, it means that Taylor expansion can be trusted as an adequate representation of F within radius Δ around λ and μ, and thus it is reasonably safe to increase Δ for the next iteration. If ρ is positive but not close to 1, we do not alter Δ, but if it is close to zero or negative, we reduce Δ.

In Figure 3, we give the algorithm for solving LD_{peri}. We start with a feasible flow satisfying (10)-(14). This can be done by finding a directed path from module 0 to module $n+1$ in each G_h and G_v and send a flow of 1 on that path. At each iteration, we solve an instance of TP under the current feasible λ and μ. The edges costs are computed based on (15)-(16). TP can be solved by any min-cost flow algorithm [1]. Once the optimal solution (δ^*,σ^*) is obtained, we evaluate ρ by (17). If $\rho < 0.25$, Δ is reduced to fourth. On the other hand, if $\rho > 0.75$ and the optimal solution to TP reaches the boundary of the region characterized by radius Δ around λ and μ, it implies that the current value of Δ interferes with the progress of the algorithm, thus its value is doubled for the next iteration. The procedure of dynamically adjusting Δ based on ρ, as well as the usage of 0.25 and 0.75, is the same as the trust region algorithm in [18]. If $\rho > \eta$, where $\eta \in [0, 0.25)$ is a given constant, we update λ and μ with $\lambda+\delta^*$ and $\mu+\sigma^*$ respectively. The optimal flow is reached when both δ^* and σ^* are 0. Finally, we use the values of the optimal flow to compute the optimal width of each module by (9). In our implementation, we choose the initial value of Δ to be 0.5 and η is slightly smaller than 0.25.

The correctness of the algorithm is given in the next theorem.

THEOREM 3. *The algorithm in Figure 3 returns an optimal solution to LD_{peri} and P_{peri}.*

PROOF. First of all, for any problem (not necessarily convex), its Lagrangian dual problem is a convex optimization problem, as stated in [3]. Therefore, any local optimum of LD_{peri} is a global optimum. Since the algorithm in Figure 3 is guaranteed to converge [18], the converged solution is a global optimum of LD_{peri}.

In addition, it is also an optimal solution to P_{peri} by Corollary 2.1 and the fact that the converged solution satisfies all non-overlapping constraints. □

[1]However, it can be verified that if $w_i = \sqrt{\frac{A_i \sum_{(i,j)\in G_v} \mu_{i,j}}{\sum_{(i,j)\in G_h} \lambda_{i,j}}}$, then $\frac{\partial F}{\partial \lambda_{i,j}}\big|_{\lambda,\mu}$ is exactly equal to w_i. So is $\frac{\partial F}{\partial \mu_{i,j}}\big|_{\lambda,\mu} = \frac{A_i}{w_i}$.

Algorithm
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_h and G_v,
$\Delta > 0$, and $\eta \in [0, 0.25)$.
Output: Optimal widths $w_1, w_2, ..., w_n$.

$\lambda, \mu \leftarrow$ a feasible flow satisfying (10)-(14);
Do
  Compute costs by (15)-(16);
  Compute (δ^*, σ^*) by solving TP;
  Evaluate ρ by (17);
  ▷*Adjust* Δ *according to* ρ
  If $(\rho < 0.25)$
    $\Delta \leftarrow \Delta/4$;
  Elsif $\Big((\rho > 0.75) \wedge$
    $\big(\exists (i,j) \in G_h \cup G_v : (|\delta^*_{i,j}| = \Delta) \vee (|\sigma^*_{i,j}| = \Delta)\big)\Big)$
    $\Delta \leftarrow 2\Delta$;
  ▷*Update* λ *and* μ
  If $(\rho > \eta)$
    $(\lambda, \mu) \leftarrow (\lambda, \mu) + (\delta^*, \sigma^*)$;
While $\big((\delta^* \neq 0) \vee (\sigma^* \neq 0)\big)$;
Compute w_i based on λ and μ by (9);
Return w;

Figure 3: Pseudocode of floorplanning algorithm.

