

Highly Efficient Gradient Computation for Density-Constrained Analytical Placement Methods

Jason Cong and Guojie Luo
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
{ cong, gluo } @ cs.ucla.edu

ABSTRACT

Recent analytical global placers use density constraints to approximate non-overlap constraints and show very successful results. In this paper we unify a wide range of density smoothing techniques that we call global smoothing, and present a highly efficient method to compute the gradient of such smoothed densities used in several well-known analytical placers [3, 5, 7]. Our method reduces the complexity of the gradient computation by a factor of n compared to a naïve method, where n is the number of modules. Furthermore, with this efficient gradient computation we can come up with an efficient nonlinear programming-based placement framework, which supercedes the existing force-directed placement methods [4, 7]. An application of our technique, as the engine of a multilevel placer, achieved 13% and 15% wirelength improvement compared with SCAMPI [13] and mPL6 [3] on IBM-HB+ benchmark [13].

Categories and Subject Descriptors

B.7.2 [Integrated Circuits]: Design Aids – *placement and routing*; G.4 [Mathematical Software]: Algorithm Design and Analysis; J.6 [Computer-Aided Engineering]: Computer-Aided Design.

General Terms

Algorithms, Design

Keywords

Mixed-size Placement, Force-directed Method

1. INTRODUCTION

Recent analytical global placers use density constraints to approximate non-overlap constraints, and show very successful results in both quality and scalability [4, 5, 7]. Differentiability of either the objective functions or constraint functions is usually required by analytical solvers. But the density function is normally not smooth, thus several smoothing techniques have been proposed and implemented to overcome this problem, including: (a) the bell-shaped function [8, 12] to replace the

rectangle-shaped modules with differential bell-shaped modules; (b) the smoothing operator defined by Helmholtz equation [4]; (c) the Gaussian smoothing [5] for the density of fixed modules; (d) the Poisson equation [7] to transform area distribution to some smoothed potential.

In this paper we consider the smoothing techniques (b), (c) and (d) listed above, which we call global smoothing techniques because the smoothed density of a single bin is correlated globally with the original density of every bin. Global smoothing techniques were used by the top placers in the ISPD06 placement contest [10], and the contest results indicate that these techniques are effective in achieving high quality solutions. However, until recently, these techniques did not completely conform to the standard nonlinear programming framework. The method in NTUplace [5] did not use Gaussian smoothing for movable modules, but only for fixed modules; The method in Kraftwerk [7] used the smoothed potential as the basis for a force-directed method, but does not follow a standard nonlinear programming framework; The method in mPL [4] generalized the force-directed method and used nonlinear programming formulation and solution technique based on the Uzawa's algorithm [2], but could only use simple approximation of the gradient computation for the smoothed density function.

To adopt these global smoothing techniques into a nonlinear programming framework, a fundamental difficulty arises because of the high complexity of gradient computation of the density penalty function. Unlike the bell-shaped function smoothing technique where the gradient of the density penalty function can be written down explicitly, the global smoothing techniques do not seem to have any simple analytical form and may require large amount of numerical computation. This difficulty motivated our work, which has resulted in the following contributions:

- We discovered the common property of the global smoothing techniques (b), (c) and (d), which makes our work extensible to handling a large class of smoothing techniques.
- We derived an equivalent expression for the gradient of the density penalty function, which leads to highly efficient numerical computation and reduces the time complexity by a factor of n , compared to a straightforward computation.
- We used our efficient density gradient method in a nonlinear programming framework. We consider this a breakthrough as this is the first time that density-constrained placement problem with global smoothing technique can be solved exactly in the general nonlinear programming framework. Moreover, we found that the resulting placement method supercedes the force-directed placement methods [4, 7].

- In particular, we applied our gradient computation to the quadratic penalty method in a multilevel placement framework, and tested this on IBM-HB+ benchmark [13]. The application leads to a 13% and 15% shorter wirelength than SCAMPI [13] and mPL6 [3], respectively. It also leads to consistent wirelength improvements in ICCAD'04 [1], ISPD'05 [11] and ISPD'06 [10] benchmarks.

The remainder of this paper is organized as follows. Section 2 describes the class of smoothing techniques we are concerned with; Section 3 defines the density penalty function under two kinds of nonlinear programming methods; Section 4 derives an equivalent expression for the gradient of density penalty function that leads to highly efficient computation; Section 5 provides further analysis on our computation and a better understanding of force-directed methods; Section 6 presents our experimental results. Finally, conclusions and future work are presented in Section 7.

2. DENSITY and SMOOTHED DENSITY

We begin with a simplified placement problem for wirelength minimization under given target density distribution constraints for all bins. These bin density constraints are usually used to replace the non-overlap constraints. If the bin density constraints are satisfied or almost satisfied, we consider the global placement stage to be completed and leave the remaining work to the detailed placement phase. Thus the global placement problem is formulated as,

$$\begin{aligned} & \text{minimize} && WL(\bar{x}, \bar{y}) \\ & \text{subject to} && D_{ij}(\bar{x}, \bar{y}) = I_{ij} \\ & && \text{for every } 1 \leq i \leq M, 1 \leq j \leq N \end{aligned}$$

In this problem there are n modules. The symbols \bar{x} and \bar{y} are short-hand notations for the n -dimension vectors (x_k) and (y_k) , respectively, where (x_k, y_k) is the placement of the k -th module with width w_k and height h_k .

We assume that the placement region is $[0, a] \times [0, b]$, which is a rectangular area with origin at $(0, 0)$, width a and height b . To measure the placement density, it is divided into $M \times N$ uniform bins B_{ij} , $1 \leq i \leq M, 1 \leq j \leq N$, with bin width $w_B = a/M$ and bin height $h_B = b/N$. $D_{ij}(\bar{x}, \bar{y})$ is the average module density in bin B_{ij} determined by the placement (\bar{x}, \bar{y}) .

The target density constraints $D_{ij}(\bar{x}, \bar{y}) = I_{ij}$ require the density in each bin B_{ij} be equal to the target density I_{ij} . In general, we can support density inequality constraints as well by transforming them to equality constraints using filler cells as done in [3].

2.1 Density

To simplify the following analysis, we assume we have infinite resolution of the bin structure; thus, the set of bin density $\{D_{ij}(\bar{x}, \bar{y})\}$ becomes the function of point density $(D(\bar{x}, \bar{y}))(u, v)$ defined in $[0, a] \times [0, b]$, which can be computed by the summation of the density contribution of each module as $\sum_{k=1}^n D_k(x_k, y_k)$, where

$$(D_k(x_k, y_k))(u, v) = \begin{cases} 1 & \begin{aligned} & u \in [x_k - w_k/2, x_k + w_k/2] \\ & \text{and} \\ & v \in [y_k - h_k/2, y_k + h_k/2] \end{aligned} \\ 0 & \text{otherwise} \end{cases}$$

A small example with two overlapping modules (with aspect ratio 2:1) for the discrete density and continuous density are shown in **Figure 1**. The discrete values represent the average bin densities.

