



Michigan Technological University
Create the Future Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's
Reports - Open

Dissertations, Master's Theses and Master's
Reports

2013

Strategic Optimization Techniques For FRTU Deployment and Chip Physical Design

Chen Liao
Michigan Technological University

Follow this and additional works at: <https://digitalcommons.mtu.edu/etds>



Part of the [Electrical and Computer Engineering Commons](#)

Copyright 2013 Chen Liao

Recommended Citation

Liao, Chen, "Strategic Optimization Techniques For FRTU Deployment and Chip Physical Design",
Dissertation, Michigan Technological University, 2013.
<https://doi.org/10.37099/mtu.dc.etds/485>

Follow this and additional works at: <https://digitalcommons.mtu.edu/etds>



Part of the [Electrical and Computer Engineering Commons](#)

STRATEGIC OPTIMIZATION TECHNIQUES FOR FRTU DEPLOYMENT AND CHIP
PHYSICAL DESIGN

By

Chen Liao

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Electrical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2013

© 2013 Chen Liao

This dissertation has been approved in partial fulfillment of the requirements for the Degree
of DOCTOR OF PHILOSOPHY in ELECTRICAL ENGINEERING.

Department of Electrical and Computer Engineering

Dissertation Advisor: *Dr. Shiyun Hu*

Committee Member: *Dr. Chee-Wooi Ten*

Committee Member: *Dr. Zhuo Feng*

Committee Member: *Dr. Cliff Chin Ngai Sze*

Department Chair: *Dr. Daniel R. Fuhrmann*

To my husband and my parents

I dedicate this dissertation to my dear family. They are my husband Sheng Hu, and my parents Xianglan Chen and Lifu Liao.

Contents

List of Figures	xi
List of Tables	xvii
Preface	xix
Acknowledgments	xxi
Abstract	xxiii
1 Introduction	1
1.1 Feeder Remote Terminal Unit (FRTU) Installation Considering Security of Secondary Distribution Network	2
1.2 Physical-Level Synthesis of Microfluidic Lab-on-a-chip	7
1.3 Approximation Scheme For Restricted Discrete Gate Sizing Targeting Delay Minimization	10
1.4 Publications	12
2 Strategic FRTU Deployment by Considering Cybersecurity in Secondary Distribution Network	15

2.1	Introduction	15
2.2	Strategic FRTU Deployment	19
2.2.1	Enhancement of System Observability	19
2.2.2	Anomaly Inference from Historical Load Profile	22
2.2.3	Prediction of Future Load Growth	30
2.3	Algorithmic Analysis	31
2.3.1	Training For Labeling	32
2.3.2	Incremental PDF Function Adjustment	37
2.3.3	FRTU Deployment Optimization	42
2.4	Simulation Results	44
2.5	Summary	52
3	Multi-Scale Variation-Aware Techniques for High Performance Digital	
	Microfluidic Lab-on-a-chip Component Placement¹	55
3.1	Introduction	56
3.2	Preliminaries	63
3.2.1	Problem Formulation	63
3.2.2	Integer Linear Programming (ILP) Formulation for Variation-Unaware Lab-on-a-chip Component Placement	65
3.3	Multi-Scale Optimization	70

¹©2011 IEEE. Reprinted, with permission, from Chen Liao and Shiyan Hu, “*Multiscale Variation-Aware Techniques for High-Performance Digital Microfluidic Lab-on-a-Chip Component Placement*”, IEEE Transactions on NanoBioscience, March 2011.

3.3.1	Grid Coarsening	71
3.3.2	Fine-Scale Tuning	72
3.4	Variation-Aware Placement Design	75
3.4.1	Variation-Aware Optimization	76
3.5	Simulation Results	78
3.5.1	Variation-Unaware Design	80
3.5.2	Variation-Aware Design	83
3.6	Summary	84

4 Physical-Level Synthesis For Digital Lab-on-a-chip Considering Variation,

	Contamination and Defect	87
4.1	Introduction	87
4.2	Preliminaries	93
4.2.1	Problem Formulation	93
4.2.2	Multi-Scale Technique For Placement Considering Variation	98
4.2.2.1	Grid Coarsening	99
4.2.2.2	Fine-Scale Tuning	99
4.3	Proposed Physical-Level Synthesis Technique Considering Variation	100
4.3.1	Maze Routing For Lab-on-a-chip Routing Considering Contamination and Defect	103
4.3.2	Physical-Level Synthesis Considering Variation	105
4.3.3	Computation of Routing Yield Using Latin Hypercube Sampling . .	110

4.4	Simulation Results	111
4.5	Summary	117
5	Approximation Scheme For Restricted Discrete Gate Sizing Targeting Delay	
	Minimization²	119
5.1	Introduction	119
5.2	Preliminaries	122
5.2.1	Notations and Definitions	122
5.2.2	Problem Formulation	127
5.3	The Discrete Gate Sizing Algorithm	129
5.3.1	Overview of the Algorithm	129
5.3.2	Oracle Construction	131
5.3.2.1	Level Based Dynamic Programming	131
5.3.2.2	Oracle Construction	136
5.3.3	The FPTAS	137
5.4	Summary	140
6	Conclusions	143
	References	147

²©Springer and the original publisher, Journal of Combinatorial Optimization, vol. 21, issue 4, 2009, pp. 497 - 510, “*Approximation Scheme For Restricted Discrete Gate Sizing Targeting Delay Minimization*”, Chen Liao and Shiyao Hu, original copyright notice is given to the publication in which the material was originally published, by adding; with kind permission from Springer Science and Business Media.

List of Figures

1.1	The structure of the distribution network with AMI.	5
1.2	The schematic of a lab-on-a-chip [1, 2, 3].	8
2.1	An FRTU example on the pole connected to a switching device.	16
2.2	Primary network feeder with FRTUs and distribution transformers in a realistic test case.	20
2.3	Load growth model.	21
2.4	Consumer electronic load profile from historical datasets.	24
2.5	An example of anomaly for the 9-th 10-minute usage data of the 5th day for load i	26
2.6	Example to determine ACI.	29
2.7	Flowchart for proposed cybersecurity planning framework.	33
2.8	An illustration of M matrix of proposed CRF model.	36
2.9	Illustration of two start points. The one starts from point 1 requiring additional iterations to than point 2.	38
2.10	The illustration of CE optimization.	42

2.11 A distribution network in stage 1 with optimization result of FRTU candidate locations.	47
2.12 A distribution network in stage 2 with predicted load growth, and optimization result of FRTU candidate locations.	48
2.13 The comparison of the ACI between the solution in Fig. 2.11 and the solution in Fig. 2.12.	50
2.14 The comparison of the number of FRTUs and the number of loads between the solution in Fig. 2.11 and the solution in Fig. 2.12.	50
2.15 An FRTU deployment solution in stage 1 using greedy algorithm.	51
3.1 An input example of the microfluidic lab-on-a-chip component placement problem [4].	59
3.2 Illustration of a lab-on-a-chip component placement.	61
3.3 An example to illustrate fine-scale tuning where 0,4,8 refer to the time. . .	72
3.4 A possible fine-scale tuning solution for 1-upward module set of Figure 3.3, where 0,4,8 refer to the time. Assume that all constraints are satisfied. . . .	75
3.5 Illustration of the overlap which is due to variations in a variation-unaware lab-on-a-chip component placement.	76
3.6 An example to illustrate the variation-aware multi-scale technique.	79
4.1 Illustration of lab-on-a-chip placement and routing.	90

4.2	An illustration of routing with timing constraint, considering contamination and defect. The routing needs to avoid the contaminated grid and defective grid. The grey route may violate the timing constraint since the routing length might be greater than t_{\max} , e.g., $t_{\max} = 10$	91
4.3	(a) An example of placement and routing. The lower left corner of a module can be placed at a grid. (b) The module could have variational height, and is touching with each other. Thus, there is no space for its routing, which leads to the failure of routing.	92
4.4	An example of non-routable placement solution for testcase vitro3-1 using [5]. The three grey modules are too close to each other, and thus the routing might fail.	95
4.5	(a) An example of grid coarsening of t -dimension, with $\kappa = 4$. For simplicity, the illustration is in 2-D, i.e., x and t coordinates. (b) A possible fine-scale tuning solution for (a). The total completion time is reduced. . . .	101
4.6	Two different fine-tuned solutions show that placement and routing interacts with each other during fine-tuning procedure. (a) An initial routing and placement solution. (b) One possible fine-tuned solution. (c) The other possible fine-tuned solution.	101
4.7	The flow chart of the proposed physical-level synthesis algorithm considering variation.	102

4.8	An illustration of 2-D maze routing (x and y coordinates). S denotes source, and D denotes destination. S and D could be lab-on-a-chip module 1 and lab-on-a-chip module 2. A black grid denotes a blocked/occupied grid. Each grid is with a unit weight except that the black grids are with a weight of g . It shows a possible minimum weight route from the source to the destination. It needs 8 grids to route from the source to the destination. . . .	104
4.9	An example to illustrate the defective grid and contaminated grid in 2-D, i.e., x and t coordinates.	106
4.10	Different module variational heights result in different physical-level synthesis samples of the fine-tuned solution based on initial solution in Figure 4.6(a). (a) A fine-scale tuning solution for Figure 4.6(a), and sample 1 of routing and placement. It satisfies the timing constraint and fluidic constraint. (b) Sample 2 violates the fluidic constraint. (c) Sample 3 has no space for routing. (d) Sample 4 satisfies the timing constraint and fluidic constraint.	107
4.11	The detailed flow chart of our front line based physical-level synthesis. . . .	109
4.12	An example to illustrate Latin Hypercube sampling. Green grids represent the probability density, blue grids represent the intervals with equal probability, and red points are Latin Hypercube samples.	111

4.13	Routing solution for vitro3-1. The droplets are transported from the sources to the destinations. A pair of source and destination, and the corresponding routing are painted by the same color.	114
(a)	One routing solution for vitro3-1 using ROUTE without V.	114
(b)	One routing solution for vitro3-1 using ROUTE with V.	114
5.1	Illustration of levels.	124
5.2	Illustration of gate delay d and arrival time t	126
5.3	An example of pruning where solution 2 is inferior to solution 1.	134

List of Tables

2.1	A sample of realistic dataset.	45
2.2	The comparison of the greedy algorithm and the proposed algorithm in stage 1.	52
3.1	Comparison of NEW and the previous work [3]. Timing refers to the overall completion time of the last module in the solution of the lab-on-a-chip component placement. CPU refers to the runtime in seconds. Timing reduction refers to the completion time improvement of NEW which is computed by comparing to the previous work [3].	80
3.2	The results of NEW w/ standard binary solution search.	82
3.3	The results of variation-aware optimizations. Timing increase is computed through comparing the completion time of NEW and NEW w/ variations. .	83

- 4.1 The comparison of ROUTE without V and ROUTE with V. Compl. time refers to the overall completion time of the last module in the solution of the lab-on-a-chip placement and routing. R. length is the total routing length which is the total number of grids needed for the transportation of droplets in this design. R. yield is routing yield. CPU refers to the runtime in seconds. Compl. time increase is computed through comparing the completion time of ROUTE without V and ROUTE with V. R. yield impr. is the routing yield improvement computed through comparing the routing yield of ROUTE without V and ROUTE with V. 115
- 4.2 The worst case design of ROUTE with V. R. length is the total routing length which is the total number of grids needed for the transportation of droplets in this design. R. yield is routing yield. 116
- 4.3 The comparison of ROUTE with V and ROUTE with V, D/C. # of D, C refers to the number of defective grid and contaminated grid. Compl. time refers to the overall completion time of the last module in the solution of the lab-on-a-chip placement and routing. R. length is the total routing length which is the total number of grids needed for the transportation of droplets in this design. R. yield is routing yield. Violation percentage refers to the number of defective/contaminated grids used in the routing over the total number of defective/contaminated grids. 117

Preface

This dissertation presents my research work in pursuing the Ph.D. degree in Electrical Engineering at Michigan Technological University. This dissertation includes previously published articles in Chapter 3, and Chapter 5. This dissertation includes an article which has been accepted in Chapter 2.

Chapter 2 contains an article has been accepted in IEEE Transactions on Smart Grid. As the first author, with the guidance of the second author Dr. Chee-Wooi Ten, I identified the problem formulation of FRTU installation considering cybersecurity for secondary distribution network. Under the guidance of my advisor Dr. Shiyang Hu (third author of this article), I proposed the algorithm and completed its implementation. This article was completed by me, Dr. Chee-Wooi Ten and Dr. Shiyang Hu.

Chapter 3 contains an article previously published in IEEE Transactions on NanoBioscience. As the first author, under the guidance of my advisor Dr. Shiyang Hu (second author of this article), I designed the algorithm flow for lab-on-a-chip component

placement. The coding part was also completed by me. This article was completed by me and Dr. Shiyan Hu.

Chapter 5 contains an article previously published in Journal of Combinatorial Optimization. As the first author, under the guidance of my advisor Dr. Shiyan Hu (second author of this article), I designed the fully polynomial time approximation scheme for restricted discrete gate sizing targeting delay minimization, and wrote the article. This article was completed by me and Dr. Shiyan Hu.

Acknowledgments

Until today, I have spent more than four years on my PhD study. While these days are with both happiness and pains, I think it is very worthwhile, and I appreciate all the persons who supported me and helped me. First and foremost, I would like to thank my advisor Dr. Shiyang Hu. He is a passionate researcher with lots of novel ideas. His encouragement and support have carried me through this study. It is impossible for me to finish my PhD study without his advice, patience, and enthusiasm.

I would like to thank Dr. Chee-Wooi Ten for his guidance and help for the project of FRTU installation. I would also like to thank all my committee members, Dr. Shiyang Hu, Dr. Chee-Wooi Ten, Dr. Zhuo Feng and Dr. Cliff Chin Ngai Sze for taking the time to review and critique my dissertation and provide me with very valuable feedback.

I want to thank my group members Xiaodan Chen, Jia Wang, Yujia Feng, Xueqian Zhao, Yonghe Guo for their help and supports. Thank my friends Ya Tian, Beini Jiang, Awadhesh Thakur, for their accompanying during those days.

Last but not least, it is not easy to express my appreciation using a single sentence to my family, but I want to thank my husband, Sheng Hu, and my parents, for the love and support in the past and future.

Abstract

Combinatorial optimization is a complex engineering subject. Although formulation often depends on the nature of problems that differs from their setup, design, constraints, and implications, establishing a unifying framework is essential. This dissertation investigates the unique features of three important optimization problems that can span from small-scale design automation to large-scale power system planning: (1) Feeder remote terminal unit (FRTU) planning strategy by considering the cybersecurity of secondary distribution network in electrical distribution grid, (2) physical-level synthesis for microfluidic lab-on-a-chip, and (3) discrete gate sizing in very-large-scale integration (VLSI) circuit.

First, an optimization technique by cross entropy is proposed to handle FRTU deployment in primary network considering cybersecurity of secondary distribution network. While it is constrained by monetary budget on the number of deployed FRTUs, the proposed algorithm identifies pivotal locations of a distribution feeder to install the FRTUs in different time horizons. Then, multi-scale optimization techniques are proposed for digital microfluidic lab-on-a-chip physical level synthesis. The proposed techniques handle the variation-aware lab-on-a-chip placement and routing co-design while satisfying all constraints, and considering contamination and defect. Last, the first fully polynomial time approximation scheme (FPTAS) is proposed for the delay driven discrete gate sizing

problem, which explores the theoretical view since the existing works are heuristics with no performance guarantee. The intellectual contribution of the proposed methods establishes a novel paradigm bridging the gaps between professional communities.

Chapter 1

Introduction

This dissertation investigates the features of three optimization problems, and proposes three strategic optimization techniques through CAD for them. The first problem is feeder remote terminal unit (FRTU) installation considering the security of secondary distribution network in smart grid, the second problem is physical-level synthesis for microfluidic lab-on-a-chip, and the third problem is discrete gate sizing in very large scale integrated (VLSI) circuit.

1.1 Feeder Remote Terminal Unit (FRTU) Installation

Considering Security of Secondary Distribution Network

With the fast development of power industry, the concerns on the energy crisis impose great challenges in the existing power system. This leads to the integration of the smart grid into the existing electrical grid system. Different from the traditional electrical grid, the smart grid allows customers or utilities to be actively involved in monitoring, controlling, and predicting the energy use. In other words, the smart grid is a modernized electricity network with the intelligent electricity device and it delivers electricity from suppliers to consumers using two-way digital technology to control appliances at customers' homes [6]. This intelligent network brings many benefits, such as saving energy, reducing cost and increasing reliability and transparency [7]. As a promising intelligence system, the smart grid is expected to be widely integrated with the traditional electrical grid in the future. There are a multitude of literatures on the smart grid. For example, [8] discusses challenges to the power system planning and operation of smart grid development, [9] mentions the evolution of smart grid and the impact on the electrical power industry, and in [10], it states the importance to define and develop the standards and protocols of smart grid for the electric power industry.

In spite of that the smart grid will bring enormous benefits, since more people can be highly involved in the communication and controlling, as well as access the network in the smart grid, it inevitably leads to a set of security and privacy concerns. The challenges arise with such factors as human behavior, commercial interests and regulatory policy and so on [6]. Some of them are similar to the traditional electrical grid, however, they are more complex to handle. The security issues are discussed in a large multitude of previous works. In [6], the description of four aspects of security issues is given, i.e., trust, communication and device security, privacy and security management. [11] compares the difference between IT network and the electrical network, and it also proposes an integrated security system to protect the smart grid against cyber attacks. [12] discusses cybersecurity in the smart grid. For example, in [13], an attack and defense modeling for the cybersecurity of the critical infrastructures is proposed. [14] proposes the vulnerability assessment of cybersecurity for supervisory control and data acquisition (SCADA) systems. [15] considers the integrated network security protocol layer for open-access power distribution systems. In [16], the security protocols against cyber attacks in the power system are discussed. [17] discusses the load altering attacks against smart power grids. Although there are a multitude of approaches for the design of the appropriate protocols and attack modeling in previous works, most of them are focused on the operation, not planning. In addition, one may note that they are mainly focused on the transmission network and primary distribution network. This is due to the lack of the real-time information from the secondary distribution network in the traditional electrical grid.

Figure 1.1 illustrates the structure of an important part of smart grid, which is a distribution network with advanced metering infrastructure (AMI). AMI is the customer service infrastructure between the customer and the distribution dispatching center (DDC). AMI consists of the customer data collection, the communication network, and DDC. It is an essential kernel part for implementing the smart grid, and makes it possible to realize the demand response based on the supply-demand mutual recognition [18]. [19] gives a brief introduction of smart integration, which includes AMI.

Since AMI demonstrates its importance, it has attracted a large amount of interests in the previous works. [18] introduces the status of AMI development in Korea, and [20] discusses the benefits for electricity grid operations and planning through a well planned AMI. [21] proposes a flexible and cost-effective AMI system for the deregulated electricity markets. [22] also considers the network optimization with AMI. In addition, [23] discusses the requirements and architectural directions intrusion detection targeting an AMI. [24] considers a few of security measures for AMI components. [25] discusses a set of constraints for smart electricity metering devices and AMI network when it considers the cybersecurity of AMI.

