UC San Diego

UC San Diego Electronic Theses and Dissertations

Title

Analytic VLSI Placement using Electrostatic Analogy

Permalink

https://escholarship.org/uc/item/8255z18z

Author

Lu, Jingwei

Publication Date

2014

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA, SAN DIEGO

Analytic VLSI Placement using Electrostatic Analogy

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science (Computer Engineering)

by

Jingwei Lu

Committee in charge:

Professor Chung-Kuan Cheng, Chair
Professor Li-Tien Cheng
Professor Ronald L. Graham
Professor Ryan Kastner
Professor Steven James Swanson

2014

Copyright

Jingwei Lu, 2014

All rights reserved.

The Dissertation of Jingwei Lu is approved and is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2014

DEDICATION

For my parents, Wen Lu and Xueming Wang.
For my wife, Yiheng Wang.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research adviser, Professor Chung-Kuan Cheng. When summarizing my achievements during the Ph.D. career, I feel quite astonished by so many positive changes made to myself, thanks to the efforts from Professor Cheng. I keep learning from him in every single research meeting, dinner conversation, email discussion, as well as many other ways. I am deeply impressed and impacted by his innovation, passion and perseverance. He has taught me how to think analytically, research theoretically and study comprehensively. All the lessons by him will remain beneficial to me throughout the rest of my life. It is a great honor of mine to study under his supervision, and it is beyond the words to express my gratitude to him.

I would like to thank all the members of my Ph.D. committee, including (in alphabetical order of the last name) Professor Li-Tien Cheng, Professor Ronald L. Graham, Professor Ryan Kastner and Professor Steven James Swanson. They have provided lots of comments and suggestions to my research work, especially from their special research area.

I would like to thank all the members of my research exam committee, including (in alphabetical order of the last name) Professor Andrew B. Kahng, Professor Daniele Micciancio and Professor Ramamohan Paturi. They have helped me a lot to the report and presentation of my research exam, which is particularly important during the beginning of my Ph.D. program.

I feel thankful to all the faculties and students I have been working with as teaching assistants, including but not limited to, Professor Chung-Kuan Cheng, Rossana Motta, Yixin Zhu and Lucia Wang, for CSE20 Discrete Mathematics, Professor Tajana Simunic Rosing, Rajib Kumar Nath, Ryan Andrew Mast, Ketan Pranav Supanekar and Josh Marxen for CSE140 Components and Design Techniques for Digital Systems. Their helps significantly improved my capability of class presentation and communica-

tion of technical ideas.

I would like to thank all the professors for the courses I have taken during my doctorate study, including (in alphabetical order of the last name) Professor Scott Baden for CSE260 Parallel Computation, Professor Kamalika Chaudhuri for CSE202 Algorithm Design and Analysis, Professor Chung-Kuan Cheng for CSE245 Computer-Aided Circuit Simulation and Verification and CSE291 High-Performance Interconnect, Professor Russell Impagliazzo for CSE200 Computability and Complexity, Professor Andrew B. Kahng for CSE241A (ECE260) VLSI Integrated Computing Circuitry and CSE248 Algorithmic and Optimization Fundamentals for VLSI CAD, Professor Sorin Lerner for CSE231 Advanced Compiler Design and CSE130 Programming Languages Principles and Paradigm, Professor Daniele Micciancio for CSE105 Introduction and Theory of Computation, Professor Alex Orailoglu for CSE243A Introduction to Synthesis Methods in VLSI CAD, Professor Steven Swanson for CSE240A Principles of Computer Architecture, Professor Yuanyuan Zhou for CSE221 Operating Systems. They have helped me quite a lot during the lectures, office hours, discussion sessions, etc., while I am able to broaden my horizon of knowledge and improve my skills of and collaboration.

I feel thankful to all the colleagues I have been working with from the research group lead by Professor Chung-Kuan Cheng, including but not limited to (in alphabetical order of the last name), Professor Quan Chen, Ryan Coutts, Peng Du, Xiang Hu, Ilgweon Kang, James Jeng-Hau Lin, Terry Chia-Hung Liu, Harry Hao Liu, Professor Yang Liu, Professor Liya Huang, Professor Weidong Huang, Professor Xiaoxia Huang, Haibin Su, Xinan Wang, Shih-Hung Weng, Professor Yun Xie, Eric Xiang Zhang, Professor Xuejun Zhang, Howard Hao Zhuang, as well as many others. I really enjoy the time when brain storming with them for research ideas. It is a fairly joyful experience of mine to work together with them.

I owe my gratitude to many staff members in the department of computer sci-

ence and engineering, including but not limited to, (in alphabetical order of the last name), Jocelyn E. Bernardo, Julie Corner, Steve Deiss, Jessica Gross, Steve Hopper, Josh Hufziger, Kathy Johns, Viera Kair, Lynne Keith-McMullin, Virginia Mcllwain, Nadyne Nawar, David Wargo, as well as many others. They have provided me with lots of advices, which helps me finish my doctorate study in a smooth way.

I would like thank all my classmates, roommates and friends in the University of California, San Diego and other California area, including but not limited to (in alphabetical order of the last name), Yue Cao, Jonas Wei-Ting Chan, Tuck-Boon Chan, Changkun Chen, Yi Chen, Xiyue Deng, Xuemei Ding, Xiaoxiao Fan, Zheng Fang, Zhou Fang, Sidi Fu, Zhe Fu, Ang Gao, Lijuan Geng, Ding Han, Kwangsoo Han, Arthur Shi Hu, Ryan Peng Huang, Yichao Huang, Kwangok Jeong, Jeff Fei Jia, Canby Xun Jiao, Xinxin Jin, Yanqin Jin, Yi Jin, Seokhyeong Kang, Brian Kim, Hongkee Kim, Lan Lan, Hyein Lee, Jiajia Li, Jihang Li, Jing Li, Jun Li, Yong Li, Bruce Chenhao Liu, Lan Liu, Lonnie He Liu, Robert Yang Liu, Yao Liu, Jiang Long, Feng Lu, Haoting Luo, Janarbek Matai, Pingfan Meng, Siddhartha Nath, Michael Nicholson, Professor Hefu Pu, Abbas Rahimi, Iris Yiying Ren, Kambiz Samadi, Vaishnav Srinivas, Hung-Wei Tseng, Mengting Wang, Chao Wei, Yen-Kuan Wu, Chen Xie, Dongyan Xu, Tianyin Xu, Fan Yang, Ruixin Yang, Yuan Yao, Sirena Yang Yu, Lelin Zhang, Liuyi Zhang, Lu Zhang, Shun Zhang, Xiang Zhang, Yi Zhang, Zhaoyu Zhang, Xiaoxiao Zheng, Yu Zheng, Linda Yuanqi Zhou, Xiang Zhou, Yuchun Zhou, Peter Xuan Zhu, as well as many others. It is a happy and productive experience of mine to take classes and discuss technical questions with them.

I feel very thankful to my old friends with whom I re-union in San Diego and other area of California, including but not limited to (in alphabetical order of last name), Yichuan Cai, Huanyu Chen, Yi Du, Qing He, Xu He, Di Hu, Rachel Lezhi Huang, Tao Li, Zhitao Li, Bin Liang, Diaochao Liu, Jie Liu, Qiang Ma, Kun Qian, Jifeng Qin, Feng

Sheng, Zhenwu Shi, Shengwei Wang, Xiang Wang, Xuan Wang, Xiaolan Xu, Feng Yuan, Cong Zhou, as well as many others. It is quite an enjoyable experience of mine to hang out with them at the spare time.

I would like to thank all my colleagues from other schools or relevant research groups, including but not limited to, Jackey Zijun Yan, Tao Lin, Natarajan Viswanathan and Professor Chris C.-N. Chu for supporting the binaries of FastPlace3.0, FastPlace-DP, POLAR, FLOP, and the modern mixed-size (MMS) benchmarks, Myung-Chul Kim and Professor Igor L. Markov for supporting the binaries of SimPL and ComPLx, Jonas Wei-Ting Chan and Professor Andrew B. Kahng for supporting the binary of APlace3, Meng-Kai Hsu and Professor Yao-Wen Chang for supporting the binaries of NTUplace3 and NTUplace3-unified, Guojie Luo and Professor Jason Cong for supporting the binary of mPL6 and mPL-R3D, as well as many other professors, doctors, researchers and students in the VLSI physical design and relevant research area.

I would like to thank all my current and prior colleagues in the company of Cadence Design Systems, Inc., including but not limited to (in alphabetical order of the last name) Charles Jay Alpert, Taufik Arifin, Chin-Chih Chang, Hongliang Chang, Jing Chen, Will Wenyong Deng, Inki Hong, Chi-Ping Hsu, Dennis Jen-Hsin Huang, Stephen Huang, Dae Hyun Kim, Kwang-Thai Lee, Jianmin Li, Zhuo Li, Pei Lin, Tao Luo, Yufeng Luo, Stefanus Mantik, Jiwoo Pak, Chin-Chi Teng, Lu Sha, Ganping Sun, Yi Wang, Yiu-Chung Wong, Linfu Xiao, Mehmet Can Yildiz, Kun Yuan, Ming Yue, Jackey Zijun Yan, Shane Shuo Zhang, Yanheng Zhang, as well as many others in placement, routing, optimization, and other relevant teams. They have substantially contributed to my research work and internship projects, I owe my deep gratitude to all of them.

I would like to sincerely acknowledge the financial support from Jacobs Fellowship, Cadence Design Systems, Inc. and National Science Foundation (NSF) CCF-1017864, for my Ph.D. program in the CSE department of UCSD, my research work on

VLSI placement, mixed-size placement, as well as many others. Without their supports, it is impossible for me to complete my Ph.D. study.

I owe the sincere gratitude to my whole family, my father Wen Lu, my mother Xueming Wang, my wife Yiheng Wang, my grandparents Songshan Lu, Jinxia Cui, Youdong Wang, Zixiu Yi, my parents-in-law Bo Wang, Ping Wang, my uncles Wu Lu, Qing Lu, Xueyu Wang, Yuenian Deng, Michael Hsuan Tao Yu, my aunts Hongwei Lu, Hongxin Lu, Xuequan Wang, Min Xiang, Michelle Xiaoqing Wang, my cousins Yuting Lu, Yuxuan Lu, Yunjie Shen, Huan Wang, Yan Lu, Xuan Zhao, Lu Zhao, Doris Jingxue Deng, Jenny Jingyi Yu, Qing Li. They have been always supporting any decisions I made, and have given me many, many helps. Although physically separated by the pacific ocean, I find their love and care always around me. Without their love, it is impossible for me to finish my Ph.D. program.

Chapter 3, in part, is a reprint of the material as it appears in "ePlace: Electrostatics based Placement using Fast Fourier Transform and Nesterov's Method" by Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Huang, Chin-Chi Teng and Chung-Kuan Cheng in ACM Transactions on Design Automation of Electronic Systems (TODAES). The dissertation author was the primary investigator and author of the paper.

Chapter 4, in part, is a reprint of the material as it appears in "ePlace-MS: Electrostatics based Placement for Mixed-Size Circuits" by Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, Dennis Huang, Yufeng Luo, Chin-Chi Teng and Chung-Kuan Cheng in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD). The dissertation author was the primary investigator and author of the paper.

VITA

2006	Bachelor of Science in Information Engineering Zhejiang University
2010	Master of Philosophy The Hong Kong Polytechnic University
2014	Doctor of Philosophy in Computer Science (Computer Engineering) University of California, San Diego

PUBLICATIONS

J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng and C.-K. Cheng, "ePlace-MS: Electrostatics based Placement for Mixed-Size Circuits", IEEE TCAD, to appear.

J. Lu, P. Chen, C.-C. Chang, L. Sha, D. Huang, C.-C. Teng and C.-K. Cheng, "ePlace: Electrostatics based Placement using Fast Fourier Transform and Nesterov's Method", ACM TODAES, to appear.

J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-S. Huang, C.-C. Teng and C.-K. Cheng, "ePlace: Electrostatics Based Placement Using Nesterov's Method", DAC 2014, pp. 1-6.

J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-S. Huang, C.-C. Teng and C.-K. Cheng, "FFTPL: An Analytic Placement Algorithm Using Fast Fourier Transform for Density Equalization", ASICON 2013, pp. 1-4.

H. Zhuang, J. Lu, K. Samadi, Y. Du and C.-K. Cheng, "Performance-Driven Placement for Design of Rotation and Right Arithmetic Shifters in Monolithic 3D ICs", ICCCAS 2013, pp. 509-513.

X. Zhang, J. Lu, Y. Liu and C.-K. Cheng, "Worst-Case Noise Area Prediction of On-Chip Power Distribution Network", SLIP 2014, pp. 1-8.

J. Lu and C.-W. Sham, “LMgr: A Low-Memory Global Router with Dynamic Topology Update and Bending-Aware Optimum Path Search”, ISQED 2013, pp. 213-238.

S. K. Han, K. Jeong, A. B. Kahng and J. Lu, “Stability and Scalability in Global Routing”, SLIP 2011, pp. 1-6.

J. Lu, W.-K. Chow and C.-W. Sham, “Clock Network Synthesis with Concurrent Gate Insertion”, PATMOS 2010, pp. 228-237.

J. Lu, W.-K. Chow, C.-W. Sham and E. F. Y. Young, “A Dual-MST Approach for Clock Network Synthesis”, ASPDAC 2010, pp. 467-473.

J. Lu, W.-K. Chow and C.-W. Sham, “Fast Power- and Slew-Aware Gated Clock Tree Synthesis”, IEEE TVLSI, 20(11) (2012), pp. 2094-2103.

J. Lu, W.-K. Chow and C.-W. Sham, “A New Clock Network Synthesizer for Modern VLSI Designs”, INTG J., 45(2) (2012), pp. 121-131.

C.-W. Sham, E. F. Y. Young and J. Lu, “Congestion Prediction in Early Stages of Physical Design”, ACM TODAES, 14(1) (2009), (12:1-18).

FIELDS OF STUDY

Major Field: Computer Science (Computer Engineering)
Studies in VLSI CAD
Professor Chung-Kuan Cheng

ABSTRACT OF THE DISSERTATION

Analytic VLSI Placement using Electrostatic Analogy

by

Jingwei Lu

Doctor of Philosophy in Computer Science (Computer Engineering)

University of California, San Diego, 2014

Professor Chung-Kuan Cheng, Chair

We develop a flat, analytic and nonlinear placement algorithm *ePlace*, which is more effective, generalized, simpler and faster than previous works. Based on the analogy between placement instance and electrostatic system, we develop a novel placement density function *eDensity*, which models every object as positive charge and the density cost as the potential energy of the electrostatic system. The electric potential and field distribution are coupled with density using a modified Poisson's equation, which is numerically solved by spectral methods using fast Fourier transform (FFT). Rather than conjugate gradient (CG) method by previous placers, we propose to use Nesterov's

method for faster convergence. The efficiency bottleneck on line search is resolved by steplength prediction through an equation of Lipschitz constant. Through empirical validation, ePlace outperforms all prior placers with better quality and efficiency. On average of ISPD 2005 benchmarks, ePlace outperforms the leading placer BonnPlace with 2.83% shorter wirelength and runs 3.05× faster. On average of ISPD 2006 benchmarks, ePlace outperforms the leading placer MAPLE with 4.59% shorter wirelength and runs 2.84× faster.

Based on the above placement prototype, we develop *ePlace-MS*, an electrostatics based placement algorithm for mixed-size circuits. The density function *eDensity* is extended to handle the mixed-size placement. We conduct detailed analysis on the correctness of the gradient formulation and the numerical solution, as well as the rationale of density equalization with its advantages over prior density functions. Nesterov's method is shown with high yet stable performance over mixed-size circuits. The steplength prediction methodology is enhanced with backtracking strategy to prevent overestimation. A nonlinear preconditioner is developed to minimize the topological and physical differences between large macros and standard cells. Besides, we devise a simulated annealer for direct macro-layout legalization. All the above innovations are integrated into our mixed-size placement prototype ePlace-MS, which outperforms all the related works in literature with better quality and efficiency. Compared to the leading-edge mixed-size placer NTUplace3, ePlace-MS produces up to 22.98% and on average 8.22% shorter wirelength over all the sixteen modern mixed-size (MMS) benchmark circuits with the same runtime.

Chapter 1

Introduction

In this chapter, we introduce the fundamentals of VLSI Placement, highlight its importance to the overall design quality, and illustrates the motivation to the research efforts on the placement and relevant topics. We then briefly discuss the main categories of algorithms for standard-cell and mixed-size placement, as well as the representative research innovations in literature. Finally, we discuss our major contributions to the research on analytic nonlinear placement, and outline the remainder of the dissertation.

1.1 Placement Basics

Placement plays an important role in the VLSI physical design automation [25, 31] for both random logic [33] and datapath intensive components [69]. Placement performance largely impacts the downstream stages of power grid design [63], clock tree synthesis [36], power optimization [37], global and detailed routing [39], post-layout simulation [18, 19] and design variability handling [67, 68]. The placement quality results highly correlate with timing [38], routability [16, 54], and power [35]. As the technology node enters the deep nanometer scale [24] with billion-transistor integration, the performance of **standard-cell placement** becomes dominant [42] on the overall quality of the design. Besides, more and more pre-designed IP blocks, macros and memory units are included in the modern IC design, in order to shorten the total turnaround. A

typical ASIC may embed up to thousands of large macros and millions of standard cells with huge topological and physical differences in between. The high design complexity and complication continuously challenge the capability of existing mixed-size placers. As a result, innovations of effective and efficient large-scale **mixed-size placement** algorithms become more and more desirable.

1.2 Prior Arts

Traditional **standard-cell placement** methods can be generally divided into four categories, namely (1) stochastic simulation (2) min-cut partition (3) quadratic minimization (4) nonlinear optimization, respectively. **Stochastic** approaches are usually based on simulated-annealing techniques, of which one representative work is Timberwolf [53]. Uphill climbing is probabilistically accepted to rescue the placer from local optima. Despite high solution quality, stochastic placement has high complexity and low convergence rate, which induces poor scalability to large circuits. **Min-cut** approaches recursively simplify the problem by partitioning the instance (netlist and placement region) into smaller sub-instances. Local optimum algorithms [4] are usually employed when the problem instance becomes sufficiently small. State-of-the-art works include Capo [52], Dragon [59] and Fengshui [3]. However, improper partitioning at early stages could induce unrecoverable quality loss to the final solution. **Quadratic** approaches approximate the net length using a quadratic function, which can be linearized by various net models [57]. The differentiability enables gradient-based minimization techniques [51]. Density equalization is performed by adding pseudo pins and nets to the physically overlapped cells with a linear term introduced to the cost function [13]. By solving the linear system, cells are iteratively dragged away from overfilled regions. State-of-the-art quadratic placers include FastPlace3.0 [61], RQL [60], SimPL [41], MAPLE [28], ComPLx [27], BonnPlace [58] and POLAR [30]. Despite

high placement efficiency, the solution quality and robustness usually lag behind nonlinear placers. **Nonlinear** approaches refer to the algorithms based on a framework of nonlinear optimization. Wirelength and density are modeled using smooth mathematical functions thus gradients can be analytically calculated. Wirelength models mainly include the log-sum-exp model [47] and the weighted-average model [23]. Density models mainly include the bell-shaped function [47], Gaussian equation [8] and Helmholtz equation [5]. The partial differential equation (PDE) can be solved by Green's function [11] or finite-difference method [5]. By Lagrange relaxation or penalty method, the grid density constraints are integrated into the objective function and solved by nonlinear CG method. State-of-the-art nonlinear placers include APlace3 [26], NTUPlace3 [8] and mPL6 [6]. Due to the high complexity of modeling functions, nonlinear approaches employ multi-level cell clustering to simplify the problem and accelerate the algorithm. However, the quality overhead is not negligible.

Prior **mixed-size placement** algorithms can be divided into three categories. **Two-stage** methods conduct placement in two separated phases, namely, floorplanning of macros followed by placement of standard cells. Location and orientation of macros are determined and fixed at the first phase without simultaneous optimization of standard cell layout. A placement follows to only optimize standard cells in the global scale. Based on an initial placement solution, MP-tree [9] packs macros along the chip boundaries to avoid overlapping with standard cells. The total amount of cell displacement and central placeable area are minimized and maximized, respectively. A constraint-graph (CG) algorithm [7] uses mathematical programming to minimize displacement while optimize macro positions and orientations. However, the limited or inaccurate information on the standard cell distribution could misguide the floorplanner at early stages, inducing suboptimal floorplan solution which inevitably degrades the overall quality. **Constructive** (floorplan-guided) approaches combine the advantages of both

floorplan and placement. The floorplanner conducts simultaneous optimization on both macros and soft blocks (clusters of standard cells). An incremental placement follows to further spread standard cells within only local scale. Capo [52] as a min-cut floorplacer combines the two steps together. A fixed-outline floorplanner is repeatedly invoked throughout the top-down placement framework providing guidance to the macro shifting. FLOP [65] groups cells into soft blocks with similar shapes and dimensions to the macros. A min-cut floorplanning approach [64] produces the initial positions for all the macros and clustered blocks with simultaneous optimization of orientations. Incremental global [61] and detailed placement further spread and legalize the standard cells within local scale. Nonetheless, the intrinsic limitation of min-cut partitioning and clustering algorithms usually induce suboptimal solutions in the placement perspective. Optimization space of standard cell placement could be substantially shrunk, while the quality loss is hard to recover. **One-stage** solution remains popular among most modern placement algorithms [22, 26, 27, 28, 30, 61]. Macros and standard cells are being placed simultaneously where the limitations discussed above can be well avoided. FastPlace3.0 [61] performs selective grid resizing to accommodate large macros with more whitespace. ComPLx [27] shreds macros into small objects with sizes similar to that of the standard cells. After placement finishes, each macro is reconstructed based on the gravity center of instances belonging to it. APlace3 [26] reshapes the smoothing curve of the density function to distinguish the smoothness of macro movement with that of standard cells. NTUplace3 [22] incorporates rotational and flipping components into the gradient function, which enables simultaneous optimization on the location of all the movable objects as well as the orientation of macros. As mentioned in [9, 65], macro and standard cell co-placement challenges the capability of modern analytic placement approaches. Despite largest search space, nevertheless, the substantial topological and physical differences between macros and standard cells might introduce gradient imbal-

ance and cause the solution hard to converge.

1.3 Our Contributions

In this dissertation, we develop a flat analytic algorithm *ePlace* [32, 33] for nonlinear global placement. ePlace is more effective, generalized, simpler and faster than previous approaches. In contrast to the multi-level framework in prior nonlinear placers, our algorithm conducts placement on the flat netlist. Moreover, we develop a novel density function *eDensity* [33] modeling the placement instance as an electrostatic system for density equalization. Unlike hierarchical density grid structures used in prior works, ePlace sticks to a flat density grid with constantly high resolution. Compared to previous nonlinear placers [6, 8, 26], ePlace avoids quality loss due to suboptimal cell clustering and low density resolution, especially at early placement iterations. The density function is formulated as the system potential energy, while the density gradient is defined to be the electric repulsive force. A modified Poisson's equation is proposed to couple the charge density with electric potential and field distribution, Neumann boundary condition is enforced to maintain the legality of the global placement solution. Based on the above definition, a fast numerical method is proposed to solve Poisson's equation using spectral methods [56] based on fast Fourier transform (FFT). it well satisfies the boundary condition and makes the local density gradient aware of global density information. The time complexity is only $O(n \log n)$ where n is the total number of movable elements. Besides, we propose to use Nesterov's method [34] for the nonlinear placement optimization. The steplength is determined as the inverse of the Lipschitz constant, which is dynamically predicted without computation overhead. The placement efficiency is improved by more than $2\times$ compared to the CG method (with line search). We further enhance the performance of the nonlinear solver using a preconditioning technique to statically approximate the Hessian matrix of the objective

function. All the above innovations are integrated into the flat nonlinear placement algorithm *ePlace*, which is validated through experiments on the ISPD 2005 [46] and ISPD 2006 [45] benchmark suites. Empirical validation shows that ePlace outperforms all the state-of-the-art placers (Capo10.5 [52], FastPlace3.0 [61], RQL [60], MAPLE [28], ComPLx [27], BonnPlace [58], POLAR [30], APlace3 [26], NTUPlace3 [8], mPL6 [6]) with much better quality and better or comparable efficiency. On average of all the eight ISPD 2005 benchmarks, ePlace outperforms the leading placer BonnPlace [58] with 2.83% shorter wirelength and runs 3.05× faster. On average of all the eight ISPD 2006 benchmarks, ePlace outperforms the leading placer MAPLE [28] with 4.59% shorter wirelength and runs 2.84× faster.

Moreover, we extend the above standard-cell placement prototype to handle large-scale mixed-size circuits, based on the infrastructure in FFTPL [33] and ePlace [32]. We name this novel mixed-size placer as *ePlace-MS* [34, 40]. As the major difficulty of mixed-size placement remains in the broad spectrum of topological and physical attributes among all the movable objects (i.e., standard cells and large macros), our innovation of nonlinear preconditioning well equalizes them in the solver's perspective. As a generalized algorithm, ePlace-MS handles standard cells and macros in exactly the same way (c.f. macro shifting and smoothing [22], soft block formation by standard cells [61, 65], special macro density smoothing [26, 28], macro shredding [27], etc.) to ensure high and stable performance over various integrated circuits with potentially quite different structures of the design. We extend our prior density function *eDensity* [32, 33] to model mixed-size integrated circuits in a generalized way. Besides, we provide detailed analysis on eDensity with (1) rationale of removing the direct-current (DC) component from the spatial density distribution (2) correctness proof of the density gradient formulation (3) correctness proof of the numerical solution (4) advantages over density functions in previous placement algorithms. We also extend Nes-

terov's method as the nonlinear solver to handle mixed-size placement, with steplength dynamically predicted via Lipschitz constant. Moreover, we use a backtracking method to effectively prevent steplength overestimation. We develop an approximated nonlinear preconditioner to resolve the substantial topological and physical gap between standard cells and macros. The solution quality is significantly improved with negligible runtime overhead. We devise an annealing-based macro legalizer providing direct control to the macro shifting. A second-phase standard cell-only global placement is proposed to resolve the quality overhead induced during macro legalization. Finally, we integrate all the innovations into *ePlace-MS*, an electrostatics based placement prototype for mixed-size circuits, with promising experimental results obtained on the modern mixed-size (MMS) [65] benchmark suite. Empirical validation shows that ePlace-MS outperforms all the state-of-the-art mixed-size placement algorithms (Capo10.5 [52], FLOP [65], FastPlace3.0 [61], ComPLx [27], mPL6 [6], NTUplace3 [22]) with much shorter wirelength and shorter or comparable runtime. Specifically, ePlace-MS outperforms the leading placer NTUplace3 [22] with up to 22.98% and on average 8.22% shorter wirelength with the same runtime over all the sixteen MMS benchmarks [65].

1.4 Dissertation Outline

The dissertation is organized as follows. Chapter 2 introduces the background knowledge and related works in literature. Chapter 3 discusses our electrostatics based density function *eDensity*, the fast numerical solution by spectral methods using FFT, the nonlinear optimization by Nesterov's method, as well as the integration into our placement prototype *ePlace*. Chapter 4 discusses our extension to the mixed-size placement prototype *ePlace-MS*, where we conduct more thorough theoretical analysis on the density gradient formulation, the rationale of density equalization, the development of nonlinear preconditioner, as well as the simulated annealing based macro legalization.

Chapter 5 summarizes all the innovations developed in our works, discusses the advantages and tradeoffs, analyzes pending problems, and points out possible directions for future research works.