6. EXPERIMENTAL RESULTS

We obtained the source code of the simulated annealing based floorplanner in [27], which used sequence pair [15] to represent a general floorplan and to generate the constraint graph in the annealing process. We then replaced their subroutine of the subgradient method with the algorithm in Figure 3 to generate our floorplanner. In our implementation, we used the min-cost flow algorithm by Goldberg [7] to solve TP. Both floorplanners were written in C and compiled using GCC 3.2 in a PC with a 2.4 GHz Xeon CPU, 512 KB 2nd level cache memory and 1GB RAM.

We kept the same setting of the simulated annealing used in [27]. More specifically, the initial temperature was chosen such that an acceptance ratio was 95% at the beginning. The temperature was lowered at a constant rate of 0.9 and the number of iterations at one temperature step was a constant. We also obtained the test files in [27], which were generated from MCNC benchmark.

We did two sets of experiments. In the first set, we ignored interconnect cost, i.e., we focused on perimeter minimization. The aspect ratio of each module was allowed to range from 0.5 to 2.0. This is a practically reasonable range. The results from both floorplanners are shown in Table 1 under column " [27]" and "ours" respectively. In particular, column "n" lists the number of modules in each test file, column "t(sec)" lists the running time in seconds, column "DS%" lists the percentage of deadspace, and column "w/h" lists the aspect ratio of the floorplan solution.

Table 1 enables a comparison between the two floorplanners from three different points of view. Firstly, although the running time of the floorplanner in [27] is less for small cases, it does not scale well with the problem size. For the largest case "ami49", it takes 2575.94 seconds to finish, which is much longer than the 270.48 seconds by our floorplanner. In other words, our floorplanner is more suitable for large designs. In addition, we observed that for

Table 1: Results under module aspect ratio bound [0.5,2.0]

Data	n	[27]			ours		
		t(sec)	DS%	w/h	t(sec)	DS%	w/h
xerox	10	2.33	0.35	0.57	19.28	0.19	1.03
apte	9	1.63	0.04	1.99	14.41	0.05	1.00
hp	11	4.35	3.05	1.41	20.65	0.16	1.02
ami33	33	860.34	1.65	1.18	381.17	0.27	1.01
ami49	49	2575.94	5.52	1.30	270.48	1.04	1.18

each test file, the number of different constraint graphs processed during annealing is same for both floorplanners. It means that the less running time by our floorplanner is due to the efficiency of solving P_{peri}. Secondly, we notice that almost every floorplan generated by our floorplanner has smaller deadspace compared with that generated by the floorplanner in [27]. For "ami49", the deadspace is 5.52% by the floorplanner in [27] and 1.04% by our floorplanner. This justifies our usage of perimeter as the objective function in P_{peri}. Further study of the source code of the floorplanner in [27] reveals that the subgradient method stops after a given number, 20, of iterations to trade-off the quality of the solution and the running time. In other words, the running time of their floorplanner will be further degraded if smaller deadspace is desired. For our algorithm in Figure 3, the average number of iterations before convergence was 22. Thirdly, the floorplan generated by our floorplanner has an aspect ratio closer to 1, which confirms Observation 2, and thus is more friendly to packaging. Figure 4 illustrates the floorplan of "ami33" generated by our floorplanner, where the dark places represent deadspace.

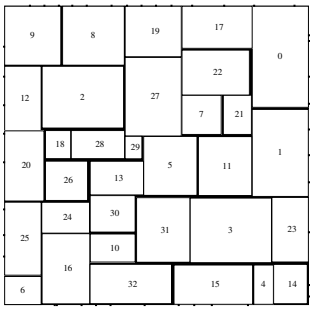

Figure 4: A floorplan of "ami33" with 0.27% deadspace and 1.01 aspect ratio under module aspect ratio bound [0.5,2.0].