The essentially important part of AMI includes the feeder remote terminal units (FRTUs) in primary distribution network and the smart electricity metering devices in secondary distribution network. FRTUs can be utilized to supervise the readings from the customers' smart electricity metering devices in real-time. FRTUs can also be remotely controlled to

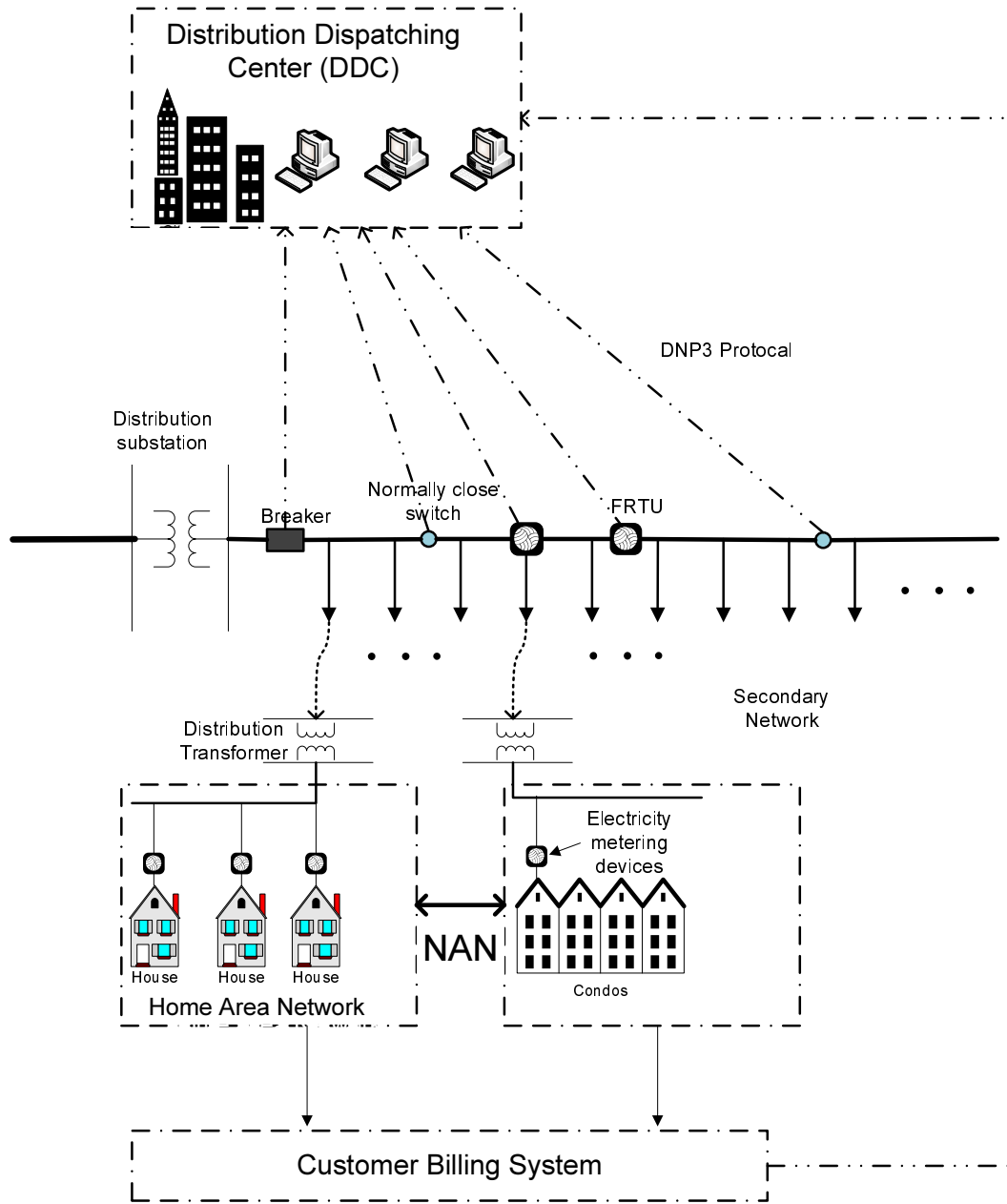


Figure 1.1: The structure of the distribution network with AMI.

perform some operation, such as open the switch. Therefore, this assists AMI to detect the attacks of the secondary distribution network and perform some appropriate operation to protect through FRTUs. However, as FRTU is a very expensive electronic device, and it is not practical to install FRTUs everywhere. The endpoint and network hardware which

includes FRTU installation takes the majority of the total AMI expense [26]. Thus, how to choose the appropriate locations to install FRTUs, such that the cost is minimized while the attacks can be effectively detected, becomes critical. On the other hand, due to the restriction of the economic budget, it is difficult to install FRTUs for all the networks in a short term. Thus, a long term investment of FRTU installation is necessary.

This dissertation proposes technique for FRTU installation considering the secondary distribution network security. With the assistance of FRTUs, AMI in the smart grid, can obtain more real-time information from the secondary distribution network. The security issues could be various, which are discussed in the previous literatures, such as [6]. In this work, it is mainly focused on the following security issue. Due to that it is convenient to revise the reading of the smart electricity metering device for the customers, there may be some stealing of electricity in the secondary distribution network or other illegal revision of the reading of the smart electricity metering device. The electric companies may want to detect this for both of their profit and the network security. Thus, how to install FRTUs in the primary network while effectively detecting the source of attacks from the secondary distribution network is considered in this dissertation. A cross entropy based technique to optimize the planning of FRTU installation in different time states is proposed, such that the economic spending is minimized while all the constraints are satisfied.

1.2 Physical-Level Synthesis of Microfluidic Lab-on-a-chip

The invention of microfluidic lab-on-a-chip provides great relief for the conventional biochemical procedure, which is often expensive and not very accurate. Microfluidic lab-on-a-chips have striking advantages that they perform multiple biochemical operations in a much cheaper way with higher sensitivity and accuracy [27, 1].

Microfluidic lab-on-a-chips have been applied in a variety of scientific research of biochemical analysis procedures such as DNA analysis [28] and proteomic analysis [29]. In addition, microfluidic lab-on-a-chip has initiated a revolution of human health related research including clinical diagnostics and drug discovery [30, 31]. For example, lab-on-a-chips have been used in clinical diagnostics on human physiological fluids [32].

The current commonly-used lab-on-a-chips, usually known as *digital microfluidic lab-on-a-chip*, is based on the manipulation of the discrete droplets which contain the biochemical samples. As shown in Figure 1.2, the droplets containing the biochemical samples are controlled by the electrohydrodynamic force which is generated from the programmed electrodes [1, 3, 2]. By this force, droplets for operations can be independently moved. The basic operations such as mixture and dilution can be performed at any place on the lab-on-a-chip. The *reconfigurability* allows different operations to share the same

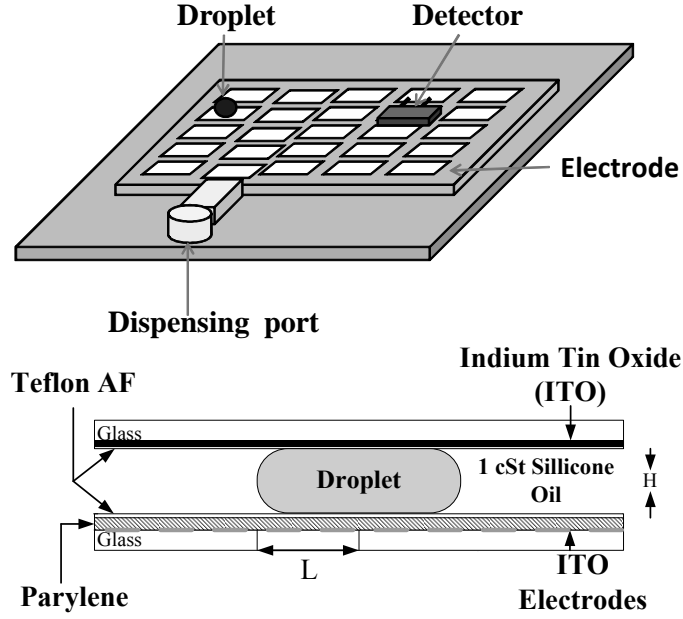


Figure 1.2: The schematic of a lab-on-a-chip [1, 2, 3].

place on the microfluidic array during different time intervals. With reconfigurability, each operation can be treated as a 3-D module whose size is determined by the space and duration the operation needs. A 3-D cell/module library can be then designed on biochemical operations [1]. With the cell/module library and design specification, CAD methodologies can be utilized to efficiently build a large-scale integrated microfluidic lab-on-a-chip.

In the lab-on-a-chip CAD flow, the lab-on-a-chip placement and routing are the key parts, which are called physical-level synthesis. Lab-on-a-chip placement is to determine the physical location and the starting time of each operation such that the overall completion time is minimized satisfying the constraints, e.g., non-overlapping constraint [5]. Lab-on-a-chip routing is to route the droplets from the source to the destination to ensure

the successful completion of the operations such that the routing distance is minimized while satisfying timing constraint and fluidic constraint. [33] proposes an algorithm based on simulated annealing and [3] improves it by applying a T-tree data structure. The lab-on-a-chip routing is not considered in some of previous works [3]. There are works showing that a good lab-on-a-chip router plays an important role in the CAD flow and techniques are proposed. [34] proposes a network flow based technique and [35] designs an integer linear programming based algorithm for lab-on-a-chip routing.

On the other hand, the biochemical reaction is sensitive to many variations [36, 34, 5]. For example, the temperature variations could lead to the operation completion time variation. [37] mentions that the biochemical operations have variability margins, which can impact the correctness of the biochemical application. [38] describes that the temperature environment is significant for the biochemical operations since some DNA would denature at inappropriate temperature. Thus, the biochemical operation completion time can have variation which necessitates the lab-on-a-chip routing considering variation. However, the traditional lab-on-a-chip placement and routing in previous works have not considered the variation. This dissertation proposes a multi-scale variation-aware optimization technique for the lab-on-a-chip component placement, and then seamlessly integrates the lab-on-a-chip routing to the placement while considering variations. The simulation results on the standard benchmark lab-on-a-chip designs demonstrate the effectiveness of the proposed techniques.

1.3 Approximation Scheme For Restricted Discrete Gate

Sizing Targeting Delay Minimization

The increasing chip density leads to the extensive use of gate sizing optimization in the combinational circuit design [39, 40, 41, 42, 43]. Gate sizing plays an important role in the very large scale integrated (VLSI) circuit design. Gate sizing has been proven to be one of the most effective approaches for power saving and delay minimization, which greatly affects the performance of the circuits. Therefore, effective algorithms for gate sizing are highly desirable to improve the design quality especially in terms of delay minimization and power saving. A large multitude of previous works with different objectives have been proposed. The standard gate sizing techniques for exploring delay and power tradeoff are proposed in [39, 40, 41, 43, 44, 45, 46, 47]. As the extensions to them, gate sizing techniques considering process variations are designed in [48, 49, 50], gate sizing techniques for cross-talk noise reduction are proposed in [51, 52, 53], a reliability driven gate sizing technique is proposed in [54], and a security aware gate sizing technique is proposed in [55].

However, most of the existing techniques such as a Lagrangian relaxation based technique in [40] and a posynomial programming based approach in [44] can only handle the continuous gate sizing problem which assumes that gate sizes can be any values within

certain range [56]. This assumption is not realistic since it is difficult and not practical to manufacture gates with continuous sizes. In practice, only a small set of gate sizes are available, which imposes a pressing need for the techniques to handle discrete gate sizes. Precisely, *the discrete gate sizing problem* is to assign a size to each gate from a given set of available gate sizes such that the circuit delay is minimized while the cost target is satisfied. This problem is known as strongly NP-hard [57]. To obtain a discrete gate sizing solution, rounding the sizes of a continuous solution to discrete sizes is fast and intuitive. However, it will result in the significant degradation of circuit delay compared to the obtained continuous gate sizing solution [56, 58]. This motivates some recent works to design combinatorial algorithms which directly handle discrete gate size, such as a continuous solution guided dynamic programming technique in [56], a network-flow based approach in [59], a parallelization and randomization based technique in [60], and a multi-dimensional gradient descent based algorithm in [61]. These algorithms are effective, however, they are all heuristics without any theoretical guarantee on the quality of their discrete gate sizing solutions. This limits the understanding of the discrete gate sizing problem in theory.

Given a minimization problem, an algorithm is said to approximate the optimal solution within a factor α if this algorithm can always produce a solution whose objective function value is at most α times the value of the optimal solution. The problem admits a fully polynomial time approximation scheme (FPTAS) if there is an algorithm which approximates the optimal solution within a factor of $(1 + \varepsilon)$ for any $\varepsilon > 0$ and runs in

time polynomial in both of the input size and $1/\varepsilon$.

This chapter aims to deepen the understanding of the discrete gate sizing problem from the theoretical point of view. It proposes the first fully polynomial time approximation scheme (FPTAS) is designed for the delay driven discrete gate sizing problem.

1.4 Publications

The publications of the author are listed as follows.

Chen Liao, Chee-Wooi Ten, Shiyan Hu, *Strategic FRTU Deployment Considering Cybersecurity in Secondary Distribution Network*, accepted to IEEE Transactions on Smart Grid, 2013.

Chen Liao, Jia Wang, Shiyan Hu, *The Power Distribution Network Expansion Planning Based on Stackelberg Minimum Weight K-Star Game*, accepted to Journal of Circuits, Systems, and Computers (JCSC), 2013.

Chen Liao and Shiyan Hu, “Discrete Wavelet Transform Based Key Sensitive Circuit Layout Fingerprinting Using Chaotic System”, Journal of Circuits, Systems and Computers (JCSC), vol. 7, no. 7, pp. 1250049-1 - 1250049-15, 2012.

Chen Liao and Shiyan Hu, “Multi-Scale Variation-Aware Techniques for High

Performance Digital Microfluidic Lab-on-a-chip Component Placement”, IEEE Transactions on NanoBioscience (TNB), vol. 10, issue 1, pp. 51 - 58, 2011.

Chen Liao and Shiyan Hu, “*Approximation Scheme for Restricted Discrete Gate Sizing Targeting Delay Minimization*”, Journal of Combinatorial Optimization, Springer, vol. 21, no. 4, pp. 497 - 510, 2011.

Jia Wang, Xiaodao Chen, **Chen Liao** and Shiyan Hu, “*The Approximation Scheme For Peak Power Driven Voltage Partitioning*”, in Proc. of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 736 - 741, 2011.

Xiaodao Chen, **Chen Liao**, Tongquan Wei and Shiyan Hu, “*An Interconnect Reliability-Driven Routing Technique For Electromigration Failure Avoidance*”, IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 5, pp. 770 - 776, 2010.

Chen Liao, Xiaodao Chen and Shiyan Hu, “*A High Performance Network Flow Based Technique for Timing-Driven Stress-Aware Chip Placement Perturbation*”, ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU Workshop), 2010.

Chen Liao and Shiyan Hu, “*Polynomial Time Approximation Schemes for Minimum Disk Cover Problems*”, Journal of Combinatorial Optimization, Springer, vol. 20, no. 4, pp. 399 - 412, 2010.

Chapter 2

Strategic FRTU Deployment by Considering Cybersecurity in Secondary Distribution Network

2.1 Introduction

The roles of labor-intensive utility workforce to periodically obtain energy consumption data from electromechanical meters have been supplemented by electronic intelligent devices [6]. The state-of-the-art cyberinfrastructure strengthens distribution modernization and overall system resilience. These devices enable bi-directional data transfer between



Figure 2.1: An FRTU example on the pole connected to a switching device.

consumer network and utility communication networks. Not only can consumers closely monitor and control their energy usages, but also can predict their energy trending based on their historical consumption and reduction of energy consumption for cost savings [7].

For decades, distribution expansion planning using computer-aided system for distribution substation and feeders has been studied with dynamic model and available technologies [62, 63, 64]. Multistage modeling for distribution planning was extensively integrated with secondary network and distributed generation [65, 66]. Recent development includes

deployment of advanced communication infrastructure for demand response as the nation's smart grid priority [18]. The design of the advanced metering infrastructure (AMI) communication backbone requires architectural innovation on intrusion detection [23]. There have been studies on cost-effective and optimization of AMI deployment, which can be constrained by the cyber-threats [21, 22, 25]. In recent years, research on economic and reliability analysis of distribution planning has been focused for the purpose of operational planning [67].

Cybersecurity of AMI system is an important issue that can impact the distribution grid [20, 19]. Despite the benefits of emerging IP-based communication infrastructure, there are challenging issues on system planning and operations [9]. Research includes vulnerability assessment [13], security protocol design [15], and cybertampering [17]. Establishing an integrated cyber-physical management framework is highly desirable in order to capture the system dynamics. This includes incorporating cybersecurity considerations on investment planning to identify pivotal nodes in primary network for validation purpose. The major issues of cybersecurity include communication protocol standardization, establishment of trustworthy network, enhancement of communication device security, and security management [10, 9, 12, 11]. There have been existing cybersecurity standards, ISO 27002 2005 (previously known as ISO IEC 17799 2005), NERC CIP-009-1 and ISA-99.02.01 [68, 69, 70], have extensively identified the required audit and improvements of critical cyber assets for power grid. The use of IP-based smart meters poses a risk of cybersecurity. The malicious intent of attackers on these smart meters

includes manipulating the metering data and archive. The first crime has been reported in 2009 that these meters are compromised [71]. Cyberattack scenarios compromising a smart meter are enumerated using attack tree [72].

Additional deployment of advanced sensors enhances system-wide observability. In primary distribution network, feeder remote terminal units (FRTUs) shown in Fig. 2.1 are important for operations [73]. FRTUs have been applied in distribution automation for fault detection, prediction, isolation, and service restoration. The FRTU also has a fault indicator transmitting discrete status to distribution dispatching center. The operator at the control center evaluates the subsystem of the distribution with other topology and identifies the fault zone [74, 75]. An FRTU-based strategy is proposed to determine the fault zone and isolate the areas separated by the boundary FRTUs [73, 76]. The communication security between FRTUs and distribution dispatching center has is suggested to establish with enhanced protocols against cyberattacks [16]. By utilizing the FRTU data measurements with AMI energy metering datasets, it can enhance the existing distribution management framework by addressing distribution grid cybersecurity. A recent development of phasor measurement units (PMU) in primary distribution network, by modernizing the grid from household to distributed generation, has been proposed to improve the distribution reliability [77].

The integration of cybersecurity management and investment planning for emerging distribution grid remain in the early stage. While the secondary network of distribution

system has been deployed with new cyberinfrastructure, modeling cybersecurity for operational planning has been nonexistent. The contribution of this work is the optimal FRTU deployment by considering the secondary distribution cybersecurity with budgetary minimization target. Multistage timeline to strategically deploy FRTUs is discussed in this paper. The remainder of this paper is organized as follows: Section 2.2 formulates the system model. Section 5.3 details the system model with an algorithmic analysis. Section 2.4 provides simulation results and section 5.4 concludes with discussions.

2.2 Strategic FRTU Deployment

Feeder remote terminal units (FRTUs), as part of the communication devices for the distribution primary network, monitor the field digital and analog measurements with remote control capability which is associated with the capacitor bank, line reclosure, line regulator, or remote controllable switch. These units have been deployed to interface with the distribution dispatching center for the purpose of monitoring and control. Due to the limited number of FRTU measurement points in the primary network, the initial estimation for all distribution transformers associated with an FRTU is estimated based on the FRTU measurements and distribution transformer ratings using allocation factor.

2.2.1 Enhancement of System Observability

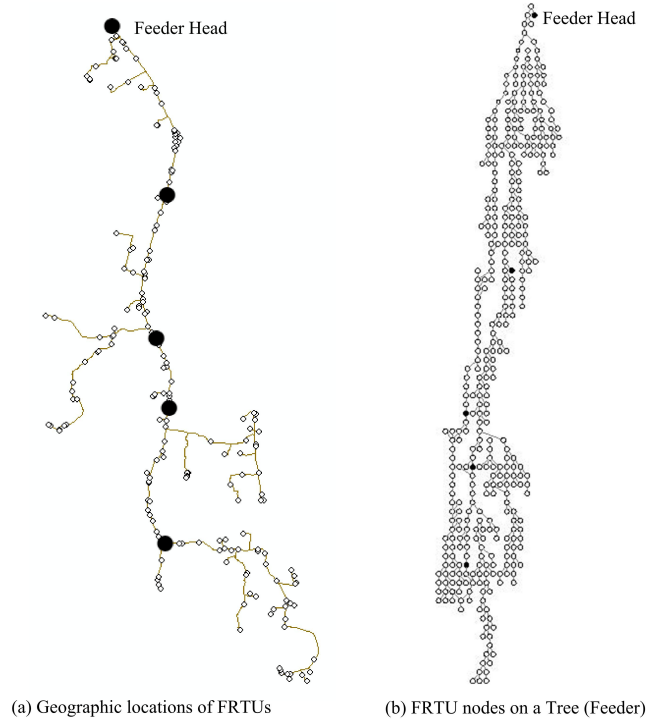


Figure 2.2: Primary network feeder with FRTUs and distribution transformers in a realistic test case.

Fig. 2.2 depicts the realistic setup of a distribution feeder with distribution transformers and FRTUs. Fig. 2.2 (a) is the geographical locations of a primary network. Fig. 2.2 (b) is the topology for the same test case. In both figures, the black points are the FRTUs and the circles are the distribution transformers in the primary network. The black nodes shown in Fig. 2.3 are the metering points that can be either FRTUs or a substation device which can provide real-time measurements. The nodes dividing the primary and secondary distribution networks are distribution transformers.

All leaf nodes are consumers' electronic meters. However, the measurement readings from

the leaf node (load) may not be with a true value as it could colorblackhave been tampered. These consistencies are categorized into 2 groups: (i) non-tampered anomaly and (ii) tampered anomaly. The first group refers to abrupt changes of energy usage, usually with additional use of home appliances, which can result in inconsistencies with the historical trending. The second one is altered metering information by malicious consumers that is inconsistent with the previous consumptions. Between the two, a false positive can occur when an anomaly detection returns a positive indication from a home metering device but the consumer is not tampered.

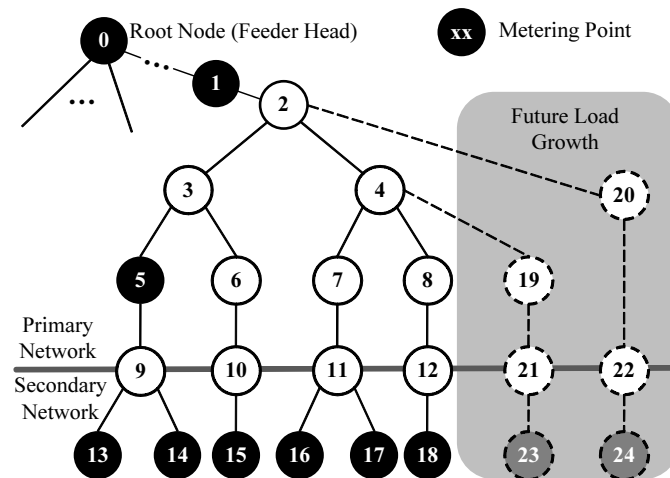


Figure 2.3: Load growth model.

A well-designed FRTU installation provides an effective solution. Since the FRTU can compare its reading and the reading of its children, *i.e.*, the loads, it determines possible irregularities if the readings do not match, and conversely, non-tampering, if matched. The FRTUs, if well deployed, serve as a trustworthy source that can be used to validate the real-time readings from all household energy meters. This section presents a system

model that illustrates the anomaly inference indices from historical load profile. It will also further discuss multistage FRTU deployment based on the deployment optimization by incorporating prediction of future load growth into the formulation.