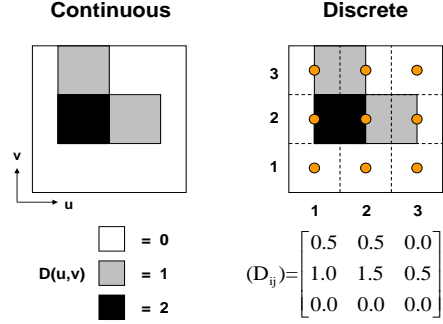


Figure 1 - Continuous Density and Discrete Density

The target density $I(u, v)$ can also be considered as $\{I_{ij}\}$ with infinite resolution. Thus we transform the constraints from the discrete bin structures to a continuous density map,

$$\begin{aligned} & \text{minimize} && WL(\bar{x}, \bar{y}) \\ & \text{subject to} && (D(\bar{x}, \bar{y}))(u, v) = I(u, v) \\ & && \text{for every } (u, v) \in [0, a] \times [0, b] \end{aligned}$$

2.2 Smoothed Density

Note that $(D(\bar{x}, \bar{y}))(u, v)$ is not differentiable in general. There have been several smoothing techniques (as introduced in Section 1), including the Helmholtz equation, the Poisson equation and Gaussian smoothing. Our study shows that all of these smoothing techniques can be generalized into one class that we shall define in Section 2.2.4, which will be used in the remainder of this paper. Here we shall first review each existing density smoothing technique.

2.2.1 Smoothing by Helmholtz Equation [4]

The smoothed density $(\hat{D}_H(\bar{x}, \bar{y}))(u, v)$ is defined as the solution of Helmholtz equation with zero-derivative boundary conditions:

$$\left(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2} - \varepsilon\right)(\hat{D}_H(\bar{x}, \bar{y}))(u, v) = -(D(\bar{x}, \bar{y}))(u, v)$$

The solution can be written down explicitly through Green's function [15].

$$(\hat{D}_H(\bar{x}, \bar{y}))(u, v) = -\int_0^a \int_0^b (D(\bar{x}, \bar{y}))(u', v') G_H(u, v, u', v') du' dv'$$

where $G_H(u, v, u', v')$

$$= \frac{1}{ab} \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{c_n c_m \cos(p_n u) \cos(q_m v) \cos(p_n u') \cos(q_m v')}{p_n^2 + q_m^2 + \varepsilon}$$

with constants c_n, c_m, p_n, q_m that only depend on n, m .

2.2.2 Smoothing by Poisson Equation [7]

The smoothed density $(\hat{D}_P(\bar{x}, \bar{y}))(u, v)$ is defined as the solution of the Poisson equation with zero-derivative at infinity:

$$\left(\frac{\partial^2}{\partial u^2} + \frac{\partial^2}{\partial v^2}\right)(\hat{D}_P(\bar{x}, \bar{y}))(u, v) = -(D(\bar{x}, \bar{y}))(u, v)$$

The solution can also be expressed explicitly by Green's function [15],

$$(\widehat{D}_p(\bar{x}, \bar{y}))(u, v) = -\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (D(\bar{x}, \bar{y}))(u', v') G_p(u, v, u', v') du' dv'$$

$$\text{where } G_p(u, v, u', v') = \frac{1}{2\pi} \ln \frac{1}{\sqrt{(u-u')^2 + (v-v')^2}}.$$

2.2.3 Gaussian Smoothing

The smoothed density $(\widehat{D}_G(\bar{x}, \bar{y}))(u, v)$ is defined as the convolution between the Gaussian function and the original density $(D(\bar{x}, \bar{y}))(u, v)$ which is

$$(\widehat{D}_G(\bar{x}, \bar{y}))(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (D(\bar{x}, \bar{y}))(u', v') G_G(u, v, u', v') du' dv'$$

$$\text{where } G_G(u, v, u', v') = \frac{1}{2\pi\sigma^2} e^{-\frac{(u-u')^2 + (v-v')^2}{2\sigma^2}}.$$

2.2.4 General Global Smoothing

We observed that the three smoothing techniques described above can be generalized into the following density smoothing operation:

$$(\widehat{D}(\bar{x}, \bar{y}))(u, v) = \int_0^a \int_0^b (D(\bar{x}, \bar{y}))(u', v') G(u, v, u', v') du' dv'$$

where the symmetry property $G(u, v, u', v') = G(u', v', u, v)$ is satisfied. Intuitively, $G(u, v, u', v')$ represents the amount of smoothed density at point (u, v) created by the original density at point (u', v') . We call this class of smoothing operation *global smoothing*, because $G(u, v, u', v')$ is usually non-zero for every pair of (u, v) and (u', v') , which indicates that the influence of the original density to the smoothed density is global.

The discussion and result in the remaining part of this paper will be based on this class of general smoothing operator. Please note that the lower limit and upper limit of the double integral are bounded in the above expression. This is only for the convenience of expression, and the following discussion is simple to extend for the unbounded cases.

Using this smoothing operator, we can transform the density constraints to the smoothed density constraints as below,

Discrete smoothed density constraints:

$$\begin{aligned} & \text{minimize } WL(\bar{x}, \bar{y}) \\ & \text{subject to } \widehat{D}_{ij}(\bar{x}, \bar{y}) = \widehat{I}_{ij} \quad (\text{OPT.1}) \\ & \text{for every } 1 \leq i \leq M, 1 \leq j \leq N \end{aligned}$$

Continuous smoothed density constraints:

$$\begin{aligned} & \text{minimize } WL(\bar{x}, \bar{y}) \\ & \text{subject to } (\widehat{D}(\bar{x}, \bar{y}))(u, v) = \widehat{I}(u, v) \quad (\text{OPT.2}) \\ & \text{for every } (u, v) \in [0, a] \times [0, b] \end{aligned}$$

The arc $\widehat{\cdot}$ is added on top of a density function to differentiate that it is a smoothed density function.

3. DENSITY PENALTY FUNCTION

The problems specified in (OPT.1) and (OPT.2) are constrained nonlinear programming problems. Their solution typically involves solving a sequence of unconstrained problems. Two commonly used methods are the quadratic penalty method and the augmented Lagrangian method, where the unconstrained problems optimize a combination of wirelength and penalty

functions on the density constraints. The details will be described in the following sections.