To test our floorplanner under a more flexible aspect ratio bound, we chose [0.1, 10.0]. The results are reported in Table 2.

Table 2: Results under module aspect ratio bound [0.1,10.0]

Data	n	[27]			ours		
		t(sec)	DS%	w/h	t(sec)	DS%	w/h
xerox	10	1.82	0.00	0.69	20.81	0.11	1.00
apte	9	1.56	0.00	1.84	15.41	0.08	1.00
hp	11	4.01	0.07	0.47	26.97	0.10	1.00
ami33	33	1062.31	2.21	0.85	223.65	0.29	0.99
ami49	49	2480.01	4.21	1.35	168.55	1.38	1.01

It can be seen that similar trends as those in Table 1 are also present in Table 2. In addition, because of the more flexible bound, almost every test case has a floorplan with aspect ratio 1.00, i.e., a perfect square. Note that since P_{peri} focuses on perimeter, a more flexible bound will lead to a smaller perimeter, but not necessarily

Table 3: Results with wire length incorporated

Data	net#	aspect ratio bound [0.5,2.0]					aspect ratio bound [0.1,10.0]				
		t(sec)	area(mm^2)	DS%	wire(mm)	w/h	t(sec)	area(mm^2)	DS%	wire(mm)	w/h
xerox	203	18.96	1.94	0.27	148.01	1.00	23.14	1.95	0.70	126.14	1.04
apte	97	17.06	4.82	0.18	133.96	0.93	21.79	4.82	0.24	125.55	1.00
hp	83	18.09	0.90	0.22	43.37	1.01	25.89	0.91	0.50	42.37	1.00
ami33	123	430.48	1.16	0.51	61.75	1.02	268.04	1.16	0.44	49.73	0.99
ami49	408	257.39	3.74	5.29	336.74	1.01	155.15	3.61	1.83	367.19	1.03

lead to smaller deadspace. We illustrate in Figure 5 the floorplan of "ami49" by our floorplanner.

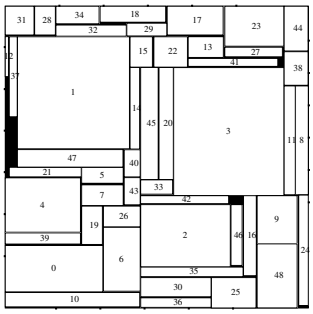

Figure 5: A floorplan of "ami49" with 1.38% deadspace and 1.01 aspect ratio under module aspect ratio bound [0.1,10.0].

In our second set of experiments, we used the summation of perimeter and the average HPWL (half-perimeter wire length) of a net as our objective. The results are shown in Table 3, where "net#" lists the number of nets in each test file, "area(mm^2)" and "wire(mm)" list the area and the total wire length of the generated floorplan respectively. We can see that when a more flexible bound is available, either the deadspace or the wire length is reduced, if not both.

It is worthy to point out a few caveats when comparing our floorplanner with other ones that do not involve solving P_{area} as a subroutine, e.g., a recent floorplanner proposed in [6]. Given horizontal and vertical constraints, the floorplanner in [6] applies simulated annealing to adjust the aspect ratio of a soft module. Although this could yield faster algorithms for both P_{area} and P_{peri}, it cannot guarantee optimal solutions. On the other hand, there is no fixed aspect ratio bound in [6]. The width of a soft module can be adjusted to match the width of any other module. As a result, the aspect ratio of some module may be impractical even though smaller deadspace could be obtained.

7. CONCLUSIONS

We clarify a misunderstanding in using Lagrangian relaxation for the floorplan area minimization problem by showing that the optimal solution cannot be obtained by solving the Lagrangian dual problem because of the non-convexity of the objective function. We then consider how to modify the objective to make it convex and propose to use perimeter. The usage of perimeter is justified by our experimental results. An algorithm is presented for the modified formulation based on min-cost flow technique and trust region method. Compared with the floorplanner in [27], our floorplanner scales better in problem size, takes much less time for large circuits, generates smaller deadspace for almost all test cases under aspect ratio bound [0.5,2.0], and produces floorplans that are more friendly to packaging.

8. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Application*. Prentice Hall, 1993.

[2] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, Inc., second edition, 1997.

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[4] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu. B*-trees: A new representation for non-slicing floorplans. In *DAC*, pages 458–463, 2000.

[5] C.-P. Chen, C. C. N. Chu, and D. F. Wong. Fast and exact simultaneous gate and wire sizing by lagrangian relaxation. *IEEE TCAD*, 18(7):1014–1025, July 1999.

[6] T.-C. Chen and Y.-W. Chang. Modern floorplanning based on fast simulated annealing. In *ISPD*, pages 104–112, 2005.

[7] A. V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22:1–29, 1997.

[8] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins, 3rd edition, 1996.

[9] P.-N. Guo, C.-K. Cheng, and T. Yoshimura. An O-tree representation of non-slicing floorplan and its applications. In *DAC*, pages 268–273, 1999.

[10] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu. Corner block list: An effective and efficient topological representation of non-slicing floorplan. In *ICCAD*, pages 8–12, 2000.

[11] M. Kang and W. W. M. Dai. General floorplanning with l-shaped, t-shaped and soft blocks based on bounded slicing grid structure. In *ASP-DAC*, pages 265–270, 1997.

[12] J.-M. Lin and Y.-W. Chang. TCG: A transitive closure graph-based representation for non-slicing floorplans. In *DAC*, pages 764–769, 2001.

[13] D. G. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, Reading, Massachusetts, 1984.

[14] 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.

[15] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. Rectangle-packing based module placement. In *ICCAD*, pages 472–479, 1995.

[16] H. Murata and E. S. Kuh. Sequence-pair based placement method for hard/soft/pre-placed modules. In *ISPD*, pages 167–172, 1998.

[17] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani. Module placement on BSG-structure and IC layout applications. In *ICCAD*, pages 484–493, 1996.

[18] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research, Springer Verlag, 1999.

[19] R. H. J. M. Otten. Automatic floorplan design. In *DAC*, pages 261–267, 1982.

[20] P. Pan and C. L. Liu. Area minimization for floorplans. *IEEE TCAD*, 14(1):123–132, January 1995.

[21] K. Sakanushi and Y. Kajitani. The quarter-state sequence (qsequence) to represent the floorplan and applications to layout optimization. In *IEEE APCCAS*, pages 829–832, 2000.

[22] L. Stockmeyer. Optimal orientations of cells in slicing floorplan designs. *Information and Control*, 59:91–101, 1983.

[23] T.-C. Wang and D. F Wong. An optimal algorithm for floorplan area optimization. In *DAC*, pages 180–186, 1990.

[24] T.-C. Wang and D. F. Wong. Optimal floorplan area optimization. *IEEE TCAD*, 2(8):992–1001, 1992.

[25] D. F. Wong and C. L. Liu. A new algorithm for floorplan design. In *DAC*, pages 101–107, 1986.

[26] B. Yao, H. Chen, C. K. Cheng, and R. Graham. Revisiting floorplan representation. In *ISPD*, pages 138–143, 2001.

[27] F. Y. Young, C. C. N. Chu, W. S. Luk, and Y. C. Wong. Handling soft modules in general non-slicing floorplan using lagrangian relaxation. *IEEE TCAD*, 20(5):687–692, May 2001.

[28] F. Y. Young, C. C. N. Chu, and Z. C. Shen. Twin binary sequences: A non-redundant representation for general non-slicing floorplan. *IEEE TCAD*, 22(4):457–469, April 2003.

[29] H. Zhou and J. Wang. Acg–adjacent constraint graph for general floorplans. In *ICCD*, pages 572–575, 2004.