2.2.2 Anomaly Inference from Historical Load Profile

Data packets of energy usages are transmitted to the consumer billing center approximately a few minutes each cycle, *e.g.*, 10 minutes or 15 minutes [78, 79]. A 10-minute cycle is utilized throughout this paper. A set of historical data from the billing center can be obtained for the purpose of system planning. Fig. 2.4 illustrates the average 10-minute usages from some historical datasets, which are in black color lines. This helps build the relational database for consumer billing center to obtain high resolution datasets that can be obtained over time. Based on the database, the anomalies of the given datasets can be defined through a density-based approach.

Since data acquisition of home metering system is archived every 10 minutes. A total number data points per day is $6 \text{ points / hour} \times 24 \text{ hours} = 144 \text{ points}$. This is quantified with respect to the number of observable day, d . The neighboring data points of q -th 10-minute usage of any day (*i.e.*, p -th) is referred to the same q -th 10-minute data point from day 1 to day d except p -th day, where $d \geq p$. To determine if a point is inconsistent with others, *e.g.*, the point at 12:00pm on Nov. 18th, comparison of this data point is made

with other points in other days at the same time, *i.e.*, the points at 12:00pm on all the other days except Nov. 18th. In general, the normal density datasets are similar to the density around its neighbors, while the anomaly density datasets are considerably different to other neighboring points.

Each household with the energy metering datasets is defined as a load i . Its q -th 10-minute usage data in ampere average of the p -th day is denoted by $\bar{I}_{p,i,q}$, which defines its reachability distance to another data \bar{I}_o denoted by [80]:

$$\phi_k(\bar{I}_{p,i,q}, \bar{I}_o) = \max\{\varphi_k(\bar{I}_{o'}), \psi_k(\bar{I}_{p,i,q}, \bar{I}_o)\}, \quad (2.1)$$

where the distance $\psi_k(\bar{I}_{p,i,q}, \bar{I}_o)$ is $|\bar{I}_{p,i,q} - \bar{I}_o|$, $\bar{I}_{o'}$ denotes the k -nearest neighbor of $\bar{I}_{p,i,q}$, k is a natural number, $\varphi_k(\bar{I}_{o'})$ is the k -distance of $\bar{I}_{p,i,q}$ such that at least for a set of k data $\bar{I}_{o''}$ satisfying that $\psi_k(\bar{I}_{p,i,q}, \bar{I}_{o''}) \leq \psi_k(\bar{I}_{p,i,q}, \bar{I}_{o'})$, and for at most a set of $k - 1$ data, $\psi_k(\bar{I}_{p,i,q}, \bar{I}_{o''}) < \psi_k(\bar{I}_{p,i,q}, \bar{I}_{o'})$. The set of k -nearest neighbors of $\bar{I}_{p,i,q}$ is denoted by $K(\bar{I}_{p,i,q})$. Based on the historical datasets, $K(\bar{I}_{p,i,q})$ can be obtained through the q -th 10-minute usage data of a few days other than the p -th day, or through the other 10-minute usage data in the p -th day. Note that 10-minute interval is an example to illustrate how it is analyzed within a time interval. The local reachability density μ_k of data $\bar{I}_{p,i,q}$ is denoted by

$$\mu_k(\bar{I}_{p,i,q}) = \left(\frac{\sum_{o \in K(\bar{I}_{p,i,q})} \phi_k(\bar{I}_{p,i,q}, \bar{I}_o)}{N_k(\bar{I}_{p,i,q})} \right)^{-1}, \quad (2.2)$$

where $N_k(\bar{I}_{p,i,q})$ is the number of data in $K(\bar{I}_{p,i,q})$, which may be larger than k since $\bar{I}_{o'}$ is unique for $\bar{I}_{p,i,q}$, while \bar{I}_o is not necessarily unique, and thus k -nearest neighbor of $\bar{I}_{p,i,q}$ may not be unique. A local anomaly factor of $\bar{I}_{p,i,q}$, denoted by $y_k(\bar{I}_{p,i,q})$, is defined as an anomaly score. It is the average ratio of local reachability density of the neighbors of $\bar{I}_{p,i,q}$.

$$y_k(\bar{I}_{p,i,q}) = \left(\frac{\sum_{o \in K(\bar{I}_{p,i,q})} \frac{\mu_k(\bar{I}_o)}{\mu_k(\bar{I}_{p,i,q})}}{N_k(\bar{I}_{p,i,q})} \right). \quad (2.3)$$

The dataset anomaly is determined when y_k is greater than 1.0. The local reachability density, $y_k(\bar{I}_{p,i,q})$, measures the difference between this data and its neighboring data. The value of data is typically similar to neighboring data, and thus the local anomaly factor is approximately 1.0. Otherwise, it is greater than 1.0.

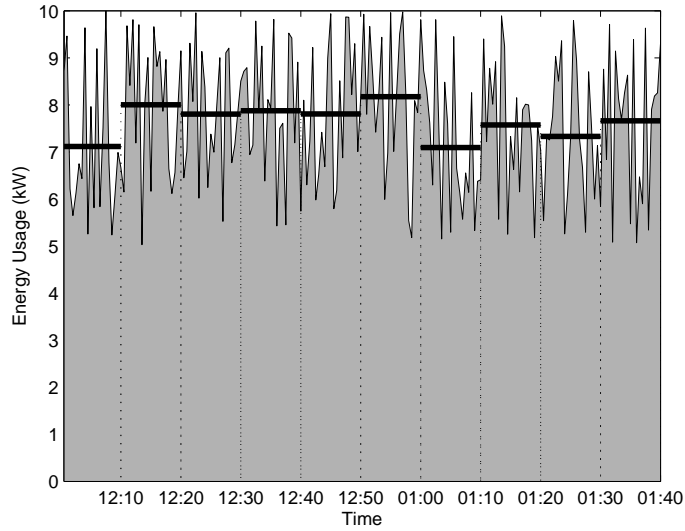


Figure 2.4: Consumer electronic load profile from historical datasets.

An example is provided in Fig. 3.3 where it reflects a real-world 10-day historical dataset in a summer for a load i from a smart meter reading. This example is arbitrarily modified with anomaly values. The 9-th 10-minute usage data of the 5th day, *i.e.*, $\bar{I}_{5,i,9}$, is 400.5A. The dataset can be determined if it is anomalous by finding all the 9-th 10-minute usage data of load i for the rest of days, *i.e.*, $\{\bar{I}_{1,i,9}, \dots, \bar{I}_{4,i,9}, \bar{I}_{6,i,9}, \dots, \bar{I}_{10,i,9}\}$. Assume that these data are {681.8, 760.9, 800.0, 734.5, 741.8, 755.4, 701.8, 732.7, 760.9} and $k = 5$. The 5-nearest neighboring data points for $\bar{I}_{5,i,9}$ is 741.8. The set of the 5 neighboring data points in $K(\bar{I}_{5,i,9})$ is {681.8, 701.8, 732.7, 734.5, 741.8}. The reachability distance to the neighboring data points in $K(\bar{I}_{5,i,9})$ is $|741.8 - 400.5| = 341.3$ using Eq. (2.1). Similarly, the reachability distances of the neighboring data points in $K(\bar{I}_{5,i,9})$, can be computed, *i.e.*, {341.3, 341.3, 341.3, 341.3, 341.3}. The local reachability density is $\mu_k(\bar{I}_{p,i,k}) = \left(\frac{341.3+341.3+341.3+341.3+341.3}{5}\right)^{-1} = 0.59 \times 10^{-3}$. One can compute the local reachability density of the neighboring data points in $K(\bar{I}_{5,i,9})$, *i.e.*, {.014, .019, .007, .008, .010}. Thus, the anomaly score $y_k(\bar{I}_{5,i,9})$ can be computed as, $y_k(\bar{I}_{5,i,9}) = \frac{0.014+0.019+0.007+0.008+0.010}{5 \times 0.59 \times 10^{-3}} = 19.66$, which is much greater than 1.0. This data is inferred as anomalous.

In the case of segregating weekdays and weekend energy consumption, it can be categorized into 2 groups in the similar way. Comparison only corresponds to each group of the two to ensure regularity of these datasets. If both cases of energy profiles are similar, then only a single analysis is required. For simplicity, a single case is interpreted in this study.

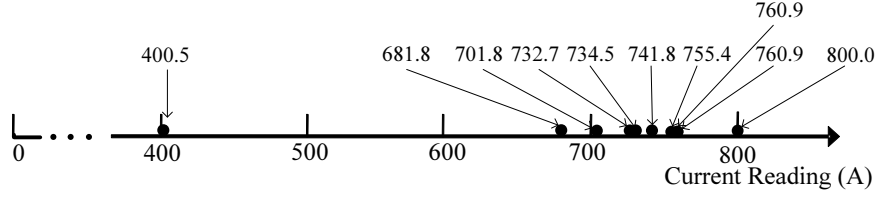


Figure 2.5: An example of anomaly for the 9-th 10-minute usage data of the 5th day for load i .

From Eq. (2.1) to Eq. (2.3), the anomalous dataset can be evaluated based on the historical data for each load can be identified. Fig. 3.3 illustrates an anomaly, *i.e.*, $y_k(I_{5,i,9})$ that can be calculated from the data $I_{5,i,9} = 19.66$. Similarly, a total number of times of anomaly in a year for any load can be computed this way.

An anomaly trending associated with a consumer is defined as follows:

$$\Lambda(\bar{I}_{p,i,q}) = \begin{cases} 1, & \text{if } y_k(\bar{I}_{p,i,q}) \gg 1.0, \\ 0, & \text{otherwise.} \end{cases} \quad \forall p, i, q. \quad (2.4)$$

The n_i is the total number of the historical anomaly readings with 10-minute cycle for node i in one year, *i.e.*,

$$n_i = \sum \Lambda(\bar{I}_{p,i,q}), \forall p, q. \quad (2.5)$$

The probability of anomaly for load i , denoted by P_i , can be estimated based on the frequency of anomaly occurrence over the total number of daily energy usage data as

follows:

$$P_i = \frac{n_i}{N_i}, \quad (2.6)$$

where N_i is the total number of all historical readings with 10-minute cycle for node i in one year. For the future loads, since there is no historical data, the probability of anomaly is set to 50%, by default.

Conceptually, the summation of current for all consumers with energy meters is close to the value of the root node of FRTU average current measurements. Due to budgetary constraints, each distribution transformer cannot be ideally assigned with one FRTU for cross validation. Fig. 2.3 shows an example of FRTU parent node with child nodes as the distribution transformers with current readings. The loads in the gray area is the possible future load growth. The current difference of the node 1 of p -th day, the q -th 10-minute cycle is $\Delta \bar{I}_{p,1,q} = \bar{I}_{p,1,q} - (\bar{I}_{p,5,q} + \bar{I}_{p,10,q} + \bar{I}_{p,16,q} + \bar{I}_{p,17,q} + \bar{I}_{p,18,q})$. Note that $\bar{I}_{p,5,q}$ is the FRTU node that provides current information of its child nodes, *i.e.*, $\bar{I}_{p,13,q}$ and $\bar{I}_{p,14,q}$. The measurements of these two child nodes are substituted with the FRTU parent node to evaluate the current. These historical datasets of FRTU current measurements are archived in a centralized database to cross-check the anomalous measurement readings from consumers.

The datasets are used to determine the pivotal FRTU location(s) for future deployment. The following Anomaly Coverage Index (ACI) is defined here as the system constraint to

determine the ratio between anomalous consumers under an FRTU monitoring and total anomalous consumers:

$$ACI = \frac{1}{O} \sum_{k=1}^O \left(\frac{(\sum_{j=1}^M \dot{P}_{j,k} | (P_{j,k} > p^*) \cap (\eta \leq \eta^*))}{\sum_{i=1}^N (\tilde{P}_{i,k} | P_{i,k} > p^*)} \right). \quad (2.7)$$

The M is a set of consumer nodes with each of their anomaly probabilities $P_{j,k} > p^*$ (with $p^* = 0$ by default) under a candidate FRTU in which the downstream of its location shall not connect with more than η^* consumer(s). The N is the total number of anomalous consumers under one feeder with $P_{i,k} > p^*$, where $M \leq N$ and $\dot{\mathbf{P}}, \tilde{\mathbf{P}} \subset \mathbf{P}$. The O is the total number of scenarios with different anomaly probabilities. The ACI^* , p^* , and η^* are user-defined constraints. The ACI constraint is, when there is some anomaly in the network, the FRTU solution is able to narrow down the location, with satisfying user-defined ACI , to at most η^* customers. Fig. 2.6 shows an example to determine ACI with $\eta^* = 3$. Nodes 3 and 4 are FRTU candidate locations. The total anomalous consumers are $N = 3$, *i.e.*, loads 13, 16, and 18. The FRTU at node 3 in the example connects $\eta = 3$ consumers, and $\eta \leq \eta^*$ for all the malicious consumer load 13. While the FRTU at node 4 connects $\eta = 4$ consumers, and $\eta > \eta^*$ for the anomalous consumer loads 16 and 18. Thus, $M = 1$.

Assuming that $O = 2$, the ACI for the network configuration is computed as:

$$\begin{aligned}
 \text{ACI} &= \frac{1}{2} \sum_{k=1}^2 \left(\frac{\sum_{j=1}^1 (\dot{P}_{j,k} | (P_{j,k} > 0) \cap (\eta \leq 3))}{\sum_{i=1}^3 (\tilde{P}_{i,k} | P_{i,k} > 0)} \right) \\
 &= \frac{1}{2} \left(\frac{(.345)}{(.345 + .132 + .537)} + \frac{(.345)}{(.345 + .132 + .537)} \right) \\
 &= 34\%.
 \end{aligned}$$

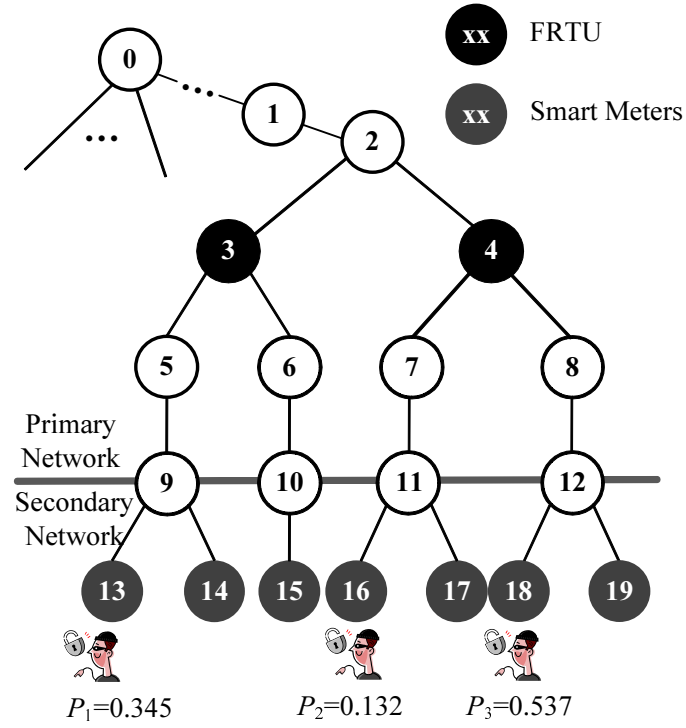


Figure 2.6: Example to determine ACI.

2.2.3 Prediction of Future Load Growth

Load growth models for the distribution grid are established and discussed in [81, 82]. In this problem formulation, future load growth is a key factor for network planning, which might involve additional loads within a feeder [82]. These additions might also have new branches connecting to the existing network in the future. Due to the properties of the load growth, *i.e.*, growth rate as the function of time, Gompertz function is utilized [83]. Generally, the load growth model based on Gompertz function is developed in planning as a function of time t [83].

$$S(t) = G \cdot e^{-be^{ct}}. \quad (2.8)$$

where the three parameters G, b, c allow considerable flexibility of data fit, G is the upper asymptote of rating capacity (kVA) for distribution transformer under each feeder, and $e = 2.718$ [83]. The Gompertz function $S(t)$ estimates the amount of load growth with the time t in kVA.

These parameters are set accordingly based on the historical datasets. Each feeder consists of an existing load set associated with a Gompertz function, *e.g.*, G is set to the maximum future loads without overload violations. The c is the sum of anomaly score of all the

historical datasets, *i.e.*,

$$c = \sum y_k(\bar{I}_{p,i,q}), \forall p, i, q, \quad (2.9)$$

which indicates that the larger anomaly score may be further investigated, as a consequence, slower rate of future growth. The b is set to be the current existing load which can be obtained from the historical datasets. It implies that the populated areas of existing loads may have less opportunities for future load growth. Fig. 2.3 depicts an example of load growth, and b is the current existing load. For example, if there currently are 10 loads in shaded area, then b is set to 10. The c is the sum of anomaly score of the historical data of the existing loads.

2.3 Algorithmic Analysis

The proposed algorithm is based on machine learning techniques. First, it requires the up-to-date input data of distribution network datasets and historical trending from household energy meters. The network topology and the historical trending are then utilized to train for labeling nodes in the primary network using conditional random field (CRF). During the process of training, each node is labeled with either Y or N . The node with label Y is associated with a larger mean of its probability density function (PDF) for the FRTU candidate location. Otherwise, node with label N is associated

with a smaller mean value. It is then followed by cross entropy (CE) method. This is an iterative process and each iteration is generated with multiple larger numbers in order to form different samples. For each sample, the selected candidate nodes for FRTUs are evaluated to ensure the constraints of the optimization formulation is within the limit, while the historical trending anomaly datasets are considered and verified if all constraints are satisfied. However, if the convergence criterion is not satisfied, the best 10% sampling datasets satisfying the constraints are selected. These mean values of the PDF functions on those nodes are updated incrementally. The module outputs the best performance samples which indicate the best FRTU locations. Fig. 2.7 shows a high-level abstraction of the proposed methodology which includes three major modules: (A) *Training for labeling*, (B) *Incremental PDF function adjustment*, and (C) *FRTU deployment optimization*.

2.3.1 Training For Labeling

Due to the size of primary distribution network to avoid total enumeration for all nodes, conditional random field (CRF) is utilized to initialize the start point for training. It models the correspondence between the neighboring parent nodes and the child nodes. Each node is assigned with a label Y or N . The Y suggests an FRTU installed, and N suggests no FRTU installed. Thus, the node with label Y has more chance to be added an FRTU, while the node with label N has less chance.

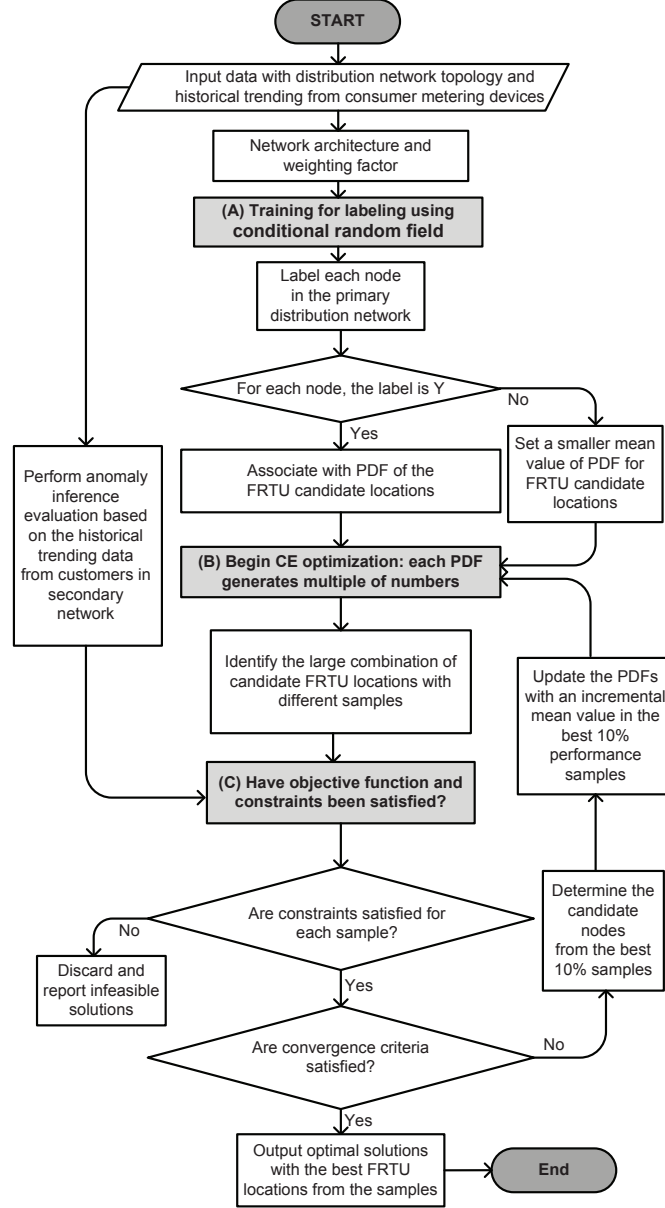


Figure 2.7: Flowchart for proposed cybersecurity planning framework.