3.1 Quadratic Penalty Method

The quadratic penalty function for the problem (OPT.1) is

$$WL(\bar{x}, \bar{y}) + \frac{\mu}{2} \sum_{i=1}^M \sum_{j=1}^N (\widehat{D}_{ij}(\bar{x}, \bar{y}) - \widehat{I}_{ij})^2 w_B h_B$$

where μ is the quadratic penalty parameter.

And the quadratic penalty function for the problem (OPT.2) is

$$Q(\bar{x}, \bar{y}; \mu) = WL(\bar{x}, \bar{y}) + \frac{\mu}{2} \int_0^a \int_0^b ((\widehat{D}(\bar{x}, \bar{y}))(u, v) - \widehat{I}(u, v))^2 dudv$$

This function can be viewed as the limit of the discrete case when the resolution of placement bins becomes infinite. But more generally, it can be considered under the theory of vector space, so that problem (OPT.2) can be rewritten as:

$$\begin{aligned} & \text{minimize } WL(\bar{x}, \bar{y}) \\ & \text{subject to } (\widehat{D}(\bar{x}, \bar{y}))(*, *) = \widehat{I}(*, *) \end{aligned}$$

$\widehat{D}(\bar{x}, \bar{y})(*, *)$, $\widehat{I}(*, *)$ are considered vectors in the vector space $L^2([0, a] \times [0, b])$, which consists of all the two-dimensional functions defined in $[0, a] \times [0, b]$. In this vector space, for any two functions $g_1(*, *), g_2(*, *) \in L^2([0, a] \times [0, b])$, we can define their inner product as follows:

$$g_1(*, *) \odot g_2(*, *) = \int_0^a \int_0^b g_1(u, v) g_2(u, v) dudv$$

The norm of $g(*, *) \in L^2([0, a] \times [0, b])$ is defined as $\|g(*, *)\| = \sqrt{g(*, *) \odot g(*, *)}$. With these definitions, the basic concepts, like the limit and the convergence, can also be defined, and the convergence of the quadratic penalty method in vector space can be proved seamlessly [9].

Under the notion of vector space, the quadratic penalty function can be written as:

$$Q(\bar{x}, \bar{y}; \mu) = WL(\bar{x}, \bar{y}) + P_Q(\bar{x}, \bar{y}; \mu)$$

$$\text{where } P_Q(\bar{x}, \bar{y}; \mu) = \frac{\mu}{2} \|\widehat{D}(\bar{x}, \bar{y}) - \widehat{I}\|^2.$$

And their gradients can be computed as follows

$$\nabla Q(\bar{x}, \bar{y}; \mu) = \nabla WL(\bar{x}, \bar{y}) + \nabla P_Q(\bar{x}, \bar{y}; \mu)$$

$$\nabla P_Q(\bar{x}, \bar{y}; \mu) = \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I}) \odot \nabla \widehat{D}(\bar{x}, \bar{y})$$

with scalar μ as the quadratic penalty parameter.

We call $P_Q(\bar{x}, \bar{y}; \mu)$ the *density penalty function* for the quadratic penalty method, and call $\nabla P_Q(\bar{x}, \bar{y}; \mu)$ the *gradient of the density penalty function*.

The algorithmic framework of the quadratic penalty method [14] is given in Algorithm 1.

3.2 Augmented Lagrangian Method

Similar to the quadratic penalty method in vector space, the augmented Lagrangian function for problem (OPT.2) can be written as:

$$L_A(\bar{x}, \bar{y}, \lambda; \mu) = WL(\bar{x}, \bar{y}) + P_A(\bar{x}, \bar{y}, \lambda; \mu)$$

$$\text{where } P_A(\bar{x}, \bar{y}, \lambda; \mu) = \lambda \odot (\widehat{D}(\bar{x}, \bar{y}) - \widehat{I}) + P_Q(\bar{x}, \bar{y}; \mu),$$

and its gradient is

Given

$\mu^{(0)} > 0$, tolerance $\tau^{(0)} > 0$,
and a starting point $(\bar{x}^{(0)}, \bar{y}^{(0)})$;

for $k = 0, 1, 2, \dots$

Starting at $(\bar{x}^{(k)}, \bar{y}^{(k)})$,

Find an approximate minimizer $(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})$,
such that $\|\nabla Q(\bar{x}^{(k+1)}, \bar{y}^{(k+1)}; \mu^{(k)})\| \leq \tau^{(k)}$;

if a convergence test is satisfied

Stop with approximate solution $(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})$;

end if

Choose new penalty parameter $\mu^{(k+1)} > \mu^{(k)}$;

Select tolerance $\tau^{(k+1)}$;

end for

Algorithm 1 – Quadratic Penalty Method

$$\nabla L_A(\bar{x}, \bar{y}, \lambda; \mu) = \nabla WL(\bar{x}, \bar{y}) + \nabla P_A(\bar{x}, \bar{y}, \lambda; \mu)$$

where $\nabla P_A(\bar{x}, \bar{y}, \lambda; \mu) = \lambda \odot \nabla \widehat{D}(\bar{x}, \bar{y}) + \nabla P_Q(\bar{x}, \bar{y}; \mu)$

$$= (\lambda + \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I})) \odot \nabla \widehat{D}(\bar{x}, \bar{y})$$

with scalar μ as the quadratic penalty parameter and $\lambda(*, *) \in L^2([0, a] \times [0, b])$ as the Lagrangian multiplier.

Similarly, we call $P_A(\bar{x}, \bar{y}, \lambda; \mu)$ the *density penalty function* for the augmented Lagrangian method, and call $\nabla P_A(\bar{x}, \bar{y}, \lambda; \mu)$ the *gradient of the density penalty function*. The algorithmic framework of the augmented Lagrangian method [14] is given in Algorithm 2.

4. GRADIENT COMPUTATION

From Section 3:

$$\nabla P_Q(\bar{x}, \bar{y}; \mu) = \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I}) \odot \nabla \widehat{D}(\bar{x}, \bar{y})$$

$$\nabla P_A(\bar{x}, \bar{y}, \lambda; \mu) = (\lambda + \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I})) \odot \nabla \widehat{D}(\bar{x}, \bar{y})$$

The gradient of the density penalty function of either the quadratic penalty method or the augmented Lagrangian method has the common form $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$, where the function $g = \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I})$ in the quadratic penalty method and $g = \lambda + \mu(\widehat{D}(\bar{x}, \bar{y}) - \widehat{I})$ in the augmented Lagrangian method. It is a $2n$ -dimension vector, whose components are $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$ and $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$ for $k = 1, 2, \dots, n$.