Chapter 2

Background

In this chapter, we first introduce the general picture and essential concepts of the abstracted placement instance. Then we formulate the problem of the analytic global placement optimization. We discuss the prior methodologies in literature, and analyze the arts and problems of each state-of-the-art quadratic and nonlinear placement algorithm.

2.1 Essential Concepts of Placement

A placement instance is formulated as a hyper-graph $G = (V, E, R)$, where V denotes the set of vertices (cells), E denotes the set of hyper-edges (nets) and R denotes the placement region, respectively. We use V_m and V_f to denote the movable cells and fixed macros in the node set V. Let $n = |V_m|$ denote the number of movable placement objects. A *legal solution* satisfies the following three requirements.

- Every cell is accommodated using enough free sites in the placement region.
- Every cell is horizontally aligned with the boundaries of one placement row.
- There is no overlap between cells or macros.

Based on the legality constraint, a placer targets minimizing the total HPWL of all the nets. Let $\mathbf{v} = (\mathbf{x}, \mathbf{y})$ denote a placement solution, where $\mathbf{x} = \{x_i | i \in V_m\}$ and $\mathbf{y} =$

$\{y_i | i \in V_m\}$ are the horizontal and vertical coordinates of all the cells. The HPWL of each net e is denoted as $HPWL_e(\mathbf{v})$ and defined in Eq. (2.1).

$$HPWL_e(\mathbf{v}) = \max_{i,j \in e} |x_i - x_j| + \max_{i,j \in e} |y_i - y_j|. \tag{2.1}$$

The total HPWL is then computed as $HPWL(\mathbf{v}) = \sum_{e \in E} HPWL_e(\mathbf{v})$ and we have the placement problem defined in Eq. (2.2).

$$\min_{\mathbf{v}} HPWL(\mathbf{v}) \text{ s.t. } \mathbf{v} \text{ is a legal solution.} \tag{2.2}$$

2.2 Definition of Global Placement

Global placement is usually regarded as a problem of constrained optimization. The placement region is uniformly decomposed into a set of $m \times m$ rectangular grids (bins) denoted as B. Based on a placement solution $\mathbf{v}$, let $\rho_b(\mathbf{v})$ denote the density of each grid b as expressed in Eq. (2.3).

$$\rho_b(\mathbf{v}) = \sum_{i \in V} l_x(b,i) l_y(b,i). \tag{2.3}$$

Here $l_x(b,i)$ and $l_y(b,i)$ denote the horizontal and vertical overlaps between the grid b and the cell i. Both $l_x(b,i)$ and $l_y(b,i)$ exhibit a rectangular shape, which is not differentiable at boundary points. As Eq. (2.4) shows, a global placement problem targets a solution $\mathbf{v}$ with minimum total HPWL subject to the constraint that the density $\rho_b(\mathbf{v})$ of all the grids are equal or below a predetermined target placement density ρ_t.

$$\min_{\mathbf{v}} HPWL(\mathbf{v}) \text{ s.t. } \rho_b(\mathbf{v}) \leq \rho_t, \forall b \in B. \tag{2.4}$$

2.3 Wirelength Smoothing

As Eq. (2.1) shows, the wirelength function $HPWL(\mathbf{v})$ is not differentiable and hard to minimize. As a result, various smoothing techniques have been developed to improve the differentiability thus convergence rate. Here we only discuss the horizontal part of the wirelength smoothing function while the vertical part can be obtained in a similar way.

Log-Sum-Exp (LSE) wirelength model is proposed in [47] and widely used in recent nonlinear placers [6, 8, 26]. For each net $e = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with n pins, the LSE function approximates the horizontal span of net e as Eq. (2.5) shows.

$$W_e(\mathbf{v}) = \gamma \left(\ln \sum_{i \in e} \exp\left(\frac{x_i}{\gamma}\right) + \ln \sum_{i \in e} \exp\left(\frac{-x_i}{\gamma}\right) \right). \tag{2.5}$$

Here γ is the smoothing parameter, which can be used to control the modeling accuracy[1]. As discussed in [62], the modeling error is upper-bounded by $\varepsilon_{LSE}(e) \leq \gamma \ln n$.

Weighted-Average (WA) wirelength model is proposed in [23]. Eq. (2.6) shows the horizontal function of net e

$$W_e(\mathbf{v}) = \left(\frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)} \right), \tag{2.6}$$

where similarly γ is used for accuracy control. [23] shows that the modeling error is upper-bounded by $\varepsilon_{WA}(e) \leq \frac{\gamma \Delta x}{1+\exp \Delta x/n}$, which is roughly half of that of $\varepsilon_{LSE}(e)$. In this work, we use the WA wirelength model for our nonlinear placement prototype *ePlace*, and use both LSE and WA wirelength models for our mixed-size placement algorithm *ePlace-MS*.

[1] The HPWL smoothing parameter γ cannot be set to arbitrarily small due to the computation precision constraint.

2.4 Density Penalty

As Eq. (2.4) shows, a legal global placement solution requires all the $|B|$ grid density constraints to be satisfied simultaneously, where $|B|$ could be of million-scale or even larger for the modern IC design. As a result, all the constraints are usually cast into a single penalty function $N(\mathbf{v})$ as shown in Eq. (2.7). By definition, all the $|B|$ density constraints will be satisfied if and only if we have $N(\mathbf{v}) = 0$.

$$\rho_b(\mathbf{v}) \leq \rho_t, \forall b \in B \Leftrightarrow N(\mathbf{v}) = 0. \tag{2.7}$$

Quadratic placement approaches usually model the density penalty as a linear or quadratic function, which can be easily integrated into their objective function. The penalty in UPlace [66] is *explicitly* devised as a weighted sum of all the frequency components of the density function. Specifically, $N(\mathbf{v}) = \sum_{u,v} w_{u,v} a_{u,v}^2$, where u and v are the discrete frequency indexes, $w_{u,v}$ are the weight factors and $a_{u,v}$ are the frequency coefficients. Notice that each frequency component is a differentiable wave function, of which the smooth curve can help direct gradient-based optimization in an effective way. The above penalty is fitted into a quadratic form and integrated into the objective function. Other quadratic placers [13, 27, 28, 30, 57, 61] modify the netlist by introducing anchor points, which *implicitly* produce the density penalty terms for the quadratic cost function.

Nonlinear placers have no constraints on the order of modeling functions thus are able to design the penalty in more flexible ways. APlace3 [26] and NTUPlace3 [8] use a quadratic penalty function with respect to grid density as Eq. (2.8) shows

$$N(\mathbf{v}) = \sum_{b \in B} \left(\widetilde{\rho}_b(\mathbf{v}) - \rho_t\right)^2. \tag{2.8}$$

As the original density function $\rho_b(\mathbf{v})$ is not differentiable and hard to optimize, a smoothed density function $\widetilde{\rho}$ is used here by employing a "bell-shape" local smoothing technique [47]. In contrast to the penalty method as discussed above, mPL6 [6] directly applies Lagrange multipliers to all the density constraints. The density function in [6] is smoothed in a global scale by using Helmholtz equation (Eq. (7) in [5]).

In this work, we model the placement instance as an electrostatic system and devise the density penalty $N(\mathbf{v})$ to be the system potential energy. In the remaining part of the dissertation, we will use $N(\mathbf{v})$ to denote both density penalty and system energy alternatively. This modeling methodology is discussed in detail in Section 3.1 regarding how the density penalty and gradient are defined. A fast numerical solution to the density and potential related Poisson's equation (Eq (3.16)) is proposed in Section 3.2.

2.5 Nonlinear Optimization Formulation

Based on the smooth wirelength function $W(\mathbf{v})$ and density penalty function $N(\mathbf{v})$, nonlinear global placers [8, 26] formulate the objective function $f(\mathbf{v})$ using a penalty factor λ as follows

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda N(\mathbf{v}). \tag{2.9}$$

As both the wirelength function and the density penalty are smoothed thus differentiable, gradient-based optimization methods [20] are used in prior nonlinear placers [8, 26] to produce high-quality numerical solutions. Alternatively, Lagrange multipliers are also used [6] to formulate the objective function in a different form as below

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \sum_{b \in B} \lambda_b \left| \widetilde{\rho}_b(\mathbf{v}) - \rho_t \right|. \tag{2.10}$$

Here λ_b denotes the multiplier on the density constraint of the bin b. This approach might consume longer runtime due to the computation demand on the multipliers. Multi-level cell clustering is employed in all the previous nonlinear placers [6, 8, 26] to accelerate the placement algorithm. Despite efficiency improvement, the quality overhead due to sub-optimal clustering is not negligible.

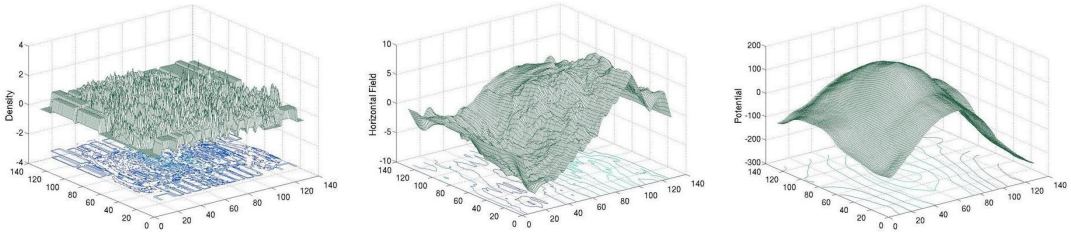
Chapter 3

ePlace: Electrostatics based Placement using Fast Fourier Transform and Nesterov's Method

In this chapter, we discuss our analytic nonlinear placement algorithm *ePlace*. We leverage the analogy between placement density constraint and electrostatic equilibrium state and develop a novel density function (*eDensity*). Spectral methods based on fast Fourier transform (FFT) is used to numerically solve the Poisson's equation. Instead of Conjugate gradient method, we use Nesterov's method as the nonlinear placement solver with steplength estimated as inverse of Lipschitz constant. The experiments validates the high and stable performance of our placement algorithm.

3.1 eDensity: A Novel Density Function by Electrostatic System Modeling

We propose a novel formulation of the density penalty and gradient function, *eDensity*, by modeling the entire placement instance as a two-dimension independent electrostatic system. The distribution of electric potential and field is determined by all the elements in the system. Each node i (a cell or a macro block) in the netlist is transformed to a positively charged particle (also denoted as i). The electric quantity q_i



(a) Electric density. **(b)** Horizontal electric field. **(c)** Electric potential.

Figure 3.1. The snapshots of electric density, horizontal field and potential distribution extracted at iteration 50. The placement is driven by only density force and conducted on the ISPD 2005 ADAPTEC1 benchmark.

of the particle is set to be the node area A_i. The motion of a movable cell i is driven by the electric force $\mathbf{F_i} = q_i \boldsymbol{\xi}_i$ formulated by Lorentz force law, where $\boldsymbol{\xi}_i$ is the local electric field. Similarly, the cell potential energy N_i is calculated as $N_i = q_i \psi_i$ where ψ_i is the electric potential at cell i. The correlation between the original placement instance and the transformed electric system is illustrated in Figure 3.2. By Coulomb's law, the electric field and potential at cell i are the superposition of the contribution from all the remaining cells in the system. An example of charge density $\rho(x,y)$, horizontal electric field $\xi_x(x,y)$ and potential $\psi(x,y)$ distribution in the entire placement region R is shown in Figure 3.1.

3.1.1 System Modeling Using Electrostatic Equilibrium

Based on the system modeling, we correlate the global placement constraint of even density distribution with the system state of electrostatic equilibrium. The electric force helps direct the charge (cell) movement towards the equilibrium state. By Gauss's law, the electric field equals the negative gradient of the potential as Eq. (3.1) shows

$$\boldsymbol{\xi}(x,y) = (\xi_x, \xi_y) = -\nabla \psi(x,y) = \left(-\frac{\partial \psi(x,y)}{\partial x}, -\frac{\partial \psi(x,y)}{\partial y} \right), \tag{3.1}$$

while the charge density equals the divergence of the electric field

$$\rho(x,y) = \nabla \cdot \boldsymbol{\xi}(x,y) = -\nabla \cdot \nabla \psi(x,y) = -\left(\frac{\partial^2 \psi(x,y)}{\partial x^2} + \frac{\partial^2 \psi(x,y)}{\partial y^2}\right). \tag{3.2}$$

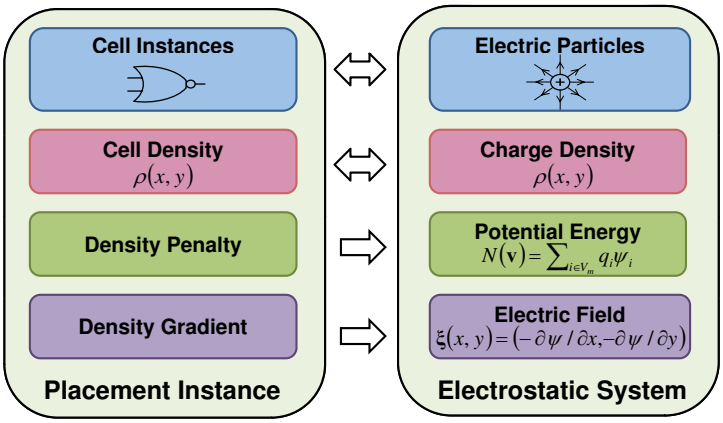
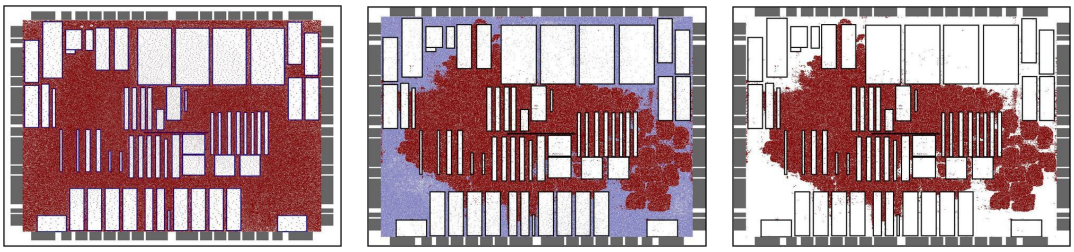


Figure 3.2. The placement instance is modeled as an electrostatic system. Each movable cell or fixed macro is transformed to a positive charge with the electric quantity set to be the node area. The density force is set as the electric force which drives cells apart from each other. The target of density equalization is equivalent to the system state of electrostatic equilibrium.

An electrostatic system with only positive charges will introduce only repulsion forces. The corresponding equilibrium state would have all the cells distributed along the chip boundaries where the global placement constraint is violated. As a result, we remove the direct-current (DC) component (i.e., the zero-frequency component) from the density distribution $\rho(x,y)$ to produce negative charges, while the integral of the density function over the placement region becomes zero. Specifically, since our density function transforms all the objects to be positive charges, a positive charge density distribution is thus produced. However, after removing the DC component from the spatial charge density distribution, under-filled placement regions with electric quantity below the original DC level become negatively charged. Meanwhile, the over-filled regions remain positively charged but with reduced electric quantity (DC is deducted from the original quantity). Cells at positively charged (i.e. highly over-filled) regions



(a) Placement without filler insertion. **(b)** Placement with filler insertion. **(c)** Placement with filler insertion (removed afterwards).

Figure 3.3. The distribution of standard cells and fillers at the end of global placement. Macros, standard cells and fillers are shown by black rectangles, red dots and blue dots, respectively. The total wirelength is shorter as fillers populate up whitespace thus squeeze cells to be placed closer. The placement is conducted on the ISPD 2005 ADAPTEC1 benchmark using Nesterov's method.

are attracted to the negatively charged regions, where the positive and negative charges neutralize with each other. Meanwhile, cells at negatively charged regions will mostly keep still. In the end, the system reaches the electrostatic equilibrium state with zero charge density over the entire placement region, while the total potential energy is reduced to zero. As a result, we model the placement density penalty and gradient using the system potential energy and electric field, respectively.

3.1.2 Density Penalty and Gradient Formulation

The total potential energy equals the sum of potential energy over all the charged elements of a new set V', which includes not only movable and fixed nodes from V, but also newly added fillers and dark nodes as discussed below.

Filler insertion: Let A_m denote the total area of all the movable nodes, while A_{ws} denotes the total area of white space. The target of even density distribution will overly spread the cells thus increase the wirelength, if we have the target density $\rho_t > \frac{A_m}{A_{ws}}$. Similar to [2, 6], we add fillers into the system, all of which are equally sized (rectangles), movable and disconnected (with zero pins). Let V_{fc} denote the set of filler cells. The

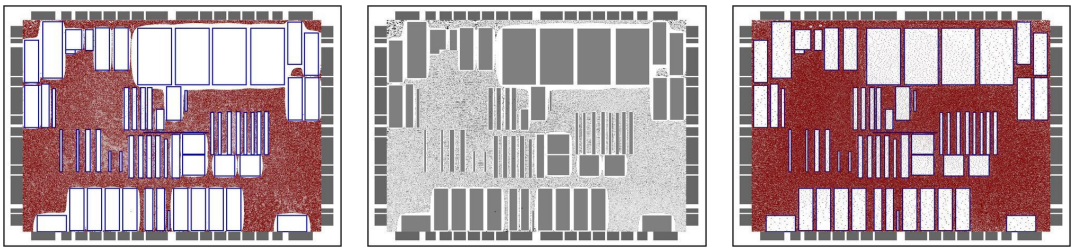
total area of filler cells is denoted as A_{fc} and defined as below.

$$A_{fc} = \rho_t A_{ws} - A_m. \tag{3.3}$$

We illustrate the effect of filler insertion in Figure 3.3. The additional density force due to filler insertion will squeeze the cells to be placed closer to their connected neighbors with density constraint still satisfied. The size of each filler i is denoted as A_i, which is determined based on the area distribution of the movable cells. Specifically, we set the filler size to be the average size of the mid 80% movable cells. The remaining top and bottom 10% largest and smallest cells are considered as noise factors and filtered out. All the fillers are removed from the final solution of global placement.

Dark node insertion: As a generalized approach, our method could handle any irregularly shaped placement region without loss on quality or efficiency. Suppose that the entire placement instance comprises a set of rectangular regions for cell placement. We impose a uniform grid R to cover all the placement regions. The total space within R but not belonging to any placement region will be decomposed into a set of rectangles, each is modeled as a *dark node*, which is processed in the same way as that of a fixed object in the problem instance. Let V_d denote the set of all the dark nodes and A_d denote the total area of all the dark nodes. Movable nodes will be stopped by the repelling force from the dark nodes when they are approaching the boundaries of any placement regions.

Density scaling: After the insertion of filler cells, we have the target density $\rho_t = \frac{A_m + A_{fc}}{A_{ws}}$. The area A_i of each fixed or dark node i must be scaled by the target density ρ_t, in order to maintain a globally equalized density distribution. Otherwise, the density force becomes higher than that of cells and fillers and repels cells away, while the whitespace around the fixed nodes is emptied with wirelength overhead induced as Figure 3.4 shows.



(a) Placement without macro density scaling.

(b) Density distribution without macro density scaling.

(c) Placement with macro density scaled by the target density ρ_t.

Figure 3.4. Without macro area scaling, the bin density at the macro blocks becomes higher than the target density ρ_t. As a result, the density force pushes the cells away from macros, inducing under-filled whitespace around macros and wirelength overhead.

Notice that our density scaling method will not introduce legalization issue. The electric quantity of each fixed or movable large macro is scaled down to the target placement density. Regions filled by small standard cells or covered by large macros will have the same charge density, there is no additional density force to drag cells away from macros. Without density scaling, it is impossible to achieve even charge density distribution over the entire domain.

Potential energy computation: Let $V' = V_m \cup V_f \cup V_{fc} \cup V_d$ denote the set of all the elements in the system. For each node $i \in V'$, let ρ_i, $\boldsymbol{\xi}_i$ and ψ_i denote the electric density, field and potential at the point where the node i locates. Given a placement solution $\mathbf{v}$ for both movable cells V_m and filler cells V_{fc}, the total potential energy N is defined in Eq. (3.4)

$$N(\mathbf{v}) = \frac{1}{2} \sum_{i \in V'} N_i = \frac{1}{2} \sum_{i \in V'} q_i \psi_i. \tag{3.4}$$

As the system energy equals the sum of mutual energy of all the pairs of charges, we have a factor of $\frac{1}{2}$ for the energy of each single charge. We cast the numerous grid density constraints into a single energy constraint of zero system energy ($N(\mathbf{v}) = 0$). Our density penalty is different from that of all the previous formulations [6, 8, 26] where it consists of a complete electrostatic system model with all the according physics laws strictly

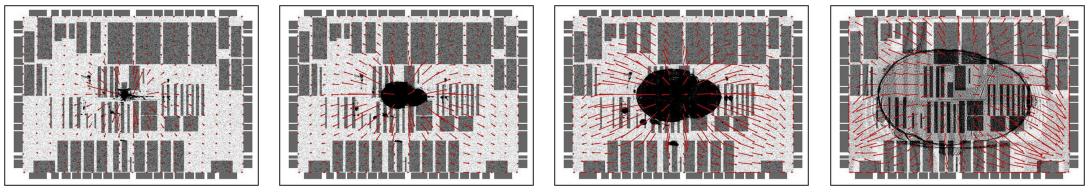
applied. By using the penalty factor λ, we could produce an unconstrained optimization problem as Eq. (3.5) shows

$$\min_{\mathbf{v}} f = W(\mathbf{v}) + \lambda N(\mathbf{v}), \tag{3.5}$$

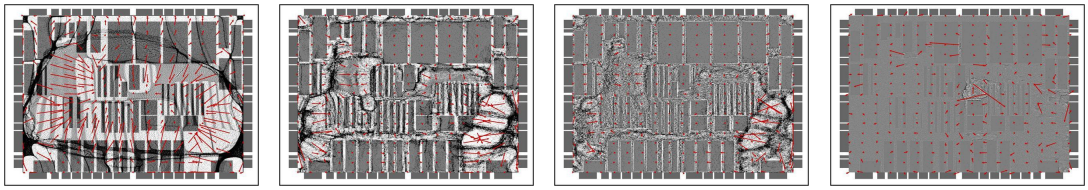
where $W(\mathbf{v})$ is by Eq. (2.6) and $f(\mathbf{v})$ is the objective cost function to minimize. As both $W(\mathbf{v})$ and $N(\mathbf{v})$ are smooth, we can generate the gradient vector by differentiating Eq. (3.5) as follows

$$\nabla f(\mathbf{v}) = \nabla W(\mathbf{v}) + \lambda \nabla N(\mathbf{v}) = \left(\frac{\partial W}{\partial x_1}, \frac{\partial W}{\partial y_1}, \cdots \right)^T - \lambda \left(q_1 \xi_{1_x}, q_1 \xi_{1_y}, \cdots \right)^T. \tag{3.6}$$

Modeling of density force orientation and magnitude remains a long-term controversial topic [42] in the analytic placement domain. For quadratic placement, it remains unclear where to introduce the anchor point for each cell in order to produce a proper dragging force. An ad-hoc force scaling is proposed in [13], while in RQL [60] the top 10% highest density force are empirically cut off to improve the quality. SimPL [41], MAPLE [28] and ComPLx [27] determine the anchor points by recursive netlist bi-partitioning, while the density force relies on initial condition and cutline determination. Without restriction on the function order, the density force formulation in nonlinear placement is of higher freedom. However, the Bell-shape smoothing technique [47] employed in [8, 26] incorporates only local information into force modeling, thus it is difficult for the placers to identify a global path of cell movement. Parameter adjustment in the smoothing function could help include remote density information but is highly case dependent and would consume more engineering effort and cause robustness issue. The algorithm in mPL6 [6] uses a more generalized approach with density force derived from potential differentiation. However, it lacks the electrostatics modeling method-



(a) Iter=1, N=1.25e12, $\tau = 97.0\%$. **(b)** Iter=5, N=6.07e11, $\tau = 93.1\%$. **(c)** Iter=6, N=2.98e11, $\tau = 79.8\%$. **(d)** Iter=7, N=5.40e10, $\tau = 42.4\%$.



(e) Iter=8, N=2.16e10, $\tau = 33.0\%$. **(f)** Iter=12, N=2.79e9, $\tau = 27.6\%$. **(g)** Iter=18, N=3.07e8, $\tau = 14.3\%$. **(h)** Iter=60, N=8.12e5, $\tau = 2.54\%$.

Figure 3.5. Snapshots of the density distribution $\rho(x,y)$ (grayscale) and the field distribution $\boldsymbol{\xi}(x,y)$ (red arrows) produced by *eDensity*. The placement is driven by only density force and conducted on the ISPD 2005 ADAPTEC1 benchmark, using Nesterov's method with preconditioning. Total potential energy and total density overflow are denoted by N and τ, respectively.

ology, which helps cast all the density constraints into one single energy function, as Eq. (3.4) shows. All of the existing problems indicate further improvement space for the density force formulation. Our analytic approach handles the problem by following the Lorentz force law, specifically

- The density force orientation on each cell aligns with that of the steepest descent of the density penalty (system potential energy).
- The density force magnitude on each cell is determined by its contribution to the reduction of the density penalty, as Eq (3.1) shows.
- The system density force vector is well balanced with the wirelength force vector using a single penalty factor, as Eq (3.5) shows.

As a result, our approach models the density force in a systematic way and it is validated by the experimental results in Section 3.5 with shorter wirelength and high efficiency.

3.1.3 Correctness of Gradient Formulation

As discussed in Section 3.1.1, we use $q_i\xi_{i_x}$ as the gradient of the density function $N(\mathbf{v})$ w.r.t. the horizontal movement of the charge i. However, by directly differentiating $N(\mathbf{v})$ w.r.t. x_i, we obtain the following formula

$$\begin{aligned}\frac{\partial N(\mathbf{v})}{\partial x_i} &= \frac{1}{2}\left(\frac{\partial N_i(\mathbf{v})}{\partial x_i} + \frac{\partial\left(\sum_{j\neq i} N_j(\mathbf{v})\right)}{\partial x_i}\right) = \frac{1}{2}q_i\frac{\partial \psi_i(\mathbf{v})}{\partial x_i} + \frac{1}{2}\sum_{j\neq i} q_j\frac{\partial \psi_j(\mathbf{v})}{\partial x_i} \\ &= \frac{1}{2}q_i\xi_{i_x}(\mathbf{v}) + \frac{1}{2}\sum_{j\neq i} q_j\frac{\partial \psi_j(\mathbf{v})}{\partial x_i},\end{aligned} \tag{3.7}$$

which is different from $q_i\xi_{i_x}$ with one extra term. By the nature of electrostatics, the potential at each charge i is the superposition of the potential contributed by all the remaining charges in the system. Let N_{i_j} denote the potential energy of charge i contributed by j, vice versa. Given a two-dimension rectangular electrostatic field (placement domain) R, we first illustrate the potential at certain distance $\mathbf{r}$ due to a charge q. By Gauss's law, we have

$$\nabla\cdot\boldsymbol{\xi} = \oint_q \boldsymbol{\xi}\,\mathrm{d}\mathbf{r} = \frac{q}{\varepsilon_0} \Rightarrow 2\pi\mathbf{r}\boldsymbol{\xi} = \frac{q}{\varepsilon_0} \Rightarrow \boldsymbol{\xi} = \frac{q}{2\pi\varepsilon_0\mathbf{r}} \tag{3.8}$$

By Poisson's equation, we have

$$\psi = -\int_{\mathbf{r_{ref}}}^{\mathbf{r}} \boldsymbol{\xi}\,\mathrm{d}\mathbf{r} = -\int_{\mathbf{r_{ref}}}^{\mathbf{r}} \frac{q}{2\pi\varepsilon_0\mathbf{r}}\mathrm{d}\mathbf{r} = -\frac{q}{2\pi\varepsilon_0}\ln\left(\frac{\mathbf{r}}{\mathbf{r_{ref}}}\right), \tag{3.9}$$

$\mathbf{r_{ref}}$ is the reference distance where ψ decreases to zero. As a result, for an electrostatic system defined on the two-dimension plane, we have

$$N_{i_j}(\mathbf{v}) = -\frac{q_i q_j}{2\pi\varepsilon_0}\ln\left(\frac{r_{i,j}(\mathbf{v})}{r_{ref}}\right) = N_{j_i}(\mathbf{v}), \tag{3.10}$$

where $r_{i,j}(\mathbf{v})$ is the physical distance between the two charges i and j based on the placement solution $\mathbf{v}$. r_{ref} is the reference distance where the potential by charge i (j) dimishes to zero, in this work we see it as the dimension of placement domain R. As a result, we have $N_{i_j}(\mathbf{v}) = N_{j_i}(\mathbf{v})$, thus the mutual potential energy of each pair of charges i and j are equivalent. By the principle of potential superposition, we have

$$N_i(\mathbf{v}) = \sum_{j \neq i} N_{i_j}(\mathbf{v}) = \sum_{j \neq i} N_{j_i}(\mathbf{v}). \tag{3.11}$$

Therefore,

$$\frac{\partial N(\mathbf{v})}{\partial x_i} = \frac{1}{2}\left(\frac{\partial N_i(\mathbf{v})}{\partial x_i} + \frac{\partial\left(\sum_{j \neq i} N_j(\mathbf{v})\right)}{\partial x_i}\right) = \frac{\partial N_i(\mathbf{v})}{\partial x_i} = q_i \frac{\partial \psi_i(\mathbf{v})}{\partial x_i} = q_i \xi_{i_x}(\mathbf{v}), \tag{3.12}$$

so $q_i \xi_{i_x}(\mathbf{v})$ is the actual gradient of $N(\mathbf{v})$ with respect to the horizontal movement Δx_i of the object i. Similarly, the density gradient of $N(\mathbf{v})$ with respect to the vertical movement of i is $q_i \xi_{i_y}(\mathbf{v})$. As a result, $q_i \boldsymbol{\xi}_i(\mathbf{v})$ is consistent with the gradient descent of the density cost (system potential energy) function $N(\mathbf{v})$.