The following part gives the CRF model, and the procedure of training to get corresponding labels, which are based on the probability of the label sequence, as shown in Fig. 2.8. For each pair of parent node and child node, there will be different combinations for the label assignment. Each combination is called a label sequence. Denote by Φ_j the j -th feature

function, denote by l_i the label of node i , by L a level index, and by \mathbf{o} the observation. The L is defined by the depth level from the bottom of the distribution feeder where the child nodes indicate $L = 1$, and the immediate parent nodes of $L = 1$ are in level 2 with $L = 2$. The level L is increasing until it reaches to the root node of the distribution feeder. The observation is the input, representing the observed knowledge, which is the level index of the nodes in this problem. The feature function captures the common features for training to determine candidate locations of deploying FRTU in a feeder. An example of the feature function is as follows.

$$\Phi_j(l_{i-1}, l_i | \mathbf{o}) = \begin{cases} 1, & \text{if } l_{i-1} = Y, l_i = N, \text{ level} = L, \\ 0, & \text{otherwise,} \end{cases} \quad (2.10)$$

The ρ denotes the weighting factor. The CRF probability is estimated through the enumerative training with a set of $n + 1$ matrices, denoted by $M_i(o)$, $i = 1, 2, \dots, n + 1$. Again, each is labeled with either a Y or N . Matrices $M_i(o)$ are defined as all different combinations of label assignment for a pair of parent node and child node, with given level index. The labels on path with largest weighting factor are used as the output labels. One has,

$$M_i(l_{i-1}, l_i | \mathbf{o}) = \exp \left(\sum_j \rho_j \Phi_j(l_{i-1}, l_i, \mathbf{o}), i \right), \quad (2.11)$$

An example is shown in Fig. 2.8. To make the graph completed and the expression simplified, a start point and an end point are added, which are l_0 and l_3 respectively. It has the matrix for the different combinations of labels along path, *e.g.*, the path along grey line, or the path along grey dashed line.

One can see that from the start point to the first label, $M_1(\mathbf{o})$ has only two elements which indicate two possible paths, *i.e.*, from start to the first Y , and from start to the first N . Thus, $M_1(\mathbf{o})$ has two elements as follows. Similarly, $M_2(\mathbf{o})$ has four elements since it has four possible paths, and $M_3(\mathbf{o})$ has two elements.

$$M_1(\mathbf{o}) = \begin{pmatrix} M_1(l_0, Y|\mathbf{o}) & M_1(l_0, N|\mathbf{o}) \end{pmatrix}, \quad (2.12)$$

$$M_2(\mathbf{o}) = \begin{pmatrix} M_2(Y, Y|\mathbf{o}) & M_2(Y, N|\mathbf{o}) \\ M_2(N, Y|\mathbf{o}) & M_2(N, N|\mathbf{o}) \end{pmatrix}. \quad (2.13)$$

For example, one possible path from l_1 to l_2 for M_2 is

$$M_2(l_1 = N, l_2 = Y|\mathbf{o}) = \exp \left(\sum_j \rho_j \Phi_j(N, Y|\mathbf{o}) \right). \quad (2.14)$$

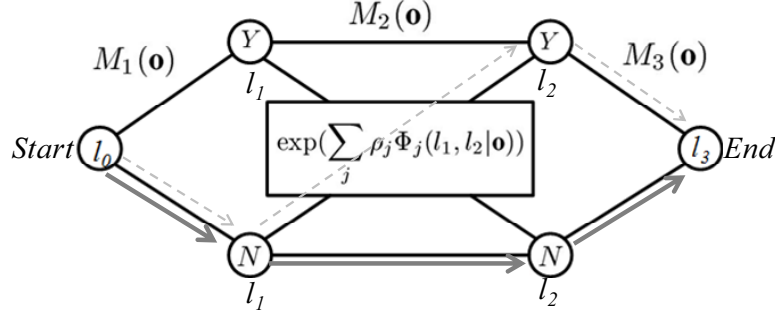


Figure 2.8: An illustration of M matrix of proposed CRF model.

Denote by $Z(\mathbf{o})$ the normalization factor. Denote by $Pl(\mathbf{l}|\mathbf{o}, \rho)$ the CRF probability corresponding to one path from the start to end, and as a result,

$$Pl(\mathbf{l}|\mathbf{o}, \rho) = \frac{1}{Z(\mathbf{o})} \prod_{i=1}^3 M_i(l_{i-1}, l_i | \mathbf{o}), \quad (2.15)$$

The proposed model, representing the correspondence between the parent node and child node, is utilized to optimize the weighting factor ρ based on the solution of the IEEE 13-bus system [84]. Quasi-Newton method can be used to compute the above parameters to train the CRF model. This provides the form of the probability of a label sequence, denoted by \mathbf{l} , given the observation sequence, denoted by \mathbf{o} , which is equal to a normalized product of potential functions, where $\mathbf{l} = \{l_1, l_2, \dots, l_n\}$ [85]. The labels of each node can be thus obtained.

2.3.2 Incremental PDF Function Adjustment

The proposed function generator is to adjust the PDFs incrementally using Gaussian distribution. A random variable is a function associating a unique numerical value with every outcome of an experiment and the value of the random variable may vary from iteration to iteration when the experiment is repeated. The density of a continuous random variable is a probability density function (PDF). Each node in stage m is associated with an initial PDF. A set of PDFs are generated that determine the tentative location for FRTU deployment with corresponding cost. If the convergence criterion is not satisfied, the PDFs of the nodes associated with the minimum cost is updated with a larger mean value. The nodes are selected from the best 10% samples replacing with a larger mean value. This procedure is iterated until the convergence criterion is satisfied. If the system already has some existing FRTUs, the probability of these nodes is set to 1, which means these nodes are selected all the time.

In CE optimization, each PDF requires a start point for its mean μ and the variance. If there is a good start point, the optimization can converge efficiently. Otherwise, the optimization might converge slowly. As illustrated in Fig. 2.9, if the initial mean μ is at start point 2, it may converge to achieve optimum at first iteration. However, if it starts at point 1, it may converge much slower to achieve its optimality, *i.e.*, it shows 4 iterations. Thus, a good start point for each PDF in CE is essential. In this paper, the CRF label provides a good

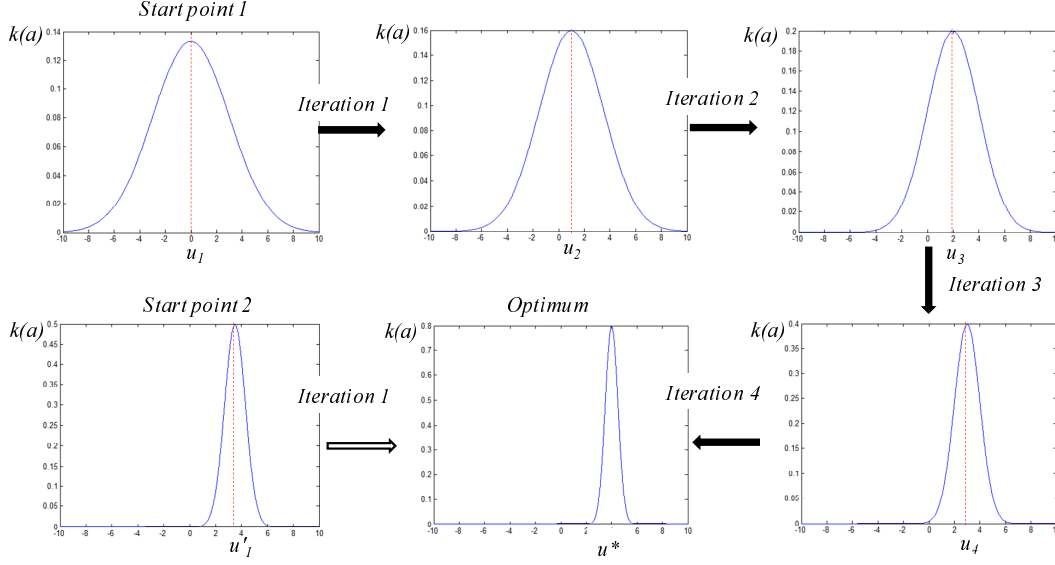


Figure 2.9: Illustration of two start points. The one starts from point 1 requiring additional iterations to than point 2.

basis to choose the start point for CE method. That is, the solution from (A) in Fig. 2.7 is utilized as a start point for the mean values of these PDFs. If a node is assigned with label Y by CRF, its initial PDF will have large mean and small variance in CE. If it is assigned with label N by CRF, its initial PDF will have small mean and large variance. This makes the approach more efficient to reach convergence value.

The $f(a)$ below is the minimization target function, where the random variable a is in a space \mathbb{A} , in which a family of probability density functions (PDFs) is distributed.

$$f^* = \min f(a), a \in \mathbb{A}. \quad (2.16)$$

The associated functions for Eq. 2.16 are denoted by $Pr(a, u)$, where u is a parameter.

With a different u , it leads to different PDF. To minimize $f(a)$, the proposed CE

method approximates the optimal solution for f such that $f(a) \leq f$ becomes a rare event. An indicator function is denoted by $Q(\cdot)$, which assists converting from the original optimization target to an optimization of a rare event function. The $Q(f(a) \leq f)$ is equal to 1 when $f(a) \leq f$. Otherwise, it is equal to 0. Using the PDF with u , a set of δ samples are generated from a , denoted by $A = a_1, a_2, \dots, a_\delta$. The $E_u(\cdot)$ is denoted as an expectation associated with $Pr(a, u)$.

The $\theta(f)$ is defined with the expectation of $Q(f(A) \leq f)$.

$$\theta(f) = E_u \left(Q(f(A) \leq f) \right), \quad (2.17)$$

Given a u , Monte Carlo method is utilized to compute θ , which first generates a large number of samples, *e.g.*, δ samples, using the PDF with the given u . For each sample, it computes $Q(f(a) \leq f)$, with the average value [86].

$$\tilde{\theta}(f) = \frac{1}{\delta} \sum_{i=1}^{\delta} Q(f(a_i) \leq f). \quad (2.18)$$

The original optimization target, $\min f(a)$, can be converted to the optimization of a stochastic, or a rare event function. That is, to maximize f such that $\theta(f)$ is approaching 0. One can see that when $\theta(f)$ approaches 0, $f(a)$ is greater than f with extremely high

possibility. The maximized f consequently leads to the minimized $f(a)$.

Since the parameter u is unknown, it is necessary to compute an appropriate value for it. As a result, it may lead to a good start point to compute θ in Eq. (2.18). An intuitive way is to search the appropriate u in a large range iteratively. However, the range can be large and it may be challenging to search it efficiently. In addition, when $\theta(f)$ approaches 0, $f(a) \leq f$ becomes a rare event, this would force Monte Carlo method to use a large number of samples for lower efficiencies. The importance sampling can be utilized to approximate u^* with a \tilde{u} for a set of samples using the PDF with \tilde{u} . This iterates until the pre-defined convergence criterion is satisfied, *e.g.*, $\theta(f) = 0.1$. This approximates for the targeted optimization.

The algorithm proceeds iteratively [86]. In our problem, each node is associated with a PDF. Each PDF has an initial mean u , and it generates some variable a . The variable a in PDF is usually continuous random variable. Thus, it takes the following technique to make it binary. If $a < 1 - u$, $a = 0$. Otherwise, $a = 1$. In this fashion, these variables form a randomly generated input sample; in other words, it provides a possible FRTU installation solution. A set of such samples are evaluated, and the good performance sample is used to incrementally update the corresponding PDFs. That is, its mean u will gradually increase at the same time the variance will gradually decrease. The PDFs will continue to generate another round of variables as the input samples. Note that the above procedure is iterative process, *i.e.*, the node with updated PDF makes this node with a higher value closer to the

variable $a = 1$ in the next iterations. Updating PDFs based on previous good performance samples provides better samples in next iteration. Finally the convergence criterion will be satisfied.

If the convergence criterion is not satisfied, the top performance samples are utilized to update the parameters in PDF. During each iteration, f is also concurrently updated for an approximation f^* . That is, f is set to the value of $f(a)$ corresponding to the top performance sample. In this case, f is updated for each iteration to the current minimum number of FRTUs which can satisfy all the constraints. The above procedures are iterated until the convergence criterion is satisfied. The module outputs the best performance samples which indicate the best FRTU locations. Note that if the system already has some existing FRTUs, the probability of these nodes is set to 1, which means that these nodes are selected all the time. The example in Fig. 2.10 illustrates the procedure. Suppose that there are two nodes, and corresponding variables are a_1, a_2 , the means are u_1, u_2 . The grey paraboloid is the PDF for the two variables. Since there are two variables, it illustrates in 2 dimensions. The grey points are generated variables which correspond to a set of randomly generated input samples, *i.e.*, if $a < 1 - u$, $a = 0$, otherwise $a = 1$, and 1 is to propose an FRTU installation while 0 is otherwise. The black dot represents the top performance solution. Then the PDF is updated according to the black point with u increasing and variance decreasing. This procedure is iterated until it reaches convergence criterion.

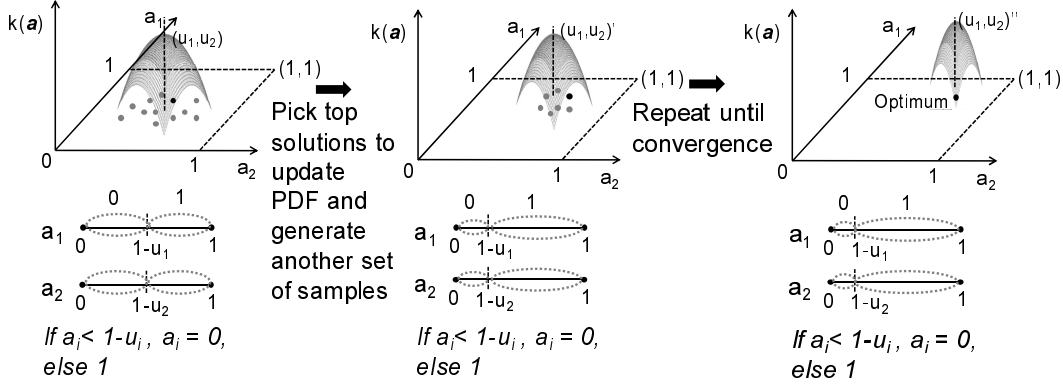


Figure 2.10: The illustration of CE optimization.

2.3.3 FRTU Deployment Optimization

Considering the anomaly indices from secondary distribution network, the optimal investment in multistage FRTU deployment is formulated as follows: The total investment cost is defined by $C(\cdot)$. For each node i in the primary network, a binary variable x_i is associated with it. Node with $x_i = 1$, indicates FRTU node or otherwise without. The entire period of timeline denotes T is the m stages that include $\{t_1, t_2, t_3, \dots, t_m\}$. Each stage has 6 months. Denote by n_j the total number of loads in stage t_j . Define $X_j = \{x_1, x_2, \dots, x_{n_j}\}$, where $1 \leq j \leq m$. Note that different time stage may have different node number due to the load growth.

The objective function of optimization is to minimize the total costs $C(X_T)$ for the entire distribution network in a given T period (with multiple m stages) while satisfying ACI

constraint.

$$\min \sum_{j=1}^m \left(\alpha C_{t,j}^{\mathbf{i}}(X_j) + \beta C_{t,j}^{\mathbf{o}}(X_j) \right) \quad (2.19)$$

where α and β are the factors related to the costs of infrastructure investment and operations defined in Eqs. (2.20) and (2.21). The $C_{t,j}^{\mathbf{i}}(X_j)$ is the investment cost of the stage t_j , and $C_{t,j}^{\mathbf{o}}(X_j)$ is the operation cost of the stage t_j , defined as Eqs. (2.22) and (2.23).

$$\alpha = (1 + r_{t,j}^{\mathbf{i}}) \times t_j^{\mathbf{i}}, \quad (2.20)$$

$j \in$ stages having investment, and

$$\beta = (1 + r_{t,j}^{\mathbf{o}}) \times t_j^{\mathbf{o}}, \quad (2.21)$$

$j \in$ stages having operations, where $r_{t,j}$ denotes the rate of interest in the stage t_j .

$$C_{t,j}^{\mathbf{i}}(X_j) = \sum_{i=1}^{n_j} C_j^{\mathbf{i}} x_i, \quad (2.22)$$

where n_j is the total number of the nodes (including the original nodes and new added

nodes) in stage t_j . The C_j^i is the investment cost for the nodes in stage t_j that includes the costs for infrastructure deployment, survey, and new network additions.

$$C_{t,j}^o(X_j) = \sum_{i=1}^{n_j} C_j^o x_i, \quad (2.23)$$

where C_j^o is the operating cost for the nodes in stage t_j , including the maintenance cost, repairing cost, the estimated power loss cost, and the operations cost. This work only considers multistage FRTU deployment based on existing network topology with predictable load growth in some areas. Only cost incurred on the efforts to deploy FRTU devices is considered.

2.4 Simulation Results

In this simulation setup, the proposed method is implemented in MATLAB and tested on a personal computer with 1.86GHz CPU and 3GB main memory. Fig. 2.11 illustrates the simulation test case of a distribution network including the current loads. This consists of 51 nodes in primary distribution network. The secondary distribution network is designed with 27 nodes in the first stage, *i.e.*, when $m = 1$ in Eq. (2.19).

The sample of this simulation is set up based on the realistic datasets from utilities given in Table 2.1. Irregular values are arbitrarily modified from the dataset for anomaly evaluation.

In stage 1, each load is with probability of anomaly in Eq. (2.6), as follows. \mathbf{P}_1 of nodes {52, 59, 60, 61, 64, 65, 71, 76, 78} = 0.6, \mathbf{P}_2 of nodes {53, 72} = 0.55, \mathbf{P}_3 of nodes {54, 55, 56, 57, 62, 63, 75, 77} = 0.1, \mathbf{P}_4 of nodes {66, 67, 69, 70} = 0.05, and the remaining nodes are with 0 probability.

All the predicted future loads in stage 2 are assigned with 0.5 probability of anomaly. The α and β from Eq. (2.19) are both set to 1.5, *i.e.*, $\alpha + \beta = 3.0$. The investment cost C_j^i for all stages is set to \$4000 per FRTU, and operating cost C_j^o for all stages is set to \$2000 per FRTU. Thus, the total cost for each FRTU is \$9000.

Table 2.1
A sample of realistic dataset.

Timestamp	Current Phase A	Current Phase B	Current Phase C
2:00PM 2/20/12 EST	177.3A	252.3A	192.2A
2:10PM 2/20/12 EST	206.4A	268.3A	221.6A
2:20PM 2/20/12 EST	214.2A	280.0A	240.8A
2:30PM 2/20/12 EST	198.8A	264.8A	208.8A
2:40PM 2/20/12 EST	201.9A	261.4A	215.6A
2:50PM 2/20/12 EST	218.0A	267.9A	206.0A
3:00PM 2/20/12 EST	199.4A	255.7A	184.9A
3:10PM 2/20/12 EST	206.8A	262.2A	198.6A
3:20PM 2/20/12 EST	214.9A	275.9A	204.0A
3:30PM 2/20/12 EST	202.8A	261.1A	199.4A

The proposed method using conditional random field (CRF) is utilized here to assign each node of a distribution feeder with a label Y or N . The node with label Y is associated with a larger mean of its probability density function (PDF) for the FRTU candidate location. Otherwise, node with label N is associated with a smaller mean value. It

is then followed by cross entropy (CE) method. This is an iterative process and every iteration is generated with multiple larger numbers in order to form different samples. For each sample, the selected nodes are evaluated to see when there is some anomaly, whether the FRTUs can narrow down the location, with user-defined ACI, to at most η consumers. If the convergence criteria are not satisfied, the best 10% sampling datasets satisfying the constraints are selected. These mean values of the PDF functions on those nodes are updated incrementally. For each iteration, f is also concurrently updated for an approximation f^* . That is, f is set to the value of $f(a)$ corresponding to the top performance sample. After each iteration, f is updated to the current minimum number of FRTUs which can satisfy all the constraints. The module outputs the best performance samples which indicate the best FRTU locations. In this simulation, η is set to 4, and ACI is set to 0.9. The convergence criteria is set to $\theta(f) = 0.1$. In this simulation example, functions from Eqs. (2.16) and (2.18) are used. Eq. (2.16) gives the minimization target and the current best value. Eq. (2.18) computes the expectation of $Q(f(A) \leq f)$, and it also corresponds to the convergence criteria $\theta(f) = 0.1$, which indicates that the current solution approaches optimal solution.

The proposed approach is tested using the stage 1 test case, which is illustrated in Fig. 2.11. There are a total of additional 8 FRTUs in this solution, which cover all the loads with more than 0.5 anomaly probability. To determine the candidate locations of FRTU deployment, a set of high probability potentially tampered by malicious consumers on each load is generated. The simulation result demonstrates it narrows down the locations to less than

or equal to 4 consumers' electronic meters with ACI of 0.9. It can thus determine if this scenario can be possibly cybertampered or false positive cases. The minimum cost is $8 \times \$9000 = \72000 .

Figure 2.11: A distribution network in stage 1 with optimization result of FRTU candidate locations.