The computation of this gradient is required to be performed hundreds or thousands of times in solving the constrained nonlinear programming problems. However, the computation is not trivial. We will explain why a naïve method is not practical and how we efficiently solve this problem. In the following sections, we assume the run time for smoothing operation is $T(n)$, and it is superlinear (of higher order than $O(n)$), as we need to consider every cell during the smoothing operation.

4.1 Naïve computation

To compute $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$, a naïve method has to compute the components $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$ and $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$ one by one, and then expand the inner product for each component:

Given

$\mu^{(0)} > 0$, tolerance $\tau^{(0)} > 0$,
A starting point $(\bar{x}^{(0)}, \bar{y}^{(0)})$ and $\lambda^{(0)}$;

for $k = 0, 1, 2, \dots$

Starting at $(\bar{x}^{(k)}, \bar{y}^{(k)})$,

Find an approximate minimizer $(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})$
such that $\|\nabla L_A(\bar{x}^{(k+1)}, \bar{y}^{(k+1)}, \lambda^{(k)}; \mu^{(k)})\| \leq \tau^{(k)}$;

if a convergence test is satisfied

Stop with approximate solution $(\bar{x}^{(k+1)}, \bar{y}^{(k+1)})$;

end if

Update $\lambda^{(k+1)} \leftarrow \lambda^{(k)} + \mu^{(k)}(\widehat{D}(\bar{x}^{(k+1)}, \bar{y}^{(k+1)}) - \widehat{I})$;

Choose new penalty parameter $\mu^{(k+1)} \geq \mu^{(k)}$;

Select tolerance $\tau^{(k+1)}$;

end for

Algorithm 2 – Augmented Lagrangian Method

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} = \int_0^a \int_0^b g(u, v) \frac{\partial (\widehat{D}(\bar{x}, \bar{y}))(u, v)}{\partial x_k} dudv$$

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial y_k} = \int_0^a \int_0^b g(u, v) \frac{\partial (\widehat{D}(\bar{x}, \bar{y}))(u, v)}{\partial y_k} dudv$$

By discretization, the double integrals are computed through double summation, and the partial derivatives are computed through a certain finite difference scheme.

The detailed description is given in Algorithm 3. In the loop between lines 04-11, there are two time-consuming parts. Line 07 consumes $T(n)$ time in each loop, and the computation of $\{\widehat{D}_{ij}\}$ cannot be reused. Lines 06 and 08 consume $O(MN) = O(n)$ time, assuming the bin number is of the same magnitude as the number of modules. Therefore, the time complexity is $O(n)(T(n) + O(n)) = O(n)T(n)$ for this naïve computation.

4.2 Efficient Computation

To avoid the time-consuming part of the naïve computation, the equivalent expressions of the inner products $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$ and $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$ are derived for efficient computation, which are given in the following theorem.

Theorem 1. Let $D(\bar{x}, \bar{y})$ and $\widehat{D}(\bar{x}, \bar{y})$ be the density and smoothed density defined in Section 2, respectively, then for any function $g \in L^2([0, a] \times [0, b])$, we have

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} = \int_{y_k - h_k/2}^{y_k + h_k/2} \left(\widehat{g}(x_k + \frac{w_k}{2}, v) - \widehat{g}(x_k - \frac{w_k}{2}, v) \right) dv \quad (\text{EQ.1})$$

$$g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial y_k} = \int_{x_k - w_k/2}^{x_k + w_k/2} \left(\widehat{g}(u, y_k + \frac{h_k}{2}) - \widehat{g}(u, y_k - \frac{h_k}{2}) \right) du \quad (\text{EQ.2})$$

Proof. Since the smoothing operator is linear, $\widehat{D}(\bar{x}, \bar{y})$ can be decomposed to $\sum_{k=1}^n \widehat{D}_k(x_k, y_k)$. Thus,

$$\begin{aligned}
\frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} &= \frac{\partial \widehat{D}_k(x_k, y_k)}{\partial x_k} \\
&= \frac{\partial}{\partial x_k} \int_0^a \int_0^b (D_k(x_k, y_k))(u', v') G(u, v, u', v') du' dv' \\
&= \frac{\partial}{\partial x_k} \int_{x_k - w_k/2}^{x_k + w_k/2} \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, u', v') du' dv' \\
&= \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k + w_k/2, v') dv' \\
&\quad - \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k - w_k/2, v') dv'
\end{aligned}$$

The last step is derived according to the Leibniz Integral Rule.

Therefore, $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$

$$= \int_0^a du \int_0^b dv g(u, v) \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} \quad (1)$$

$$= \int_0^a du \int_0^b dv g(u, v) \left(\int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k + w_k/2, v') dv' - \int_{y_k - h_k/2}^{y_k + h_k/2} G(u, v, x_k - w_k/2, v') dv' \right) \quad (2)$$

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} dv' \left(\int_0^a du \int_0^b dv g(u, v) G(u, v, x_k + w_k/2, v') - \int_0^a du \int_0^b dv g(u, v) G(u, v, x_k - w_k/2, v') \right) \quad (3)$$

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} dv' \left(\int_0^a du \int_0^b dv g(u, v) G(x_k + w_k/2, v', u, v) - \int_0^a du \int_0^b dv g(u, v) G(x_k - w_k/2, v', u, v) \right) \quad (4)$$

$$= \int_{y_k - h_k/2}^{y_k + h_k/2} (\widehat{g}(x_k + w_k/2, v') - \widehat{g}(x_k - w_k/2, v')) dv' \quad (5)$$

Step (1) is the inner product defined in Section 3.1; Step (2) substitutes $\partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$ by the previous computation; Step (3) changes the order of integrals; Step (4) exchanges the variables of function G because of its symmetric property defined in Section 2.2.4; Step (5) applies the definition of smoothed density as in Section 2.2.4.

In the same way, we can also derive the expression (EQ.2) for $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k$.

The key insight of the equation (EQ.1) (or (EQ.2)) is that the integral of the product $g(u, v) \cdot (\partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k)(u, v)$ over the placement region is equivalent to the difference of the integrals of $\widehat{g}(u, v)$ along a pair of opposing edges on the module boundary. The naïve computation has to compute $\partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k$ for each k , but with the equation (EQ.1), we only need to compute $\widehat{g}(u, v)$ once and reuse it for all the modules. Thus we give a highly efficient computation of $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$ in Algorithm 4. The function `interpolate({gij}, (x, y))` computes $g(x, y)$ at any (x, y) by bilinear interpolation. And the numerical integration is computed by the function `integrate({gij}, (x1, y1), (x2, y2))` for (EQ.1) and (EQ.2).

Theorem 2. The computational complexity of Algorithm 4 is $T(n)$, no greater than the complexity of smoothing operation.