3.2 Poisson's Equation and Numerical Solution

Based on our eDensity formulation in Section 3.1, we propose Poisson's equation to couple the charge density with electric potential and field. Neumann boundary condition is used to enforce the legality of the global placement solution. The Poisson's equation is numerically solved using spectral methods with high accuracy yet low complexity. Moreover, we propose a technique to locally smooth the density over discrete grids.

3.2.1 Well-Defined Poisson's Equation

By Gauss' law, the electric potential distribution $\psi(x,y)$ can be coupled with the density function $\rho(x,y)$ using Poisson's equation as Eq. (3.13) shows.

$$\nabla \cdot \nabla \psi(x,y) = -\rho(x,y),\ (x,y) \in R. \tag{3.13}$$

Here the density function equals the negative of the divergence of the gradient vector of the potential function. Let $\hat{\mathbf{n}}$ denote the outer normal vector of the placement region R and ∂R denote the boundary. When cells are moving towards the borderline of the placement region, the movement should be slowdown or stopped in order to prevent cells from moving outside. The electric (density) force is thus diminishing towards zero while approaching the boundary of the density function domain. As a result, we use the Neumann boundary condition which requires zero boundary gradient as Eq. (3.14) shows

$$\hat{\mathbf{n}} \cdot \nabla \psi(x,y) = \mathbf{0},\ (x,y) \in \partial R. \tag{3.14}$$

Besides, the integral of the density function $\rho(x,y)$ and the potential function $\psi(x,y)$ over the entire placement region R is set to be zero, as Eq. (3.15) shows

$$\iint_R \rho(x,y) = \iint_R \psi(x,y) = 0. \tag{3.15}$$

Therefore, all the constant factors introduced by the indefinite integration from density to field and potential become zero. Moreover, Eq. (3.15) ensures the unique solution to the partial differential equation (PDE) in Eq. (3.13). The problem due to the ill-defined PDE in [13] is thus overcome. Based on all the above definitions, we have our well-

defined Poisson's equation constructed as below

$$\begin{cases} \nabla \cdot \nabla \psi(x,y) = -\rho(x,y), \\ \hat{\mathbf{n}} \cdot \nabla \psi(x,y) = \mathbf{0},\ (x,y) \in \partial R, \\ \iint_R \rho(x,y) = \iint_R \psi(x,y) = 0. \end{cases} \tag{3.16}$$

There are several quadratic placement works [13, 57] in literature, of which the Poisson's equation is used. However, the PDE solution is only used to determine the location of anchor points. Some nonlinear placers [6] use Helmholtz equation to include two orders of derivatives to the smoothed density function. To guarantee unique PDE solution, a linear term is added to the equation with a self-tuned multiplier. Unlike all the previous PDE-based placement approaches, our method is based on a complete system model. The density penalty is formally formulated as the system potential energy. The Poisson's equation is used to compute the electric field, which together with electric quantity determine the density gradient by strictly following the Lorentz force law. The uniqueness of our PDE solution is promised by enforce zero integral of the potential, which not only simplifies the integration but also avoid the introduction of extra noise due to the linear term in [6].

3.2.2 Fast Numerical Solution using Spectral Methods

We propose a numerical solution using spectral methods [56] to effectively and efficiently solve the Poisson's equation in Eq. (3.16). Spectral methods express the solution to some PDE as the summation of basis functions (e.g., sinusoid and cosine waveforms) and choose the coefficients in the sum to satisfy the PDE and boundary conditions. A sinusoid function is an odd and periodic function. It diminishes to zero at the boundary of each period, which could naturally satisfy the Neumann condition as

stated in Eq. (3.14). As a result, we use sinusoid wave function as the basis function to express the electric field. As the density and potential functions are the derivative and integral of the field function, we use cosine wave as basis function to express them. Based on such decomposition at frequency domain, we use spectral methods to solve the Poisson's equation.

For expression using discrete cosine transformation (DCT), we modify the original density function $\rho(x,y)$ to an even and periodic form $\rho_{DCT}(x,y)$. Therefore, the new function can be decomposed into a group of cosine waveforms oscillating at different frequencies and constructed by DCT. Electric field and potential functions can be constructed by DCT and discrete sinusoidal transform (DST) in a similar way. The specific modification to the density function is as follows. Suppose the placement region R is uniformly decomposed into an $m \times m$ grid structure, thus the density function $\rho(x,y)$ is defined within the domain of $[0,m-1] \times [0,m-1]$. We mirror the density wave to the negative half-plane, such that the function domain is extended to $[-m,m-1] \times [-m,m-1]$, while the density function becomes even. Then we periodically extend the domain of the density function to $[-\infty,+\infty] \times [-\infty,+\infty]$. Based on these two modifications, the new density function $\rho_{DCT}(x,y)$ can be expressed using DCT as follows.

Let u and v denote integer indexes ranging from 0 to $m-1$. The frequency components are defined as $w_u = 2\pi\frac{u}{m}$ and $w_v = 2\pi\frac{v}{m}$, respectively. We use $a_{u,v}$ to denote the coefficient of each basis wave function of DCT. By definition, all the $m \times m$ coefficients can be generated by the integral of the density function multiplied by the basis wave functions over the 2D grid. The solution to each coefficient is shown in Eq. (3.17).

$$a_{u,v} = \frac{1}{m^2} \sum_{x=0}^{m-1} \sum_{y=0}^{m-1} \rho(x,y) \cos(w_u x) \cos(w_v y). \tag{3.17}$$

All the above coefficients can be rapidly computed by invoking FFT library only once.

Using these cosine coefficients, the new density function $\rho_{DCT}(x,y)$ can be expressed as a sum of cosine waves as Eq. (3.18) shows

$$\rho_{DCT}(x,y) = \sum_{u=0}^{m-1}\sum_{v=0}^{m-1} a_{u,v}\cos(w_u x)\cos(w_v y), \tag{3.18}$$

which can also be rapidly computed using one time of inverse FFT library invocation.

Based on Eq. (3.13), (3.15) and the cosine expression of the density function in Eq. (3.18), we have the solution to the potential function $\psi_{DCT}(x,y)$ as Eq. (3.19) shows

$$\psi_{DCT}(x,y) = \sum_{u=0}^{m-1}\sum_{v=0}^{m-1} \frac{a_{u,v}}{w_u^2+w_v^2}\cos(w_u x)\cos(w_v y), \tag{3.19}$$

which well satisfies Eq. (3.13). By Gauss's law, the electric field vector is the negative gradient of the potential function as Eq. (3.1) shows. Based on the solution to the potential function in Eq. (3.19), we can obtain the solution to the electric field $\boldsymbol{\xi}(x,y) = (\xi_{X_{DSCT}}, \xi_{Y_{DCST}})$ in the form of DCT and DST as Eq. (3.20) shows.

$$\begin{cases} \xi_{X_{DSCT}} = \sum_u \sum_v \frac{a_{u,v} w_u}{w_u^2+w_v^2}\sin(w_u x)\cos(w_v y), \\ \xi_{Y_{DCST}} = \sum_u \sum_v \frac{a_{u,v} w_v}{w_u^2+w_v^2}\cos(w_u x)\sin(w_v y). \end{cases} \tag{3.20}$$

Notice that the horizontal component $\xi_{X_{DSCT}}$ is constructed by sinusoid waves for the horizontal field, which diminishes to zero while reaching the end of a period thus the horizontal boundary of the placement region. Similar construction is conducted on the vertical field $\xi_{Y_{DCST}}$. Library support to the above numerical solutions can be found in various FFT packages [50].

UPlace [66] also employs DCT to transform the density function into the frequency domain. They form the density penalty using a weighted sum of all the frequency components, where the biased weights between different frequencies would help

improve the density equalization. In our approach, the DCT and DST are used in spectral methods to generate the solution to the partial differential equations, where density penalty and gradient are modeled as system potential energy and electric force. As a result, our approach is different from UPlace in the formulation of both density penalty and gradient.

3.2.3 Correctness of Numerical Solution

Poisson's equation in Eq. (3.16) is solved via spectral methods, which uses the fast Fourier transform (FFT) applied to the two-dimension (2D) spatial domain. Sinusoidal waveform approaches zero at the end of each function period, such behavior well matches the Neumann condition $\hat{\mathbf{n}} \cdot \nabla \psi(x,y) = \mathbf{0}$, $\forall (x,y) \in \partial R$ in Eq. (3.16), which requires zero gradient along the boundaries. As a result, we apply discrete sinusoidal transformation (DST) to the spatial field distribution $\boldsymbol{\xi}(x,y)$. As the electric potential and density distribution are the integral and derivative of the field, i.e., $\nabla \psi(x,y) = -\boldsymbol{\xi}(x,y)$ and $\rho(x,y) = \nabla \cdot \boldsymbol{\xi}(x,y)$, we reconstruct them via discrete cosine transformation (DCT). Based on an even mirroring and periodic extension, we have the DCT coefficients $a_{j,k}$ of the spatial density distribution $\rho(x,y)$ as

$$a_{j,k} = \frac{1}{m^2} \sum_{x=0}^{m-1} \sum_{y=0}^{m-1} \rho(x,y) \cos(w_j x) \cos(w_k y), \tag{3.21}$$

where w_j and w_k are frequency components. The density $\rho(x,y)$ can then be spatially expressed as

$$\rho(x,y) = \sum_{j=0}^{m-1} \sum_{k=0}^{m-1} a_{j,k} \cos(w_j x) \cos(w_k y). \tag{3.22}$$

As $\nabla \cdot \nabla \psi(x,y) = -\rho(x,y)$, we have the spatial potential distribution expressed as

$$\psi(x,y) = \sum_{j=0}^{m-1} \sum_{k=0}^{m-1} \frac{a_{j,k}}{w_j^2 + w_k^2} \cos(w_j x) \cos(w_k y). \tag{3.23}$$

Notice that for every pair of horizontal and vertical frequency components $\cos(w_j x)$ and $\cos(w_k y)$ from density $\rho(x,y)$ and potential $\psi(x,y)$, we have the Poisson's equation in Eq. (3.16) well satisfied in the numerical perspective as shown below.

$$\begin{aligned}
&\nabla \cdot \nabla \psi(x,y) = \frac{\partial^2 \psi(x,y)}{\partial x^2} + \frac{\partial^2 \psi(x,y)}{\partial y^2} \\
=&\frac{\partial^2 \left(\sum_j \sum_k \frac{a_{j,k}}{w_j^2+w_k^2} \cos(w_j x)\cos(w_k y) \right)}{\partial x^2} + \frac{\partial^2 \left(\sum_j \sum_k \frac{a_{j,k}}{w_j^2+w_k^2} \cos(w_j x)\cos(w_k y) \right)}{\partial y^2} \\
=&-\sum_j \sum_k \frac{a_{j,k} w_j^2}{w_j^2+w_k^2} \cos(w_j x)\cos(w_k y) - \sum_j \sum_k \frac{a_{j,k} w_k^2}{w_j^2+w_k^2} \cos(w_j x)\cos(w_k y) \\
=&-\sum_j \sum_k a_{j,k} \cos(w_j x)\cos(w_k y) = -\rho(x,y)
\end{aligned} \tag{3.24}$$

We remove the DC component ρ_{avg} from $\rho(x,y)$ by setting $a_{0,0} = 0$. The spatial field distribution is similarly expressed as below

$$\begin{cases}
\xi_x(x,y) = \sum_j \sum_k \frac{a_{j,k} w_j}{w_j^2+w_k^2} \sin(w_j x)\cos(w_k y), \\
\xi_y(x,y) = \sum_j \sum_k \frac{a_{j,k} w_k}{w_j^2+w_k^2} \cos(w_j x)\sin(w_k y),
\end{cases} \tag{3.25}$$

which also satisfies Eq. (3.16) in the numerical perspective as we have

$$\begin{aligned}
&\nabla \psi(x,y) = \left(\frac{\partial \psi(x,y)}{\partial x}, \frac{\partial \psi(x,y)}{\partial y} \right) \\
=&\left(\frac{\partial \left(\sum_j \sum_k \frac{a_{j,k}}{w_j^2+w_k^2} \cos(w_j x)\cos(w_k y) \right)}{\partial x}, \frac{\partial \left(\sum_j \sum_k \frac{a_{j,k}}{w_j^2+w_k^2} \cos(w_j x)\cos(w_k y) \right)}{\partial y} \right) \\
=&\left(-\sum_j \sum_k \frac{a_{j,k} w_j}{w_j^2+w_k^2} \sin(w_j x)\cos(w_k y), -\sum_j \sum_k \frac{a_{j,k} w_k}{w_j^2+w_k^2} \sin(w_j x)\cos(w_k y) \right) \\
=&(-\xi_x(x,y), -\xi_y(x,y)) = -\boldsymbol{\xi}(x,y)
\end{aligned} \tag{3.26}$$

Given $|V| = n'$ movable objects (standard cells, macros and fillers) in the netlist, ePlace-MS decomposes the placement region R into $m \times m$ grids, where $m = \sqrt{n'}$, to have

one object per grid on average. The above 2D-FFT computation thus costs exactly $O(n' \log n')$ runtime per iteration. As the number of fillers is at essentially the same order of the number of all the standard cells and movable macros, the complexity is essentially $O(n \log n)$. The well-formulated density gradient, global density smoothness and low computational complexity enables ePlace-MS to conduct placement on the flat netlist and the flat density grid with constantly high resolution. Compared to all the prior mixed-size nonlinear placers [6, 22, 26] with multi-level netlist clustering and grid coarsening, ePlace-MS avoids quality loss due to the suboptimal clustering and low density resolution, especially at early iterations.

3.2.4 Convergence

Our density function formulation is based on the analogy between an electrostatic system and a placement instance. For general case, the traditional bin packing problem has been proved to be NP-hard [10] thus it is intractable to prove its convergence. However, for homogeneous case, i.e., all the objects are of equal size, we can show the convergence through analogy of the charge distribution. Assume the final density distribution is not even, from Eq. (3.17) we know that there must be some density frequency coefficients $a_{u,v} \neq 0$. As a result, we have the respective electric field coefficients $\frac{a_{u,v} w_u}{w_u^2 + w_v^2} \neq 0$ and $\frac{a_{u,v} w_v}{w_u^2 + w_v^2} \neq 0$, which means that ξ_x and ξ_y are not zero. The electric force will then keep pushing the system potential energy to drop by gradient descent till finally a globally even density distribution is achieved. As a result, our density function has guaranteed convergence.

3.2.5 Behavior and Complexity Analysis

An example of discrete density and field distribution in a two-dimension plane is shown in Figure 3.5. The distribution of the electric field changes across different

iterations according to the variation of the density distribution. Therefore, the electric field dynamically directs the cells to the under-filled regions. From the figure we can also find that the electric field diminishes at the boundaries of the placement region. As also shown in Figure 3.1(b), such behavior satisfies the Neumann condition and the demand of global placement.

Suppose that we totally have n' cells ($n' = |V_m| + |V_{fc}|$) and an $m \times m$ grid imposed on the placement region. The total complexity of our numerical solution has two sources of contribution (1) density computation (2) potential and field computation.

Density computation: At each iteration, the density function is generated by the following two steps.

- Traversing all the bins in B to clear the cell density and cell area occupation of each bin to zero.
- Traversing all the cells in $V_m \cup V_{fc}$ to determine the area contribution of each cell to the according bins which overlap with the cell.

The first step consumes $O(m^2)$ time while the second step consumes $O(n')$ time. Totally it would consume $O(n' + m^2)$ time to generate the density distribution at each iteration.

Potential and field computation: At each iteration, we need to invoke FFT library for four times to solve Eq. (3.17), (3.19) and (3.20), respectively. Each 2D FFT library call consumes $O(m^2 \log m^2) = O(2m^2 \log m) = O(m^2 \log m)$ time, thus the total complexity is $O(m^2 \log m)$.

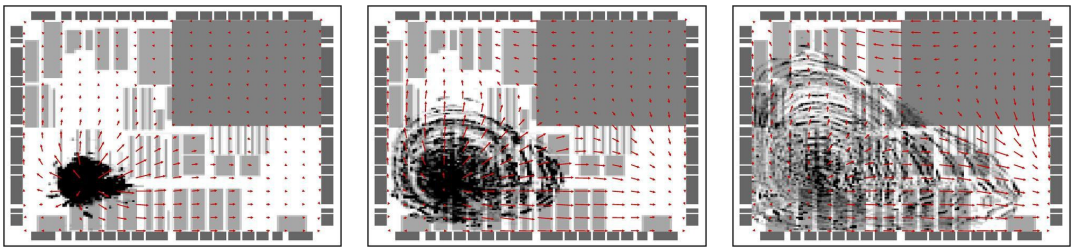
In general, our numerical solution has the computation complexity of $O(n' + m^2 \log m)$ for each placement iteration. As the number of grid is usually at the same scale of the number of cells (to ensure accuracy after discretization), we have $O(n') = O(m^2)$ and the total complexity is essentially $O(m^2 \log m)$ or $O(n' \log n')$. Addition of fillers could slightly increase the computation time but would not change the overall

complexity. All the fillers are equally sized towards the average of size of standard cells and will all be upsized to that of a single bin if utilization is small, thus the total number of fillers will not exceed $O(m^2)$. Moreover, as the number of fillers is at essentially the same order of that of movable placement objects, we have $n = O(n')$, thus the overall complexity is still $O(n \log n)$, where n is the number of movable placement objects.

There are many numerical solutions used in literature for the placement density function. Green's function is used in [13] to solve the PDE using 2D convolution. However, the computation complexity is high with $O(n^2)$ total runtime consumed. Bell-shape density smoothing is used in [8, 26], where by default the density gradient is aware of only local information. Global density variation could be included in local gradient computation by parameter adjustment in the smoothing function. However, as the gradient computation on each cell would take $O(m^2) = O(n)$ time, the total time is still $O(n^2)$. Our PDE solution with spectral methods provides better performance than the above numerical solutions, as it is aware of global density information while only takes $O(n \log n)$ time for each iteration. The density variation could be instantly propagated to all the placement grids due to the frequency decomposition in Eq. (3.18). As shown in Figure 3.6, local density gradient could be immediately adjusted based on the cell redistribution at remote area.

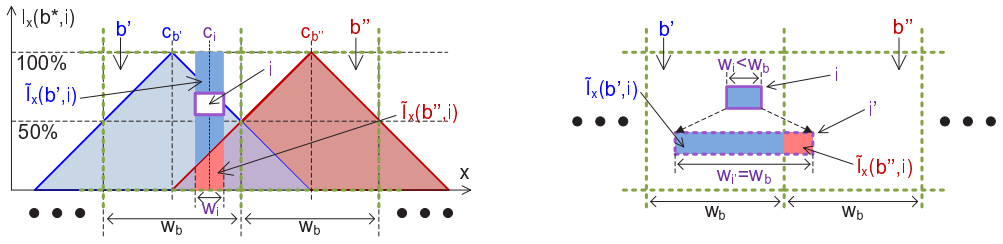
3.2.6 Local Smoothness Over Discrete Grids

Global smoothness by eDensity is achieved via Eq. (3.1) and Eq. (3.2). However, as the physical dimension of each density bin is usually larger than that of cells, local cell movement within a bin cannot be reflected in the density cost function, where smoothness is degraded. As a result, we propose a local smoothing technique to handle this issue, such that the density function by Eq. (3.4) could well reflect any infinitely small movement of cells within each bin. A one-dimension example is shown in Fig-



(a) **(b)** **(c)**

Figure 3.6. The spatial distribution of density force across different placement iterations. All the cells are initially squeezed into the lower-left subregion with an obstacle placed at the upper-right subregion. The local density gradient could immediately respond to the remote density variation and identify a global motion path for each overlapped cell to some remaining whitespace on the chip.



(a) **(b)**

Figure 3.7. A one-dimension illustration of our local density smoothing technique. Here the cell width is smaller than the bin width ($w_i < w_b$). We enlarge the cell to the dimension of one bin. As a result, movement of the cell i at any time will always change the overlaps between itself and the two bins b' and b'', thus change the density of b' and b'' simultaneously. There is no local smoothing applied when $w_i \geq w_b$.

ure 3.7. Here w_i and w_b are the widths of cell i and bin b, c_i and c_b are the coordinates of the centers of cell i and bin b, respectively. $l_x(i,b)$ and $\widetilde{l}_x(i,b)$ are the original and smoothed horizontal overlaps between the cell and the bin, so we have

$$\widetilde{l}_x(i,b) = \begin{cases} \left(1.0 - \frac{c_i - c_b}{w_b}\right) \times w_i & : c_i \in [c_b - w_b, c_b + w_b] \\ 0 & : c_i \in (-\infty, c_b - w_b) \cup (c_b + w_b, +\infty) \end{cases} \tag{3.27}$$

As the cell is being shifted rightwards, the contribution to the density of b' is linearly reduced, while the contribution to the density of b'' is linearly increased, respectively.

The total contribution of i to the two neighboring bins (b' and b'') is constant and equals w_i when the center of the cell c_i locates between the centers of the two bins $c_{b'}$ and $c_{b''}$. The smoothing effect is equivalent to the combination of cell dimension stretching and cell density lowering, which keeps the objective cost function analytic. Specifically, for each cell i, we conduct the local density smoothing as follows.

- If $w_i < w_b$, stretch the cell width from w_i to w_b and reduce the cell density from 1.0 to w_i/w_b.
- If $w_i \geq w_b$, keep the original cell width and density.

As a result, this smoothing technique is consistent over different granularity and cell dimensions. Notice that our local smoothing technique is being used at every iteration when updating the density map. It costs constant time for each object since only finite neighboring bins are affected by each object, thus the computation complexity is not changed.

3.2.7 Advantage Analysis

Density force modeling remains quite a controversial problem [42] in **quadratic placement**, where the best location of anchor point for each object is usually unclear. RQL [60] nullifies the top 10% density force vectors to suppress over-spreading of standard cells, while the empirical tuning lacks theoretical support and may not guarantee convergence. [13] uses **Green's function** to determine appropriate positions of cell anchors. The two-dimension convolution makes the complexity to be $O(n^2)$ thus is computationally expensive. Kraftwerk2 [57] determines the anchor position via solution to the Poisson's equation. Due to the function order restriction in the quadratic placement infrastructure, the density cost is degraded from exponential to linear, which helps achieve convexity and efficiency but loses quality. SimPL [41] and ComPLx [27]

determine the anchor position via recursive bi-partitioning, while convergence is theoretically promised via the primal-dual framework. Nevertheless, the solution quality is sensitive towards the initial solution. Moreover, it is hard to tell how much the optimum solution would follow the initial layout with minimum wirelength yet high overlap.

Nonlinear placement has no restriction on function orders thus ensures more flexibility in density modeling. However, the non-convexity of the density function remains a headache to the nonlinear solvers. **Bell-shape** method [47] covers only adjacent grids in the local scale. Iterative grid uncoarsening is usually conducted in prior nonlinear placers [22, 26] to keep consistent with the scale of clustered netlist. However, the quality degradation due to low density resolution is not negligible. Besides, such local density smoothness would force objects to detour around obstacles thus inevitably lower the convergence rate. Notice that bell-shape method could realize fully global density smoothness by parameter adjustment, nevertheless, the regarding complexity scales up to $O(n^2)$, which is numerically expensive. **Helmholtz equation** in [6] smooths the density in global scale with only $O(n \log n)$ runtime complexity. However, sub-optimality in the choice of the linear factor ε in the Helmholtz equation (Eq. (7) of [5]) introduces noises. Moreover, there is no formulation of density gradient functions in [6], where up to millions of constraints are simultaneously applied to all the grid density, which complicates the problem, degrades the placement quality and efficiency.

eDensity concisely formulates the placement density problem using the closed-form equation in Eq. (3.4). By differentiating it, we derive the gradient vector to direct density cost reduction, where by Eq. (3.5) only one penalty factor is needed for force balancing with wirelength. eDensity numerically solves the partial differential equation via the spectral methods in Eq. (3.23) and (3.25). Based on the nice properties of fast Fourier transform, it consumes exactly only $O(n \log n)$ runtime per iteration. At each grid, the local electric potential and field are impacted by the global density distribution,

while objects driven by density forces are able to freely move over blockages or macros, as Figure 4.4 shows. Moreover, the global smoothness enables all the movable objects in over-filled regions to detect whitespace at remote area, as illustrated in Figure 4.3, which helps quickly converge to the objective of even density. To this end, unlike all the prior methodologies in literature, eDensity approaches density equalization via directly simulating the behavior of a real electrostatic system, which in reality will always transfer towards the states of lower potential energy (until the energy decreases to zero), therefore theoretically guarantees the global convergence of eDensity. The nature of simulation enables us to use constantly high density resolution throughout the whole global placement, without any potential misguidance to the nonlinear solver.

3.3 Nonlinear Optimization

Global placement is proved to be an NP-complete problem [15]. Development of prior heuristics are mostly directed by mathematical derivation for quality and efficiency. As Eq. (2.9) shows, the objective function consists of a convex wirelength function [23] and usually a non-convex density function [47], where the property of non-convexity challenges the performance of modern convex programming methods. In this section, we first briefly introduce the Conjugate Gradient (CG) method [20] which is widely used in previous nonlinear placement works [8, 26], and discuss the efficiency bottleneck on the line search. Then we propose Nesterov's method to solve the nonlinear problem and illustrate our technique of Lipschitz constant prediction, which determines the steplength in constant time. To the best of our knowledge, our work is the first one in literature to incorporate Nesterov's method and Lipschitz constant prediction into global placement optimization. A comparison of placement quality and efficiency by using these two optimization methods in ePlace is shown in Section 3.5, where Nesterov's method could outperform CG method with 2.28% shorter wirelength and 2.21×

speedup on average of all the ISPD 2005 benchmarks. In the end, we discuss our preconditioning technique.