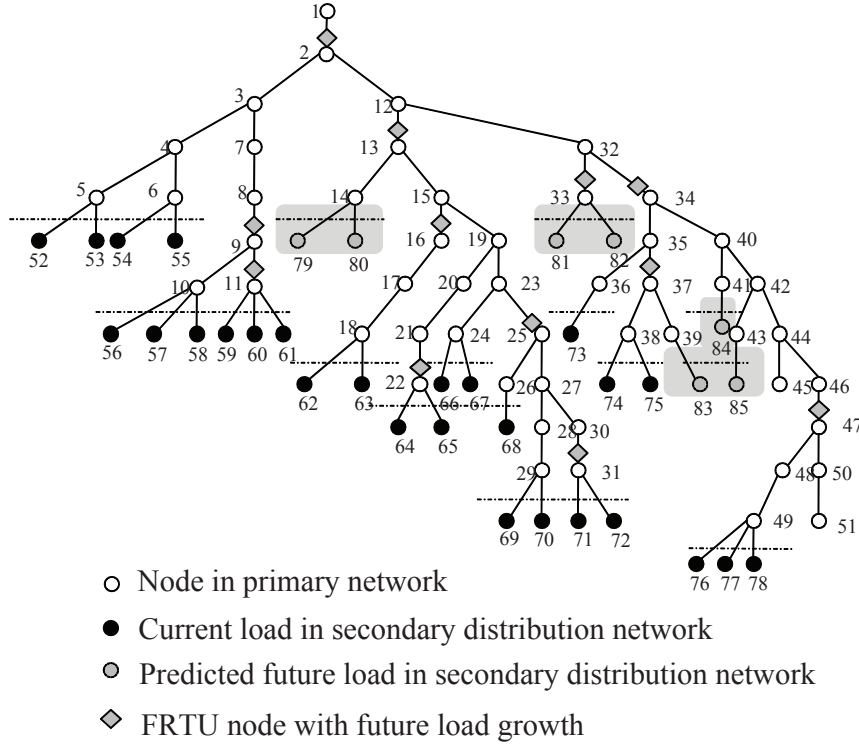


Figure 2.12: A distribution network in stage 2 with predicted load growth, and optimization result of FRTU candidate locations.

stage, *i.e.*, when $m = 2$, shown as the gray nodes in Fig. 2.12.

Fig. 2.12 also illustrates the FRTU optimization solution by considering the predicted load growth in the feeder. The solution consists of 12 FRTUs, which results in the total cost of $12 \times \$9000 = \108000 . It can be observed that the proposed solution can monitor all loads. Simultaneously with potential anomaly and load growth model, the simulation results shows that it can narrow down the location of the anomaly to at most 4 loads with ≥ 0.9 ACI. Although this solution may have more cost than the one in Fig. 2.11, it considers the load growth possibility. In contrast, the solution shown in Fig. 2.11 which does not consider load growth possibility is not able to detect the anomaly from the future loads,

e.g., from nodes 79 and 80. These nodes might result in the high false positive rate. On the other hand, a larger η can result in a higher ACI. For example, for the test case in Fig. 2.11, if adding one FRTU at node 2, and it may narrow down to the location at most 5 loads with 1.0 probability, *i.e.*, $\eta = 5$ and $ACI = 1.0$.

The Figs. 2.13 and 2.14 compare the two solutions in Fig. 2.11 and Fig. 2.12. Fig. 2.13 shows the comparison of the ACI, which is computed through the times that the FRTU deployment can narrow down the anomaly to at most 4 nodes over the total times of anomalies in all rounds. It can be seen that in stage 1, both solutions can achieve a satisfying ACI, which is greater than $ACI = 0.9$. However, in stage 2, the solution without considering the future load growth (the solution in Fig. 2.11) has a much worse ACI compared to that with considering the future load growth (the solution in Fig. 2.12), and it is not able to satisfy the constraint of ACI. In contrast, it is effective to consider the future load growth when performing the FRTU optimization. Fig. 2.14 provides the numerical comparison in term of additional FRTUs and loads to be monitored. The solution without considering the future load growth requires additional 8 FRTUs and the one considering the future load growth requires additional 12 FRTUs. Although the solution with future load growth requires more FRTUs, it can monitor more loads with better ACI. Thus, one can see the solution with future load growth outperforms the other one.

Comparison between proposed method and greedy algorithm is provided for analysis. Ideally, the greedy algorithm recommends every 4 loads under a common parent node

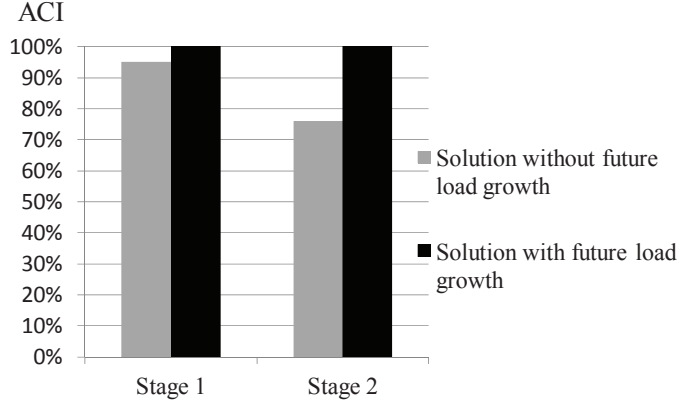


Figure 2.13: The comparison of the ACI between the solution in Fig. 2.11 and the solution in Fig. 2.12.

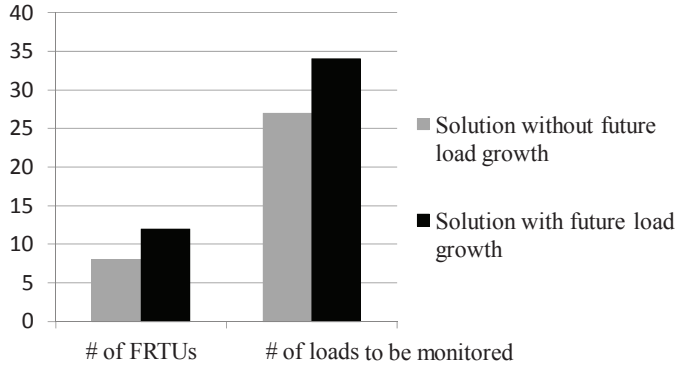


Figure 2.14: The comparison of the number of FRTUs and the number of loads between the solution in Fig. 2.11 and the solution in Fig. 2.12.

shall be added with one FRTU. With this, it could minimize the cost of FRTU, *i.e.*, each load with ACI of 0.9 is grouped with the other offspring nodes under an FRTU parent node.

However, its optimality depends on the topology of a feeder. As shown in Fig. 2.15 in stage 1, if each parent node is chosen with 4 child nodes from the left side, it will have these sets of nodes: $\{52, 53, 54, 55\}$, $\{56, 57, 58, 59\}$, $\{60, 61, 62, 63\} \dots$. The greedy algorithm selects node 4 for the first set, node 9 for the second set, and node 2 for the third set, and so on, which have 7 FRTUs in total. However, this will lead to violation of constraint by

having node 9 needs to take over 6 nodes, and node 25 takes over 5 nodes. This can result in optimal solutions but may violate the ACI constraint.

In contrast, the proposed algorithm addresses this issue by considering both minimization target and the constraint. As shown in Table 4.1, it compares the greedy algorithm and the proposed algorithm in stage 1. It is observed that the proposed algorithm outperforms the greedy algorithm.

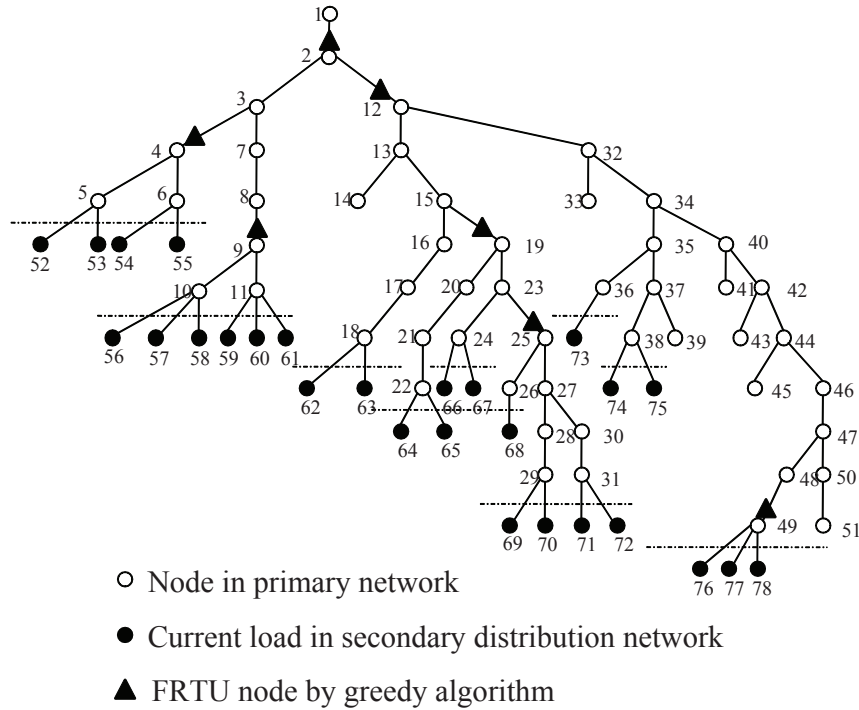


Table 2.2

The comparison of the greedy algorithm and the proposed algorithm in stage 1.

	# of FRTUs	ACI	Cost
Greedy Algorithm	7	52%	\$63000
Proposed Method	8	93%	\$72000

2.5 Summary

Integration of cyber-physical modeling for distribution planning has been a critical issue for strategic cyberinfrastructure investment. With the rapid development of IP-based communication on secondary network, it is crucial to provide an assisted tool for planning engineers to identify the system bottleneck for cross-checking the malicious activities in secondary distribution network. The proposed method aims to perform FRTU deployment in primary network by considering secondary network cybersecurity.

Due to the complexity of combinatorial enumeration for multistage deployment, adopting the existing optimization techniques may not suggest systematic identification of FRTU candidate locations. The proposed iterative algorithm recommends three major modules, which combines the combinatorial nature of the problem with potential multiple stages optimization based on the budget and historical anomaly datasets.

The simulation results demonstrate the feasibility of the proposed method by providing

a practical solution for multistage investment planning of distribution cyberinfrastructure enhancement. Future work includes cost recovery of cyberinfrastructure for utilities to better project the return of investment.

Chapter 3

Multi-Scale Variation-Aware Techniques for High Performance Digital Microfluidic Lab-on-a-chip Component Placement¹

3.1 Introduction

Traditional biochemical laboratory procedures are often very expensive to perform and the invention of microfluidic lab-on-a-chip alleviate the burden. Lab-on-a-chip integrates

¹©2011 IEEE. Reprinted, with permission, from Chen Liao and Shiyan Hu, “*Multiscale Variation-Aware Techniques for High-Performance Digital Microfluidic Lab-on-a-Chip Component Placement*”, IEEE Transactions on NanoBioscience, March 2011.

miniaturized components for implementing various functions in biochemical analysis into a chip using microfluidic technology [27, 1]. By lab-on-a-chips, biochemical experiments can be performed in a much cheaper way while having higher sensitivity and accuracy in detection [87, 88]. It can be effectively utilized as a test tube for the real biochemical experiment. Lab-on-a-chip has been successfully applied to many different areas. For example, the important genetic applications and biochemical analysis procedures such as DNA analysis and proteomic analysis [28]. In an ongoing world-wide research collaboration known as the Human Genome Project, lab-on-a-chips also significantly improve the speed to identify the estimated 80,000 genes in human DNA [89]. The microfluidic lab-on-a-chip is also applied in the electrochemical real-time monitoring of glucose and oxygen in [90], and the development of microfluidic lab-on-a-chip for vitro hepatotoxicity is discussed in [91]. In addition, it becomes indispensable for conducting human health related research including clinical diagnostics and drug discovery [31, 30]. For example, lab-on-a-chip has been successfully used in clinical diagnostics on human physiological fluids [32]. The proteomic analysis for cervical cancer based on microfluidic lab-on-a-chips is given in [29]. The lab-on-a-chip has more applications, such as being used in toxicological, protein and so on.

A lab-on-a-chip is a set of microarrays, which are miniaturized test benches, integrated on a solid substrate. It allows multiple biochemical operations/reactions to be performed simultaneously to achieve high throughput and speed, and also with the high accuracy. Usually, as shown in Figure 1.2, a lab-on-a-chip has no larger area than one square

centimeter. Similar to the traditional electronic integrated chip which has a very high computation speed, it can perform thousands of biochemical operations in a few seconds, such as sample holding, reagent mixing, separation and detection and so on [89]. The first generation of microfluidic lab-on-a-chip is continuous flow based and consists of permanently etched micropumps, microvalves, and microchannels. It is not accurate and not simple to operate. In contrast, the prevailing second generation of microfluidic lab-on-a-chip is droplet based which means that the liquids are manipulated as discrete microdroplets [1]. A few methods to manipulate microdroplets have been designed, such as those with structured surfaces, thermocapillarity, electrochemical effects and electrostatic actuation [92]. A popular lab-on-a-chip technology is based on electrowetting where each nanoliter droplet is controlled by the electric-field-induced electrohydrodynamic force generated from the programmed electrodes (refer to Figure 1.2 [1, 2, 3]). The reconfigurability allows different operations to share the same place on the microfluidic array during different time intervals. With systematic electrical signal programming, each biochemical operation can be treated as a three dimensional (3-D) module/cell whose size is decided by the space and the duration the operation needs. Together with the reconfigurability offered in lab-on-a-chips, a three-dimensional module/cell library can be designed [1]. With the cell/module library and design specification, CAD methodologies can be utilized to efficiently build a large-scale integrated microfluidic lab-on-a-chip (refer to Figure 3.1 [4]) [1, 93]. In other words, the cell-library based design methodology enables us to build a large-scale integrated microfluidic lab-on-a-chip efficiently using CAD

methodologies [1, 93].

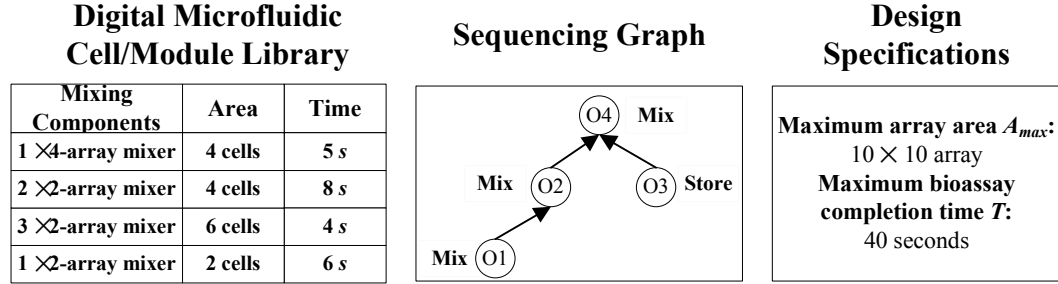


Figure 3.1: An input example of the microfluidic lab-on-a-chip component placement problem [4].

Similar to the case of IC design, the increasing complexity of the microfluidic lab-on-a-chips results in the increase of complexity of the physical-level synthesis of lab-on-a-chip. Thus, the high quality module placer and router are necessary for the lab-on-a-chip CAD. In the lab-on-a-chip CAD flow, the physical-level synthesis, which consists of *lab-on-a-chip component placement* and *lab-on-a-chip routing*, is the crucial component which has significant impact on the the whole flow. In detail, lab-on-a-chip component placement determines the physical location and the starting time for each operation such that the total completion time is minimized while all constraints are satisfied. Lab-on-a-chip routing is to seek the best routes to transport droplets such that the total length of routes is minimized while the distance constraint is satisfied. Similar to VLSI circuit, novel techniques to efficiently compute accurate solutions for lab-on-a-chip component placement and routing problems are in great demand in biochip CAD.

In contrast to the manual design, CAD uses the computer technology in the design process to handle device miniaturization and increasing design complexity. In the lab-on-a-chip CAD flow, *physical component placement of digital microfluidic lab-on-a-chip* is a crucial part whose solution may significantly impact the whole flow.

This chapter is mainly focused on the lab-on-a-chip placement. Precisely, it determines the physical location and the starting time of each operation such that the overall completion time determined by the last operation is minimized (see Figure 3.2 for an example). Similar to VLSI placement, designing a high quality module placer is quite challenging in lab-on-a-chip CAD. There are a large multitude of previous works for VLSI placement. For example, [94] proposes a quadratic based placement algorithm. A partition-based placement approach is designed in [95]. [96] proposes the constraint graph based technique for global placement and the greedy method based technique for the detailed placement. All the above techniques are effective in VLSI placement. One may want to directly migrate the techniques from VLSI placement to lab-on-a-chip component placement. However, there are critical differences. For example, lab-on-a-chip component placement handles resource constraint and scheduling constraint while VLSI placement only considers location optimization. On the other hand, VLSI placement is usually for two dimension while lab-on-a-chip component placement is for three dimension due to the reaction time. Thus, simple migration of the techniques from VLSI design to lab-on-a-chip design is difficult to lead to high quality solutions.

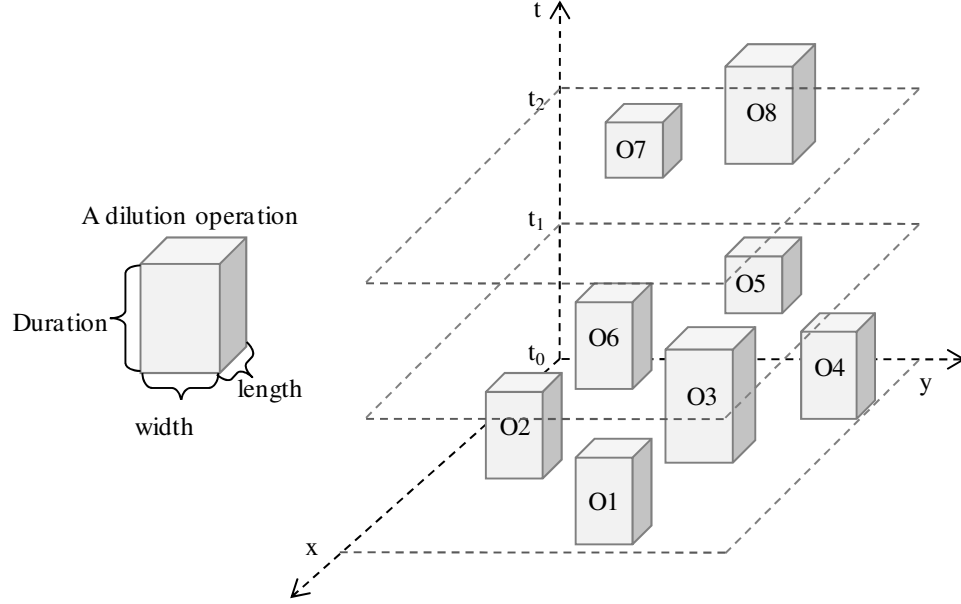


Figure 3.2: Illustration of a lab-on-a-chip component placement.

In spite of wide application of lab-on-a-chips, existing research works on lab-on-a-chip CAD are quite limited. For the lab-on-a-chip component placement, there are not much previous works [33, 3]. [33] proposes a simulate annealing based technique and [3] improves it by the incorporation of T -tree data structure. These algorithms are effective, however, they are not always able to compute a lab-on-a-chip component placement satisfying all design constraints due to that they are based on simulated annealing [3]. On the other hand, the timing of a biochemical reaction is sensitive to variations such as temperature variations, which necessitates the variation-aware lab-on-a-chip component placement. However, no previous work considers this issue. Thus, effective algorithmic techniques to compute high quality variation-aware lab-on-a-chip component placement satisfying various design constraints are in demand.

In this chapter, a multi-scale variation-aware optimization technique is proposed for the

lab-on-a-chip component placement. Our technique is always able to return high-quality placement and it is the first technique addressing the variation-aware lab-on-a-chip component placement. Our simulation results demonstrate that without considering variations, the proposed technique significantly outperforms the previous state-of-the-art approach [3] and is able to achieve up to 65.9% reduction in completions time. When considering variations, the variation-unaware design has the average yield of 2%, while our variation-aware technique can always compute the designs satisfying the yield constraint with only 7.7% completion time increase. The main contribution of this chapter is summarized as follows.

† Without considering variations, the multi-scale technique which consists of grid coarsening stage and front-line based fine-scale tuning stage is proposed to efficiently compute the solution for the variation-unaware lab-on-a-chip component placement.

† In contrast to the previous simulated annealing based works which cannot always return the solution satisfying all constraints, our proposed technique can obtain the nearly optimal solutions while satisfying all the constraints.

† To the best of the authors' knowledge, this is the first work addressing the variation-aware lab-on-a-chip component placement. A novel multi-scale variation-aware optimization technique is proposed. Latin Hypercube sampling technique is also integrated in the multi-scale technique for efficiency.

† The simulation results on the standard benchmark lab-on-a-chip designs demonstrate

that without considering variations, the proposed technique always satisfies the design constraints, and it outperforms the state-of-the-art approach [3] with up to 65.9% reduction in completion time.

† When considering variations, the variation-unaware design has the average yield of 2%, while the design by our variation-aware technique can always satisfy the yield constraint with only 7.7% completion time increase.

The rest of the chapter is organized as follows. Section 3.2 presents the problem formulation and integer linear programming formulation for lab-on-a-chip component placement. Section 3.3 introduces our multi-scale optimization technique for variation-unaware placement. Section 3.4 describes our multi-scale variation-aware technique to efficiently compute the variation-aware placement. Section 4.4 presents the simulation results with analysis. A summary of work is given in Section 4.5.