Proof. Lines 01-03 consume $T(n)$ time for smoothing. Integrals are computed numerically in each loop between line 04 and line 09. Because the lower limit and upper limit of these integrals

<p>Input: $(\bar{x}, \bar{y}), \{g_{ij}\}$ Output: $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$ Algorithm: 01: $\{D_{ij}\} \leftarrow \text{compute_density}(\bar{x}, \bar{y})$; 02: $\{\widehat{D}_{ij}\} \leftarrow \text{smooth}(\{D_{ij}\})$; 03: Select small $\Delta x, \Delta y$; 04: for $k = 1, 2, 3, \dots, n$ 05: $x_k \leftarrow x_k + \Delta x$; 06: $\{D_{ij}\} \leftarrow \text{compute_density}(\bar{x}, \bar{y})$; 07: $\{\widehat{D}_{ij}\} \leftarrow \text{smooth}(\{D_{ij}\})$; 08: $g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial x_k} \leftarrow \sum_{i=1}^M \sum_{j=1}^N g_{ij} \frac{\widehat{D}'_{ij} - \widehat{D}_{ij}}{\Delta x} w_B h_B$; 09: $x_k \leftarrow x_k - \Delta x$; 10: Compute $g \odot \frac{\partial \widehat{D}(\bar{x}, \bar{y})}{\partial y_k}$ as in lines 05-09; 11: end for</p>

Algorithm 3 –

Naïve Gradient Computation for the Density Penalty Function

<p>Input: $(\bar{x}, \bar{y}), \{g_{ij}\}$ Output: $g \odot \nabla \widehat{D}(\bar{x}, \bar{y})$ Algorithm: 01: $\{D_{ij}\} \leftarrow \text{compute_density}(\bar{x}, \bar{y})$; 02: $\{\widehat{D}_{ij}\} \leftarrow \text{smooth}(\{D_{ij}\})$; 03: $\{\widehat{g}_{ij}\} \leftarrow \text{smooth}(\{g_{ij}\})$; 04: for $k = 1, 2, 3, \dots, n$ 05: $x_L \leftarrow x_k - w_k/2, x_R \leftarrow x_k + w_k/2$; 06: $y_B \leftarrow y_k - h_k/2, y_T \leftarrow y_k + h_k/2$; 07: $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial x_k \leftarrow \text{integrate}(\{\widehat{g}_{ij}\}, (x_R, y_B), (x_R, y_T)) - \text{integrate}(\{\widehat{g}_{ij}\}, (x_L, y_B), (x_L, y_T))$; 08: $g \odot \partial \widehat{D}(\bar{x}, \bar{y}) / \partial y_k \leftarrow \text{integrate}(\{\widehat{g}_{ij}\}, (x_L, y_T), (x_R, y_T)) - \text{integrate}(\{\widehat{g}_{ij}\}, (x_L, y_B), (x_R, y_B))$; 09: end for</p>

<p>Function <code>integrate({g_{ij}}, (x₁, y₁), (x₂, y₂))</code> 01: Select the number M of intervals dividing the segment $(x_1, y_1) - (x_2, y_2)$; 02: $\Delta x \leftarrow (x_2 - x_1) / M, \Delta y \leftarrow (y_2 - y_1) / M$; 03: $I \leftarrow \text{interpolate}(\{g_{ij}\}, (x_1, y_1)) / 2 + \text{interpolate}(\{g_{ij}\}, (x_2, y_2)) / 2$; 04: for $k = 1, 2, 3, \dots, M - 1$ 05: $x \leftarrow x_1 + k\Delta x, y \leftarrow y_1 + k\Delta y$; 06: $I \leftarrow I + \text{interpolate}(\{g_{ij}\}, (x, y))$; 07: end for 08: $I \leftarrow I \cdot \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} / M$; 09: return I.</p>
--

<p>Function <code>interpolate({g_{ij}}, (x, y))</code> assume $(a_i, b_j) - (a_{i+1}, b_{j+1})$ is the bin containing (x, y); return $g_{i,j}(a_{i+1} - x)(b_{j+1} - y) / (w_B h_B) + g_{i,j+1}(a_{i+1} - x)(y - b_j) / (w_B h_B) + g_{i+1,j}(x - a_i)(b_{j+1} - y) / (w_B h_B) + g_{i+1,j+1}(x - a_i)(y - b_j) / (w_B h_B)$.</p>
--

Algorithm 4 –

Efficient Gradient Computation for the Density Penalty Function

depend on the module size, which are constants in the problem, they only require $O(1)$ time to compute each integral, and $O(n)$ time in total. Therefore $T(n) + O(n) = T(n)$ is the time complexity for Algorithm 4. \blacksquare

The advantage of Algorithm 4 is that $\{\hat{g}_{ij}\}$ can be reused in every loop, so that we only need to smooth $\{g_{ij}\}$ once. This algorithm reduces the run time by a factor of n in terms of computational complexity compared to the naïve method.

5. FURTHER ANALYSIS

In this section we perform further analysis on the theorems discovered in Section 4, and show that they provide the algorithmic foundation to explain why the heuristics used in the existing force-directed methods work in the nonlinear programming framework. We also show why our approach of using the quadratic penalty method or augmented Lagrangian method with efficient gradient computation of the density penalty function is more general, stable, and theoretically sound.

5.1 Spreading Effect

If we solve the problem by the quadratic penalty method (Section 3.1), we have to solve $\|\nabla Q(\bar{x}, \bar{y}; \mu)\| \leq \tau$ multiple times, where $-\nabla WL(\bar{x}, \bar{y}) - \nabla P_Q(\bar{x}, \bar{y}; \mu)$ is the steepest descent direction. The term $-\nabla WL(\bar{x}, \bar{y})$ is the direction to reduce wire length, and the term $-\nabla P_Q(\bar{x}, \bar{y}; \mu)$ has a spreading effect that leads to a density feasible solution.