3.3.1 Conjugate Gradient Method with Line Search

Details of the CG method in one iteration is illustrated in Algorithm 1. Polak-Ribiere method is used to update β_k for correlation with previous search directions as line 2 shows. β_k is reset to zero when the conjugacy is lost. The search direction is computed at line 3. We use line search to determine the steplength, the best solution along the search path $\mathbf{d}_k$ and within the search interval α_k^{max} is obtained. In our approach, golden section search (GSS) is used to implement line search[1] as line 4 shows. The new solution for the current iteration is computed at line 5 and used as the initial solution for the next iteration, while CG would converge after a number of such iterations. CG

Algorithm 1. CG-Solver at kth iteration

Require: initial solution $\mathbf{v}_k$
objective function $f_k = f(\mathbf{v}_k)$ maximal and minimal search interval α_k^{max} and α_k^{min}
Ensure: local optimal solution $\mathbf{v}_{k+1}$
1: gradient vector $\nabla f_k = \nabla f(\mathbf{v}_k)$
2: Polak-Ribiere parameter $\beta_k = \max\left\{\frac{\nabla f_k^T(\nabla f_k - \nabla f_{k-1})}{\|\nabla f_{k-1}\|^2}, 0\right\}$
3: search direction $\mathbf{d}_k = -\nabla f_k + \beta_k \mathbf{d}_{k-1}$
4: steplength $\alpha_k = GSS\left(\mathbf{v}_k, f_k, \mathbf{d}_k, \alpha_k^{max}, \alpha_k^{min}\right)$
5: new solution $\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k$
6: **return** $\mathbf{v}_{k+1}$

targets optimization of locally quadratic functions. The closer f is to a quadratic form, the faster CG would converge. Otherwise, CG would easily lose the conjugacy with β reset to zero (line 2). As discussed in [55], the local error rate of CG method is bounded as $\|e_{(k)}\| \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e_{(0)}\|$, where $\|e_{(k)}\|$ is the error at the kth iteration and κ is the condition number of the Hessian matrix of the objective function, respectively. On the

[1] Within one iteration, the length of the search interval is recursively reduced by the golden ratio 0.618 in each step until the interval length is below α_k^{min}.

other side, the global convergence rate by CG method cannot exceed $O(1/k)$ [48]. Despite the wide usage of CG in previous nonlinear placers, there are still several existing problems.

- The major runtime bottleneck of nonlinear placement lies on the line search at line 4, where the cost function is repeatedly evaluated at different points along the search direction. Profile statistics in Section 3.5 show that on the placement of ISPD 2005 ADAPTEC1, line search takes about 63% of the total runtime of global placement and about 50% of the total placement turnaround, respectively. As a result, line search becomes a roadblock to the pursuit of higher placement efficiency.

- At each iteration, the CG method requires the steplength to be at the zero gradient point along the search direction. However, GSS could only locate the local minimal point, while the actual zero gradient point may fall beyond the range of the search interval. As a result, such inaccurate steplength would prevent the CG method from matching its expected performance.

- The objective function of placement is highly nonlinear where the local cost behavior is usually far from a quadratic form. It becomes fairly easy to lose the conjugacy with respect to previous search directions, while the current search direction is repeatedly reset to that of the negative gradient ($\beta_k = 0$ at line 2), degrading the performance of the CG method to that of the gradient descent method.

As line search is usually time consuming and could dominate the efficiency of the entire nonlinear placement [26], there are attempts in literature to use steplength prediction [8] instead. Specifically, as shown by Eq. (12) in [8], steplength is modeled as $\alpha_k = \frac{sw_b}{\|\mathbf{d}_k\|_2}$ where w_b is the bin dimension and $\|\mathbf{d}_k\|_2$ is the Euclidean norm of the search direction vector. s is a constant factor which is tuned between 0.2 and 0.3 to obtain a good tradeoff

between runtime and quality. In this work, we propose a novel and systematic approach to dynamically estimate the steplength, based on the local smoothness of the gradient function. Specifically, we use Nesterov's method as the nonlinear solver and Lipschitz constant prediction to determine the steplength. The optimizer could be beneficial from both convergence rate and solution quality simultaneously. The results in Section 3.5 show that our approach could outperform [26] and [8] by roughly 14% and 10% shorter wirelength and 10× and 1.5× speedup on average of all the ISPD 2005 and ISPD 2006 benchmarks.

3.3.2 Nesterov's Method with Lipschitz Constant Prediction

We propose to use Nesterov's method for nonlinear global placement optimization. Similar to the CG method, Nesterov's method requires only first-order gradient and linear memory cost with respect to the problem size. Nesterov's method targets solving a convex programming problem in Hilbert space H. Unlike most convex programming methods, Nesterov's method constructs a minimizing sequence of points $\{\mathbf{u}_k\}_0^\infty$ which is not relaxational. Algorithm 2 illustrates one iteration of the method on a typical problem $\min\{f(\mathbf{u})|\mathbf{u}\in H\}$ with a non-empty set U^* of minima. Here $\mathbf{u}$ is the solution to the

Algorithm 2. Nesterov-Solver at kth iteration

Require: major solution $\mathbf{u}_k$, reference solution $\mathbf{v}_k$, optimization parameter a_k and objective function $f_k = f(\mathbf{y}_k)$.
Ensure: new solutions $\mathbf{u}_{k+1}$ and $\mathbf{v}_{k+1}$
1: gradient vector $\nabla f_k = \nabla f(\mathbf{v}_k)$
2: steplength $\alpha_k = \arg\max_\alpha \{f_k - f(\mathbf{v}_k - \alpha\nabla f_k) \geq 0.5\alpha\|\nabla f_k\|^2\}$
3: new solution $\mathbf{u}_{k+1} = \mathbf{v}_k - \alpha_k\nabla f_k$
4: parameter update $a_{k+1} = \left(1+\sqrt{4a_k^2+1}\right)/2$
5: new reference solution $\mathbf{v}_{k+1} = \mathbf{u}_{k+1} + (a_k - 1)(\mathbf{u}_{k+1} - \mathbf{u}_k)/a_{k+1}$
6: **return** $\mathbf{u}_{k+1}$

convex programming problem, $\mathbf{v}$ is a reference solution which determines the steplength,

a is an optimization parameter and α is the steplength, respectively. At the beginning ($k = 0$), the method starts from an initial solution $\mathbf{v}_0 \in H$ and sets $a_0 = 1$, $\mathbf{u}_0 = \mathbf{v}_0$ and $\alpha_0 = \frac{\|\mathbf{v_0}-\mathbf{z}\|}{\|\nabla f(\mathbf{v}_0)-\nabla f(\mathbf{z})\|}$, respectively. $\mathbf{z}$ is an arbitrary point in H and $\mathbf{z} \neq \mathbf{v}_0$. All the above vectors and scalars will be iteratively updated. At line 2, the steplength α_k is maximized in order to accelerate the convergence. The new solution $\mathbf{u}_{k+1}$ is updated at line 3 based on the initial reference solution $\mathbf{v}_k$. The new optimization parameter a_{k+1} is updated at line 4, while the new reference solution $\mathbf{v}_{k+1}$ is updated at line 5 based on the solution $\mathbf{u}$ and parameter a.

The convergence rate of Nesterov's method in Algorithm 2 is proved to be $O(1/k^2)$ in [49] where k is the number of iterations. Notice that Nesterov's method [49] is the first one in literature to achieve $O(1/k^2)$ convergence rate, which is proved to be the upper-bound of convergence rate for the first-order optimization methods [48]. The expected convergence rate requires that the steplength α_k satisfies Eq. (3.28) at every single iteration.

$$f(\mathbf{v}_k) - f(\mathbf{v}_k - \alpha_k \nabla f(\mathbf{v}_k)) \geq 0.5\alpha_k \|\nabla f(\mathbf{v}_k)\|^2 \tag{3.28}$$

An upper-bounded error rate of Nesterov's method is shown in Eq. (3.29).

Theorem 1. *Suppose $f(\mathbf{u})$ is a convex function in $C^{1,1}(H)$ and $U^* \neq \emptyset$, where $C^{1,1}(H)$ means that the gradient function $\nabla f(\mathbf{u})$ is of Lipschitz continuity. We have $\mathbf{u}^* \in U^*$ and L is the Lipschitz constant of the gradient function $\nabla f(\mathbf{u})$. The following assertion is true based on the solution $\mathbf{u}_k$ output by Algorithm 2.*

$$f(\mathbf{u}_k) - f(\mathbf{u}^*) \leq \frac{4L\|\mathbf{v}_0 - \mathbf{u}^*\|^2}{(k+2)^2} \tag{3.29}$$

Here we define the Lipschitz constant L of the gradient function ∇f as follows.

Definition 1. *Given $f \in C^{1,1}(H)$, L is the Lipschitz constant of ∇f, if $\forall \mathbf{u}, \mathbf{v} \in H$ we have*

$$\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\|. \tag{3.30}$$

$\nabla f(\mathbf{u})$ is thus of Lipschitz continuity. The inequality in Eq. (3.28) must be iteratively satisfied to achieve $O\left(1/k^2\right)$ convergence rate. Similar to line search in CG method, [49] uses bisection search to determine the maximum steplength. At each iteration, the objective function would be evaluated for $O(\log L)$ times, which increases the complexity to $O(n \log n \log L)$. Instead of line search, we use steplength prediction to accelerate our placement algorithm. As discussed in [49], if the Lipschitz constant of the gradient function is known, we can set the steplength as the inverse of Lipschitz constant to satisfy Eq. (3.28) without convergence overhead. However, to estimate the exact Lipschitz constant for the objective function of global placement is difficult due to the following issues.

- The objective function is non-convex due to the energy (density) function, thus the requirement for Theorem 1 is not satisfied.
- The wirelength function is iteratively changed due to the dynamically adjusted smoothing coefficient (γ in Eq. (2.6)).
- The penalty factor (λ in Eq. (3.5)) on the energy (density) function is iteratively changed for the runtime force balancing between wirelength and density.

As a result, we propose a method to dynamically approximate the Lipschitz constant $\widetilde{L}_k$. Based on Eq. (3.30), we select $\mathbf{u}$ to be the current reference solution ($\mathbf{v}_k$) and $\mathbf{v}$ to the reference solution at the last iteration ($\mathbf{v}_{k-1}$). The Lipschitz constant for $\nabla f(\mathbf{v}_k)$ is

approximated as follows

$$\widetilde{L}_k = \frac{\|\nabla f(\mathbf{v}_k) - \nabla f(\mathbf{v}_{k-1})\|}{\|\mathbf{v}_k - \mathbf{v}_{k-1}\|}. \tag{3.31}$$

Our approximation method is effective and efficient because

- There is no additional computation cost introduced as both $\nabla f(\mathbf{v}_k)$ and $\nabla f(\mathbf{v}_{k-1})$ are known.

- The two solutions $\mathbf{v}_k$ and $\mathbf{v}_{k-1}$ are supposed to be close to each other. Therefore $\|\mathbf{v}_k - \mathbf{v}_{k-1}\|$ is relatively small compared to $\|\mathbf{u} - \mathbf{v}\|$ by randomly selecting $\mathbf{u}$ and $\mathbf{v}$. This prevents underestimation of $\widetilde{L}_k$ thus overestimation of the steplength α_k.

The results in Section 3.5 show that our placement algorithm using Nesterov's method with Lipschitz constant prediction could simultaneously improve the runtime and wirelength by 2.21× and 2.28% on average of all the ISPD 2005 benchmarks, compared to that by CG method together with line search.

3.3.3 Preconditioning

Preconditioning reduces the condition number of a problem, which is transformed to be more suitable for numerical solution. Traditional preconditioning techniques compute and inverse the Hessian matrix ($\mathbf{H_f}$) of the objective function (f). Preconditioning has very wide applications in quadratic placers [27, 30, 41, 60, 61] but zero attempts in nonlinear placers [6, 8, 26], because the density function is not convex. A preconditioned gradient vector $\nabla f_{pre} = \mathbf{H_f}^{-1} \nabla f$ can smooth the numerical optimization to converge in fewer iterations. Nevertheless, the objective function of global placement is highly nonlinear and iteratively changed. Moreover, the problem instance is usually of millions of objects, where the complexity of $O(n^2)$ makes the iterative computation

of Hessian matrix fairly expensive and indeed impractical. As a result, we select Jacobi preconditioner with only diagonal terms of the Hessian matrix being used as Eq. (3.32) shows.

$$\mathbf{H}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \approx \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 f}{\partial x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} = \widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} \quad (3.32)$$

We have similar definition on $\widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{y},\mathbf{y}}}$ and can construct $\widetilde{\mathbf{H}}_{\mathbf{f}}$ based on them. By Eq. (3.5) we have $\frac{\partial^2 f(\mathbf{v})}{\partial x_i^2} = \frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} + \lambda \frac{\partial^2 N(\mathbf{v})}{\partial x_i^2}$, and we concisely approximate $\frac{\partial^2 W(\mathbf{v})}{\partial x_i^2}$ and $\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2}$ to ensure functionality of the preconditioner. Differentiating the wirelength function in Eq. (2.6) by two orders is computationally expensive and we use the vertex degree of object i instead,

$$\frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} = \sum_{e \in E_i} \frac{\partial^2 W_e(\mathbf{v})}{\partial x_i^2} \Rightarrow |E_i|, \quad (3.33)$$

where E_i denotes the net subset incident to the object i. The non-convexity of the density function in Eq. (3.4) disables the traditional preconditioner to achieve the expected performance. Eq. (3.34) shows its two-order differentiation

$$\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} = q_i \frac{\partial^2 \psi_i(\mathbf{v})}{\partial x_i^2} = q_i \frac{-\partial \xi_{i_x}(\mathbf{v})}{\partial x_i} \Rightarrow q_i. \quad (3.34)$$

As a result, we use the linear term q_i as the density preconditioner and the Hessian

matrix is approximated as below

$$\widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} = \begin{pmatrix} |E_1| + \lambda q_1 & 0 & \cdots & 0 \\ 0 & |E_2| + \lambda q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |E_n| + \lambda q_n \end{pmatrix}. \tag{3.35}$$

Therefore, we have the preconditioned gradient as $\nabla f_{pre} = \widetilde{\mathbf{H}}_{\mathbf{f}}^{-1} \nabla f$. In Section 3.5 it shows that our preconditioner could improve the wirelength by 2.42% with essentially the same runtime on average of all the ISPD 2005 benchmarks.

3.4 Global Placement Algorithm

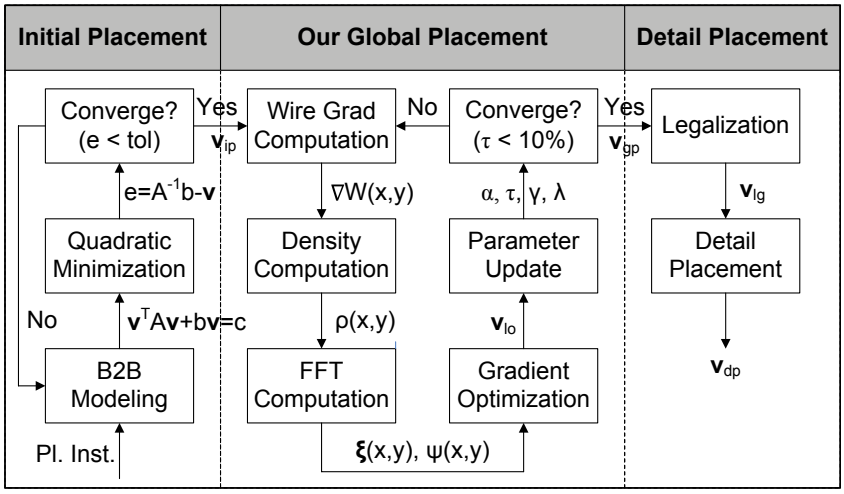


Figure 3.8. The entire flow of ePlace, including initial quadratic wirelength minimization, our novel global placement algorithm, and detailed placement with legal solution generated.

The entire flow of ePlace is shown in Figure 3.8, where our algorithm accounts for the middle stage of global placement. The global placement is based on the input solution $\mathbf{v}_{ip}$ from the initial placement stage, where the quadratic wirelength is minimized using the bound-to-bound (B2B) net model [57]. A linear CG solver is used with

Jacobi preconditioning for acceleration [41]. After global placement completes, all the fillers are removed from the solution $\mathbf{v}_{gp}$, which is then legalized and discretely optimized using the detailed placer from [8]. As discussed in Section 3.3, both CG method and Nesterov's method are used to solve the unconstrained optimization problem in Eq. (3.5). A self-adaptive parameter adjustment method (introduced in Section 3.4.1) is incorporated to improve the quality and convergence rate. Finally, we discuss the global placement algorithm in Section 3.4.2.

3.4.1 Self-Adaptive Parameter Adjustment

Grid dimension: ePlace uses fixed grid dimension throughout the entire global placement. There is naturally a trade-off between granularity and efficiency. Coarser grid induces higher efficiency but lower accuracy, vice versa. From experiments we observe that coarser grid causes additional problems. For instance, more cells are undertaking the same density force. These cells clot together and motion in the same trace. This induces density oscillation between adjacent regions and impedes cell spreading. In our approach, we determine the grid dimension based on the number of cells in the netlist and inserted fillers. As the FFT package from [50] requires that the grid dimension m to be a power of 2, we set $m = \lceil \log_2 \sqrt{n'} \rceil$ and upper-bound m by 1024 due to efficiency concerns.

Steplength: As discussed in Section 3.3, in Nesterov's method the steplength is determined by the inverse of the approximated Lipschitz constant as shown in Eq. (3.31). In CG method, the steplength is determined by line search which locates the local minimal cost along the conjugate search direction within a interval. The length of the search interval α_k^{max} is dynamically adjusted as follows. The initial value is determined as linearly proportional to the bin dimension, specifically, $\alpha_0^{max} = \kappa w_b$, where w_b is the grid width. In practice, we set $\kappa = 0.044$ to achieve the best placement quality. α_k^{max} is iteratively

updated based on the optimal steplength α_{k-1} as Eq. (3.36) shows.

$$\alpha_k^{max} = \max(\alpha_0^{max}, 2\alpha_{k-1}),\ \alpha_k^{min} = 0.01\alpha_k^{max} \tag{3.36}$$

α_k is the steplength for the kth iteration generated by GSS, as line 4 of Algorithm 1 shows. Notice that in practice, α_k may not be the exact local optimal, as line search will stop when the interval reduces to α_k^{min}. Moreover, if $f(x)$ is a multi-modal function within the search interval, GSS may perform like a "random perturbation" and even increase the cost under pathological conditions. Despite its sub-optimality, such occasional "random perturbation" will be actually useful. Solutions could escape from local optimum with uphill climbing actions due to GSS. As a result, GSS remains an effective and efficient line search option.

Penalty factor: In our approach, we set the initial value of the penalty factor λ_0 by Eq. (3.37), in order to balance the forces of wirelength and density. This method is also used in [8, 26]. Here $W_{x_i} = \frac{\partial W}{\partial x_i}$ and $W_{y_i} = \frac{\partial W}{\partial y_i}$, while ξ_{x_i} and ξ_{y_i} denote the horizontal and vertical electric field at node i, respectively.

$$\lambda_0 = \frac{\sum_{i \in V_m'} \left(|W_{i_x}| + \left|W_{i_y}\right| \right)}{\sum_{i \in V_m'} q_i \left(|\xi_{i_x}| + \left|\xi_{i_y}\right| \right)}. \tag{3.37}$$

Traditional approaches usually multiply the penalty factor λ by a constant number (2.0 in [8, 26]), when the optimization converges locally. However, as wirelength and density are changed at every iteration, the penalty factor should be updated immediately in order to remain adaptive. In our approach, we iteratively update the penalty factor by setting $\lambda_k = \mu_k \lambda_{k-1}$. The multiplier μ_k is based on the iterative HPWL variation $\Delta HPWL_k = HPWL(\mathbf{v}_k) - HPWL(\mathbf{v}_{k-1})$ as Eq. (3.38) shows

$$\mu_k = \mu_0^{-\frac{\Delta HPWL_k}{\Delta HPWL_{ref}} + 1.0}, \tag{3.38}$$

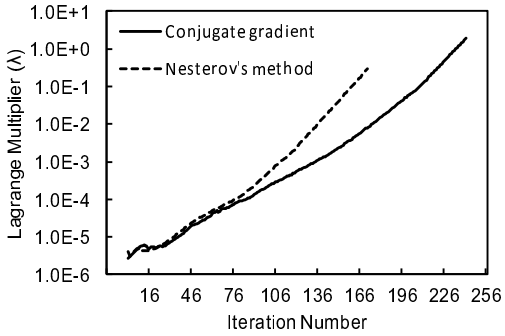


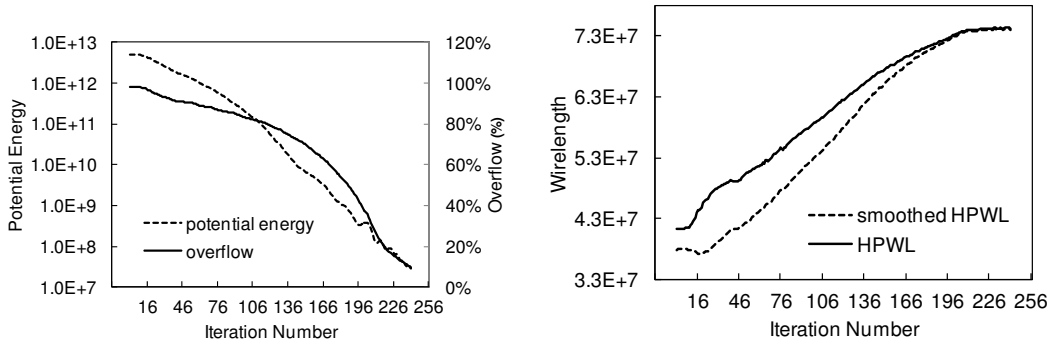
Figure 3.9. The illustration of the iterative variation of penalty factor using CG method with line search and Nesterov's method with Lipschitz constant approximation. The penalty factor increases almost monotonically under the nonlinear optimization by both methods. The placement is conducted on the ISPD 2005 ADAPTEC1 benchmark, where Nesterov's method consumes fewer iterations than CG method.

where μ_0 is a pre-determined fixed number and $\Delta HPWL_{ref}$ is the expected wirelength increase per iteration. In practice, we set $\mu_0 = 1.1$ and $\Delta HPWL_{ref} = 3.5 \times 10^5$ for best quality. The multiplier μ_k is upper- and lower-bounded by 1.1 and 0.75 in order to damp out the transient noise during the optimization flow. The experimental results show that the penalty factor iteratively increases under the nonlinear optimization of both CG method and Nesterov's method as illustrated in Figure 3.9.

Density overflow: Global placement usually terminates when the overlap is sufficiently small. The remaining work is handled by the downstream legalizer and detail placer. Similar to NTUPlace3 [8] and mPL6 [11], we use the density overflow τ defined in Eq. (3.39) as the stopping criterion.

$$\tau = \frac{\sum_{b \in B} \max(\rho'_b - \rho_t, 0) A_b}{\sum_{i \in V_m} A_i}. \tag{3.39}$$

Here A_b is the area of grid b, while A_i is the area of movable cell i. ρ'_b denotes the density of grid b due to only movable cells. The global placer terminates when the overflow τ is less than τ_{min}. The experimental results show that the total potential energy N is well correlated with the density overflow τ as illustrated in Figure 3.10(a). The potential



(a) Total overflow ratio τ and potential energy N. **(b)** $HPWL$ and the smoothed wirelength cost W.

Figure 3.10. The overflow decreases in a linear rate while the potential energy decreases in an exponential rate. The smoothed wirelength cost approximates HPWL better when the density overflow approaches the lower limit (τ_{min}). The global placement uses CG method and is conducted on the ISPD 2005 ADAPTEC1 benchmark.

energy decreases exponentially while the density overflow decreases linearly.

Wirelength coefficient: In our approach, we use the WA model [23] in Eq. (2.6) to smooth the wirelength function. WA outperforms the traditional LSE model with about $2\times$ accuracy. The experiments show that quality and convergence are sensitive to the smoothing parameter γ. Our approach relaxes the smoothing parameter at early iterations, such that more cells are encouraged to be globally moved out of the high-density regions. At later stages, when local movement dominates, the parameter is reduced to make the smoothed wirelength W approach $HPWL$. Meanwhile, the density of a smaller grid is more sensitive towards cell movement, vice versa. Therefore, we set the smoothing parameter γ to be the function of both the density overflow τ and the grid size w_b. By reducing the smoothing parameter, we only enable the motion of HPWL-insensitive cells which are locally shifted to resolve the remaining overlap. Here for HPWL-insensitive cells we are referring to those cells whose movement will not change the HPWL of their incident nets, i.e., cells locate relatively far away to the boundaries of net bounding box. At later iterations, we only expect minor perturbation to the placement layout, such that solution will converge smoothly. As a result, we determine to

enhance the accuracy of wirelength modeling, the respective wirelength force becomes stronger to allow only small-scale cell movement thus minor layout perturbation. The iterative correlation of the smoothed wirelength to the HPWL is shown in Figure 3.10(b), where the smoothed wirelength converges to HPWL in the end. Our empirical studies show that modeling γ as a linear function of bin dimension w_b yet an exponential function of density overflow τ achieves the best quality. As the density overflow usually starts from around 100% and end with 10% (our stopping criterion), we set $\gamma(\tau = 1.0) = 80w_b$ and $\gamma(\tau = 0.1) = 0.8w_b$ by empirical tuning. The function of the smoothing parameter γ in terms of density overflow τ is then modeled as

$$\gamma(\tau) = 8.0w_b \times 10^{k\tau+b}. \tag{3.40}$$

Based on the value of $\gamma(1.0)$ and $\gamma(0.1)$ as mentioned above, it is easy to derive that $k = \frac{20}{9}$ and $b = -\frac{11}{9}$, respectively.

3.4.2 Global Placement

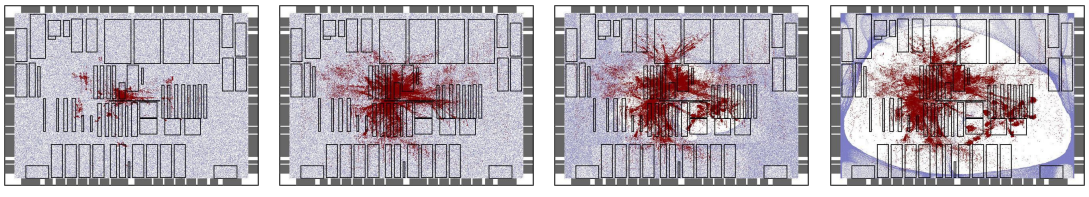
The detail flow of our global placement method ePlace is shown in Algorithm 3. The objective function f_k is formulated at line 5. The wirelength gradient ∇W_k and density distribution $\rho_k(x,y)$ are computed at line 6. The FFT library [50] is invoked at line 7 to generate the distribution of field $\boldsymbol{\xi}_k(x,y)$ and potential $\psi_k(x,y)$. The density (energy) gradient ∇N_k is computed at line 8, while the total gradient ∇f_k is computed at line 9. The nonlinear solver (NL-Solver) is invoked at line 10 with current solution $\mathbf{v}_k$. The solution $\mathbf{v}_{k+1}$ for the next iteration is output by the nonlinear solver and used to update the parameters at line 11. The stopping criterion is evaluated at line 12 to determine whether the solution converges or not. Finally, the global placement solution $\mathbf{v}_{gp}$ is output to the legalizer and detailed placer at line 17.