3.2 Preliminaries

3.2.1 Problem Formulation

A lab-on-a-chip is to manipulate droplets which contain biochemical reactants to perform biochemical reactions/operations. As a basic element in a lab-on-a-chip, each biochemical

operation can be treated as a three-dimensional cell/module whose volume is decided by the space ($x - y$ plane) and the duration (t -dimension) the operation needs. For a module m , denoted by $l(m)$ its length, by $w(m)$ its width and by $h(m)$ its height which is the time needed to perform the corresponding biochemical operation, respectively. Note that the timing of a biochemical reaction is sensitive to variations such as temperature variations. That is, the variations may impact the height of a module $h(m)$, which results in the *variational height*. In practice, droplet movements are manipulated at discrete intervals and the biochemical operations are scheduled at discrete time. A lattice is laid onto the solution space, which means that a module can be placed only at a grid point. The lab-on-a-chip component placement is to determine the physical location and the starting time of each operation/module/component.

In the lab-on-a-chip component placement, modules are to be packed such that *scheduling constraint*, *spacing constraint*, and *resource constraint* are satisfied [3]. Scheduling constraint, also called *precedence constraint*, specifies temporal relationship between operations by a sequencing graph. Spacing constraint, also called *non-overlapping constraint*, ensures that no operations can be performed during the same scheduled time period at the same location. For those modules sharing some resources, they can be scheduled at the same time only if there are enough available resources. This is called resource constraint. Given n modules, the lab-on-a-chip component placement problem targets to compute module placement solution with minimum completion time (determined by the last module) subject to the above constraints. When considering variations such as

temperature variations, some modules in the above lab-on-a-chip component placement may overlap due to the change of module size in some operating condition. Given a lab-on-a-chip component placement, we call the placement under an operating condition a sample. Given a large enough number of samples, yield is defined as the ratio of the number of samples not leading to any overlap over the total number of samples. In the variation-aware lab-on-a-chip component placement problem, a yield constraint is given and one targets to compute a placement solution satisfying the yield constraint. Our problem is formulated as follows.

Performance Driven Variation-Aware Lab-on-a-chip Component Placement: Given a set of three-dimensional modules, each of which is specified by some length, width, and variational height, and a fixed die area, to compute a solution for the lab-on-a-chip component placement, i.e., to decide the physical locations of the module with minimum overall completion time such that the non-overlapping, resource and scheduling constraints are satisfied, and the yield constraint is also satisfied.

3.2.2 Integer Linear Programming (ILP) Formulation for Variation-Unaware Lab-on-a-chip Component Placement

We first introduce the ILP formulation for variation-unaware lab-on-a-chip component placement in this subsection. The variation-aware lab-on-a-chip component placement will

be presented in Section 3.4. In contrast to the previous approach for the lab-on-a-chip component placement [3] which migrates VLSI circuit placement/floorplanning techniques, our technique directly solves the lab-on-a-chip component placement as a constrained three dimensional packing problem. For this, the problem of the lab-on-a-chip component placement is formulated as an integer linear programming (ILP) problem. We associate with each module i and each three-dimensional grid point (x, y, t) a binary variable $a_{x,y,t,i}$, i.e., $a_{x,y,t,i} \in \{0, 1\}$. Each module is indexed by its lower left corner. Thus, $a_{x,y,t,i} = 1$ means that the lower left corner of the i -th module is placed at (x, y, t) . Given n modules, the objective is to minimize the timing T subject to the following constraints.

The first constraint is to ensure that each module can be placed at only one location. That is,

$$\sum_{(x,y,t)} a_{x,y,t,i} = 1, \forall i = 1, 2, \dots, n. \quad (3.1)$$

The non-overlapping constraint is handled as follows. For any grid point, at most one module can be placed to cover it, i.e., the corresponding operation is active at the grid point. Formally, a module m_i covers (x, y, t) if its lower left index is in $[x - l(m_i), x] \times [y - w(m_i), y] \times [t - h(m_i), t]$, i.e., any of $a_{x',y',t'}$ is 1 where $x - l(m_i) < x' \leq x$, $y - w(m_i) < y' \leq y$ and $t - h(m_i) < t' \leq t$. Thus,

$$\sum_{\forall \text{ module } i \text{ covers } (x,y,t)} a_{x,y,t,i} \leq 1, \forall (x, y, t). \quad (3.2)$$

To handle the resource constraint, assume that there are v types of resources and there are $r_j, j = 1, 2, \dots, v$ available units of resource j . Denote by the non-negative constant $s_{i_{r_j}}$ the required units of resource j for performing operation i . For all the modules performed at time t , total required resources need to be no greater than the available resources for each resource type. Thus,

$$\sum_{\forall \text{ module } i \text{ covers } t} s_{i_{r_j}} \cdot a_{x,y,t,i} \leq r_j, \forall t, j, \quad (3.3)$$

where a module covers a time t if the corresponding biochemical operation is active at t . A module m_i covers t if the t -coordinate of its lower left index is in $[t - h(m_i), t]$. To handle precedence/scheduling constraint, if module i needs to be performed before module j as specified in the sequencing graph, we have

$$a_{x,y,t,j} + \sum_{\forall (x',y',t') \text{ with } t' \geq t} a_{x',y',t',i} \leq 1, \forall (x,y,t). \quad (3.4)$$

Note that the above t' needs to be iterated to T (subject to the following boundary issues) which is the maximum completion time. Further note that the boundary issue needs to be handled. An operation with the corresponding module m cannot be started at time later than $T - h(m)$ since all operations need to be completed by T . This means that all t for module m in the above formulations can be up to $T - h(m)$. x and y for module m are similarly handled. For the simplicity of presentation, assume that all boundary issues have been taken care of in the following formulation.

The objective of the linear program (LP) is to minimize T . Since one does not know T and the precedence/scheduling constraints cannot be formulated without it, the above mathematical program is cast to a decision problem rather than an optimization problem. The complete *decision integer linear programming formulation* is shown as follows, assuming that the boundary issues have been taken care of for simplicity.

$$\begin{aligned}
& \min 1 \\
& s.t. \\
& \sum_{(x,y,t)} a_{x,y,t,i} = 1, \forall i = 1, 2, \dots, n \\
& \sum_{\forall \text{ module } i \text{ covers } (x,y,t)} a_{x,y,t,i} \leq 1, \forall (x, y, t) \\
& \sum_{\forall \text{ module } i \text{ covers } t} s_{i r_j} \cdot a_{x,y,t,i} \leq r_j, \forall t, j \\
& a_{x,y,t,j} + \sum_{\forall (x',y',t') \text{ with } t' \geq t} a_{x',y',t',i} \leq 1, \forall (x, y, t) \\
& a_{x,y,t,i} \in \{0, 1\}.
\end{aligned} \tag{3.5}$$

Suppose that there is an approach to solve this integer linear program which will be presented soon. The optimal completion time, denoted by T^* , can be then found by performing a binary search within the upper bound, denoted by \bar{T} , and lower bound, denoted by \underline{T} . Each time, $T = \frac{\bar{T} + \underline{T}}{2}$ is used to formulate the above decision linear program.

Subsequently, either the lower bound or the upper bound will be set to T according to the decision result. That is, if the integer linear program is not feasible, the lower bound \underline{T} is set to T . Otherwise, the upper bound \bar{T} is set to T . The above process is iterated until the ratio between \bar{T} and \underline{T} is smaller than a user specified parameter. Denote by B the ratio between the initial upper and lower bounds. The above technique needs $O(\log B)$ iterations.

This process can be accelerated through performing the following logarithmic scale binary search proposed in [97] within the upper and lower bounds of T^* . For this, each time, $T = \sqrt{\bar{T} \cdot \underline{T}}$ is used to formulate the decision linear program. The decision problem is then solved and either the lower bound or the upper bound will be set to T as above. It can be shown that in this way, the number of iterations is reduced to $O(\log \log B)$ as follows. After solving each decision problem, if the upper bound is reduced to T , the ratio between the new upper and lower bounds will be $\frac{T}{\underline{T}} = \frac{\sqrt{\bar{T} \cdot \underline{T}}}{\underline{T}} = \sqrt{\frac{\bar{T}}{\underline{T}}}$. If the lower bound is increased to T , the new ratio will be $\frac{\bar{T}}{T} = \frac{\bar{T}}{\sqrt{\bar{T} \cdot \underline{T}}} = \sqrt{\frac{\bar{T}}{\underline{T}}}$. In either case, the ratio of new upper and lower bounds is reduced to \sqrt{B} . After the next iteration, the ratio will be reduced to $B^{1/4}$. In general, one can prove that after i oracle queries, the ratio between upper and lower bounds is $B^{\frac{1}{2^i}}$. Clearly, for this ratio to be below a ratio b , it needs $O(\log(\frac{\log B}{\log b}))$ iterations. For any constant ratio b , the above logarithmic scale binary search needs to perform $O(\log \log B)$ iterations. In our simulations, since T needs to be an integer by recalling that each module can only be placed in discrete position (i.e., at grid point), instead of using a fixed ratio b , we perform the logarithmic scale binary search until $\bar{T} - \underline{T} \leq 1$. As demonstrated in the simulation results, the logarithmic scale solution search is efficient.

The classic sequential rounding proposed in [98] is adopted to solve our ILP. One first relaxes the integer constraint $a_{x,y,t,i} \in \{0, 1\}$ in Eqn. (3.5) to be $a_{x,y,t,i} \in [0, 1]$. The resulting linear program, called *relaxed linear program*, does not have any integer constraint. It is well known that in practice, the linear programming problem can be solved in quadratic time in terms of the number of variables and constraints. The solution for the relaxed linear program will be rounded to integers in an iterative manner.

In each iteration, a linear program is solved. In the solution, all $a_{x,y,t,i} > 1 - \delta$ are fixed to 1, where δ is a user specified parameter. If no new variable is fixed, the one with the smallest rounding error will be fixed to 1. These can be accomplished by adding some additional constraints $a_{x,y,t,i} = 1$ to Eqn. (3.5). The new linear program will then be solved. Thus, each time, at least one variable is rounded. The above process is repeated until all variables have the integer values. This guarantees that the rounding procedure will compute an integer solution. This process is integrated into the logarithmic solution search as described above to solve the ILP.

3.3 Multi-Scale Optimization

Given a large lab-on-a-chip, the linear program may contain many variables and constraints, which degrades the efficiency in computation. This motivates us to explore the multi-scale optimization technique for the problem of the lab-on-a-chip component placement. This

technique consists of grid coarsening for speedup and fine-scale tuning for completion time improvement. Note that similar multi-scale techniques are popular in VLSI placement, however, our algorithm is different from them.

3.3.1 Grid Coarsening

In grid coarsening stage, the grid size is first coarsened by a factor of ρ . That is, the three-dimensional modules are only allowed to be placed at grid location which is a multiple of ρ . For example, setting $\rho = 4$ along t -dimension means that each module can only be placed at a coordinate of $0, 4, 8, \dots$ along t -dimension (refer to Figure 3.3). In this way, the number of variables will be significantly reduced. For example, there is no $a_{x,y,1,i}, a_{x,y,2,i}, a_{x,y,3,i}$ in the linear program when setting $\rho = 4$. One can similarly set ρ along the x, y dimensions. As a result, the number of constraints and the complexity of the linear program will be significantly reduced. Note that there is no such restriction on T . Together with the fact that modules can have various heights, T does not need to be a multiple of ρ . Solving the lab-on-a-chip component placement at a coarse scale introduces speedup but degrades the solution quality. Varying ρ , different tradeoff between runtime and solution quality can be obtained.

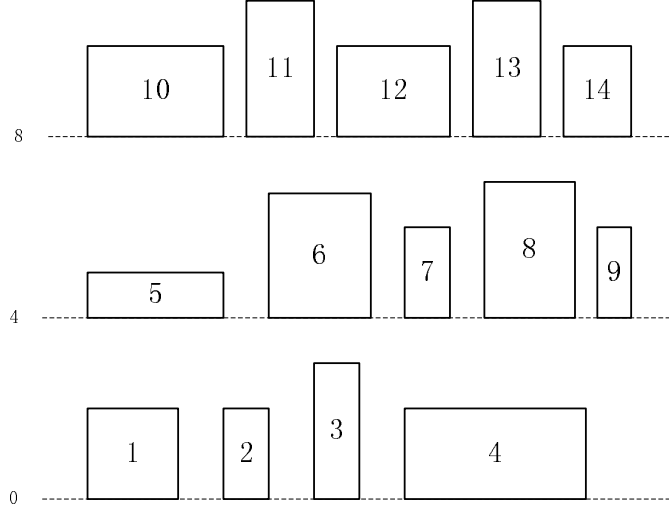


Figure 3.3: An example to illustrate fine-scale tuning where 0, 4, 8 refer to the time.

3.3.2 Fine-Scale Tuning

To recover the solution quality loss, the following fine-scale tuning technique is proposed. For simplicity of presentation, let us just focus on the completion time reduction along t -dimension, and other dimensions could be handled similarly. After obtaining the integer linear program solution at a coarse scale, we will attempt to move each module downward along t -dimension. Consider an example in Figure 3.3 which shows an integer linear program solution with $\rho = 4$. The modules starting at $t = 4$ can be certainly pushed downward to reduce the completion time. In contrast to only moving the modules along t -dimension, in our algorithm, more solution space will be explored. Take module 5 as an example. Suppose that its lower left corner is placed at $(1, 2, 4)$, i.e., $a_{1,2,4,5} = 1$. A new linear program will be formulated as follows. All the binary variables corresponding to

the module 5 are limited to those $a_{x,y,t',5}$ such that (x,y) spans the lab-on-a-chip base area and t' only spans over $\{1,2,3,4\}$, i.e., at a fine scale. Recall that in the previous linear problem in Eqn. (3.5), t in $a_{x,y,t,5}$ spans over the whole t -dimension ($\leq T$ in the decision LP) when grid coarsening is not applied. With grid coarsening, t can only take the values of $0, 4, 8, \dots, T$ to save the runtime. In contrast, in the fine-scale tuning stage, one can afford to set t as every possible integer value since it is restricted to a small local region around the solution of the coarsened integer linear program. The whole strategy makes sense since the coarsened integer linear program solution should be a good starting point for further tuning.

The fine-scale tuning technique is formally described as follows. Define the *i-front-line module set* as a set of modules with starting time at $i \cdot \rho$ in the coarsened integer linear program solution. The 0-front-line module set is the set $\{m_1, m_2, m_3, m_4\}$ for the example shown in Figure 3.3. Given the *i-front-line module set*, define the *i-upward module set* as the union of all *q-front-line module set* where $q \geq i$. For example, the 1-upward module set consists of all the modules except those in the 0-front-line module set (which have the starting time at 0). It is the set $\{m_5, m_6, \dots, m_{14}\}$ for the example shown in Figure 3.3. Similarly, define the *i-downward module set* as the union of all *q-front-line module set* where $q < i$. For example, the 1-downward module set consists of the modules in the 0-front-line module set (which have the starting time at 0). It is the set $\{m_1, m_2, m_3, m_4\}$ for the example shown in Figure 3.3. Clearly, the union of any *i-th* upward and downward module set forms the set containing all modules.

Starting with the 1-downward module set, for every module m_i in the set, fix it as in the coarsened integer program solution. In Figure 3.3, this means that all $a_{x,y,0,i}$ for $i = 1, 2, 3, 4$ are fixed to the values in the solution of the coarsened integer program. Denote by $t_c(m)$ the current t of module m in an integer linear program solution. Thus, $a_{x,y,t_c(m),m} = 1$ for some x, y . For each module m_i in the 1-upward module set, the variables are constructed to span all x, y while t is restricted to $\{t_c(m) - \rho + 1, t_c(m) - \rho + 2, \dots, t_c(m)\}$. We then formulate all the non-overlapping, resource, and scheduling constraints over these variables. The resulting integer linear program will be solved using sequential rounding technique described in Section 3.2.2 (without grid coarsening since it is the fine-scale tuning stage and the computation is affordable in local region). Note that the new decision program is feasible when the completion time is set to the same T as the previous coarsened integer linear program. This is the case since one can assign the values of a according to the solution of the previous coarsened integer linear program. The purpose of the tuning is to reduce T . For this, a loop is formed and each time T is decremented until the integer program becomes infeasible. For example, one possible fine-scale tuning solution for 1-upward module set of Figure 3.3 is shown as Figure 3.4. One can see that the locations of some modules are moved downward and the completion time is reduced.

After this, we will proceed to the 2-downward module set and 2-upward module set. Basically, fix the module location for the second downward set and seek the improvement over the upward second module set. This process is repeated until the current best solution reaches the last front-line. Our simulation results indicate the completion time can be

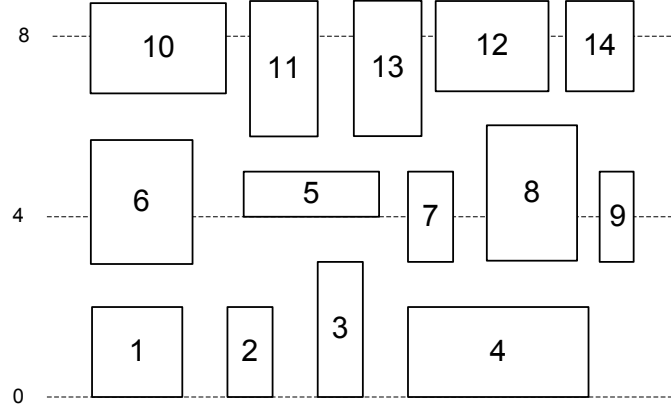


Figure 3.4: A possible fine-scale tuning solution for 1-upward module set of Figure 3.3, where 0, 4, 8 refer to the time. Assume that all constraints are satisfied.

significantly reduced by this fine-scale tuning technique. Refer to Section 4.4 for the details.

3.4 Variation-Aware Placement Design

In reality, the biochemical operations are quite sensitive to the variations. For example, the environment with different temperature results in the uncertain duration of operations which refers to the variational height of the modules. With variational heights of modules, it becomes a significant challenge to design a high-performance microfluidic lab-on-a-chip. Figure 3.5 shows an example of variation-unaware solution for the lab-on-a-chip component placement without any overlap, which is illustrated in two dimension for simplicity. With the variational height of module 2, an overlap between module 2 and module 5 could be introduced, which means that the design is sensitive

to variations. It is clear that the quality of microfluidic lab-on-a-chip design is highly dependent on variations. However, no previous works on lab-on-a-chip design consider the variation-aware lab-on-a-chip component placement. It is desirable to design an effective technique for the problem.

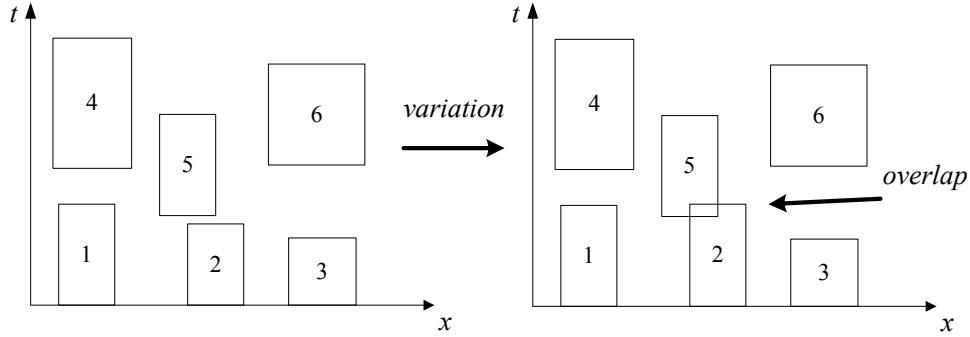


Figure 3.5: Illustration of the overlap which is due to variations in a variation-unaware lab-on-a-chip component placement.

3.4.1 Variation-Aware Optimization

A novel multi-scale variation-aware optimization technique is proposed based on the multi-scale technique in Section 3.3. We will first introduce the yield evaluation technique and then the variation-aware optimization technique.

Recall that a placement under an operating condition is called a sample. Given a large enough number of samples, yield is defined as the ratio of the number of samples not leading to any overlap over the total number of samples. Monte Carlo simulations are performed to evaluate the yield [99]. For this, a large number (e.g., 10,000) of samples

could be generated according to the probabilistic distributions on variational heights. However, it degrades the efficiency. To address this, Latin Hypercube sampling technique, which is first proposed in [100], is integrated in the multi-scale variation-aware placement technique. It is a popular technique in improving the simulation efficiency and it has been successfully used in a few VLSI CAD problems such as [101]. The advantage of Latin Hypercube sampling is that one can use only a few samples to approximate the simulation results with a large number of samples. This significantly improves the efficiency in computation. We refer the interested readers to [100, 101] for the details of Latin Hypercube sampling technique.

In the variation-unaware linear programming formulation, i.e., Eqn. (3.5), the heights of modules are all fixed. While in variation-aware optimization, duration following Gaussian distribution is introduced to our linear programming formulation. To solve it, our approach is motivated from [102] which formulates and solves the robust Knapsack and Portfolio problems using uncertain data based linear programming. In Eqn. (3.5), the height $h(m)$ of each module m is replaced by $h(m) + \beta \times \hat{h}(m)$, where $\hat{h}(m)$ is defined as the *variational range* according to Gaussian distribution, and β is a parameter to control the variation. Varying β , different placement solutions will be obtained. For example, when β is set to 1, the placement solution is the worst-case design, and when β is set to 0, the placement solution is variation-unaware design. Denote by $\tilde{h}(m)$ the variational height, i.e., $\tilde{h}(m) = h(m) + \beta \times \hat{h}(m)$. Note that after this transformation, each $\tilde{h}(m)$ is a constant for any fixed β . To explore the best trade-off between the yield and completion time, in our simulations,

the search of β (from 0.1 to 1.0 with a step size of 0.1) is performed.