It can be proven that there exists a smoothing operator such that the smoothed density \hat{D} satisfies $\hat{D} = \hat{D}_H$ where \hat{D}_H is the smoothed density by Helmholtz equation defined in Section 2.2.1. For small-size modules, the spreading effect $-\nabla P_Q(\bar{x}, \bar{y}; \mu)$ for the k -th module in x -direction can be approximated as follows:

$$\begin{aligned} & -(\hat{D}(\bar{x}, \bar{y}) - \hat{I}) \odot \frac{\partial \hat{D}(\bar{x}, \bar{y})}{\partial x_k} \\ &= -\int_{y_k - h_k/2}^{y_k + h_k/2} (\hat{D}(\bar{x}, \bar{y}) - \hat{I})(x_k + w_k/2, v) dv' \\ & \quad + \int_{y_k - h_k/2}^{y_k + h_k/2} (\hat{D}(\bar{x}, \bar{y}) - \hat{I})(x_k - w_k/2, v) dv' \\ &\approx -h_k (\hat{D}(\bar{x}, \bar{y}) - \hat{I})(x_k + w_k/2, y_k) \\ & \quad + h_k (\hat{D}(\bar{x}, \bar{y}) - \hat{I})(x_k - w_k/2, y_k) \\ &\approx -h_k w_k \partial (\hat{D}(\bar{x}, \bar{y})) (x_k, y_{ki}) / \partial u \\ &\approx -h_k w_k \partial (\hat{D}_H(\bar{x}, \bar{y})) (x_k, y_k) / \partial u \end{aligned}$$

This last expression is exactly the force computation in mPL6 [3], where $\partial (\hat{D}_H(\bar{x}, \bar{y})) (x_k, y_k) / \partial u$ was approximated by a finite difference scheme, and the module area $h_k w_k$ was the multiple factor used in mPL6 as a heuristic. From the assumptions made to reach these approximations, we know that this heuristic may fail while the module size variation is large. Compared to the heuristic force computed in [3], the spreading effect of our method is more general and is able to handle both small modules and large modules.

5.2 Holding Effect

Hold forces are used in Kraftwerk [7, 16] for stability and support of ECO. By using our gradient computation with the augmented Lagrangian method, we also discover the holding effect in our method.

If we solve the problem by the augmented Lagrangian method (Section 3.2), the multiplier λ at the k -th iteration is updated by $\lambda^{(k)} = \lambda^{(k-1)} + \mu^{(k-1)} (\hat{D}(\bar{x}^{(k)}, \bar{y}^{(k)}) - \hat{I})$. Since $(\bar{x}^{(k)}, \bar{y}^{(k)})$ is the solution from the $(k-1)$ -th iteration, we have $\|\nabla L_A(\bar{x}^{(k)}, \bar{y}^{(k)}, \lambda^{(k-1)}; \mu^{(k-1)})\| \approx 0$ and $\|\nabla L_A(\bar{x}^{(k)}, \bar{y}^{(k)}, \lambda^{(k-1)}; \mu^{(k-1)})\|$

$$\begin{aligned} &= \|\nabla WL(\bar{x}^{(k)}, \bar{y}^{(k)}) + \nabla P_A(\bar{x}^{(k)}, \bar{y}^{(k)}, \lambda^{(k-1)}; \mu^{(k-1)})\| \\ &= \|\nabla WL(\bar{x}^{(k)}, \bar{y}^{(k)}) + (\lambda^{(k-1)} + \mu^{(k-1)} (\hat{D}(\bar{x}^{(k)}, \bar{y}^{(k)}) - \hat{I})) \odot \nabla \hat{D}(\bar{x}^{(k)}, \bar{y}^{(k)})\| \\ &= \|\nabla WL(\bar{x}^{(k)}, \bar{y}^{(k)}) + \lambda^{(k)} \odot \nabla \hat{D}(\bar{x}^{(k)}, \bar{y}^{(k)})\| \end{aligned}$$

Therefore, $-\lambda^{(k)} \odot \nabla \hat{D}(\bar{x}^{(k)}, \bar{y}^{(k)}) \approx \nabla WL(\bar{x}^{(k)}, \bar{y}^{(k)})$.

In Kraftwerk, the hold force at the k -th iteration is defined as $F_{hold}^{(k)} = \nabla WL(\bar{x}^{(k)}, \bar{y}^{(k)})$ where $(\bar{x}^{(k)}, \bar{y}^{(k)})$ is the solution from the $(k-1)$ -th iteration. It is clear that this holding force is approximately $-\lambda^{(k)} \odot \nabla \hat{D}(\bar{x}^{(k)}, \bar{y}^{(k)})$.

Moreover, the steepest descent direction of the augmented Lagrangian function at the k -th iteration is:

$$-\nabla WL(\bar{x}, \bar{y}) - \lambda^{(k)} \odot \nabla \hat{D}(\bar{x}, \bar{y}) - \nabla P_Q(\bar{x}, \bar{y}; \mu^{(k)})$$

where $-\nabla P_Q(\bar{x}, \bar{y}; \mu^{(k)})$ is the spreading effect as analyzed in the previous section and $-\lambda^{(k)} \odot \nabla \hat{D}(\bar{x}, \bar{y})$ is approximately the holding force as in Kraftwerk at the starting point $(\bar{x}^{(k)}, \bar{y}^{(k)})$. The difference between Kraftwerk and our method is that the holding force is a constant during each iteration but $-\lambda^{(k)} \odot \nabla \hat{D}(\bar{x}, \bar{y})$ is not. The study of its effect will be left for future work.

By using the augmented Lagrangian method, we also separate the two kinds of effects, where the holding effect is related to the Lagrangian multipliers and the spreading effect is related to the penalty parameters.

6. EXPERIMENTAL RESULTS

We implemented a nonlinear programming solver for problem (OPT.2) defined in Section 2.2.4, by the quadratic penalty method referred in Section 3.1. The global placer is implemented in a multilevel placement framework as mPL6 [3], except that the Uzawa solver in each placement level is replaced by our quadratic penalty solver. Since the efficient gradient computation is implemented in the solver, we name this placer “mPL-g”.

The first experiment was performed on the IBM-HB+ benchmark [13], which consists of hard instances with large module size variation (as the analysis in Section 5.1 shows that it is under such conditions that the exact smoothed density gradient computation is superior to the heuristic used in mPL6). The placement results of SCAMPI, mPL6 and mPL-g are shown in Table 1. “HPWL-gp”, “ovl-gp” and “time-gp” are respectively the wire length, bin area overflow and run time after the global placement phase. “HPWL” and “time” are the final wirelength and total run time after the detailed placement phase. The global placement by either mPL6 or mPL-g is fed into the detailed placer of NTUplace-DP [5] for the final placement. The overlap-free condition was verified for the final placement. The results are averaged by using the geometric mean, and are normalized by setting our result to 1. We compared the mPL-g results with

SCAMPI [13] and mPL6 [3]. The stopping criterion of mPL6 and mPL-g was set to 2% constraint violation, which terminates the global placer if the total overflow of bin densities is less than 2%. With the same stopping criterion for this set of hard instances, mPL-g converges 2.8X faster than mPL6. The wirelength is 13% shorter and 15% shorter than SCAMPI and mPL6, respectively.