Algorithm 3. ePlace

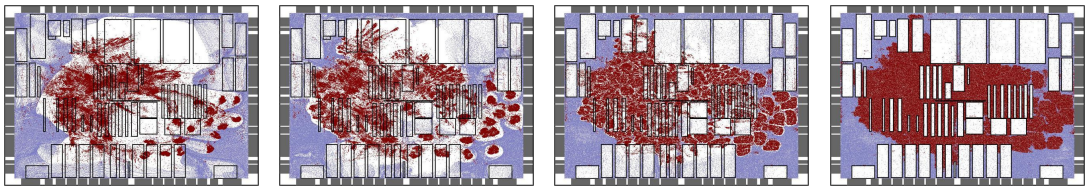
Require: initial placement solution $\mathbf{v}_0 = \mathbf{v}_{ip}$
uniform chip decomposition into $m \times m$ grid
minimum overflow τ_{min}
maximum iterations $k_{max} = 3000$

Ensure: global placement solution $\mathbf{v}_{gp}$

1: $m \times m$ decomposition over R
2: initialize λ_0 by Eq. (3.37)
3: initialize $\alpha_0^{max} = 0.044 w_b$
4: **for** $k = 1 \rightarrow k_{max}$ **do**
5: $f_k = f(\mathbf{v}_k) = W(\mathbf{v}_k) + \lambda_k N(\mathbf{v}_k)$
6: compute wirelength gradient ∇W_k and density ρ_k
7: (ψ_k, ξ_k) =FFT-Solver(ρ_k)
8: compute energy (density) gradient $\nabla N_k = \mathbf{q}\boldsymbol{\xi}_k$
9: $\nabla f_k = \nabla W_k + \lambda_k \nabla N_k$
10: $\mathbf{v}_{k+1}$ =NL-Solver$\left(\mathbf{v}_k, f_k, \nabla f_k, \alpha_k^{max}, 0.01\alpha_k^{max}\right)$
11: update α_{k+1}^{max}, λ_{k+1}, τ_{k+1}, γ_{k+1} by Eq. (3.36), (3.38), (3.39), (3.40)
12: **if** $\tau_{k+1} \leq \tau_{min}$ **then**
13: $\mathbf{v}_{gp} = \mathbf{v}_{k+1}$
14: break
15: **end if**
16: **end for**
17: **return** $\mathbf{v}_{gp}$



(a) Iter=1, HPWL=4.27e7, τ = 96.2%, W=2.10e7, N=1.25e12.
(b) Iter=50, HPWL=6.54e7, τ = 81.5%, W=1.77e7, N=5.30e11.
(c) Iter=150, HPWL=6.89e7, τ = 78.4%, W=2.00e7, N=2.13e11.
(d) Iter=175, HPWL=7.25e7, τ = 72.2%, W=2.62e7, N=7.18e10.



(e) Iter=200, HPWL=7.56e7, τ = 63.6%, W=3.59e7, N=1.74e10.
(f) Iter=225, HPWL=7.76e7, τ = 51.8%, W=4.90e7, N=1.94e9.
(g) Iter=250, HPWL=7.64e7, τ = 33.1%, W=6.20e7, N=1.46e8.
(h) Iter=286, HPWL=7.42e7, τ = 9.95%, W=6.95e7, N=2.01e6.

Figure 3.11. Snapshots of cell and filler distribution during global placement progression. Standard cells, macros and fillers are shown by red points, black rectangles and blue points, respectively. The placement is conducted on the ISPD 2005 ADAPTEC1 benchmark by ePlace using Nesterov's method with preconditioning.

We illustrate the process of global placement in Figure 3.11 using snapshots of cell and filler distribution extracted from eight intermediate iterations. Nesterov's method is used for the nonlinear optimization with dynamic prediction of Lipschitz constant to determine the steplength. The initial placement solution $\mathbf{v}_{ip}$ is shown in Figure 3.11(a), where standard cells are placed at the central region while filler cells are randomly distributed over the entire placement region R. At later iterations, standard cells are spreading away from over-filled regions, the density force pushes the disconnected fillers towards the boundary of the placement region. In the end, all the standard cells converge to a stable location with acceptable system energy (density penalty) and wirelength overhead.

3.5 Experiments and Results

We implement our algorithm using C programming language and execute the program in single-thread mode on a Linux machine with Intel i7 920 2.67GHz CPU and 12GB memory. In our experiments, we use the benchmark suites from [46] and [45], which are published in the ISPD 2005 and ISPD 2006 placement contests, respectively. As denoted in [45, 46], the benchmark circuits preserve the physical structure of real ASIC designs. We also use the evaluation policies and scripts in [46] and [45], as they have become common criteria and are widely admitted in modern placement works, to rank the performance of different placers in our experiments. Besides, we set the minimum density overflow $\tau_{min} = 10\%$ as the stopping criterion of global placement for all the benchmarks. We apply the same setting of parameters to all the testcases, in other words, there is no parameter tuning towards specific benchmarks.

Global placement plays the dominant role on the overall placement solution quality. However, global placement result is illegal and it is relatively hard to tell how much wirelength penalty will be introduced when legalizing it. There are placers in literature, e.g. SimPL [41] and ComPLx [27], which consists of only global placement algorithm development, and they invoke detailed placement engine from other works to legalize and discretely optimize their solutions. As a result, we follow the custom in literature to conduct performance comparison between legalized solutions. Specifically, we use the detailed placer in [8] to perform legalization and detailed placement on our global placement solution.

We compare the performance of our work with ten cutting-edge placers of different categories: Capo10.5 [52] (min-cut), FastPlace3.0 [61], RQL [60], MAPLE [28], ComPLx (v13.07.30) [27], BonnPlace [58], POLAR [30], (quadratic), APlace3 [26], NTUPlace3 [8] and mPL6 [6] (nonlinear). We have applied and obtained the source

Table 3.1. Circuit statistics of the ISPD 2005 placement benchmark suite [46].

Circuits	# Objects	# Standard Cells	# Movable Macros	# Fixed Macros	# Nets	Density (%)	Util. (%)	Density Bound (%)
ADAPTEC1	211447	210904	0	543	221142	75.71	57.34	100
ADAPTEC2	255023	254457	0	566	266009	78.59	44.32	100
ADAPTEC3	451650	450927	0	723	466758	74.58	33.68	100
ADAPTEC4	496045	494716	0	1329	515951	62.71	27.18	100
BIGBLUE1	278164	277604	0	560	284479	54.19	44.67	100
BIGBLUE2	557866	534782	0	23084	577235	61.88	37.90	100
BIGBLUE3	1096812	1093034	2485	1293	1123170	85.52	56.23	100
BIGBLUE4	2177353	2169183	0	8170	2229886	65.14	44.06	100

code or binaries from seven of the above ten placers, each of them is compiled and executed in our local machine. The executable of RQL, MAPLE and BonnPlace are not available due to their industrial use and other issues. As a result, their solution quality and runtime results are cited from the according publications [28, 58, 60]. The performance of Capo10.5 and APlace3 on the ISPD 2006 benchmark suite is obtained from the respective contest result [45].

3.5.1 Results on ISPD 2005 Benchmark Suite

The circuit statistics of ISPD 2005 benchmark suite are shown in Table 3.1. Notice that the design scale is up to of two million cells, which well represents the modern IC design complexity. As there is no specific density constraint, the density upper-bound in Table 3.1 is set as 100% for every benchmark. Notice that one out of the totally eight circuits (BIGBLUE3) has movable macros, of which the physical dimension and logic effort differ quite a lot from standard cells. Such objects further challenge the existing placers to provide stable performance under different circuit characteristics.

All the experimental results are shown in Table 3.2 and Table 3.3 with HPWL in $\times 10^6$ and CPU in minutes. The experiments are executed in the single-thread mode (except for POLAR, which consumes up to four CPUs simultaneously) with the solution quality evaluated using the official scripts from [46]. For our placement framework ePlace, we include three different configurations to study its performance in detail.

- **CG**: ePlace using conjugate gradient method for nonlinear optimization and line search for steplength determination.

- **Nes**: ePlace using Nesterov's method for nonlinear optimization and Lipschitz constant prediction for steplength determination.

- **Nes-Pre**: ePlace using Nesterov's method for nonlinear optimization, Lipschitz constant prediction for steplength determination and preconditioning for search space reshaping.

It is relatively difficult for ePlace-CG and ePlace-Nes to handle large macros (e.g., BIGBLUE3) as density force is linearly proportional to the object area by Eq. (3.6), while movable macros significantly differ from standard cells with much higher magnitude of gradient. As a result, unpreconditioned gradient makes macros with large area and high incidence degree bounce between opposite placement boundaries, causing the solution to oscillate and hard to converge within limited number of iterations[2]. To prevent divergence of nonlinear placement optimization, in ePlace-CG and ePlace-Nes, we disable the movement of objects with area larger than 500$\times$ of the average objects area. By preconditioning, we relieve the imbalance between object gradient and make the search space more spherical, thus all the objects are allowed to move in ePlace-Nes-Pre. Among all the above three options, ePlace-CG has the worst solution quality and placement efficiency, where ePlace-Nes could outperform it by roughly 2.28% shorter wirelength and 2.21$\times$ speedup on average. Using preconditioning could further reduce the wirelength by 2.42%, while the runtime is not increased. By default, we use Nesterov's method together with gradient preconditioning in ePlace.

Compared to the performance of all the ten placers from our local experiments or according publications as shown in Table 3.2, ePlace-Nes-Pre generates the best place-

[2]In ePlace, we set 3000 as the upper limit of iterations.

Table 3.2. HPWL ($\times 10^6$) on the ISPD 2005 benchmark suite [46]. CP=Capo, FP=FastPlace, MPE=MAPLE, CPx=ComPLx, BPL=BonnPlace, AP=APlace, NP=NTUPlace. Cited results are marked with *. HPWL and legality of all the solutions are evaluated by the official scripts [46]. Average results are normalized to ePlace-Nes-Pre.

Categories	Min-Cut	Quadratic						Nonlinear			ePlace (nonlinear)		
Benchmarks	CP10.5	FP3	RQL*	MPE*	CPx	BPL*	POLAR	AP3	NP3	mPL6	CG	Nes	Nes-Pre
ADAPTEC1	87.80	78.34	77.82	76.36	77.73	76.87	77.21	78.35	80.29	77.93	76.46	74.66	**74.63**
ADAPTEC2	102.66	93.47	88.51	86.95	88.84	86.36	86.16	95.70	90.18	92.04	85.57	89.00	**84.84**
ADAPTEC3	234.27	213.48	210.96	209.78	203.45	202.00	201.30	218.52	233.77	214.16	202.16	201.76	**194.57**
ADAPTEC4	204.33	196.88	188.86	179.91	183.16	181.53	182.37	209.28	215.02	193.89	185.83	183.43	**179.02**
BIGBLUE1	106.58	96.23	94.98	93.74	94.41	94.85	94.67	100.02	98.65	96.80	91.64	91.05	**90.99**
BIGBLUE2	161.68	154.89	150.03	144.55	145.33	144.21	143.85	153.75	158.27	152.34	145.54	143.31	**141.83**
BIGBLUE3	403.36	369.19	323.09	323.05	337.66	317.17	324.53	411.59	346.33	344.25	359.00	326.77	**308.77**
BIGBLUE4	945.77	834.04	797.66	775.71	788.33	781.79	781.06	871.29	829.09	829.44	805.90	763.14	**753.20**
Average	21.14%	10.00%	5.40%	3.21%	4.50%	2.83%	3.08%	14.33%	12.05%	8.33%	4.70%	2.42%	0.00%

ment solutions with the shortest total wirelength in all the eight benchmarks. On average, ePlace-Nes-Pre improves the total wirelength by 21.14%, 10.00%, 5.40%, 3.21%, 4.50%, 2.83%, 3.08%, 14.33%, 12.05%, and 8.33% over Capo10.5, FastPlace3.0, RQL, MAPLE, ComPLx, BonnPlace, POLAR, APlace3, NTUPlace3 and mPL6, respectively.

ePlace is faster than all the previous nonlinear placers. Specifically, ePlace-Nes-Pre outperforms APlace3, NTUPlace3 and mPL6 with 9.13×, 1.40× and 3.78× speedup, even if they are using multi-level clustering for problem simplification while we are conducting placement on the original flat netlist. At the coarsest level, a hierarchical placer will usually place about only 1000 clusters, which is 0.1% of that of the original netlist. However, despite zero netlist coarsening, our placer runs faster than the previous multi-level works. Such performance validates the efficiency of our placement algorithm. ePlace-Nes-Pre is slower than some of the previous quadratic placement approaches. This is mainly because all the computation intensive steps in nonlinear optimization are not included in quadratic placers. For instance, in nonlinear placement the objective cost and gradient function are both of very high order, they consume most portion of the runtime at each iteration. However, in quadratic placement, these two functions are of only second and first orders, of which numerical solution can be computed much faster. Specifically, ePlace-Nes-Pre runs 0.53×, 0.91× and 0.52× slower than FastPlace3.0, RQL and ComPLx, while the respective wirelength improvement is 10.00%, 5.40% and 4.50%. MAPLE, BonnPlace and POLAR have the best published results on the ISPD 2005 benchmark suite in literature. As Table 3.3 shows, the average runtime of MAPLE, BonnPlace and POLAR is 2.84×, 3.05× and 0.52× that of our placer ePlace-Nes-Pre, respectively, while our wirelength improvement over these three placers are 3.21%, 2.83% and 3.08%, respectively.

As Table 3.2 and Table 3.3 shows, on average of all the ISPD 2005 benchmarks, preconditioning produces 2.42% shorter wirelength and consumes essentially the same

Table 3.3. Runtime (minutes) on the ISPD 2005 benchmark suite [46]. CP=Capo, FP=FastPlace, MPE=MAPLE, CPx=ComPLx, AP=APlace, NP=NTUPlace. Cited results are marked with *. Average results are normalized to ePlace-Nes-Pre.

Categories	Min-Cut	Quadratic						Nonlinear			ePlace (nonlinear)		
Benchmarks	CP10.5	FP3	RQL*	MPE*	CPx	BPL*	POLAR	AP3	NP3	mPL6	CG	Nes	Nes-Pre
ADAPTEC1	48.33	2.92	5.19	18.46	2.97	18.28	2.78	48.88	7.17	23.27	9.17	4.65	4.28
ADAPTEC2	61.63	4.13	7.71	25.26	3.90	26.40	4.67	68.07	8.22	24.75	12.67	7.48	4.90
ADAPTEC3	133.43	9.53	16.80	61.25	8.62	48.50	7.72	186.67	18.53	73.97	45.40	20.77	13.73
ADAPTEC4	141.85	8.75	14.57	58.68	7.38	41.12	7.83	209.60	23.53	71.03	34.33	14.66	15.00
BIGBLUE1	77.90	4.57	7.27	27.47	4.93	26.22	3.72	64.05	14.30	30.05	23.63	5.38	6.25
BIGBLUE2	150.15	8.00	12.82	54.05	7.58	43.13	7.48	136.43	35.10	79.00	30.83	7.31	10.50
BIGBLUE3	373.87	21.05	31.13	117.00	20.75	114.68	20.95	289.78	38.77	104.63	107.75	32.69	28.68
BIGBLUE4	779.22	40.13	78.54	303.93	40.18	258.68	50.87	730.42	106.08	238.82	165.00	43.08	63.70
Average	8.94×	0.53×	0.91×	2.84×	0.52×	3.05×	0.52×	9.13×	1.40×	3.78×	2.21×	0.99×	1.00×

Table 3.4. Circuit statistics of the ISPD 2006 placement benchmark suite [45].

Circuits	# Objects	# Standard Cells	# Movable Macros	# Fixed Macros	# Nets	Density (%)	Util. (%)	Density Bound (%)
ADAPTEC5	843128	842482	0	646	867798	78.62	49.85	50
NEWBLUE1	330474	330037	64	337	338901	70.69	70.69	80
NEWBLUE2	441516	436516	3723	1277	465219	86.15	61.66	90
NEWBLUE3	494011	482833	0	11178	552199	84.71	26.33	80
NEWBLUE4	646139	642717	0	3422	637051	65.82	46.47	50
NEWBLUE5	1233058	1228177	0	4881	1284251	74.43	49.26	50
NEWBLUE6	1255039	1248150	0	6889	1288443	59.26	38.70	80
NEWBLUE7	2507954	2481372	0	26582	2636820	76.36	49.06	80

runtime compared to the original placement. There are some special testcase, such as BIGBLUE3 of ISPD05, which causes our placer fail to converge without preconditioning, i.e., the runtime would approach infinity. As discussed before, we disable the movement of objects with size above certain threshold to enforce the convergence.

3.5.2 Results on ISPD 2006 Benchmark Suite

The circuit statistics of the ISPD 2006 benchmark suite [45] are shown in Table 3.4. Notice that BonnPlace [58] is specifically designed for the ISPD 2005 benchmark suite, while its binary or results on the ISPD 2006 benchmarks are not available. As a result, we do not include it in the experiments. Similar to ISPD 2005, the design scale is up to of 2.5 million objects and this represents the complexity of modern ASIC design. In contrast to ISPD 2005, there is a benchmark-specific density constraint. Violation of such constraint in placement solutions (i.e., exceeding the density upper-bound) would induce penalty on the total wirelength. Here two out of the totally eight circuits (NEWBLUE1 and NEWBLUE2) have movable macros which challenge the placement performance stability across different object dimensions.

All the experimental results are shown in Table 3.5 and Table 3.6 with the scaled HPWL (sHPWL) in $\times 10^6$ and CPU in minutes. Here we just include ePlace-Nes-Pre (denoted as ePlace) in this experiment. Following the contest protocol, we define the

scaled wirelength as

$$sHPWL = HPWL \times (1 + 0.01 \times \tau_s). \tag{3.41}$$

Here τ_s is the density penalty on the wirelength, it denotes the scaled density overflow per bin as defined below

$$\tau_s = \left(\frac{\tau_{tot} A_{b'} \rho_t}{400 \sum_{i \in V_m} A_i} \right)^2. \tag{3.42}$$

Here $A_{b'}$ is the area of each uniform bin b', which is defined by the contest organizer [45] with both width and height equal to ten times the placement row height of each benchmark. A_i is the area of each movable object i. τ_{tot} is the total density overflow amount, which is defined as

$$\tau_{tot} = \sum_{b' \in B'} \max(\rho'_{b'} - \rho_t, 0) A_{b'} \tag{3.43}$$

using the contest specified bin structure B' as mentioned above.

Compared to the quality of all the ten placers as shown in Table 3.5, ePlace generates the best placement solution (with the shortest scaled wirelength) in seven out of the totally eight benchmarks. On average, our placer improves the total wirelength by 43.73%, 16.25%, 7.99%, 4.59%, 4.86%, 7.16%, 18.38%, 7.74% and 10.11% over Capo, FastPlace3.0, RQL, MAPLE, ComPLx, POLAR, APlace3, NTUPlace3 and mPL6, respectively.

The runtime of all the placers on the ISPD 2006 benchmark suite is shown in Table 3.6. Notice that the runtime of RQL and MAPLE is not available as the authors did not release them in the respective publications [28, 60]. Compared to all the three prior nonlinear placers, ePlace improves the efficiency by up to 10.21×. As discussed before, nonlinear placers lag behind quadratic placers in efficiency due to the computation of high-order gradient functions. However, such gap is largely reduced by ePlace, i.e., on

Table 3.5. Scaled HPWL (sHPWL) ($\times 10^6$) and original HPWL (in parenthesis) on the ISPD 2006 benchmark suite [45]. CP=Capo, FP=FastPlace, MPE=MAPLE, CPx=ComPLx, AP=APlace, NP=NTUPlace. Cited results are marked with *. The scaled HPWL, legality and density penalty of the solutions are all evaluated by the official scripts [45]. Average results are normalized to ePlace.

Categories	Min-Cut	Quadratic					Nonlinear			
Benchmarks	CP10.5*	FP3	RQL*	MPE*	CPx	POLAR	AP3*	NP3	mPL6	ePlace
ADAPTEC5	494.64 (491.60)	472.72 (437.01)	443.28 (405.73)	407.33 (N/A)	415.77 (407.26)	438.47 (389.82)	520.97 (449.61)	444.41 (345.82)	428.31 (423.96)	**397.53** (394.71)
NEWBLUE1	98.48 (98.35)	74.11 (73.34)	64.43 (64.21)	69.25 (N/A)	64.75 (64.10)	67.52 (66.11)	73.31 (73.26)	**61.01** (60.58)	72.62 (66.61)	62.31 (62.13)
NEWBLUE2	309.53 (308.64)	206.04 (204.00)	199.60 (196.74)	191.66 (N/A)	193.06 (191.07)	191.25 (187.81)	198.24 (197.42)	194.24 (191.31)	201.91 (199.05)	**182.69** (181.38)
NEWBLUE3	361.25 (361.21)	297.45 (295.83)	269.33 (269.13)	268.07 (N/A)	273.42 (270.91)	271.28 (267.64)	273.64 (273.63)	275.08 (274.94)	285.26 (283.40)	**266.80** (266.61)
NEWBLUE4	362.40 (358.28)	308.33 (295.86)	308.75 (268.07)	282.49 (N/A)	292.82 (288.65)	305.14 (273.96)	384.12 (377.55)	296.62 (260.98)	298.20 (293.22)	**276.13** (272.86)
NEWBLUE5	659.57 (657.40)	621.47 (579.67)	537.49 (473.14)	515.04 (N/A)	507.74 (498.94)	521.85 (462.18)	613.86 (545.90)	537.92 (446.89)	535.80 (528.02)	**492.62** (489.55)
NEWBLUE6	668.66 (668.33)	549.89 (544.31)	515.69 (494.30)	494.82 (N/A)	501.05 (495.39)	512.06 (483.99)	522.73 (522.58)	534.96 (533.45)	523.47 (516.21)	**464.44** (462.60)
NEWBLUE7	1518.75 (1518.49)	1105.58 (1091.20)	1057.80 (1031.33)	1032.60 (N/A)	1041.21 (1026.80)	1045.20 (998.95)	1098.90 (1098.26)	1096.16 (1074.54)	1085.68 (1072.86)	**989.96** (987.45)
Average sHPWL (Average HPWL)	43.73% (44.04%)	16.25% (13.40%)	7.99% (2.72%)	4.59% (N/A)	4.86% (4.01%)	7.16% (1.21%)	18.38% (14.67%)	7.74% (0.60%)	10.11% (8.28%)	0.00% (0.00%)

Table 3.6. Runtime (minutes) on the ISPD 2006 benchmark suite [45]. CP=Capo, FP=FastPlace, CPx=ComPLx, AP=APlace, NP=NTUPlace. Cited results are marked with *. Average results are normalized to ePlace.

Categories	Min-Cut	Quadratic			Nonlinear			
Benchmarks	CP10.5*	FP3.0	CPx	POLAR	AP3*	NP3	mPL6	ePlace
ADAPTEC5	161.97	21.00	16.70	14.48	337.78	64.53	97.30	34.18
NEWBLUE1	42.70	5.18	4.15	5.95	71.72	12.57	24.48	9.62
NEWBLUE2	94.03	8.80	9.70	8.53	92.22	22.80	61.28	10.10
NEWBLUE3	101.27	10.10	8.58	9.02	208.38	21.00	102.23	14.38
NEWBLUE4	115.43	13.22	11.05	10.17	249.70	38.92	67.75	22.92
NEWBLUE5	347.57	28.70	25.85	23.78	546.65	76.82	127.38	54.83
NEWBLUE6	308.08	20.85	20.52	22.27	485.40	67.60	120.83	52.33
NEWBLUE7	916.03	40.97	50.65	48.23	914.20	149.30	307.03	86.27
Average	6.68×	0.59×	0.55×	0.69×	10.21×	1.63×	3.71×	1.00×

Table 3.7. Scaled density overflow on the ISPD 2006 benchmark suite [45]. CP=Capo, FP=FastPlace, MPE=MAPLE, CPx=ComPLx, AP=APlace, NP=NTUPlace. Cited results are marked with *. All the results are evaluated by the official scripts [45]. Average results are normalized to ePlace.

Categories	Min-Cut	Quadratic					Nonlinear			
Benchmarks	CP10.5*	FP3	RQL*	MPE*	CPx	POLAR	AP3*	NP3	mPL6	ePlace
ADAPTEC5	0.62	8.17	9.25	4.76	1.93	12.48	15.87	28.51	1.03	0.71
NEWBLUE1	0.13	1.04	0.34	1.05	1.02	2.13	0.06	0.70	9.02	0.28
NEWBLUE2	0.29	1.00	1.45	1.01	1.05	1.83	0.42	1.82	1.44	0.68
NEWBLUE3	0.01	0.55	0.07	0.77	0.93	1.36	0.00	0.05	0.66	0.07
NEWBLUE4	1.15	4.22	15.2	5.86	1.45	11.38	1.74	13.66	1.70	1.20
NEWBLUE5	0.33	7.21	13.6	4.05	1.76	12.91	12.45	20.37	1.47	0.63
NEWBLUE6	0.05	1.02	4.33	1.08	1.14	5.80	0.03	0.28	1.41	0.40
NEWBLUE7	0.02	1.30	2.57	1.70	1.40	4.63	0.06	2.01	1.19	0.25
Average	0.45×	5.90×	9.08×	5.46×	4.19×	13.77×	5.58×	12.29×	7.14×	1.00×

average of all the eight ISPD 2006 circuits, state-of-the-art quadratic placers consumes roughly 60% runtime of that by ePlace.

Besides, we also include the results of the scaled density overflow and original wirelength for comparison between all the placers, as shown in Table 3.7 and Table 3.5 (in parenthesis), respectively. Our placer could outperform eight out of the totally nine placers with smaller scaled density overflow. For Capo10.5 with better density overflow, ePlace produces 44.04% shorter original wirelength, where their respective scaled wirelength still lag behind ours by 43.73%. In terms of original HPWL, our placer outperforms eight out of the totally nine placers in comparison. POLAR and NTUplace3

lag behind ePlace with 0.60% and 1.21% shorter original HPWL, however, their scaled density overflow is 13.77× and 12.29× than that of ePlace. As a result, the scaled wirelength of ePlace is 7.16% and 7.74% shorter than that of POLAR and NTUplace3, respectively.

3.5.3 Placement Runtime Breakdown

We use the timing profile of our placement algorithm ePlace on ADAPTEC1 to analyze the runtime bottleneck. The placement region is uniformly decomposed into 512×512 grids. Using CG method with line search (ePlace-CG), we find that 5.62% of the total runtime is consumed by initial placement (quadratic wirelength minimization), 14.68% is consumed by legalization and detailed placement, while the remaining 79.70% is due to our global placement. A breakdown of the global placement execution shows that the runtime bottlenecks lie on the computation of wirelength gradient (6.89%), density gradient (20.88%) and function evaluation in line search (63.22%), while remaining operations take 9.01% runtime. To improve the efficiency, steplength prediction can be used to replace the line search, and we use Nesterov's method to solve the runtime bottleneck. The steplength is predicted based on our method of dynamic Lipschitz constant approximation, of which the runtime overhead is negligible. After replacing CG method with Nesterov's method, the total runtime of ePlace-Nes is improved by 2.21×. Specifically, the runtime consumed by global placement is reduced to 54.72%, while the initial placement and detail placement consume 11.44% and 33.84%, respectively. The remaining bottlenecks mainly lie on the computation of wirelength gradient (19.72%) and density gradient (59.6%), while other miscellaneous operations totally cost 20.68% time. To further accelerate the placement engine, we can extend the gradient computation to a parallel platform. The symmetric structure of the FFT algorithm for density gradient computation as well as the nature of the wirelength gradient

computation [12] would well fit the architecture of graphics processing unit (GPU) [43] and distributed systems.