With a fixed β , one can solve the integer linear program in Eqn. (3.5). Based on this solution, the multi-scale placement optimization technique is enhanced to consider the variations for yield improvement. Recall that in fine-scale tuning, when proceeding to i -front-line, the technique perturbs the current placement to compute a new placement solution for the $(i + 1)$ -upward module set in the small local region. In contrast to directly using this new placement solution, the yield of this solution will be computed. If the yield is greater than the yield constraint, this solution will be taken. Otherwise, the original solution will be kept. The algorithm will proceed to the next front-line in a similar way. As the example shown in Figure 3.6, the yield constraint is set to 99%. A fine-scale tuning solution for 1-upward module-set is computed from a grid-coarsening solution. The yield of the new solution is less than 99%. Thus, this solution is discarded and the original one is kept. The algorithm will proceed to the 2-upward module-set in a similar way.

3.5 Simulation Results

In the simulations, the proposed multi-scale integer linear programming based lab-on-a-chip component placement algorithm is implemented in C++, and is tested on a computer with 2.5GHz CPU and 4GB main memory. We conduct the simulation on a set of standard lab-on-a-chip benchmarks used in [3] and compare our new algorithm

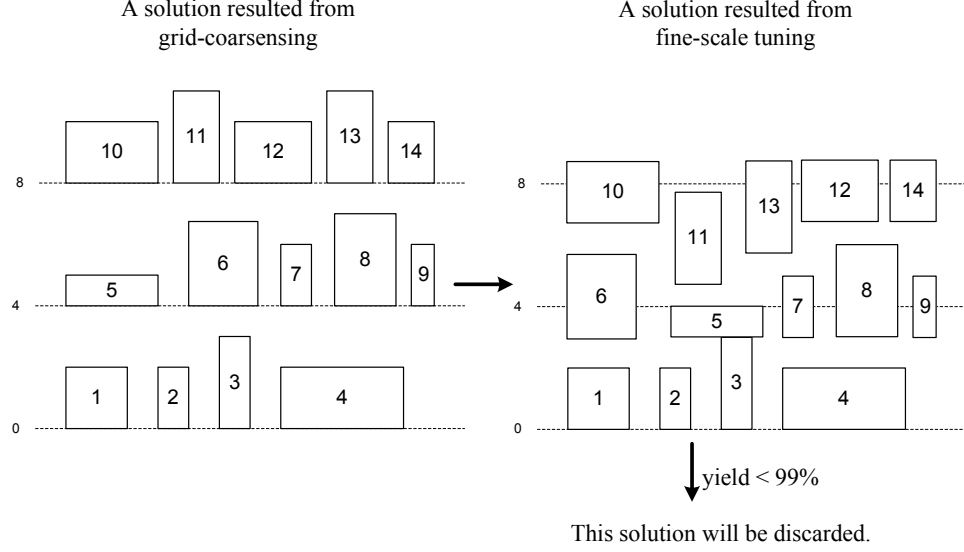


Figure 3.6: An example to illustrate the variation-aware multi-scale technique.

without considering variations to [3]. We also conduct the simulation to compare our variation-aware optimization with the variation-unaware optimization. In our simulation, the yield constraint is set to 99% and Gaussian variation is assumed with the variational range \hat{h} set to 0.1 which is equal to 3σ in Gaussian distribution. Note that our proposed technique could also handle other distributions. Techniques used in comparisons are summarized as follows.

- † NEW: the proposed multi-scale variation-unaware optimization technique.
- † NEW w/o Fine-Tune: NEW except that the fine-scale tuning is turned off.
- † NEW w/ standard binary solution search: NEW with standard binary solution search instead of logarithmic scale solution search.
- † NEW w/ variations: the proposed multi-scale variation-aware optimization

technique.

3.5.1 Variation-Unaware Design

We first compare our algorithm with the previous work [3]. The results are summarized in Table 3.1. Since our algorithm is a multi-scale based technique, it is interesting to investigate the performance when the fine-scale tuning is turned off, i.e., NEW w/o Fine-Tune. We make the following observations.

Table 3.1

Comparison of NEW and the previous work [3]. Timing refers to the overall completion time of the last module in the solution of the lab-on-a-chip component placement. CPU refers to the runtime in seconds. Timing reduction refers to the completion time improvement of NEW which is computed by comparing to the previous work [3].

Testcase		Previous work [3]		NEW w/o Fine-Tune			NEW			
Name	Area Constraint	Timing	CPU(s)	Timing	CPU(s)	Timing Reduction	Timing	Timing Reduction	CPU(s)	Yield
vitro1-1	9×9	80	36.8	76	137.7	5.0%	56	30.0%	190.0	0.5%
vitro1-2	8×8	100	20.8	69	95.8	31.0%	54	46.0%	135.5	0%
vitro1-3	7×7	107	31.2	69	209.0	36.7%	54	49.5%	261.0	0%
vitro1-4	6×6	N/A	N/A	69	81.8	N/A	54	N/A	109.8	0%
vitro2-1	8×8	78	18.4	47	94.5	39.7%	42	46.2%	101.1	2%
vitro2-2	7×7	64	13.6	45	54.9	29.7%	42	34.4%	65.5	1.5%
vitro2-3	6×6	129	6.8	48	78.0	62.8%	44	65.9%	73.6	1.0%
vitro3-1	7×7	64	9.6	45	22.6	29.7%	44	31.3%	27.0	4.5%
vitro3-2	6×6	63	14.6	47	29.4	35.4%	44	30.2%	34.2	10.5%
vitro3-3	5×5	112	12.6	45	14.6	59.8%	42	62.5%	17.7	0%
Average								44.0%		2.0%

† The previous work [3] cannot always return a solution satisfying the chip area constraint due to the nature of simulated annealing. It is the case for the benchmark design vitro1-4. The best design we can get after 20 runs of [3] needs the

lab-on-a-chip area of 7×6 with completion time 96. This violates the area constraint of 6×6 . Some other results we can get are $9 \times 8 \times 118$ and $12 \times 6 \times 72$ which violate the area constraint even more.

† NEW is always able to meet the design constraints. Our solution is much better than the previous work. On average, the completion time is reduced by 44%. For the design vitro2-3, 65.9% completion time reduction is achieved.

† The multi-scale technique in NEW consists of grid coarsening and fine-scale tuning. Note that T is not rounded to the multiple of ρ in the linear programming formulation. To choose ρ , extensive simulations are performed and the following parameter setting gives the best tradeoff between solution quality and runtime. We coarsen the time dimension by a factor of $\rho = 8$ on t for the first 4 testcases and $\rho = 4$ on t for the last 6 testcases.

† Fine-scale tuning can significantly improve the solution quality. Comparing NEW and NEW w/o Fine-Tune, one can achieve up to $1 - 56/76 = 26.3\%$ completion time improvement.

† It can be seen that [3] saves some runtime over NEW, however, its solution quality is much worse. As mentioned above, the completion time by NEW can be about half of that of [3] in some testcases. Current lab-on-a-chip design technology is still in the early stage without very large scale integration. It is the time to compute high quality design solutions for promoting the technology advances in lab-on-a-chip as much as

possible. That is, the solution quality, rather than efficiency, is of the top importance.

With regard to the proposed technique, the obtained large improvement in solution quality clearly outweighs the runtime slowdown.

NEW contains many advanced techniques including logarithmic scale solution search proposed in [97]. It is interesting to investigate the performance of our algorithm (with $T = \sqrt{\bar{T} \cdot \underline{T}}$ compared to the traditional binary search (with $T = \frac{\bar{T} + \underline{T}}{2}$). The results are summarized in Table 3.2. Note that the solution could be changed compared to NEW since each time the approximation result to the decision integer linear program may be different. However, one can see that the solution quality difference is slight, but the runtime difference is quite significant. For example, the logarithmic solution search runs up to $543.5/135.5 = 4.0\times$ faster for the design vitro1-2.

Table 3.2
The results of NEW w/ standard binary solution search.

Testcase		NEW w/ standard binary solution search	
Name	Area Constraint	Timing	CPU(s)
vitro1-1	9×9	54	291.6
vitro1-2	8×8	56	543.5
vitro2-1	8×8	42	219.8
vitro2-2	7×7	44	93.2
vitro2-3	6×6	44	79.8
vitro3-1	7×7	44	35.6
vitro3-2	6×6	44	34.3
vitro3-3	5×5	42	17.8

3.5.2 Variation-Aware Design

We compare the variation-aware placement with the variation-unaware placement. To explore the best trade-off between the yield and completion time, a search for the best solution satisfying the yield constraint is applied by varying the parameter β . The yield constraint is set to 99%. The results with best β are summarized in Table 3.3, where the timing increase is computed as the ratio of the completion time of NEW w/ variations over that of NEW minus 1. From Table 3.1 and Table 3.3, we make the following observations.

Table 3.3

The results of variation-aware optimizations. Timing increase is computed through comparing the completion time of NEW and NEW w/ variations.

Testcase		Variation aware design			
Name	Area Constraint	Timing	Timing Increase	CPU(s)	Yield
vitro1-1	9×9	58	3.6%	2889.2	100%
vitro1-2	8×8	58	7.1%	1933.2	100%
vitro1-3	7×7	60	11.1%	1544.2	100%
vitro1-4	6×6	61	13.0%	1955.3	100%
vitro2-1	8×8	45	7.1%	1690.9	100%
vitro2-2	7×7	47	11.9%	1130.0	100%
vitro2-3	6×6	47	6.8%	931.7	100%
vitro3-1	7×7	45	2.3%	523.4	100%
vitro3-2	6×6	45	2.3%	528.3	100%
vitro3-3	5×5	47	11.9%	275.7	100%
Average			7.7%		100%

† For all testcases, the yields of the variation-unaware design computed by NEW are very small. For example, the yield of testcase of vitro3-3 is 0, and the largest yield is

10.5% for testcase of 3-2. The average yield is only 2%. When considering Gaussian variations, for each module m with an original height of $h(m)$, the variational height has a possibility of $\frac{1}{2}$ to be greater than $h(m)$, which could lead to overlap if the computed variation-unaware placement is quite compact (i.e., with small amount of empty space). A rough analysis would then show that the possibility of an overlap (when considering variations in a variation-unaware placement solution) could reach $1 - (1/2)^n$ for n modules. Although this number would not be accurate, one can get the sense on why the yield is very small. This also demonstrates that without considering variations, it is difficult to design a high-performance lab-on-a-chip.

† For all testcases, our variation-aware design can satisfy the yield constraint. It demonstrates that these designs are significantly insensitive to variations, which means that the NEW w/ variations is effective. On the other hand, the completion time of the design computed by NEW is smaller than that of NEW w/ variations. However, the great improvement of the yield outweighs the increase of completion time.

3.6 Summary

In this chapter, a variation-aware optimization technique is proposed for the lab-on-a-chip component placement. As a large number of variables and constraints significantly impact

the efficiency in computation, the multi-scale technique including the grid coarsening and fine-scale tuning is explored. Meanwhile, since the quality of microfluidic lab-on-a-chip design is highly dependent on the variations, this chapter designs the variation-aware technique for lab-on-a-chip component placement using Latin Hypercube sampling based Monte Carlo simulation. The simulation results on standard benchmark designs demonstrate that the proposed technique is effective and outperform the state-of-the-art work, with up to 65.9% reduction in completion time. In addition, our variation-aware technique can always satisfy the yield constraint with small degradation in completion time.

Chapter 4

Physical-Level Synthesis For Digital Lab-on-a-chip Considering Variation, Contamination and Defect

4.1 Introduction

As mentioned in Chapter 3, the microfluidic lab-on-a-chip leads to the automation of laboratory procedures in biochemistry [103, 37]. It gradually replaces traditional biochemistry procedures and has been widely used due to its advantages in, e.g., sample holding, reagent mixing, separation and detection etc. [87, 88]. This novel technology has

been successfully applied in different areas. For example, the microfluidic lab-on-a-chip is applied in the electrochemical real-time monitoring of glucose and oxygen in [90], and the development of microfluidic lab-on-a-chip for vitro hepatotoxicity is discussed in [91]. Meanwhile, there are also some works on the human health related research. For example, the proteomic analysis for cervical cancer based on microfluidic lab-on-a-chips is given in [29].

The prevailing digital microfluidic lab-on-a-chip is based on the manipulation of discrete liquid droplets which contain biochemical samples for the reactions [1]. Microfluidic processing is performed on unit-sized packets of fluid which are transported, stored, mixed, reacted or analyzed in a discrete manner using a standard set of basic instructions. The complex procedures, such as chemical synthesis or biological assays, can be built up step by step. Refer to Figure 1.2 [1, 2, 3, 5]. The nanoliter droplets containing the biochemical samples are between two electrified plates. Through the systematic electrical signal programming, the droplets can be transported to the desired places for the biochemical operations.

It is well known that compared to manual IC design, the computer-aided design (CAD) methodologies has largely reduced the design efforts of the integrated circuit (IC) design. CAD methodologies have not been widely used in commercial lab-on-a-chips. Due to the increasing complexity and decreasing size, it is necessary to apply CAD methodologies for the large-scale integrated lab-on-a-chip design [1, 93, 5]. As mentioned in Chapter 3,

in the lab-on-a-chip CAD flow, the lab-on-a-chip placement and routing are the key parts, which are called *physical-level synthesis*. Lab-on-a-chip placement is to determine the physical location and the starting time of each operation such that the overall completion time is minimized satisfying the precedence constraint, non-overlapping constraint and resource constraint [5]. Lab-on-a-chip routing is to route the droplets from the source to the destination to ensure the successful completion of the operations such that the routing distance is minimized while satisfying timing constraint and fluidic constraint.

This chapter is mainly focused on physical-level synthesis, including both of lab-on-a-chip placement and routing. Refer to Figure 4.1. Each operation can be treated as a three-dimensional (3-D) module, decided by x, y and t , where $x - y$ plane is its space and t is the time duration of this operation. Together with the reconfigurability of lab-on-a-chips, a 3-D module library can be designed on biochemical operations [1]. These operations/modules need to be placed at the 3-D space, and the droplets are transported from the source to destination, where both of the source and destination can be either an operation or a reservoir/dispensing port.

Contamination and defect also need to be considered during lab-on-a-chip physical-level synthesis [104]. Contamination in lab-on-a-chip is caused by liquid residue of the transportation between different operations [104]. The contaminated grids usually need to wait for a certain time before they can be used for routing again. Defect in lab-on-a-chip is usually due to the lab-on-a-chip manufacturing imperfection, which are affected by

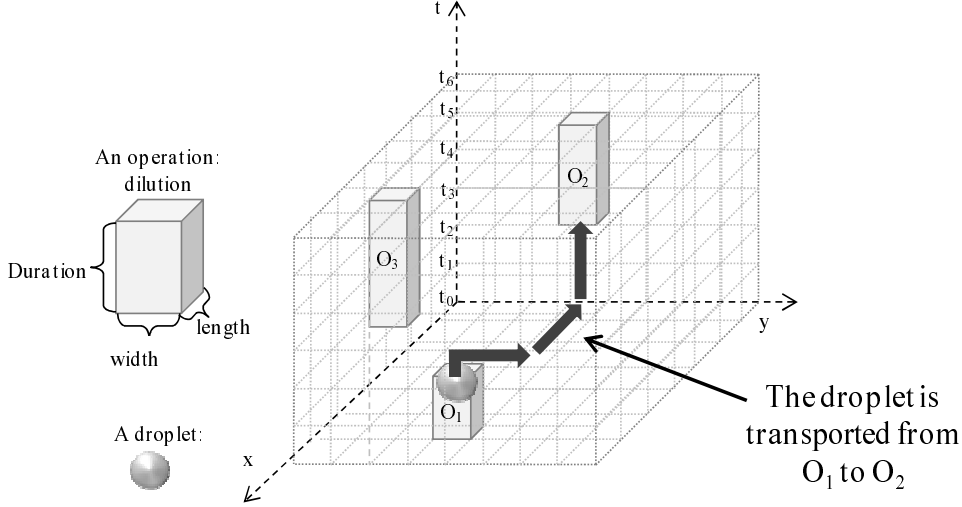


Figure 4.1: Illustration of lab-on-a-chip placement and routing.

the shrinking processes, new materials and so on [105]. The defective grids can not be used for routing at any time. Refer to Figure 4.2 for an illustration of contaminated grid and defective grid. Contamination aware and defect tolerant technique in microfluidic lab-on-a-chip has attracted much attentions in previous literatures. A contamination aware algorithm for droplet routing in lab-on-a-chip is proposed in [104]. [106] considers the cross-contamination problems on pin-constrained lab-on-a-chips during the lab-on-a-chip design flow. [107] proposes a unified synthesis method that simultaneously consider defect tolerant architectural synthesis and defect aware physical design for lab-on-a-chip. [105] designs a defect tolerant methodology based on graceful degradation and dynamic reconfiguration for lab-on-a-chip design.

On the other hand, the biochemical reaction is sensitive to many variations [36, 34, 5]. For example, the temperature variations could lead to the operation completion time variation. [37] mentions that the biochemical operations have variability margins, which can impact

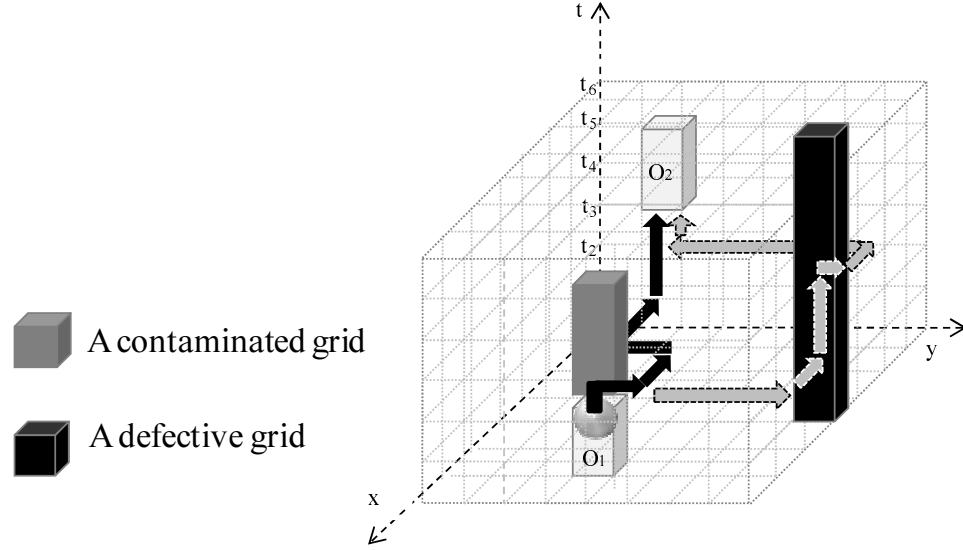


Figure 4.2: An illustration of routing with timing constraint, considering contamination and defect. The routing needs to avoid the contaminated grid and defective grid. The grey route may violate the timing constraint since the routing length might be greater than t_{\max} , e.g., $t_{\max} = 10$.

the correctness of the biochemical application. [38] describes that the temperature environment is significant for the biochemical operations since some DNA would denature at inappropriate temperature. Thus, the biochemical operation completion time can have variation which necessitates the lab-on-a-chip routing considering variation. Refer to Figure 4.3. Without considering variation, a grid which is supposed to be used for routing may be occupied by the module with variational height, and may lead to the failure of routing [34, 35]. However, the traditional lab-on-a-chip routing does not consider the variation. Note that [5] only considers placement with variation, which has impact on routing, but it neglects the routing.

This is the first work to propose the optimization technique for lab-on-a-chip physical-level synthesis considering variation, contamination and defect. The main contribution of this

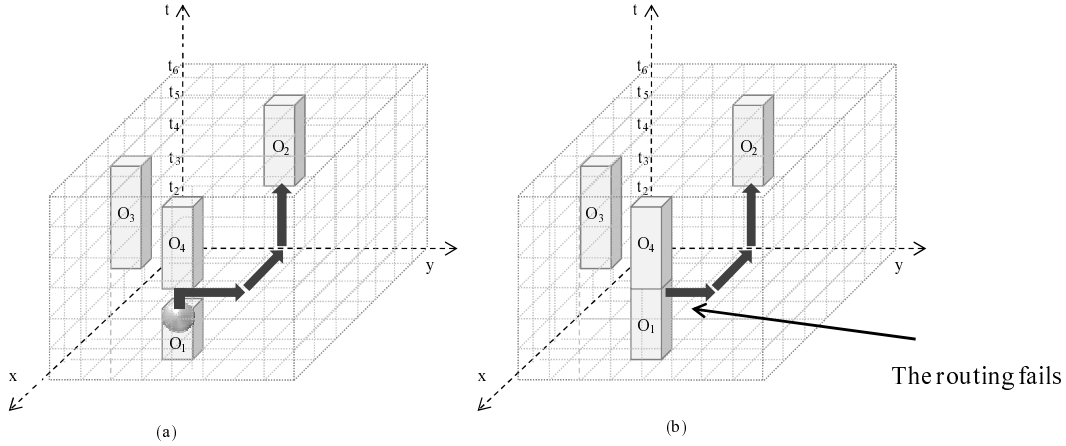


Figure 4.3: (a) An example of placement and routing. The lower left corner of a module can be placed at a grid. (b) The module could have variational height, and is touching with each other. Thus, there is no space for its routing, which leads to the failure of routing.

chapter is summarized as follows.