The second experiment was performed on the ICCAD'04 mixed-size placement benchmark [1]. In Table 2, We compare the placement results of mPL-g and mPL6, also with NTUplace-DP as the detailed placer. We can see that the improvement of mPL-g is consistent in every circuit, either after the global placement phase or after the detailed placement phase. For some of the circuits, mPL-g got more than 6% wirelength reduction compared to mPL6. The results reveal that the method of exact gradient computation leads the solver converge to a better solution.

We also experimented with the ISPD'05 [11] and ISPD'06 [10] placement contest benchmarks, showed in Table 3. As most macrocells are fixed in these benchmarks, the exact gradient computation in mPL-g does not result in significantly better wirelength, but it is still shorter than mPL6 in general, and shorter by more than 5% for some circuits.

The consistency of the improvement in the global placement phase is checked by using another detailed placer XDP [6]. The results (data omitted) are slightly better (about 1%) than using NTUplace-DP.

7. CONCLUSIONS

In this paper, we introduced a general class of density smoothing operators, and developed the theory and efficient algorithms for computing the gradient of density penalty functions in the nonlinear programming framework. This is the first time that the density-constrained placement problem with global smoothing technique can be solved exactly in the general nonlinear programming framework. We showed that such an approach supercedes the existing force-directed placement methods. The experiment on IBM-HB+ benchmark shows the effectiveness of our technique.

Moreover, our approach with efficient computation of the density gradient shows promise in the application to the following problems: (1) 3D placement with non-overlap constraints and Through-Silicon Via density constraints, and (2) thermal-aware placement with temperature constraints.

We expect that the use of smoothed densities and our gradient computation technique will be a general framework to study the placement problems with more complicated constraints, and to enable the application of the more complex nonlinear programming algorithms such as the augmented Lagrangian algorithm.

8. ACKNOWLEDGMENTS

This research is partially supported by National Science Foundation under CCF-0430077 and CCF-0528583. The authors would like to thank Prof. Tony Chan, Prof. Lieven Vandenberghe, John Lee and Eric Radke for the inspiring discussions.

9. REFERENCES

- [1] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa, and I. L. Markov, "Unification of Partitioning, Placement and Floorplanning," *Proceedings of the 2004 IEEE/ACM International Conference on Computer-Aided Design*, pp. 550-557, November 2004.
- [2] K. Arrow, L. Huriwicz, and H. Uzawa. *Studies in Nonlinear Programming*, Stanford University Press, Stanford, CA, 1958.
- [3] T. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie, "mPL6: Enhancement Multilevel Mixed-Size Placement with Congestion Control," in *Modern Circuit Placement*, ed. G.-J. Nam and J. Cong, Springer Publishers, 2007.
- [4] T. Chan, J. Cong, and K. Sze, "Multilevel Generalized Force-directed Method for Circuit Placement," *Proceedings of the 2005 International Symposium on Physical Design*, San Francisco, CA, pp. 185-192, April 2005.
- [5] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "A High-quality Mixed-size Analytical Placer Considering Preplaced Blocks and Density Constraints," *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, pp. 187-192, November 2006.
- [6] J. Cong and M. Xie, "A Robust Detailed Placement for Mixed-size IC Designs," *Proceedings of the 2006 Conference on Asia South Pacific Design Automation*, Yokohama, Japan, pp. 188-194, January 2006.
- [7] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," *Proceedings of the 35th Annual Conference on Design Automation*, San Francisco, CA, pp. 269-274, June 1998.
- [8] A. B. Kahng, S. Reda, and Q. Wang, "Architecture and Details of a High Quality, Large-scale Analytical Placer," *Proceedings of the 2005 IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, pp. 891-898, November 2005.
- [9] D. G. Luenberger. *Optimization by Vector Space Methods*, John Wiley & Sons, Inc., 1969.
- [10] G.-J. Nam, "ISPD 2006 Placement Contest: Benchmark Suite and Results," *Proceedings of the 2006 International Symposium on Physical Design*, San Jose, CA, pp. 167-167, April 2006.
- [11] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ISPD2005 Placement Contest and Benchmark Suite," *Proceedings of the 2005 International Symposium on Physical Design*, San Francisco, CA, pp. 216-220, April 2005.
- [12] W. C. Naylor, R. Donnelly, and L. Sha, *Non-linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer*, US Patent 6301693, October 2001.
- [13] A. N. Ng, I. L. Markov, R. Aggarwal, and V. Ramachandran, "Solving Hard Instances of Floorplacement," *Proceedings of the 2006 International Symposium on Physical Design*, San Jose, CA, pp. 170-177, April 2006.
- [14] J. Nocedal and S. J. Wright. *Numerical Optimization* 2nd ed., Springer, 2006.
- [15] A. D. Polyani. *Handbook of Linear Partial Differential Equations for Engineers and Scientists*, Chapman & Hall/CRC, 2002.
- [16] P. Spindler and F. M. Johannes, "Fast and Robust Quadratic Placement Combined with an Exact Linear Net Model," *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, San Jose, CA, pp. 179-186, November 2006.

IBM-HB+	SCAMPI		mPL6 + NTUplace-DP				mPL-g + NTUplace-DP					
	HPWL (e+06)	time (s)	HPWL-gp (e+06)	ovl-gp (%)	time-gp (s)	HPWL (e+06)	time (s)	HPWL-gp (e+06)	ovl-gp (%)	time-gp (s)	HPWL (e+06)	time (s)
ibm01	3.40	62.0	2.79	1.42%	42.2	3.31	47.1916	2.53	1.34%	26.6	3.01	29.6
ibm02	8	139.6	6.36	1.35%	248.4	7.41	257.371	5.21	1.08%	74.0	6.06	89.0
ibm03	9.5	104.6	8.55	1.12%	124.0	9.23	127.954	7.78	1.10%	60.5	8.59	69.5
ibm04	12.3	144.1	10.38	1.65%	261.9	11.20	275.889	9.54	1.28%	81.6	10.58	92.6
ibm06	11	170.0	8.86	0.89%	228.2	9.21	232.245	8.28	0.87%	63.1	8.69	70.1
ibm07	15.7	99.0	14.53	1.61%	159.0	16.63	198.04	13.85	1.62%	95.3	15.79	134.3
ibm08	20.5	188.4	19.46	1.80%	441.8	20.56	445.772	16.62	1.02%	116.8	18.48	126.8
ibm09	22.2	182.0	18.90	1.54%	412.8	21.09	417.799	14.17	1.54%	104.6	16.07	109.6
ibm10	55.2	319.9	53.33	8.60%	624.7	70.56	664.665	38.34	1.57%	197.4	43.75	223.4
ibm11	27.8	144.7	27.26	1.97%	540.5	30.03	555.49	23.47	1.64%	145.8	27.54	160.8
ibm12	67.6	406.1	54.24	1.47%	439.4	56.37	445.428	45.83	1.47%	178.2	48.56	188.2
ibm13	42.2	209.6	35.34	1.96%	575.2	39.11	582.192	31.46	1.60%	130.5	35.63	137.5
ibm14	66.4	268.3	69.43	1.83%	580.0	75.24	593.014	57.13	1.41%	204.2	67.01	234.2
ibm15	88.2	375.9	99.96	1.62%	1266.1	103.53	1274.09	75.82	1.05%	297.2	81.18	310.2
ibm16	106.2	306.3	97.32	1.98%	636.7	102.27	645.712	87.28	1.99%	251.0	95.45	267.0
ibm17	152.7	385.7	149.87	2.00%	1031.8	163.16	1043.76	135.75	1.76%	357.4	149.82	371.4
ibm18	77.8	192.3	81.07	1.98%	930.5	90.98	940.536	68.58	1.99%	233.2	83.54	250.2
GEO-MEAN	1.13	N/A				1.15	2.82				1.0	1.0