3.6 Summary

In this chapter, we propose a flat nonlinear global placement algorithm *ePlace*. Based on the development of a novel placement density formulation *eDensity*, the placement instance is converted to an electrostatic system to model the density cost as the system potential energy. The electric potential and field distribution are correlated with the spatial density distribution via a well-defined Poisson's equation, and we use spectral methods based on fast Fourier transform to produce fast and accurate numerical solution. We propose to use Nesterov's method as the nonlinear placement solver, which outperforms CG solver with better quality and efficiency. A novel heuristic is developed to dynamically approximate the Lipschitz constant for steplength prediction. Our nonlinear preconditioning technique further enhance the solution quality with negligible runtime overhead. The experimental results on the ISPD 2005 and ISPD 2006 benchmarks validate the high performance of ePlace. More details on ePlace framework and solutions can be found at [14].

Chapter 3 includes the published content in the journal "ePlace: Electrostatics based Placement using Fast Fourier Transform and Nesterov's Method" by Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Huang, Chin-Chi Teng and Chung-Kuan Cheng in ACM Transactions on Design Automation of Electronic Systems (TODAES). The dissertation author was the primary investigator and author of the paper.

Chapter 4

ePlace-MS: Electrostatics based Placement for Mixed-Size Circuits

In this chapter, we discuss our develpement of a novel mixed-size placer *ePlace-MS*, which is based on the prototype proposed in Chapter 3. Macros and standard cells are equalized by our nonlinear preconditioning methodology and smoothly co-optimized by Nesterov's method. We also provide more thorough analysis and insights on the density function *eDensity*, which shows high performance on mixed-size circuits. Experiments validate the high performance of our mixed-size placer ePlace-MS.

4.1 Placement Overview

Figure 4.1 shows the flowchart of ePlace-MS. Given a placement instance, it quadratically minimizes the total wirelength at the first stage of mixed-size initial placement (mIP). The initial solution $\mathbf{v}_{mIP}$ is of low wirelength but high overlap. Based on the target density ρ_t, our mixed-size global placer (mGP) populates extra whitespace with unconnected fillers, then iteratively co-optimizes all the objects (standard cells, macros and fillers) together. After mGP, we remove all the fillers, fix the standard-cell layout, then invoke the annealing engine mLG to legalize the location of all the macros. In the second-phase global placement (cGP), we retrieve all the fillers and distribute them

appropriately, then free standard cells and co-place them with fillers to further reduce the wirelength. Finally, in the standard-cell detailed placement (cDP), we invoke the detailed placer in [22] to legalize and discretely optimize the standard-cell layout.

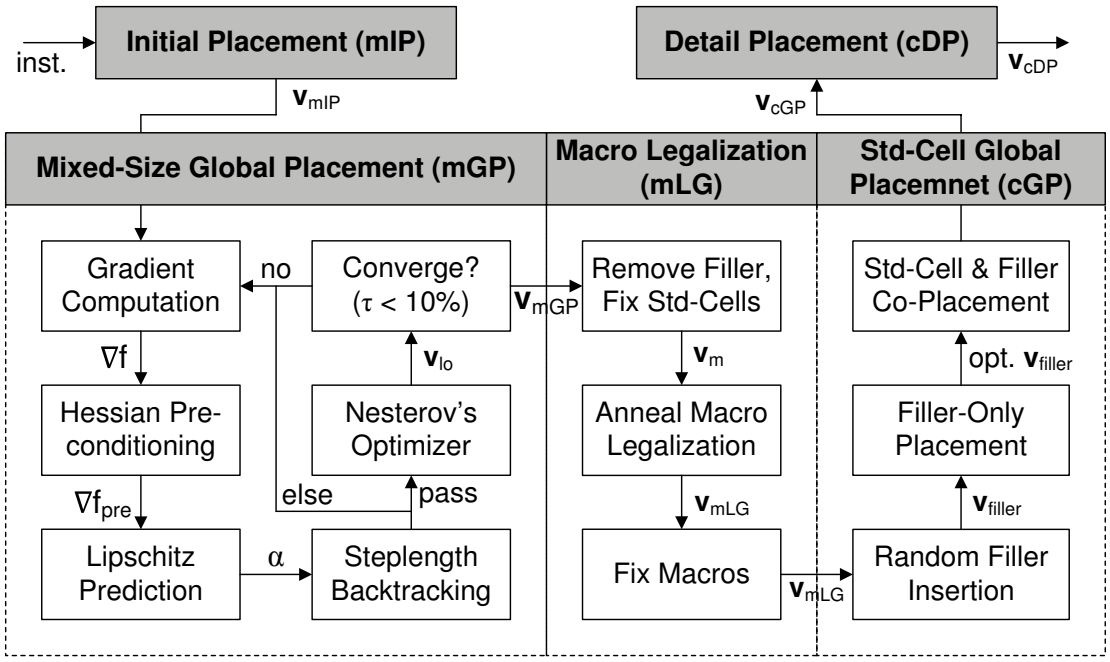


Figure 4.1. The flowchart of ePlace-MS.

ePlace-MS does not allow rotation or flipping of any objects due to the lithography issue. However, it has the flexibility to smoothly integrate the rotational and flipping gradients [22] to guide placement optimization iteratively. Deadspace allocation is also not considered in this work, while it can be effectively realized in ePlace-MS via appropriate macro inflation.

ePlace-MS maximally expands the design space for mGP with the major optimization effort budgeted on mixed-size global placement, since all the objects (standard cells, macros, fillers) are allowed to move and can be optimized simultaneously. In contrast, the design spaces for mLG and cGP are relatively shrunk, as only macros or standard cells are allowed to move with other objects fixed thus acting as constraints, which actually constrains the search space of mixed-size placement solution. Specifi-

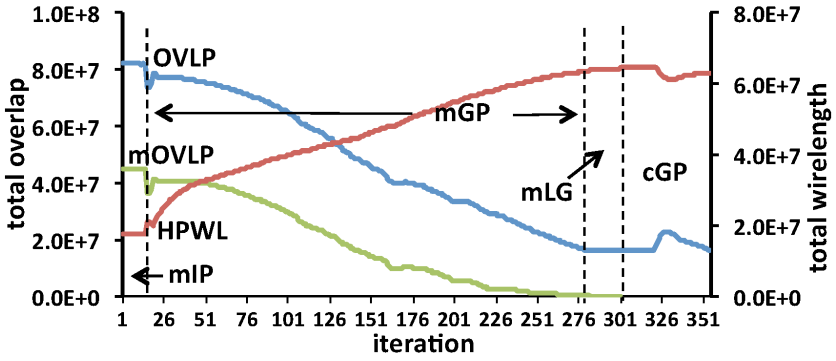


Figure 4.2. Total HPWL, total object overlap (OVLP) and total macro overlap (mOVLP) at different stages and iterations of ePlace-MS-WA on the MMS ADAPTEC1 benchmark. Overlap between macros are cleaned at macro legalization (mLG) where mOVLP decreases to zero. Remaining OVLP will all be cleaned at cDP (following cGP).

cally, only minor layout perturbation is expected to perform changes within local scale. As Figure 4.2[1] shows, the constrained optimization focuses on the mGP stage and terminates when overlap is small enough. The entire placement framework is built upon our recent work of FFTPL [33] with similar initialization and iterative adjustment of parameters. **Grid dimension** m is statically determined as $m = \lceil \log_2 \sqrt{n'} \rceil$ and upper-bounded by 1024, where $n' = |V'|$ is the number of movable macros, standard cells and fillers [33]. **Penalty factor** λ is initially set as Eq. (10) of [8]. We iteratively update $\lambda_k = \mu_k \lambda_{k-1}$ in mGP to balance the wirelength and density forces, where $\mu_k = 1.1^{-\frac{\Delta HPWL_k}{\Delta HPWL_{REF}}+1.0}$ based on the HPWL variation $\Delta HPWL_k = HPWL(\vec{v}_k) - HPWL(\vec{v}_{k-1})$. In practice, we set $\Delta HPWL_{REF} = 3.5 \times 10^5$ and bound μ_k by $[0.75, 1.1]$. **Density overflow** τ is used as the stopping criterion. We terminate mGP when $\tau \leq 10\%$ and cGP when $\tau \leq 7\%$, respectively. **Wirelength coefficient** γ is used to smooth the HPWL. We set the smoothing parameter as $\gamma = 8.0 w_b \times 10^{20/9 \times (\tau - 0.1) - 1.0}$ to encourage global movement at early iterations and convergence at later iterations. **Fillers** are used to balance the electrostatic direct current (DC) component in the global scale. The total area of fillers equals the

[1]Here OVLP denotes physical overlap among all the objects. Computation costs $O(n \log n)$ time via scanline and segment-tree data structure.

total whitespace multiplies target density then subtracted by the total area of all the movable objects. All the fillers are equally sized to be the average physical dimensions of all the standard cells. More details of parameter adjustment or filler formation can be found in [33].

4.2 Density Function for Mixed-Size Placement

Given the high performance of ePlace on standard cell circuits, we extend the density function to handle the mixed-size placement in a generalized way. Figure 4.3 shows the progression via a density-only mixed-size placement by ePlace-MS, where standard cells and macros are smoothly co-optimized towards even density distribution. Table 4.2 and 4.3 show that our density function has the best performance with shortest wirelength and smallest density overflow versus all the mixed-size placers in literature [22, 27, 61, 65].

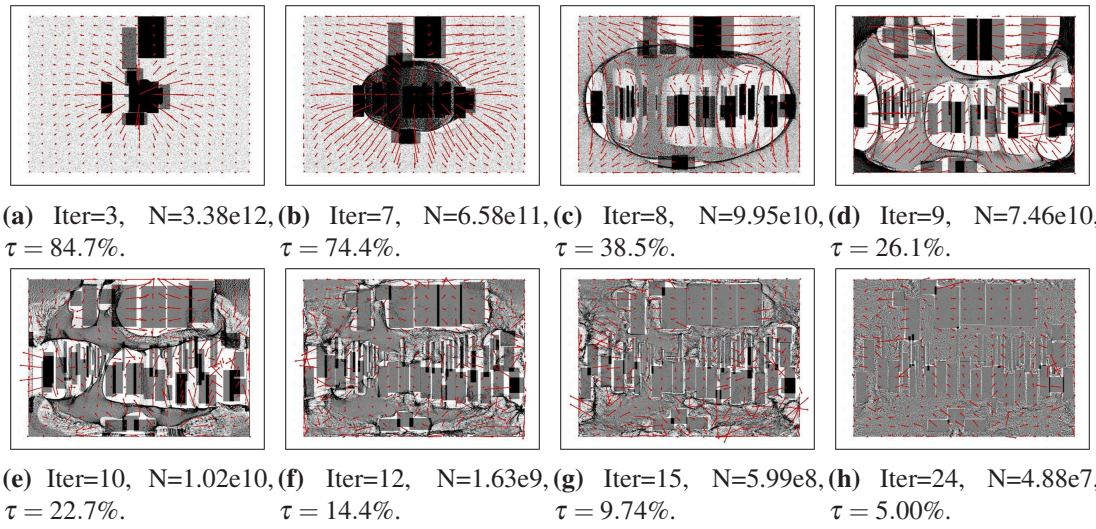
A placement density function is developed in our prior work [33] and discussed in Section 3.1 based on the electrostatic analogy, which is therefore named *eDensity*. Modeling every object as a positive charge, the density function $N(\mathbf{v})$ shown in Eq. (4.1) is modeled as the total electric potential energy. The electric force keeps spreading all the charges apart from each other, thus reducing the total potential energy towards zero in the end. The electrostatic equilibrium state is coupled with even placement density distribution and will be eventually reached. Compared to all the previous mixed-size placement algorithms [6, 22, 26, 27, 30, 52, 61, 65], our density function achieves the minimum density overflow as shown in Table 4.3, indicating the fewest violations to the target density thus the best performance of our density function

$$N(\mathbf{v}) = \frac{1}{2}\sum_{i \in V} N_i(\mathbf{v}) = \frac{1}{2}\sum_{i \in V} q_i \psi_i(\mathbf{v}). \tag{4.1}$$

Here q_i is the electric quantity of the charge i, it equals the area of the respective object i. ψ_i is the local potential. Also, as the system energy equals the sum of mutual potential energy between all the pairs of charges, we have a factor of $\frac{1}{2}$ for the energy of each single charge. A well-defined Poisson's equation in Eq. (4.2) correlates the density distribution $\rho(x,y)$ with the potential distribution $\psi(x,y)$, where x and y are spatial coordinates. We enforce Neumann boundary condition (i.e., zero gradient at the boundary of the density function or placement domain) to prevent objects from moving outside the placement region R. Specifically, the horizontal density gradients along the two vertical boundaries are equivalent to zero, vice versa, such that movement towards the placement boundaries will be gradually slowed down and finally stopped.

$$\begin{cases} \nabla \cdot \nabla \psi(x,y) = -\rho(x,y), \\ \hat{\mathbf{n}} \cdot \nabla \psi(x,y) = \mathbf{0},\ (x,y) \in \partial R, \\ \iint_R \rho(x,y) = \iint_R \psi(x,y) = 0. \end{cases} \tag{4.2}$$

Here $\hat{\mathbf{n}}$ is the outer normal vector at the boundary ∂R. We use $\boldsymbol{\xi}(x,y) = \nabla\psi(x,y)$ to denote the electric field distribution. The electric force on each charge i equals $q_i\boldsymbol{\xi}_i(\mathbf{v})$, where $\boldsymbol{\xi}_i = (\xi_{i_x}, \xi_{i_y})$ is the local field vector and can be decomposed into its horizontal (ξ_{i_x}) and vertical (ξ_{i_y}) components. Our density function $N(\mathbf{v})$ is generalized. In contrast to prior nonlinear placers [22, 26], there is no special handling or smoothing applied to movable macros or fixed blocks. Please refer to Section 3.2.7 for a more detailed advantage analysis. The global smoothness of $N(\mathbf{v})$ (by Eq. (4.1) and (4.2)) indicates that the local movement of any object will change the potential map in the global scale. The potential energy of all the objects will thus be changed by the movement of every single object i.



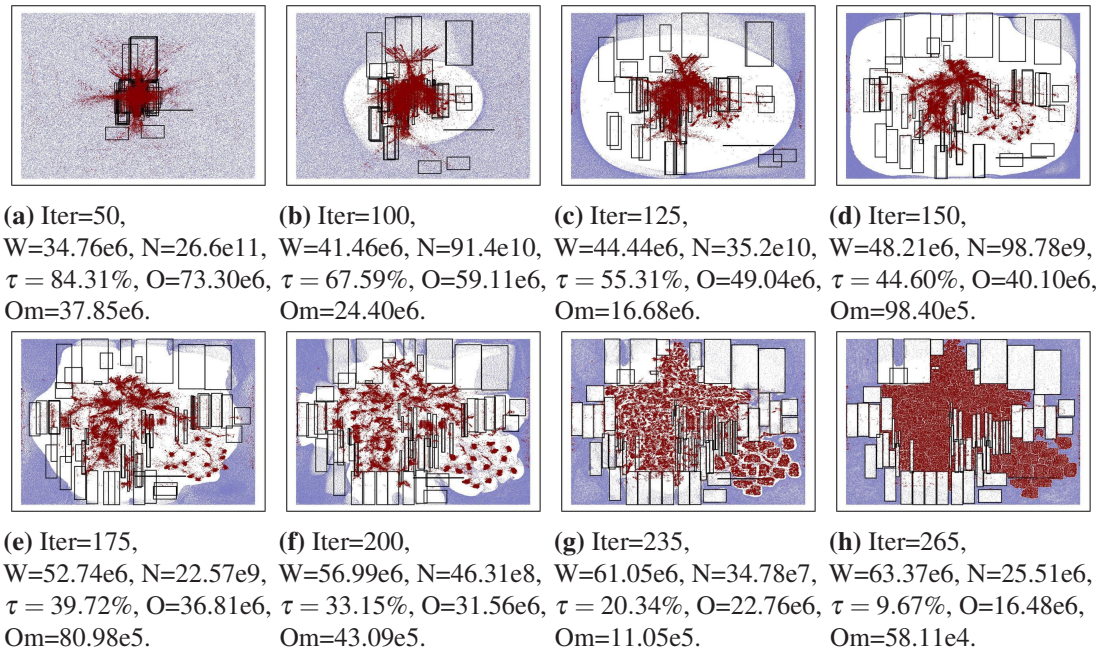
(a) Iter=3, N=3.38e12, $\tau = 84.7\%$. **(b)** Iter=7, N=6.58e11, $\tau = 74.4\%$. **(c)** Iter=8, N=9.95e10, $\tau = 38.5\%$. **(d)** Iter=9, N=7.46e10, $\tau = 26.1\%$.

(e) Iter=10, N=1.02e10, $\tau = 22.7\%$. **(f)** Iter=12, N=1.63e9, $\tau = 14.4\%$. **(g)** Iter=15, N=5.99e8, $\tau = 9.74\%$. **(h)** Iter=24, N=4.88e7, $\tau = 5.00\%$.

Figure 4.3. Snapshots of the density distribution by eDensity via mixed-size placement on the MMS ADAPTEC1 benchmark. The placement is driven by only density forces (denoted by red arrows) with the magnitude of the grid density characterized by grayscale. Total potential energy and total density overflow are denoted by N and τ, respectively.

4.3 Nonlinear Optimization for Mixed-Size Global Placement (mGP)

Our prior work shows high performance of Nesterov's method on placing standard cell based circuits. In this section, we extend it to handle mixed-size placement, where we observe consistently good performance as shown by the experimental results in Section 4.7. In the framework of ePlace-MS (Figure 4.1), mGP uses Nesterov's method to smoothly conducts simultaneous optimization on both macros and standard cells, as Figure 4.4 shows. As a generalized approach, mGP handles macros and standard cells in exactly the same way (c.f. macro shifting at each netlist declustering level [22], formation of soft blocks by standard cells [61, 65], special density smoothing of macros [26, 28], macro shredding [27], etc.). In each iteration, we compute the gradient and preconditioner, predict the Lipschitz constant, and adjust steplength via backtracking. Nesterov's method solves the nonlinear problem iteratively till conver-



(a) Iter=50, W=34.76e6, N=26.6e11, $\tau = 84.31\%$, O=73.30e6, Om=37.85e6.

(b) Iter=100, W=41.46e6, N=91.4e10, $\tau = 67.59\%$, O=59.11e6, Om=24.40e6.

(c) Iter=125, W=44.44e6, N=35.2e10, $\tau = 55.31\%$, O=49.04e6, Om=16.68e6.

(d) Iter=150, W=48.21e6, N=98.78e9, $\tau = 44.60\%$, O=40.10e6, Om=98.40e5.

(e) Iter=175, W=52.74e6, N=22.57e9, $\tau = 39.72\%$, O=36.81e6, Om=80.98e5.

(f) Iter=200, W=56.99e6, N=46.31e8, $\tau = 33.15\%$, O=31.56e6, Om=43.09e5.

(g) Iter=235, W=61.05e6, N=34.78e7, $\tau = 20.34\%$, O=22.76e6, Om=11.05e5.

(h) Iter=265, W=63.37e6, N=25.51e6, $\tau = 9.67\%$, O=16.48e6, Om=58.11e4.

Figure 4.4. Snapshots of mGP progression in ePlace-MS-WA on the MMS ADAPTEC1 benchmark with standard cells, macros and fillers shown by red points, black rectangles and blue points. Total wirelength, total potential energy, total density overflow, total object overlap and total macro overlap are denoted by W, N, τ, O and Om, respectively.

gence is reached.

4.3.1 Existing Problems

Line search remains the major runtime bottleneck in the Conjugate Gradient method[2], which is widely used in prior nonlinear placers [26]. In practice, it is not guaranteed that the steplength output by line search could satisfy the conjugacy requirement [20]. Specifically, the vector of current search direction may not be orthogonal (w.r.t. the Hessian matrix of the cost function) to all the previous vectors. Therefore, theoretical convergence rate of conjugate gradient method, $2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k$, can not be guaranteed. Instead of line search, [8] statically determines the steplength via upper-bounding the Euclidean distance of objects movement per iteration by a constant number. Such

[2]Our empirical studies on FFTPL [33] show that line search takes more than 60% of the total runtime on placing ADAPTEC1 of ISPD 2005.

static prediction assumes underestimation of steplength, which in general slows down the placement convergence rate. Moreover, steplength overestimation could still occur at some special area of the search space where the gradient changes sharply, therefore degrades the solution quality. As a result, a systematic solution with dynamic steplength adjustment and theoretical support becomes quite necessary.

4.3.2 Nesterov's Method

The flow of Nesterov's method used in ePlace-MS is illustrated in Algorithm 4. We use Lipschitz constant prediction together with steplength backtracking to control the speed of optimization. a_k is an optimization parameter which is iteratively updated. There are two concurrently updated solutions, $\mathbf{u_k}$ and $\mathbf{v_k}$, where only $\mathbf{u}$ is output as the final solution (at the end of mGP and cGP), while $\mathbf{v}$ is used for steplength prediction. ∇f_{pre} denotes the preconditioned gradient vector, which will be discussed in Section 4.4. Initially, we set $a_0 = 1$ and have both $\mathbf{u}_0$ and $\mathbf{v}_0$ set as $\mathbf{v}_{mIP}$. $BkTrk$ denotes steplength backtracking as shown in Section 4.3.4. The convergence rate of Nesterov's method is proven to be $O(1/k^2)$ in [49], on condition that the steplength α_k satisfies Eq. (4.3) at every single iteration k.

$$f(\mathbf{v}_k) - f(\mathbf{v}_k - \alpha_k \nabla f(\mathbf{v}_k)) \geq 0.5\alpha_k \|\nabla f(\mathbf{v}_k)\|^2 \tag{4.3}$$

Bisection search is suggested by [49] to generate the maximal α_k without violating the inequality in Eq. (4.3). Similar to line search in the Conjugate Gradient method, such steplength search usually introduces significant runtime overhead. As [49] claims, the function $f(\mathbf{v}_k - \alpha \nabla f(\mathbf{v}_k))$ would be evaluated by $O(\log L)$ times along the search direction for a single iteration, increasing the complexity to $O(n \log n \log L)$. Here L is the Lipschitz constant as defined in Definition 2. As a result, step length prediction

becomes necessary to accelerate the optimization process.

Algorithm 4. Nesterov's method in ePlace-MS

Require: a_k, $\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{v}_{k-1}$, $\nabla f_{pre}(\mathbf{v}_k)$, $\nabla f_{pre}(\mathbf{v}_{k-1})$
Ensure: $\mathbf{u}_{k+1}$, $\mathbf{v}_{k+1}$, a_{k+1}
1: $\alpha_k = BkTrk(\mathbf{v}_k, \mathbf{v}_{k-1}, \nabla f_{pre}(\mathbf{v}_k), \nabla f_{pre}(\mathbf{v}_{k-1}))$
2: $\mathbf{u}_{k+1} = \mathbf{v}_k - \alpha_k \nabla f_{pre}(\mathbf{v}_k)$
3: $a_{k+1} = \left(1 + \sqrt{4a_k^2 + 1}\right)/2$
4: $\mathbf{v}_{k+1} = \mathbf{u}_{k+1} + (a_k - 1)(\mathbf{u}_{k+1} - \mathbf{u}_k)/a_{k+1}$
5: **return**

4.3.3 Lipschitz Constant Prediction

Instead of line search, we compute the steplength through a closed-form formula of the Lipschitz constant of the gradient, which is defined as below.

Definition 2. *Given a multivariate convex function $f(\mathbf{v}) \in C^{1,1}(H)$, $\exists L > 0$ s.t. $\forall \mathbf{u}, \mathbf{v} \in H$,*

$$\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\|. \tag{4.4}$$

H as Hilbert space is a generalized notion of Euclidean space, $C^{1,1}(H)$ requires $f(\mathbf{v})$ with Lipschitz continuous gradient. As our objective is non-convex, we leverage Nesterov's method in an approximate way. [49] states that $\alpha_k = L^{-1}$ satisfies the steplength requirement specified in Eq. (4.3) but lacks a formal proof. The rationale behind is that smaller Lipschitz constant indicates higher smoothness of the gradient thus faster convergence can be achieved via larger steplength, vice versa. Here we provide a proof to the statement that $\alpha_k = L^{-1}$ always satisfies Eq. (4.3) as Theorem 2.

Theorem 2. *Given convex $f \in C^{1,1}(H)$ and L defined in Definition 2, $\alpha \leq L^{-1}$ satisfies Eq. (4.3).*

Proof. $\forall \mathbf{u}, \mathbf{v} \in H$, we have

$$
\begin{aligned}
& f(\mathbf{v}) - f(\mathbf{u}) - \langle \nabla f(\mathbf{u}), \mathbf{v} - \mathbf{u} \rangle \\
=& \int_{\mathbf{u}}^{\mathbf{v}} \nabla f(\mathbf{v}') d\mathbf{v}' - \langle \nabla f(\mathbf{u}), \mathbf{v} - \mathbf{u} \rangle \\
=& \int_0^1 \nabla f(\mathbf{u} + \tau(\mathbf{v} - \mathbf{u})) d\left(\tau(\mathbf{v} - \mathbf{u})\right) - \langle \nabla f(\mathbf{u}), \mathbf{v} - \mathbf{u} \rangle \\
=& \int_0^1 \langle \nabla f(\mathbf{u} + \tau(\mathbf{v} - \mathbf{u})) - \nabla f(\mathbf{u}),\ \mathbf{v} - \mathbf{u} \rangle d\tau \\
\leq& \int_0^1 \|\nabla f(\mathbf{u} + \tau(\mathbf{v} - \mathbf{u})) - \nabla f(\mathbf{u})\| \cdot \|\mathbf{v} - \mathbf{u}\| d\tau \\
\leq& \int_0^1 L \cdot \|\tau(\mathbf{v} - \mathbf{u})\| \cdot \|\mathbf{v} - \mathbf{u}\| d\tau \\
=& 0.5L \|\mathbf{v} - \mathbf{u}\|^2,
\end{aligned}
\tag{4.5}
$$

where the first and second inequalities hold based on the Cauchy-Schwartz inequality [1] and the definition of Lipschitz constant in Eq. (4.4), respectively. Eq. (4.5) indicates that

$$
f(\mathbf{v}) \leq f(\mathbf{u}) + \langle \nabla f(\mathbf{u}), \mathbf{v} - \mathbf{u} \rangle + 0.5L \|\mathbf{v} - \mathbf{u}\|^2. \tag{4.6}
$$

Let $\mathbf{u} = \mathbf{v}_k$ and $\mathbf{v} = \mathbf{v}_k - \alpha_k \nabla f(\mathbf{v}_k)$, based on Eq. (4.6) we have

$$
\begin{aligned}
& f(\mathbf{u}) - f(\mathbf{v}) = f(\mathbf{v}_k) - f(\mathbf{v}_k - \alpha_k \nabla f(\mathbf{v}_k)) \\
\geq& \langle \nabla f(\mathbf{u}), \mathbf{u} - \mathbf{v} \rangle - 0.5L \|\mathbf{v} - \mathbf{u}\|^2 \\
=& \langle \nabla f(\mathbf{v}_k), \alpha_k \nabla f(\mathbf{v}_k) \rangle - 0.5\alpha_k^2 L \|\nabla f(\mathbf{v}_k)\|^2 \\
=& \alpha_k \|\nabla f(\mathbf{v}_k)\|^2 - 0.5\alpha_k^2 L \|\nabla f(\mathbf{v}_k)\|^2 \\
\geq& (\alpha_k - 0.5\alpha_k^2 \alpha_k^{-1}) \|\nabla f(\mathbf{v}_k)\|^2 \\
=& 0.5\alpha_k \|\nabla f(\mathbf{v}_k)\|^2,
\end{aligned}
\tag{4.7}
$$

where the second inequality holds if we have $L \leq \alpha_k^{-1}$. □

As a result, L^{-1} can be used as the steplength to accelerate the algorithm without convergence penalty. Exact Lipschitz constant is very expensive to compute (even more time consuming than line search). Moreover, static estimation will be invalidated through iterative change of the cost function, as both the wirelength coefficient γ in Eq. (2.5) and penalty factor λ in Eq. (3.5) are being iteratively adjusted in ePlace-MS (more details can be found in [33]). As a result, we approximate the Lipschitz constant and steplength as follows

$$\widetilde{L}_k = \frac{\|\nabla f(\mathbf{v}_k) - \nabla f(\mathbf{v}_{k-1})\|}{\|\mathbf{v}_k - \mathbf{v}_{k-1}\|},\ \alpha_k = \widetilde{L}_k^{-1}, \tag{4.8}$$

where only $\mathbf{v}$ is used for Lipschitz constant prediction. The computation overhead is negligible since both $\nabla f(\mathbf{v}_{k-1})$ and $\nabla f(\mathbf{v}_k)$ are known thus there is no extra gradient or cost computation.