Table 1 – Experimental Results on IBM-HB+ benchmark

ICCAD'04	mPL6 + NTUplace-DP				mPL-g + NTUplace-DP							
	HPWL-gp (e+06)	time-gp (min)	HPWL-dp (e+06)	time-tot (min)	HPWL-gp (e+06)	time-gp (min)	HPWL-dp (e+06)	time-tot (min)				
ibm01	2.11	3.05	2.26	3.17	1.97	93.36%	3.35	109.84%	2.17	96.05%	3.50	110.53%
ibm02	4.83	4.45	4.84	4.68	4.54	94.00%	5.02	112.81%	4.58	94.56%	5.25	112.17%
ibm03	6.54	4.87	6.56	5.15	6.43	98.32%	5.28	108.42%	6.49	99.02%	5.53	107.31%
ibm04	7.38	5.39	7.67	5.72	7.05	95.53%	6.19	114.84%	7.75	101.02%	6.47	113.10%
ibm05	9.43	6.23	9.36	6.51	9.27	98.30%	6.72	107.87%	9.25	98.89%	6.99	107.27%
ibm06	5.72	6.82	5.71	7.20	5.67	99.13%	7.85	115.10%	5.77	100.99%	8.23	114.30%
ibm07	9.96	9.38	9.95	9.91	9.71	97.49%	10.25	109.28%	9.81	98.54%	10.78	108.78%
ibm08	12.26	13.89	13.63	14.42	11.42	93.15%	15.89	114.40%	12.80	93.89%	16.41	113.75%
ibm09	12.41	14.84	12.38	15.39	11.78	94.92%	17.09	115.16%	11.96	96.63%	17.67	114.84%
ibm10	28	19.02	29.08	19.90	27.44	98.00%	20.51	107.83%	28.83	99.15%	21.61	108.57%
ibm11	18.02	18.73	17.89	19.53	17.12	95.01%	21	112.12%	17.33	96.87%	21.78	111.54%
ibm12	32.49	20.68	32.88	21.61	31.56	97.14%	22.64	109.48%	32.34	98.37%	23.54	108.91%
ibm13	22.85	23.44	22.61	24.42	22.19	97.11%	26.11	111.39%	22.90	101.31%	27.06	110.80%
ibm14	37.14	36.7	36.16	38.68	35.52	95.64%	37.55	102.32%	36.16	99.99%	39.65	102.50%
ibm15	47.28	46.35	45.81	48.72	47.2	99.83%	48.58	104.81%	46.91	102.40%	50.90	104.47%
ibm16	57.93	62.71	56.58	65.03	54.02	93.25%	70.7	112.74%	53.92	95.29%	73.38	112.85%
ibm17	67.3	54.3	65.38	56.83	64.96	96.52%	56.62	104.27%	66.61	101.88%	59.27	104.29%
ibm18	44.23	55.96	42.48	58.94	42.49	96.07%	58.63	104.77%	42.05	98.98%	61.46	104.28%
MEAN						96.26%		109.86%		98.55%		109.46%

Table 2 – Experimental Results on ICCAD'04 benchmark

ISPD'05 ISPD'06	mPL6 + NTUplace-DP				mPL-g + NTUplace-DP							
	HPWL-gp (e+07)	time-gp (hour)	HPWL-dp (e+07)	time-tot (hour)	HPWL-gp (e+07)	time-gp (hour)	HPWL-dp (e+07)	time-tot (hour)				
adaptec2	9.35	0.81	9.29	0.85	9.05	96.79%	0.95	117.59%	9.00	96.87%	0.99	116.75%
adaptec4	19.75	2.65	19.60	2.74	19.70	99.73%	2.93	110.83%	19.59	99.97%	3.02	110.52%
bigblue1	10.17	0.97	9.82	1.00	9.94	97.80%	1.16	118.83%	9.64	98.10%	1.19	118.28%
bigblue2	15.67	2.65	15.27	2.79	15.65	99.88%	3.14	118.70%	15.30	100.17%	3.28	117.76%
bigblue3	36.00	3.54	34.81	3.73	36.04	100.11%	4.36	123.14%	35.05	100.67%	4.57	122.46%
bigblue4	87.91	8.33	84.33	8.89	87.76	99.83%	10.52	126.41%	84.55	100.25%	11.09	124.73%
adaptec5	34.67	3.73	33.72	3.88	34.40	99.22%	4.64	124.47%	33.70	99.96%	4.78	123.37%
newblue1	6.52	0.84	6.34	0.91	6.27	96.15%	1.15	136.24%	6.25	98.51%	1.22	133.98%
newblue2	20.11	2.43	19.97	2.50	20.08	99.81%	3.02	123.94%	20.02	100.27%	3.09	123.28%
newblue3	29.30	2.44	28.39	2.67	29.32	100.08%	3.16	129.51%	28.28	99.61%	3.41	127.78%
newblue4	25.61	2.96	24.93	3.10	24.82	96.90%	3.56	120.30%	23.62	94.74%	3.70	119.40%
newblue5	43.81	6.46	42.87	6.78	43.25	98.71%	6.95	107.54%	42.85	99.95%	7.26	107.10%
newblue6	52.31	5.57	50.36	5.96	51.71	98.84%	6.05	108.70%	50.06	99.40%	6.44	108.09%
newblue7	109.74	7.82	108.42	8.33	108.86	99.19%	9.99	127.82%	109.31	100.82%	10.56	126.66%
MEAN						98.79%		121.00%		99.23%		120.01%

Table 3 – Experimental Results on ISPD'05 and ISPD'06 placement contest benchmarks