4.3.4 Steplength Backtracking

We develop a backtracking method to enhance the prediction accuracy via preventing potential steplength overestimation by Eq. (4.8), which would unexpectedly misguide the nonlinear solver. Being used to generate $\mathbf{v}_{k+1}$, however, α_k by Eq. (4.8) is predicted using $\mathbf{v}_k$ and $\mathbf{v}_{k-1}$. Instead, our backtracking method predicts α_k using $\mathbf{v}_k$ and $\mathbf{v}_{k+1}$. At line 1 of Algorithm 5, we set the steplength computed by Eq. (4.8) as a temporary variable $\hat{\alpha}_k$. The respective temporary solution $\hat{\mathbf{v}}_{k+1}$ (line 3) is used to produce a *reference steplength*. If it is exceeded by $\hat{\alpha}_k$ (line 4), we update $\hat{\alpha}_k$ and $\hat{\mathbf{v}}_{k+1}$ at lines 5 and 7 and do the backtracking circularly until the inequality at line 4 is satisfied. $\mathbf{v}_k$ and $\mathbf{v}_{k-1}$ are the placement solutions for the current iteration k and the past iteration $k-1$. $\mathbf{u}_k$ is the other solution (at iteration k) simultaneously updated with $\mathbf{v}_k$, as shown in Algorithm 4. $\varepsilon = 0.95$ is the scaling factor to encourage earlier return of function *BkTrk*

Algorithm 5. *BkTrk*

Require: a_k, a_{k+1}, $\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{v}_{k-1}$, $\nabla f_{pre}(\mathbf{v}_k)$, $\nabla f_{pre}(\mathbf{v}_{k-1})$
Ensure: α_k
1: $\hat{\alpha}_k = \frac{\|\mathbf{v}_k - \mathbf{v}_{k-1}\|}{\|\nabla f_{pre}(\mathbf{v}_k) - \nabla f_{pre}(\mathbf{v}_{k-1})\|}$
2: $\hat{\mathbf{u}}_{k+1} = \mathbf{v}_k - \hat{\alpha}_k \nabla f_{pre}(\mathbf{v}_k)$
3: $\hat{\mathbf{v}}_{k+1} = \hat{\mathbf{u}}_{k+1} + (a_k - 1)(\hat{\mathbf{u}}_{k+1} - \mathbf{u}_k)/a_{k+1}$
4: **while** $\hat{\alpha}_k > \varepsilon \left(\frac{\|\hat{\mathbf{v}}_{k+1} - \mathbf{v}_k\|}{\|\nabla f_{pre}(\hat{\mathbf{v}}_{k+1}) - \nabla f_{pre}(\mathbf{v}_k)\|} \right)$ **do**
5: $\quad \hat{\alpha}_k = \frac{\|\hat{\mathbf{v}}_{k+1} - \mathbf{v}_k\|}{\|\nabla f_{pre}(\hat{\mathbf{v}}_{k+1}) - \nabla f_{pre}(\mathbf{v}_k)\|}$
6: $\quad \hat{\mathbf{u}}_{k+1} = \mathbf{v}_k - \hat{\alpha}_k \nabla f_{pre}(\mathbf{v}_k)$
7: $\quad \hat{\mathbf{v}}_{k+1} = \hat{\mathbf{u}}_{k+1} + (a_k - 1)(\hat{\mathbf{u}}_{k+1} - \mathbf{u}_k)/a_{k+1}$
8: **end while**
9: $\alpha_k = \hat{\alpha}_k$
10: **return**

thus prevent over-backtracking, which could consume too much runtime with limited accuracy improvement. The runtime overhead is zero if the first check at line 2 is passed, since the newly computed gradient $\nabla f(\hat{\mathbf{v}}_{k+1})$ can be reused at the following iteration. Experiments show that the average number of backtracks per iteration over all the sixteen MMS benchmarks [65] is only 1.037, indicating less than 4% runtime overhead on mGP. Disabling backtracking causes ePlace-MS-WA (using the weighted-average wirelength model) to fail on MMS BIGBLUE4 and increase wirelength by 43.12% on average of the remaining 15 MMS benchmarks, showing the substantial importance of our steplength backtracking method for the mixed-size placement.

4.4 Nonlinear Preconditioning

This section introduces our development of the nonlinear preconditioner, which is used by Nesterov's method in Algorithm 4 and steplength backtracking in Algorithm 5. Preconditioning reduces the condition number of a problem, which is transformed to be more suitable for numerical solution. Traditional preconditioning techniques compute the inverse of the Hessian matrix $\mathbf{H_f}$ of the objective function f. Preconditioning has

broad application in quadratic placers [27, 30, 60, 61] but none attempts in nonlinear placers [6, 22, 26], mainly due to the non-convexity of the density function. In this work, we approximate the original Hessian $\mathbf{H_f}$ with a positive definite diagonal matrix $\widetilde{\mathbf{H}}_\mathbf{f}$ as the preconditioner. We multiply it to the gradient vector and use $\nabla f_{pre} = \widetilde{\mathbf{H}}_\mathbf{f}^{-1} \nabla f$ to direct the nonlinear placement optimization. A preconditioned gradient vector ∇f_{pre} is used to stretch the function space to be more spherical in order to smooth and accelerate the numerical optimization. However, as the objective function of global placement is of large scale (usually millions of objects to place) and highly nonlinear, to compute the Hessian matrix becomes very expensive and indeed computationally impractical. As a result, we choose the Jacobi preconditioner using only the diagonal terms of the Hessian matrix $\mathbf{H_f}$, as Eq. (4.9) shows

$$\mathbf{H}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} \approx \widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 f}{\partial x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}. \tag{4.9}$$

By Eq. (3.5), we have

$$\widetilde{\mathbf{H}}_\mathbf{f} = \begin{pmatrix} \widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} & 0 \\ 0 & \widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{y},\mathbf{y}}} \end{pmatrix} = \widetilde{\mathbf{H}}_\mathbf{W} + \lambda \widetilde{\mathbf{H}}_\mathbf{N}. \tag{4.10}$$

As $\frac{\partial^2 f(\mathbf{v})}{\partial x_i^2} = \frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} + \lambda \frac{\partial^2 N(\mathbf{v})}{\partial x_i^2}$, we need to separately compute or estimate $\frac{\partial^2 W}{\partial x_i^2}$ and $\frac{\partial^2 N}{\partial x_i^2}$ at every iteration.

4.4.1 Wirelength

Based on the LSE wirelength modeling equation shown in Eq. (2.5), we differentiate it to derive the gradient function of the wirelength of net e w.r.t. x_i as shown below.

$$\begin{aligned}\frac{\partial W_e^{LSE}(\mathbf{v})}{\partial x_i} =& \frac{\gamma}{\sum_{j\in e}\exp\left(x_j/\gamma\right)} \times \frac{\partial \sum_{j\in e}\exp\left(x_j/\gamma\right)}{\partial x_i} + \\ & \frac{\gamma}{\sum_{j\in e}\exp\left(-x_j/\gamma\right)} \times \frac{\partial \sum_{j\in e}\exp\left(-x_j/\gamma\right)}{\partial x_i} \\ =& \frac{\exp\left(x_i/\gamma\right)}{\sum_{j\in e}\exp\left(x_j/\gamma\right)} - \frac{\exp\left(-x_i/\gamma\right)}{\sum_{j\in e}\exp\left(-x_j/\gamma\right)}\end{aligned} \tag{4.11}$$

Via further differentiating Eq. (4.11) w.r.t. x_i, we are able to derive the second-order gradient of the LSE function as below.

$$\begin{aligned}\frac{\partial^2 W_e^{LSE}(\mathbf{v})}{\partial x_i^2} =& \frac{\exp\left(x_i/\gamma\right)\left\{\sum_{j\in e}\exp\left(x_j/\gamma\right) - \exp\left(x_i/\gamma\right)\right\}}{\gamma\{\sum_{j\in e}\exp\left(x_j/\gamma\right)\}^2} + \\ & \frac{\exp\left(-x_i/\gamma\right)\left\{\sum_{j\in e}\exp\left(-x_j/\gamma\right) - \exp\left(-x_i/\gamma\right)\right\}}{\gamma\{\sum_{j\in e}\exp\left(-x/\gamma\right)\}^2}\end{aligned} \tag{4.12}$$

Similarly, we can derive the gradient function of the WA wirelength model by differentiating Eq. (2.6), as below shows

$$\begin{aligned}\frac{\partial W_e^{WA}(\mathbf{v})}{\partial x_i} =& \frac{\sum_{j\in e}\exp\left(x_j/\gamma\right)\left(\exp\left(x_i/\gamma\right) + \left(x_i/\gamma\right)\exp\left(x_i/\gamma\right)\right)}{\left(\sum_{j\in e}\exp\left(x_j/\gamma\right)\right)^2} - \\ & \frac{\left(\exp\left(x_i/\gamma\right)/\gamma\right)\left(\sum_{j\in e} x_j \exp\left(x_j/\gamma\right)\right)}{\left(\sum_{j\in e}\exp\left(x_j/\gamma\right)\right)^2} + \\ & \frac{\sum_{j\in e}\exp\left(-x_j/\gamma\right)\left(\exp\left(-x_i/\gamma\right) - \left(x_i/\gamma\right)\exp\left(-x_i/\gamma\right)\right)}{\left(\sum_{j\in e}\exp\left(-x_j/\gamma\right)\right)^2} + \\ & \frac{\left(\exp\left(-x_i/\gamma\right)/\gamma\right)\left(\sum_{j\in e} x_j \exp\left(-x_j/\gamma\right)\right)}{\left(\sum_{j\in e}\exp\left(-x_j/\gamma\right)\right)^2}\end{aligned} \tag{4.13}$$

However, further differentiation of Eq. (4.13) is complicated, moreover, quite computa-

tionally expensive. As a result, we use the vertex degree of object i instead,

$$\frac{\partial^2 W_e^{WA}(\mathbf{v})}{\partial x_i^2} = \sum_{e \in E_i} \frac{\partial^2 W_e(\mathbf{v})}{\partial x_i^2} \approx |E_i|, \tag{4.14}$$

where E_i denote the set of all the nets incident to the object i. We have the second-order derivative of the wirelength function $W(\mathbf{v})$ w.r.t. the horizontal movement of object i (i.e. x_i) expressed as below.

$$\frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} = \frac{\partial^2 \sum_{e \in E_i} W_e(\mathbf{v})}{\partial x_i^2} = \sum_{e \in E_i} \frac{\partial^2 W_e(\mathbf{v})}{\partial x_i^2} \tag{4.15}$$

Since $W(\mathbf{v})$ in both LSE and WA are strongly convex [21, 47] and globally differentiable, the Hessian matrices are also positive definite with straightly positive eigenvalues. As a result, we can use the closed-form formula $\frac{\partial^2 W}{\partial x_i^2}$ in Eq. (4.15) as the nonlinear wirelength preconditioner.

4.4.2 Density

By differentiating the density gradient function in Eq. (3.12), we could obtain the second-order derivative as below

$$\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} = -q_i \frac{\partial \xi_{i_x}(\mathbf{v})}{\partial x_i} = -q_i \rho_{i_x}(\mathbf{v}), \tag{4.16}$$

where $\rho_i = \rho_{i_x} + \rho_{i_y}$. However, the density function $N(\mathbf{v})$ by Eq. (4.1) is based on a repulsive force dominant system, thus it is non-convex. As a result, we could have $\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} < 0$ for some object i. Negative preconditioner will invert the direction of gradient, causing the cost to increase and the placement solution to diverge. To avoid this, we concisely

approximate the density preconditioner as below.

$$\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} = q_i \frac{\partial^2 \psi_i(\mathbf{v})}{\partial x_i^2} \approx q_i \tag{4.17}$$

Such operation actually helps decompose charges of different electric quantities all into unit charges, the electric force applied onto each charge is uniquely determined by the local electric field, while placement oscillation due to imbalance of density forces is avoided. The rationale behind is similar to the mechanical movement, where the motion velocity of each object depends on its acceleration, which is uniquely determined by the respective field (electrostatic, gravitational, etc.) but not the mass of the object. As a result, our density equalization method is indeed a simulation of the behavior of a real electrostatic system. Such system in the real world will always progress towards states of lower energy, which guarantees the convergence in the end achieving the even density distribution. The performance comparison of the three density preconditioners (no preconditioner, Eq. (4.17) and Eq. (4.16)) is shown in Figure 4.5. Compared to the other two options, our proposed preconditioner using charge quantity q_i (object area) achieves the highest effectiveness and efficiency in the convergence of the density cost minimization.

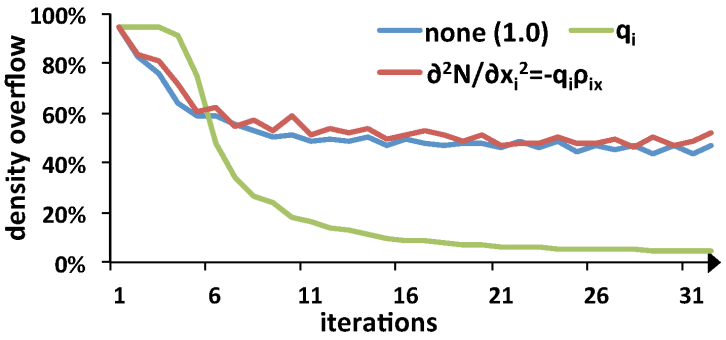


Figure 4.5. Performance comparison of the three candidate density preconditioners via a density-only placement on the MMS ADAPTEC1 benchmark.

4.4.3 Summary

As a result, we use $\frac{\partial^2 f(\mathbf{v})}{\partial x_i^2} \approx \frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} + \lambda q_i$ to approximate the ith diagonal term of the placement preconditioner $\widetilde{\mathbf{H}}_{\mathbf{f_{x,x}}}$ w.r.t. horizontal charge movement, while the preconditioner for the vertical charge movement can be derived in a similar way. Disabling the preconditioner causes ePlace-MS to fail on nine out of the totally sixteen MMS benchmarks, since macros significantly differ from standard cells with much higher magnitude of gradients. As a result, unpreconditioned gradient makes macros with large area and high incidence degree to bounce between opposite placement boundaries, causing the solution to oscillate and hard to converge within limited number of iterations[3]. On average of the remaining seven MMS benchmarks, the wirelength is increased by 24.63%, indicating the high effectiveness of our preconditioner for the mixed-size placement.

4.5 Macro Legalization (mLG)

Based on the mGP solution $\mathbf{v}_{mGP}$, mLG legalizes the macro layout via a simulated annealing (SA) [29] based approach, as Figure 4.7 shows. Unlike traditional SA based floorplanners and macro placers [7, 9, 44, 61] which perturb floorplan expression then physically realize it, mLG uses SA to directly control macro motion.

- We expect a high-quality solution from mGP. Only local macro shifts are expected in mLG, the shrunk design space can be well explored by SA.
- Our SA engine is more efficient with only minor position change to each single macro.
- After each random perturbation of floorplan expression, the respective floorplan realization may cause significant layout change, which is time consuming and could induce unexpected quality degradation.

[3] We set 3000 as the upper limit of iterations in ePlace-MS.

Similar to Timberwolf [53], however, mLG legalizes macros rather than detailedly place cells. As Figure 4.6 shows, mLG can be decomposed into two levels. At each iteration

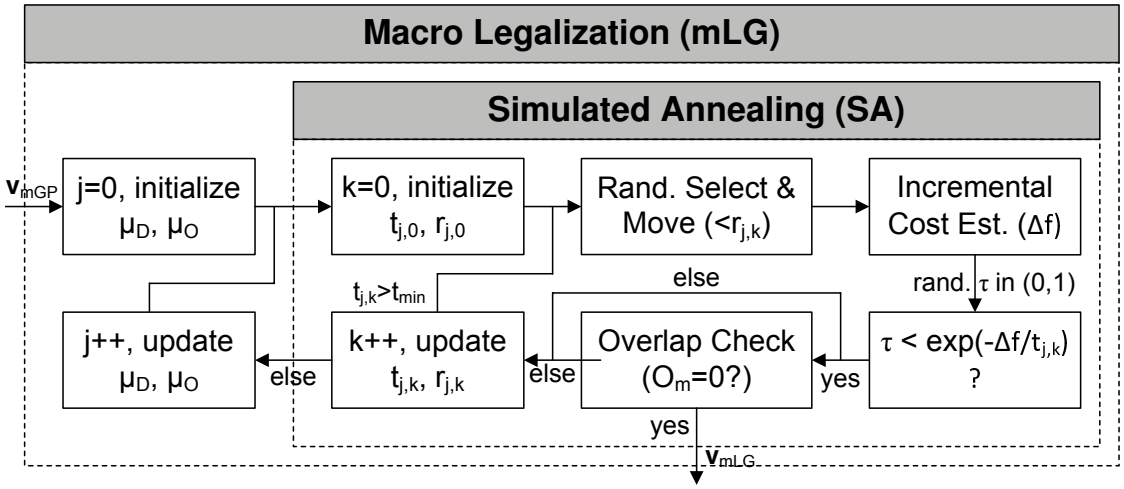


Figure 4.6. Our two-level annealing-based macro legalizer.

of the outer loop (mLG iteration), we update the cost function $f_{mLG}(\mathbf{v})$ by

$$f_{mLG}(\mathbf{v}) = HPWL(\mathbf{v}) + \mu_D D(\mathbf{v}) + \mu_O O_m(\mathbf{v}), \tag{4.18}$$

where $HPWL(\mathbf{v})$, $D(\mathbf{v})$ and $O_m(\mathbf{v})$ denote the total wirelength, total standard-cell area covered by macros and total macro overlap, respectively. We set mLG as a constrained optimization.

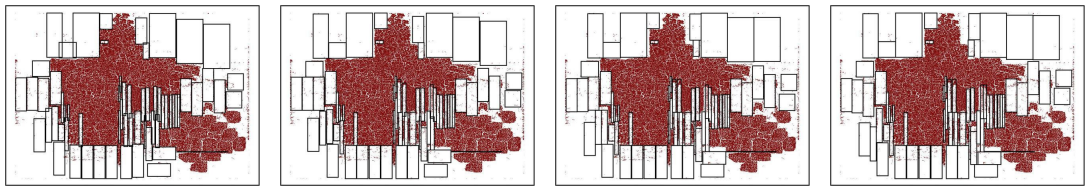
- **Objective** is to minimize $HPWL(\mathbf{v}) + \mu_D D(\mathbf{v})$. Since penalty on $D(\mathbf{v})$ will be transformed to wirelength during cGP and cDP, we treat them equally in mLG thus statically set $\mu_D = \frac{HPWL(\mathbf{v})}{D(\mathbf{v})}$.

- **Constraint** is zero macro overlap ($O_m(\mathbf{v}) = 0$). We set μ_O as the penalty factor and initialize it as $(HPWL(\mathbf{v}) + \mu_D D(\mathbf{v}))/O_m(\mathbf{v})$. μ_O is multiplied by β at each mLG iteration to make the legalizer more aggressive on macro overlap reduction.

At each iteration of the inner loop shown in Figure 4.6 (SA iteration), the annealer randomly picks a macro and randomly determine its motion vector within the search range. The cost difference Δf is then incrementally evaluated and we generate a random number $\tau \in (0,1)$ to determine whether the new layout will be accepted or not by checking if $\tau < \exp\left(-\frac{\Delta f}{t_{j,k}}\right)$. Here j and k denote the mLG and SA iteration indices. The **temperature** $t_{j,k}$ at each iteration (j,k) is determined based on the maximum cost increase $\Delta f_{max}(j,k)$ that will be accepted by more than 50% probability, thus we set $t_{j,k} = \frac{\Delta f_{max}(j,k)}{\ln 2}$. We set $\Delta f_{max}(j,0)$ ($\Delta f_{max}(j,k_{max})$) as $0.03 \times \beta^j$ ($0.0001 \times \beta^j$), denoting that cost increase by less than 3% (0.01%) at the first (last) SA iteration will be accepted by more than 50% probability. These parameters appear small but fit well into our framework, since only minor layout change is expected in mLG. Meanwhile, they are scaled up per mLG iteration to adapt to the enhancement of the penalty factor μ_O. We initialize $\Delta f_{max}(j,k)$ by $\Delta f_{max}(j,0)$ and linearly decrease it towards $\Delta f_{max}(j,k_{max})$. The **radius** $r_{j,k}$ of macro motion range is dependent on both the penalty factor and the amount of macros. Given m macros to legalize, we set $r_{j,0} = \frac{R_x}{\sqrt{m}} \times 0.05 \times \beta^j$, which means the entire placement region R can be decomposed into m sub-regions, every macro can be moved within 5% of its assigned region at each time. Similar to the temperature, the radius is scaled by β at each mLG iteration. In practice, we set $\beta = 1.5$ to achieve good tradeoff between quality and efficiency.

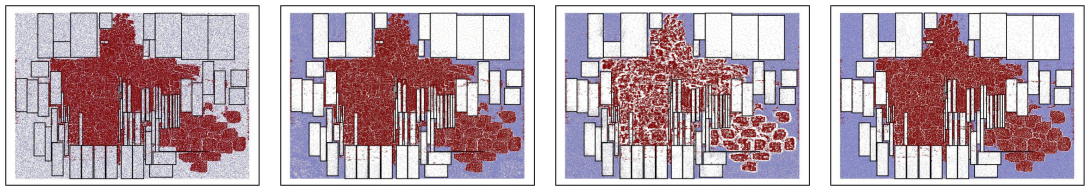
4.6 Standard Cell-Only Global Placement (cGP)

Based on the fixed macro layout produced by mLG, cGP mitigates the quality overhead due to mLG via a second-phase global placement performed on only standard cells. With the presence of all the macros fixed, however, cGP uses the same nonlinear algorithm as that of mGP. In contrast, as Figure 4.8 shows, cGP introduces only small changes to the standard-cell layout and converges much faster than mGP. It consists



(a) j=0, k=0, W=63.37e6, D=12.29e5, O=16.48e6, Om=60.94e4. **(b)** j=0, k=1, W=63.61e6, D=13.57e5, O=16.17e6, Om=17.14e4. **(c)** j=0, k=9, W=64.19e6, D=14.55e5, O=16.03e6, Om=2.16e4. **(d)** j=3, k=1, W=64.36e6, D=14.83e5, O=16.08e6, Om=0.

Figure 4.7. Distribution of macros (a) before mLG (b) 1st mLG iteration (c) 2nd mLG iteration (d) after mLG by ePlace-MS-WA on the MMS ADAPTEC1 benchmark with fixed standard-cell layout and all the fillers removed. Total wirelength, total standard-cell area covered by macros, total object overlap and total macro overlap are denoted by W, D, O and Om, respectively.



(a) Iter=0, W=64.36e6, N=42.05e9, $\tau = 9.84\%$, O=16.09e6. **(b)** Iter=20, W=64.36e6, N=78.27e6, $\tau = 9.84\%$, O=16.09e6. **(c)** Iter=28, W=61.30e6, N=21.70e7, $\tau = 19.96\%$, O=22.83e6. **(d)** Iter=51, W=63.04e6, N=15.29e6, $\tau = 9.81\%$, O=16.29e6.

Figure 4.8. Distribution of standard cells and fillers (a) before cGP (b) after filler redistribution (c) standard cell and filler co-optimization (d) after cGP by ePlace-MS-WA on the MMS ADAPTEC1 benchmark with the fixed macro layout produced by mLG. Total wirelength, total potential energy, total density overflow and total object overlap are denoted by W, N, τ and O, respectively. Total macro overlap remains zero and is not shown here.

of three steps (1) filler insertion (2) filler-only placement (3) standard cell & filler co-placement.

As Figure 4.1 shows, mLG is unaware of existing fillers in $\mathbf{x}_{cGP}$ and may introduce substantial macro-to-filler overlap. As a result, we retrieve all the fillers and randomly distribute them in the placement region. With all the standard cells fixed, a filler-only placement is conducted for 20 iterations in order to relocate fillers to their best sites. The resulting solution with minimal density cost ensures the following placement of standard cells not to sacrifice wirelength to compensate density cost due to improper filler distribution. Experiments show that on average of all the MMS benchmarks, the

wirelength will be increased by 6.53% if we remove filler-only placement.

cGP co-optimizes standard cells with fillers after distributing all the fillers to their best location. The initial penalty factor λ_{cGP}^{init} is determined based on the penalty factor λ_{mGP}^{last} at the last mGP iteration. As λ will be multiplied by up to 1.1 for maximal aggressiveness enhancement, we set $\lambda_{cGP}^{init} = \lambda_{mGP}^{last} \times 1.1^m$ denoting that m buffering iterations are budgeted for cGP to recover the aggressiveness of mGP. This is shown in the cGP section of Figure 4.2, where the wirelength (overlap) first reduces (increases) sharply to approach a low-wirelength initial solution for cGP (similar to what mIP does). By increasing λ_{cGP} iteratively, cGP reduces the existing overlap with well controlled wirelength overhead. In practice, we set m as the number of mGP iterations divided by ten to achieve good performance with short runtime.

4.7 Experiments and Results

We implement ePlace-MS using C programming language and execute the program in a Linux machine with Intel i7 920 2.67GHz CPU and 12GB memory. To validate the performance of ePlace-MS, we conduct experiments on the modern mixed-size (**MMS**) benchmarks [65], as shown in Table 4.1. MMS benchmarks inherit the same netlists and density constraints ρ_t from ISPD 2005 [46] and ISPD 2006 [45] benchmarks but have all the macros freed to place. There are also fixed IO blocks inserted within the placement domain in order to maintain the uniqueness of the analytic solution. Following the contest policy in ISPD 2006 [45], there is a benchmark-specific density upper-bound ρ_t for eight out of the totally sixteen circuits. This target density ρ_t helps produce whitespace among circuit objects to accommodate interconnect and buffers, therefore facilitate the following design stages of routing, timing correction, etc. By the benchmark protocol [45], exceeding ρ_t will penalize the wirelength by $sHPWL = HPWL \times (1 + 0.01 \times \tau_{avg})$, where τ_{avg} denotes the scaled density overflow

per bin and *sHPWL* is the scaled wirelength. More detailed circuit statistics of MMS benchmarks can be found in [65]. After cGP is completed, ePlace-MS invokes the detailed placer in [22] for the legalization and detailed placement of only standard cells (cDP). There is no benchmark specific parameter tuning in our work, and we use the official scripts from [65] to evaluate the performance of all the placers in our experiments.

Seven state-of-the-art mixed-size placers covering two categories of algorithms (as discussed in Chapter 1) are included in the experiments for the performance comparison, namely, Capo10.5 [52], FastPlace3.0 [61], ComPLx (v13.07.30) [27], POLAR [30], mPL6 [6], FLOP [65], NTUplace3-unified [22]. We have obtained the binaries of four placers and executed them on our machine. FLOP is not available due to IP and other issues, thus we cite their performance from [65]. Capo10.5 and mPL6 fail to work with MMS benchmarks in our machine, so we cite the respective results also from [65] instead. Also, APlace3 [26] crashes on every MMS circuit as reported in [65] thus is not included in the results. MP-tree [9] and CG [7] are not available due to the industrial copyrights, while their results on MMS benchmarks are also not available. However, as both of them have been outperformed by NTUplace3-unified [22] with on average 21% and 9% shorter wirelength (reported in Table V of [22]), we do not include them in our experiments.

The experimental results of HPWL and scaled HPWL (sHPWL) on the MMS circuits are shown in Table 4.2. As shown in Table 4.1, there are no target density constraints for the first eight circuits (i.e., 100%) thus no density penalty on the wirelength. In other words, HPWL equals sHPWL for the first eight MMS testcases in Table 4.2 as marked with †. NTUplace3-unified-NR (with macro rotation and flipping disabled) fails on two MMS benchmarks (NEWBLUE3 and NEWBLUE7) with the average wirelength, density overflow and runtime computed based on the other fourteen benchmarks. Compared to all the placers in the experiments, ePlace-MS produces the best solutions

Table 4.1. Statistics of the MMS benchmark suite [65].

Circuits	# Objects	# Mov. Objects	# Std. Cells	# Macros	# Fixed I/Os	# Nets	# Pins	Target Den. (ρ_t)
ADAPTEC1	211447	210967	210904	63	480	221142	944053	100 %
ADAPTEC2	255023	254584	254457	127	439	266009	1069482	100 %
ADAPTEC3	451650	450985	450927	58	665	466758	1875039	100 %
ADAPTEC4	496054	494785	494716	69	1260	515951	1912420	100 %
BIGBLUE1	278164	277636	277604	32	528	284479	1144691	100 %
BIGBLUE2	557866	535741	534782	959	22125	577235	2122282	100 %
BIGBLUE3	1096812	1095583	1093034	2549	1229	1123170	3833218	100 %
BIGBLUE4	2177353	2169382	2169183	199	7970	2229886	8900078	100 %
ADAPTEC5	843128	842558	842482	76	570	867798	3493147	50 %
NEWBLUE1	330474	330137	330073	64	337	338901	1244342	80 %
NEWBLUE2	441516	440264	436516	3748	1252	465219	1773855	90 %
NEWBLUE3	494011	482884	482833	51	11127	552199	1929892	80 %
NEWBLUE4	646139	642798	642717	81	3341	637051	2499178	50 %
NEWBLUE5	1233058	1228268	1228177	91	4790	1284251	4957843	50 %
NEWBLUE6	1255039	1248224	1248150	74	6815	1288443	5307594	80 %
NEWBLUE7	2507954	2481533	2481372	161	26421	2636820	10104920	80 %

Table 4.2. HPWL (marked with †) and scaled HPWL ($\times 10^6$) on the MMS benchmark suite [65]. Mac=Macros, CP=Capo, FP=FastPlace, CPx=ComPLx, NP3U=NTUplace3-unified, NR="no rotation or flipping of macros". Cited results are marked with *. All the results are evaluated by the official scripts [65].

Categories	Constructive			One-Stage						ePlace-MS	
Benchmarks	CP10.5*	FLOP-NR*	FLOP*	FP3.0	CPx	POLAR	mPL6*	NP3U-NR	NP3U	LSE	WA
ADAPTEC1†	84.77	77.18	76.83	82.39	79.05	92.17	77.84	75.92	75.55	66.99	**66.82**
ADAPTEC2†	92.61	87.17	84.14	88.53	99.11	149.43	88.40	84.89	78.50	**76.74**	76.76
ADAPTEC3†	202.37	182.21	175.99	187.98	175.78	197.48	180.64	170.88	169.74	161.63	**161.55**
ADAPTEC4†	202.38	166.55	161.68	187.50	156.75	175.19	162.02	167.13	166.68	**145.89**	147.04
BIGBLUE1†	112.58	95.45	94.92	104.91	96.18	99.12	99.36	96.42	96.57	87.27	**86.29**
BIGBLUE2†	149.54	150.66	153.02	145.89	147.19	157.72	144.37	148.12	147.17	132.72	**130.06**
BIGBLUE3†	583.37	372.79	346.24	400.40	344.63	420.28	319.63	324.39	338.47	287.34	**284.39**
BIGBLUE4†	915.37	807.53	777.84	775.43	772.53	814.07	804.00	797.17	799.66	660.17	**656.68**
ADAPTEC5	565.88	381.83	357.83	338.77	338.67	380.45	376.30	295.24	**294.24**	304.68	312.86
NEWBLUE1	110.54	73.36	67.97	73.91	65.26	70.68	66.93	61.13	61.25	**60.43**	61.87
NEWBLUE2	303.25	231.94	187.40	197.15	187.87	197.65	179.18	164.27	163.76	**159.11**	162.98
NEWBLUE3	1282.19	344.71	345.99	325.72	**269.47**	601.17	415.86	N/A	280.92	287.69	304.16
NEWBLUE4	300.69	256.91	256.54	270.70	256.97	277.60	277.69	231.59	229.36	**226.29**	229.20
NEWBLUE5	570.32	516.71	510.83	500.09	453.05	450.69	515.49	414.81	420.46	**392.77**	392.93
NEWBLUE6	609.16	502.24	493.64	512.19	452.83	475.78	482.44	471.51	474.86	414.56	**409.28**
NEWBLUE7	1481.45	1113.07	1078.18	1016.10	1010.00	1107.59	1038.66	N/A	1100.84	**889.18**	895.11
Avg. (s)HPWL	66.14%	20.16%	15.46%	19.47%	12.04%	32.03%	17.25%	8.61%	8.22%	−0.57%	0.00%

Table 4.3. Scaled average density overflow per bin on the MMS benchmark suite [65]. Mac=Macros, CP=Capo, FP=FastPlace, CPx=ComPLx, NP3U=NTUplace3-unified, NR="no rotation or flipping of macros". Cited results are marked with *.

Categories	Constructive			One-Stage						ePlace-MS	
Benchmarks	CP10.5*	FLOP-NR*	FLOP*	FP3.0	CPx	POLAR	mPL6*	NP3U-NR	NP3U	LSE	WA
ADAPTEC5	N/A	N/A	4.19	2.41	1.00	5.48	N/A	4.59	5.34	0.09	0.75
NEWBLUE1	N/A	N/A	1.14	1.03	1.05	2.39	N/A	0.53	1.35	0.06	0.04
NEWBLUE2	N/A	N/A	0.87	0.07	0.19	0.02	N/A	0.12	0.05	0.05	0.03
NEWBLUE3	N/A	N/A	1.02	0.01	0.01	0.00	N/A	N/A	0.00	0.00	0.00
NEWBLUE4	N/A	N/A	4.94	2.62	1.35	10.43	N/A	8.69	10.1	0.35	0.29
NEWBLUE5	N/A	N/A	2.85	1.21	1.08	7.68	N/A	7.80	9.14	0.17	0.17
NEWBLUE6	N/A	N/A	1.34	1.11	1.06	5.10	N/A	1.53	2.09	0.27	0.23
NEWBLUE7	N/A	N/A	1.48	0.60	0.99	1.88	N/A	28.51	0.33	0.14	0.09
Avg. Den. Ovf.	N/A	N/A	27.64×	7.49×	7.69×	23.99×	N/A	17.65×	17.99×	1.15×	1.00

Table 4.4. Runtime (minutes) on the MMS benchmark suite [65]. Mac=Macros, CP=Capo, FP=FastPlace, CPx=ComPLx, NP3U=NTUplace3-unified, NR="no rotation or flipping of macros". Cited results are marked with *.

Categories	Constructive			One-Stage						ePlace-MS	
Benchmarks	CP10.5*	FLOP-NR*	FLOP*	FP3.0	CPx	POLAR	mPL6*	NP3U-NR	NP3U	LSE	WA
ADAPTEC1	92.78	12.03	13.73	3.32	5.43	4.85	40.07	5.63	6.50	5.25	5.47
ADAPTEC2	122.88	20.32	19.87	3.28	21.55	9.73	47.83	8.45	6.30	7.58	7.43
ADAPTEC3	282.88	32.38	35.27	6.73	14.07	10.72	99.72	15.10	10.57	26.22	27.23
ADAPTEC4	291.15	39.97	40.45	6.80	16.87	12.52	99.52	9.40	8.97	56.40	29.35
BIGBLUE1	140.97	29.60	33.45	5.05	4.25	4.95	47.15	12.07	10.90	7.85	7.82
BIGBLUE2	294.12	39.55	51.83	5.32	49.80	12.28	203.37	17.08	17.68	13.97	13.70
BIGBLUE3	91165.35	117.28	990.62	20.27	176.38	46.07	159.13	47.30	58.13	82.20	72.98
BIGBLUE4	1829.75	230.27	327.85	36.73	126.02	56.63	397.80	115.95	92.17	141.37	204.15
ADAPTEC5	399.28	84.13	54.88	15.22	13.15	21.37	377.27	53.78	46.07	50.27	48.35
NEWBLUE1	52.22	22.80	16.87	5.03	4.58	6.20	52.85	11.85	10.75	11.70	10.87
NEWBLUE2	135.93	44.10	40.23	6.52	51.02	21.93	100.73	13.55	15.00	51.12	62.40
NEWBLUE3	1222.32	38.93	45.95	12.08	36.30	39.27	293.72	N/A	58.08	30.57	17.53
NEWBLUE4	109.82	42.92	40.92	9.05	10.57	13.58	162.20	28.05	32.07	28.27	29.73
NEWBLUE5	275.80	146.68	152.72	21.60	35.78	30.62	413.43	82.03	77.50	55.47	63.40
NEWBLUE6	301.27	157.50	159.38	18.37	21.25	28.02	218.53	62.97	65.73	112.62	69.65
NEWBLUE7	723.10	312.75	418.40	50.53	73.75	66.77	528.00	N/A	116.03	392.02	191.47
Avg. CPU	13.71×	1.96×	2.06×	0.36×	1.09×	0.67×	5.92×	0.90×	1.00×	1.18×	1.00×

with the shortest wirelength for fourteen out of the totally sixteen testcases. Besides, it outperforms the leading-edge mixed-size placer NTUplace3 [22] by up to 22.98% shorter wirelength[4] and on average 8.22% shorter wirelength over all the MMS circuits. Notice that unlike NTUplace3, ePlace-MS does not allow macro rotation or flipping, which indicates further improvement space thus potentially better solution quality. The statistics of density overflow (i.e. the amount of violations to the testcase dependent target density ρ_t as specified in Table 4.1) is shown in Table 4.3. The respective results of Capo10.5, FLOP-NR and mPL6 are not available from respective publications [65]. ePlace-MS obtains consistently the lowest density overflow at all the eight testcases (with predefined target density), showing the best performance of our density modeling method *eDensity*. The runtime statistics is shown in Table 4.4. On average of all the sixteen MMS benchmarks, ePlace-MS runs faster than Capo10.5, FLOP, ComPLx, mPL6, and shows essentially the same efficiency with NTUplace3. Despite longer runtime than FastPlace3.0 and POLAR, ePlace-MS produces on average 19.47% and 32.03% shorter wirelength. In general, ePlace-MS outperforms all the mixed-size placement algorithms in literature and achieves good results on both LSE and WA wirelength models, showing that our density function and nonlinear optimization algorithm have high and stable performance, which are not dependent on specific wirelength models.

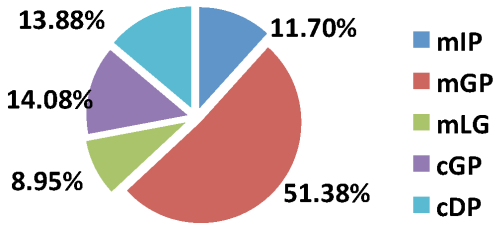


Figure 4.9. The runtime breakdown of ePlace-MS-WA on average of all the sixteen MMS benchmarks.

[4]ePlace-MS produces 22.98% shorter wirelength than NTUplace3 on NEWBLUE7, which is the largest design in the MMS benchmark suite with roughly 2.5 million components.

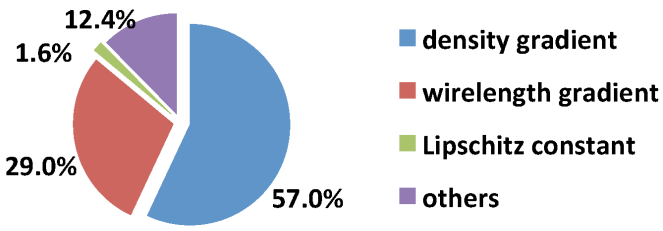


Figure 4.10. The runtime breakdown of mGP of ePlace-MS-WA on average of all the sixteen MMS benchmarks.

Figure 4.9 shows the **CPU breakdown** of ePlace-MS-WA on average of all the MMS benchmarks. mGP is the most effective placement stage (as Figure 4.2 shows) and consumes the longest runtime. A further breakdown of mGP by Figure 4.10 illustrates that computation of density and wirelength gradients and other operations (Lipschitz constant prediction, parameter update, etc.) consume 57%, 29% and 14% runtime of mGP.

4.8 Summary

ePlace-MS is a generalized and effective placement algorithm to handle mixed-size circuits of very large scale. Using the density function *eDensity* based on electrostatics analogy, macros and standard cells are equalized by preconditioning and smoothly co-optimized by Nesterov's method. Steplength is determined via Lipschitz continuity together with a backtracking strategy to prevent overestimation. Unlike all the approaches in literature, ePlace-MS treats standard cells and macros in exactly the same way. The experimental results on MMS benchmarks validate its high and stable performance.

Chapter 4 includes the content to appear in the journal "ePlace-MS: Electrostatics based Placement for Mixed-Size Circuits" by Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, Dennis Huang, Yufeng Luo, Chin-Chi Teng and Chung-Kuan Cheng in IEEE Transactions on Computer-

Aided Design of Integrated Circuits and Systems (TCAD). The dissertation author was the primary investigator and author of the paper.

Chapter 5

Conclusion

ePlace is a generalized and effective nonlinear placement algorithm. It resolves the traditional bottlenecks in nonlinear placement (low efficiency due to line search, sub-optimality of netlist clustering, quality degradation via coarse density grid at early stage, etc.), and shows that nonlinear placement has the capability to outperform cutting-edge quadratic placement algorithms [27, 28, 30, 58, 60] with better solution quality and comparable or even shorter runtime. Compared to the state-of-the-art research innovations in placement literature, such as the linear formulation of density for quadratic placement [13], or the exponential formulation of wirelength and Bell-shape quadratic formulation of density for nonlinear placement [47], and etc., ePlace looks into this traditional problem in a new angle. Specifically, we well study and leverage the analogy between placement and electrostatics, while eDensity is actually conducting a simulation on the behavior of the equivalent electrostatic system. As a result, we could have global smoothness, fast convergence and high quality all be achieved in a promising way.

By extension, we develop ePlace-MS, a generalized and effective placement algorithm, to handle large-scale mixed-size circuits. Macros and standard cells are well equalized by our nonlinear preconditioning methodology. All the circuit objects, despite of huge topological and physical differences, are smoothly co-optimized by Nesterov's

method with steplength estimated via Lipschitz continuity together with a novel backtracking strategy. Unlike all the approaches in literature, ePlace-MS treats standard cells and macros in exactly the same way, which saves engineering efforts and enhances performance stability. The experimental results on MMS benchmarks validate the high and stable performance of ePlace-MS over circuits of very different structures.

In future, We will explore opportunities in the parallel computing platform, while gradient computation can be well accelerated via distributed system [17] or graphics processing units [43]. As Figure 4.10 shows, the major runtime bottlenecks of mixed-size global placement lie in the computation of density and wirelength gradients, where complete independence between placement objects can be identified within the wirelength formulation and the FFT structure, indicating ultra-high parallelism of ePlace-MS thus huge potential of acceleration via multi-core, graphics processing unit or distributed systems. Moreover, the internal analytic infrastructure of ePlace-MS allows it to smoothly interface with other optimization stages in the entire VLSI back-end design flow, while integration of other design objectives (timing, routability, thermal, etc.) can be effectively realized through well formulated gradient functions together with appropriate adjustment of balancing ratios.

Bibliography

[1] K. M. Abadir and J. R. Magnus. *Matrix Algebra*. Cambridge University Press, 2005.

[2] S. N. Adya, I. L. Markov, and P. G. Villarrubia. On Whitespace and Stability in Mixed-Size Placement. In *ICCAD*, pages 311–318, 2003.

[3] A. R. Agnihorti, S. Ono, C. Li, M. C. Yildiz, A. Khathate, C.-K. Koh, and P. H. Madden. Mixed Block Placement Via Fractional Cut Recursive Bisection. *IEEE TCAD*, 24(5):748–761, 2005.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Optimal Partitioners and End-case Placers for Standard-cell Layout. *IEEE TCAD*, 19(11):1304–1313, 2000.

[5] T. Chan, J. Cong, and K. Sze. Multilevel Generalized Force-directed Method for Circuit Placement. In *ISPD*, pages 185–192, 2005.

[6] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. mPL6: Enhanced Multi-level Mixed-Size Placement. In *ISPD*, pages 212–214, 2006.

[7] H.-C. Chen, Y.-L. Chunag, Y.-W. Chang, and Y.-C. Chang. Constraint Graph-Based Macro Placement for Modern Mixed-Size Circuit Designs. In *ICCAD*, pages 218–223, 2008.

[8] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. NTUPlace3: An Analytical Placer for Large-Scale Mixed-Size Designs with Preplaced Blocks and Density Constraint. *IEEE TCAD*, 27(7):1228–1240, 2008.

[9] T.-C. Chen, P.-H. Yuh, Y.-W. Chang, F.-J. Huang, and D. Liu. MP-Trees: A Packing-Based Macro Placement Algorithm for Modern Mixed-Size Designs. *IEEE TCAD*, 27(9):1621–1634, 2008.

[10] E. G. Coffman, M. R. Garey, and D. S. Johnson. *Approximation Algorithms for Bin Packing: A Survey*. PWS Publishing Co., Boston, MA, USA, 1997.

[11] J. Cong, G. Luo, and E. Radke. Highly Efficient Gradient Computation for Density-Constrained Analytical Placement. *IEEE TCAD*, 27(12):2133–2144, 2008.

[12] J. Cong and Zou Y. Parallel Multi-level Analytical Global Placement on Graphics Processing Unit. In *ICCAD*, pages 681–688, 2009.

[13] H. Eisenmann and F. M. Johannes. Generic Global Placement and Floorplanning. In *DAC*, pages 269–274, 1998.

[14] ePlace Homepage. http://vlsi-cuda.ucsd.edu/~ljw/ePlace/index.html.

[15] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some Simplified NP-Complete Graph Problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[16] S. K. Han, K. Jeong, A. B. Kahng, and J. Lu. Stability and Scalability in Global Routing. In *SLIP*, pages 1–6, 2011.

[17] Q. He, W. Au, A. Korobkov, and S. Venkateswaran. Parallel Power Grid Analysis Using Distributed Direct Linear Solver. In *EMC(S)*, 2014.

[18] Q. He, D. Chen, and D. Jiao. From Layout Directly to Simulation: A First-Principle Guided Circuit Simulator of Linear Complexity and Its Efficient Parallelization. *IEEE CPMT*, 2(4):687–699, 2012.

[19] Q. He, H. Gan, and D. Jiao. Explicit Time-Domain Finite-Element Method Stablized for an Arbitrarily Large Time Step. *IEEE AP*, 60(11):5240–5250, 2013.

[20] M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[21] M.-K. Hsu, V. Balabanov, and Y.-W. Chang. TSV-Aware Analytical Placement for 3D IC Designs Based on a Novel Weighted-Average Wirelength Model. *IEEE TCAD*, 32(4):497–509, 2013.

[22] M.-K. Hsu and Y.-W. Chang. Unified Analytical Global Placement for Large-Scale Mixed-Size Circuit Designs. *IEEE TCAD*, 31(9):1366–1378, 2012.

[23] M.-K. Hsu, Y.-W. Chang, and V. Balabanov. TSV-Aware Analytical Placement for 3D IC Designs. In *DAC*, pages 664–669, 2011.

[24] ITRS. http://www.itrs.net/Links/2012ITRS/Home2012.htm. 2012.

[25] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer, 2010.

[26] A. B. Kahng and Q. Wang. A Faster Implementation of APlace. In *ISPD*, pages 218–220, 2006.

[27] M.-C. Kim and I.L. Markov. ComPLx: A Competitive Primal-dual Lagrange Optimization for Global Placement. In *DAC*, pages 747–752, 2012.

[28] M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji. MAPLE: Multilevel Adaptive Placement for Mixed-Size Designs. In *ISPD*, pages 193–200, 2012.

[29] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[30] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev. POLAR: Placement based on Novel Rough Legalization and Refinement. In *ICCAD*, pages 357–362, 2013.

[31] J. Lu. *Fundamental Research on Electronic Design Automation in VLSI Design - Routability*. M.Phil. Thesis, The Hong Kong Polytechnic University, 2010.

[32] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. Huang, C.-C. Teng, and C.-K. Cheng. ePlace: Electrostatics based Placement using Fast Fourier Transform and Nesterov's Method. *ACM TODAES*, 2014.

[33] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. FFTPL: An Analytic Placement Algorithm Using Fast Fourier Transform for Density Equalization. In *ASICON*, 2013.

[34] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. ePlace: Electrostatics based Placement using Nesterov's Method. In *DAC*, 2014.

[35] J. Lu, W.-K. Chow, and C.-W. Sham. Clock Network Synthesis with Concurrent Gate Insertion. In *PATMOS*, pages 228–237, 2010.

[36] J. Lu, W.-K. Chow, and C.-W. Sham. A New Clock Network Synthesizer for Modern VLSI Designs. *Integration*, 45(2):121–131, 2012.

[37] J. Lu, W.-K. Chow, and C.-W. Sham. Fast Power- and Slew-Aware Gated Clock Tree Synthesis. *IEEE TVLSI*, 20(11):2094–2103, 2012.

[38] J. Lu, W.-K. Chow, C.-W. Sham, and E. F.-Y. Young. A Dual-MST Approach for Clock Network Synthesis. In *ASPDAC*, pages 467–473, 2010.

[39] J. Lu and C.-W. Sham. LMgr: A Low-Memory Global Router with Dynamic Topology Update and Bending-Aware Optimum Path Search. In *ISQED*, pages 231–238, 2013.

[40] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C.-K. Cheng. ePlace-MS: Electrostatics based Placement for Mixed-Size Circuits. *IEEE TCAD*, 2014.

[41] D.-J. Lee M.-C. Kim and I. L. Markov. SimPL: An Effective Placement Algorithm. *IEEE TCAD*, 31(1):50–60, 2012.

[42] I. L. Markov, J. Hu, and M.-C. Kim. Progress and Challenges in VLSI Placement Research. In *DAC*, 2012.

[43] K. Moreland and E. Angel. The FFT on a GPU. *Graphics Hardware*, 2003.

[44] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI Module Placement Based on Rectangle-Packing by the Sequence Pair. *IEEE TCAD*, 15(12):1518–1524, 1996.

[45] G.-J. Nam. ISPD 2006 Placement Contest: Benchmark Suite and Results. In *ISPD*, pages 167–167, 2006.

[46] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. The ISPD2005 Placement Contest and Benchmark Suite. In *ISPD*, pages 216–220, 2005.

[47] W. C. Naylor, R. Donelly, and L. Sha. Non-Linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer. In *US Patent 6301693*, 2001.

[48] A. S. Nemirovskii and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley and Sons Ltd., 1983.

[49] Y. E. Nesterov. A Method of Solving A Convex Programming Problem with Convergence Rate $O(1/k^2)$. *Soviet Math*, 27(2):372–376, 1983.

[50] Takuya Ooura. General Purpose FFT Package, http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html. 2001.

[51] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[52] J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov. Min-Cut Floorplacement. *IEEE TCAD*, 25(7):1313–1326, 2006.

[53] C. Sechen and A. Sangiovanni-Vincentelli. TimberWolf3.2: A New Standard Cell Placement and Global Routing Package. In *DAC*, pages 432–439, 1986.

[54] C.-W. Sham, E. F.-Y. Young, and J. Lu. Congestion Prediction in Early Stages of Physical Design. *ACM TODAES*, 14(1):12:1–18, 2009.

[55] J. Shewchuk. An Introduction to the Conjugate Gradient Method without the Agonizing Pain. In *CMU-CS-TR-94-125,*, 1994.

[56] G. Skollermo. A Fourier Method for the Numerical Solution of Poisson's Equation. *Mathematics of Computation*, 29(131):697–711, 1975.

[57] P. Spindler, U. Schlichtmann, and F. M. Johannes. Kraftwerk2 - A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model. *IEEE TCAD*, 27(8):1398–1411, 2008.

[58] M. Struzyna. Sub-Quadratic Objectives in Quadratic Placement. In *DATE*, pages 1867–1872, 2013.

[59] T. Taghavi, X. Yang, and B.-K. Choi. Dragon2005: Large-Scale Mixedsize Placement Tool. In *ISPD*, pages 245–247, 2005.

[60] N. Viswanathan, G.-J. Nam, C. J. Alpert, P. Villarrubia, H. Ren, and C. Chu. RQL: Global Placement via Relaxed Quadratic Spreading and Linearization. In *DAC*, pages 453–458, 2007.

[61] N. Viswanathan, M. Pan, and C. Chu. FastPlace3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control. In *ASPDAC*, pages 135–140, 2007.

[62] L.-T. Wang, Y.-W. Chang, and K.-T. Cheng. *Electronic Design Automation: Synthesis, Verification and Test*. Morgan Kaufmann, 2009.

[63] X. Wang, W. Yueh, D. B. Roy, S. Narasimhan, Y. Zheng, S. Mukhopadhyay, D. Mukhopadhyay, and S. Bhunia. Role of Power Grid in Side Channel Attack and Power-Grid-Aware Secure Design. In *DAC*, pages 1–9, 2013.

[64] J. Z. Yan and C. Chu. DeFer: Deferred Decision Making Enabled Fixed-Outline Floorplanner. In *DAC*, pages 161–166, 2008.

[65] J. Z. Yan, N. Viswanathan, and C. Chu. Handling Complexities in Modern Large-Scale Mixed-Size Placement. In *DAC*, pages 436–441, 2009.

[66] B. Yao, H. Chen, C.-K. Cheng, N.-C. Chou, L.-T. Liu, and P. Suaris. Unified Quadratic Programming Approach for Mixed Mode Placement. In *ISPD*, pages 193–199, 2005.

[67] Y. Zheng, A. Basak, and S. Bhunia. CACI: Dynamic Current Analysis Towards Robust Recycled Chip Identification. In *DAC*, pages 1–6, 2014.

[68] Y. Zheng, M. Hashemian, and S. Bhunia. RESP: A Robust Physical Unclonable Function Retrofitted into Embedded SRAM Array. In *DAC*, pages 1–9, 2013.

[69] H. Zhuang, J. Lu, K. Samadi, Y. Du, and C.-K. Cheng. Performance-Driven Placement for Design of Rotation and Right Arithmetic Shifters in Monolithic 3D ICs. In *ICCCAS*, pages 509–513, 2013.