- † To the best of the authors' knowledge, this is the first work to consider all the impacts of variation, contamination and defect in lab-on-a-chip physical-level synthesis.
- † It proposes the maze routing based, variation, contamination and defect aware droplet routing technique, and it seamlessly integrates the routing to the placement technique in [5]. It improves the placement solution in order to satisfy all the constraints for routing, which is the main reason for the routing yield improvement. This realizes the placement and routing co-optimization and enhances the lab-on-a-chip CAD tool.
- † The simulation results on a set of standard testcases demonstrate that, without considering variation, the routing yield is very small which may lead to the failure of the design. For example, the design of vitro2-1 only has the routing yield of 41.5%.

† With considering variation, our technique can successfully route the droplets while satisfying the routing yield constraint. It significantly improves the average routing yield by 51.2% with only 3.5% increase in completion time.

† There is no violation of using the defective/contaminated grids through the proposed technique, while the technique without considering contamination and defect uses 17% of the defective/contaminated grids.

The rest of the chapter is organized as follows: Section 4.2 presents the problem formulation, and gives the definition of the terminologies. It also briefly introduces the technique proposed for placement in [5]. Section 4.3 proposes our technique for physical-level synthesis considering variation, contamination and defect. Section 4.4 presents the simulation results with analysis. A summary of work is given in Section 4.5.

4.2 Preliminaries

4.2.1 Problem Formulation

A lab-on-a-chip is to perform biochemical reactions/operations through manipulating droplets which contain the biochemical reactants. Recall that each biochemical operation can be considered as a three-dimensional (3-D) module decided by x, y and t , where

$x - y$ plane is its space and t is its duration. In practice, the movement of droplets is manipulated at discrete intervals without unexpected overlapping with other droplets, and the biochemical operations are scheduled at discrete time. Thus, a lattice is laid onto the 3-D space and divides the space to a set of unit grids, which are the lab-on-a-chip routing grids. The droplets can only be moved through these grids, and a module can only be placed at a grid. Refer to Figure 4.1 for illustration.

In the lab-on-a-chip placement, it is to determine the physical location and the starting time of each operation such that the overall completion time is minimized while satisfying the precedence constraint, non-overlapping constraint and resource constraint. Precedence constraint defines the temporal relationship between operations through a sequencing graph. Non-overlapping constraint ensures that no operations can be performed during the same time period at the same location. Resource constraint is that the operations sharing some resources can be scheduled at the same time period only if there are enough available resources [5].

In the traditional lab-on-a-chip routing, given a set of sources and destinations, the droplets are manipulated to transport from its source to the destination, while satisfying *timing constraint* and *fluidic constraint*. The source and destination can be either a module or a reservoir/dispensing port. Dispensing port is a place on the chip for some droplet generation. Timing constraint specifies the maximum timing of a droplet transporting from the source to the destination [108]. Fluidic constraint states that the minimum spacing

between two droplets is one grid [108].

As the routing procedure is neglected when performing the lab-on-a-chip placement in some previous works [3, 5], the placement solution may have no space for routing, which results in the negative effect on the physical-level synthesis. Figure 4.4 shows such an example. Thus, if a placement solution has no space to route, one could improve the placement to make it routable, which means that there is space for routing and the updated placement satisfies the precedence constraint, non-overlapping constraint and resource constraint.

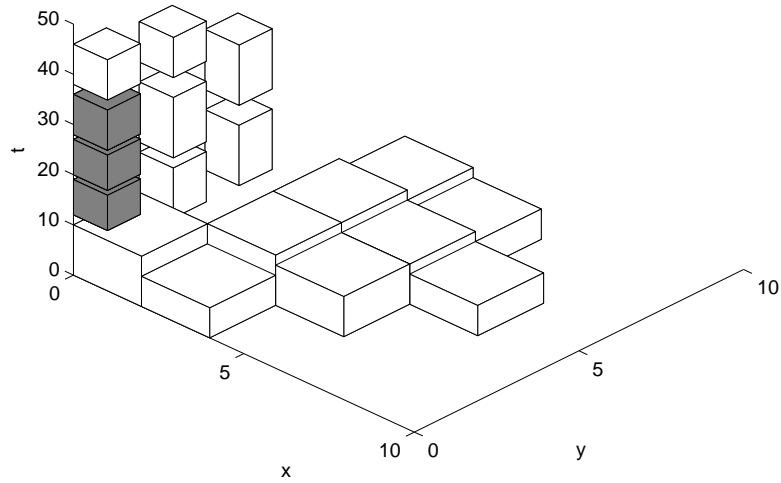


Figure 4.4: An example of non-routable placement solution for testcase vitro3-1 using [5]. The three grey modules are too close to each other, and thus the routing might fail.

Given an initial placement solution consisting of a set of modules specified by some

lengths, widths and heights, and the sources and destinations, our physical-level synthesis problem without considering variation aims to improve the placement solution to be routable, and route each droplet from its source and destination, such that the timing constraint and fluidic constraint are satisfied. In addition, the contamination and defect also need to be considered during routing.

On the other hand, the biochemical reaction is sensitive to many variations [34]. Thus, the case that the biochemical reaction completion time can have variation necessitates the lab-on-a-chip routing considering variation. Since each module can be considered as a 3-D module as mentioned above, for a module m , it can be decided by $l(m) \times w(m) \times h(m)$, where $l(m)$, $w(m)$ and $h(m)$ respectively denote its length, width and height. Note that the height is the time needed to perform the corresponding biochemical operation. Refer to Figure 4.3. The completion time variation affects the height of the module $h(m)$. Define the resultant height to be *variational height*, denoted by $\tilde{h}(m)$. The model in [5] defines

$$\tilde{h}(m) = h(m) + \beta \times \hat{h}(m), \quad (4.1)$$

where $\hat{h}(m)$ is the *variational range* for the height of the module following Gaussian distribution, β is a user defined parameter which can adjust the induced variation to the module, $0 \leq \beta \leq 1$. Larger β , larger variation. When $\beta = 0$, it leads to a design without considering variation, and when $\beta = 1$, it leads to worst-case design. Note that through Eqn. (4.1), with any fixed β , the height of module \tilde{h} will be a constant [5]. To explore the

best tradeoff between the routing yield and CAD design solution quality using different β , a search needs to be applied through varying β . In our simulation, the search of β is performed from 0.1 to 1.0 with a step size of 0.1.

Given a physical-level synthesis solution, which consists of the routing solution integrated with a placement, we call the routing with the placement under an operating condition a *physical-level synthesis sample*. Note that in the placement problem in [5], a sample is only associated with a placement solution, while in our problem, a sample is associated with a physical-level synthesis solution. Recall that given a large enough number of those physical-level synthesis samples, routing yield is defined as the ratio of the number of samples not violating any constraint over the total number of samples. The high routing yield means that it is good for most operating conditions. In the physical-level synthesis problem considering variation, a *routing yield constraint* is given, which is between 0% and 100%, and one targets to compute a physical-level synthesis solution satisfying the routing yield constraint. It is also clear that the variation is different from contamination and defect. Our problem is formulated as follows.

Physical-Level Synthesis For Lab-on-a-chip Considering Variation, Contamination and Defect: Given an initial placement solution consisting of a set of modules specified by lengths, widths, and variational heights, and the sources and destinations, our physical-level synthesis problem aims to improve the placement solution to be routable, and decide the transporting route for each droplet from its source to the destination such that the timing

constraint, fluidic constraint and the routing yield constraint are satisfied, while avoiding the defective and contaminated grids.

4.2.2 Multi-Scale Technique For Placement Considering Variation

Chapter 3 introduces the multi-scale variation aware technique for placement. We first briefly overview it since our technique will be seamlessly integrated to the initial placement solution using its technique, which is also introduced in [5]. Recall that [5] only considers variation aware placement but not routing.

In Chapter 3, the placement problem is defined as follows. Given a set of 3-D modules, each of which is specified by some length, width, and variational height, and a fixed die area, to decide the physical locations of the module with minimum overall completion time such that the non-overlapping, resource and scheduling constraints are satisfied, and the placement yield constraint is also satisfied. Note that placement yield in [5] refers to the ratio of the number of placement samples not leading to any overlap over the total number of placement samples. This placement yield is different from our routing yield. In its proposed technique, the lab-on-a-chip placement problem is formulated as an integer linear programming (ILP) problem. A lattice is laid onto the 3-D space and divides the space to a set of unit grids. The left lower corner of each module will be placed at one and only one grid, satisfying all the constraints. Each grid can be occupied by no greater

than one module. In the ILP in [5], there are a large number of variables associated with grids, modules and constraints. Refer to [5] for details of the ILP formulation. Given a large lab-on-a-chip, the ILP may contain too many variables and constraints, which largely degrades the efficiency in computation [5]. Thus, the multi-scale technique consisting of grid coarsening for speedup and fine-scale tuning to improve the completion time is proposed.

4.2.2.1 Grid Coarsening

In the original ILP, the lower left corner of each module allows to be placed at any grid, as long as the constraints are satisfied. In grid coarsening, the grid size is coarsened by a factor of κ , which means that the modules are only allowed to be placed at the grid location with the coordinates of a multiple of κ . For example, if set $\kappa = 4$ along t -dimension, the modules can be only placed at a coordinate of 0, 4, 8, 12... along t -dimension. The coarsening of other coordinate is the same. Refer to Figure 4.5(a) for illustration. There is a tradeoff between the runtime and solution quality.

4.2.2.2 Fine-Scale Tuning

Fine-scale tuning is utilized to recover the solution quality loss in grid coarsening. Take t -dimension for example. After grid coarsening, the ILP is solved and a coarsened

placement solution can be obtained. The solution quality may be negatively impacted by grid coarsening. However, refer to the example of coarsened placement solution in Figure 4.5(a). Some modules may be able to be pushed down to improve solution quality. Thus, a set iterations of tuning is performed. Suppose the lower left corner of a module is placed at $i \cdot \kappa$ in t -dimension. Define the *local region* of this module to be between $(i - 1) \cdot \kappa$ and $i \cdot \kappa$ in t -dimension, where i is an integer and $i \leq 1$. For example, the local region of module 5 in Figure 4.5(a) is between 0 and 4 in t -dimension. For the i -th iteration, define all the lines at $i \cdot \kappa$ in t -dimension the i -th *front line*. In the i -th iteration, the modules whose lower left corners are placed below $i \cdot \kappa$ are fixed, and those placed above $i \cdot \kappa$ will search a location with smaller completion time in their local region while satisfying all constraints. Consider Figure 4.5 as an example. In the first iteration, module 1, module 2, module 3 and module 4 are fixed as the coarsened placement solution and the other modules will be tuned. Each front line will be handled in an iteration, until all front lines are handled. An improved placement will be obtained. Refer to [5] for details.

4.3 Proposed Physical-Level Synthesis Technique

Considering Variation

A novel optimization technique for lab-on-a-chip physical-level synthesis considering variation, contamination and defect is proposed. There are two key parts in our technique.

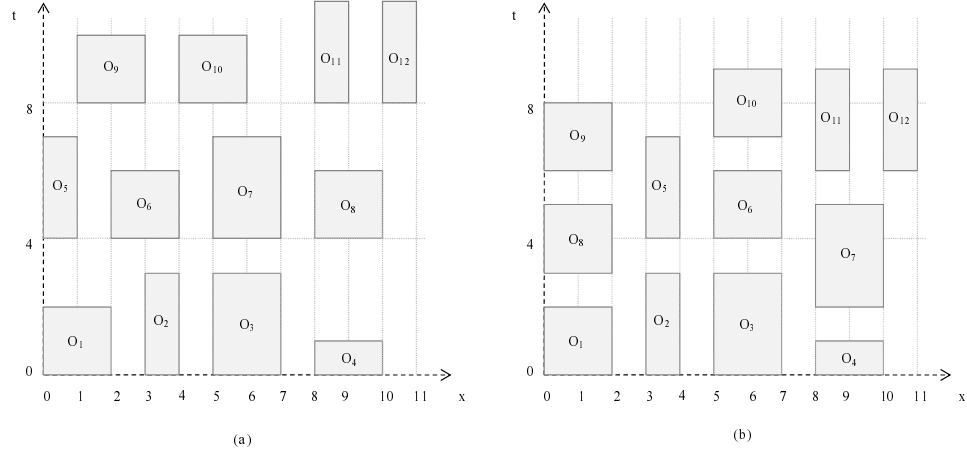


Figure 4.5: (a) An example of grid coarsening of t -dimension, with $\kappa = 4$. For simplicity, the illustration is in 2-D, i.e., x and t coordinates. (b) A possible fine-scale tuning solution for (a). The total completion time is reduced.

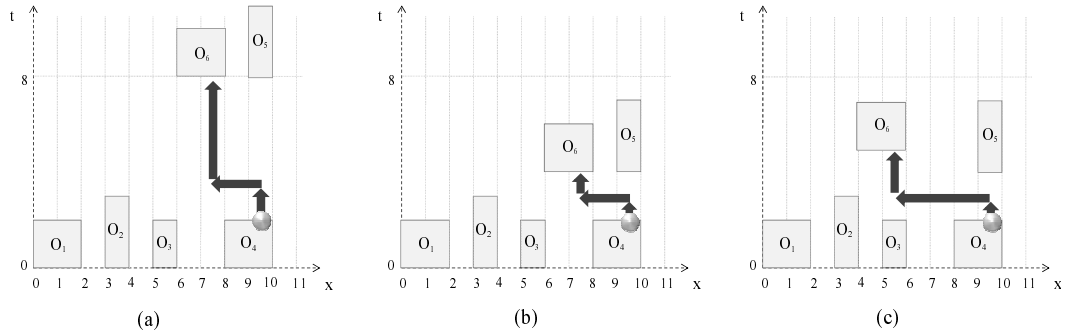


Figure 4.6: Two different fine-tuned solutions show that placement and routing interacts with each other during fine-tuning procedure. (a) An initial routing and placement solution. (b) One possible fine-tuned solution. (c) The other possible fine-tuned solution.

It considers variation to satisfy the routing yield constraint, and also handles contamination and defect. It is seamlessly integrated to the placement technique, which enhances the lab-on-a-chip CAD tool. Note that in the physical-level synthesis, the routing and placement largely impact each other, and our technique performs routing during placement optimization procedures in order to improve the solution. Figure 4.6 shows an example that

the placement and routing interacts with each other during fine-tuning. Figure 4.7 gives the flow chart of the algorithm. Basic routability check is to check whether the grid coarsening placement solution has space for routing and satisfies all the constraints for placement. If not, the perturbation will be performed to make it pass the basic routability check.

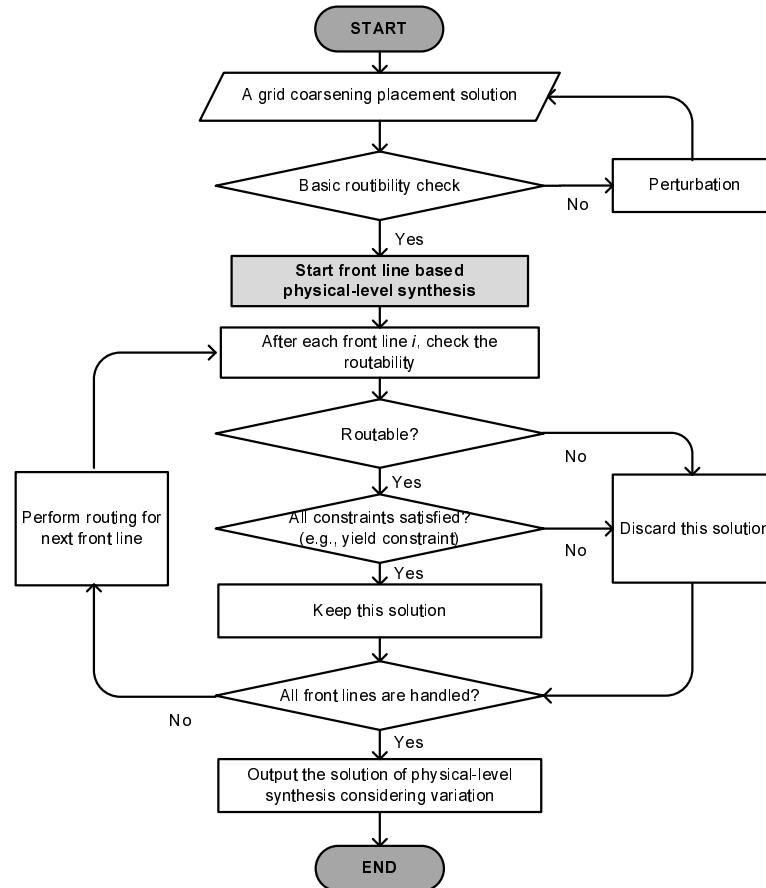


Figure 4.7: The flow chart of the proposed physical-level synthesis algorithm considering variation.

4.3.1 Maze Routing For Lab-on-a-chip Routing Considering Contamination and Defect

Maze routing is a standard technique in VLSI CAD design, and it has also been used for biochip routing, such as [109, 104, 110]. Maze routing divides the entire routing space as a set of unit grids, and parts of which might be blocked or already occupied by the existing routes. It continues to increase the concentric circles centering at the source until one circle reaches the destination. A path can be then obtained by counting downward through the circles [111]. If each grid is associated with a weight, the maze routing is also able to compute a minimum weight route. Figure 4.8 illustrates the maze routing procedure. For simplicity, it is a 2-D maze routing, and each grid is with equal weight except the blocked/occupied grids. Starting at the source, the radius of the circles grows from 1 to 8 until the circle reaches the destination. Tracing back in a circular fashion, a minimum weight route can be obtained. [110] also utilizes maze routing for lab-on-a-chip routing. However, it does not consider variation, contamination and defect.

In our problem, given a source and a destination, 3-D maze routing is utilized to compute the route. Each grid in the 3-D space is associated with an equal weight unless specified. If the grid is occupied by a module or an existing route, a very large weight will be then assigned such that no other route through this grid will be the minimum weight route. A route with the minimum weight among all possible routes will be chosen as the best/shortest

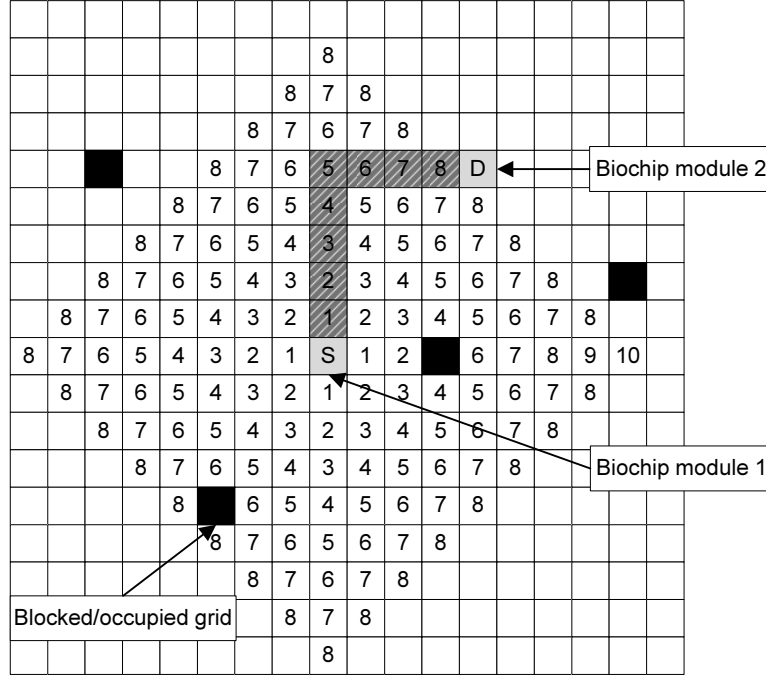


Figure 4.8: An illustration of 2-D maze routing (x and y coordinates). S denotes source, and D denotes destination. S and D could be lab-on-a-chip module 1 and lab-on-a-chip module 2. A black grid denotes a blocked/occupied grid. Each grid is with a unit weight except that the black grids are with a weight of g . It shows a possible minimum weight route from the source to the destination. It needs 8 grids to route from the source to the destination.

route for transporting the droplet. Note that in our case there is a restriction, i.e., when route along the t -dimension, it chooses the route from down to up, but not up to down. This is due to that t -dimension represents the time.

The proposed technique can handle the fluidic constraint. Recall that fluidic constraint states that the minimum spacing between two droplets is one grid. When a route is chosen to transport the droplet, all the grids adjacent to the grids on this route will be assigned with a very large weight. It guarantees that these grids will not be chosen any more such that the fluidic constraint is satisfied. The proposed technique can also handle the timing

constraint. Given the timing constraint t_{\max} , when routing length in a route is greater than t_{\max} , this route will be discarded. Refer to Figure 4.2. There are two routes from module 1 to module 2, i.e., the black one and the grey one. The grey one may violate the timing constraint since the routing length might be greater than t_{\max} . Thus, the grey one will not be kept.

The proposed technique can handle the contamination and defect. Given a set of defective grids with certain coordination x, y in the lab-on-a-chip, these grids can not be used for routing for all the time, i.e., any possible t . Given a set of contaminated grids with certain coordinations, these grids need to be disable for routing for a certain time. For example, a contaminated grid is at $\{x, y, t\}$, and the wash time to clear the contamination is t_c , then the grids $\{x, y, t\}, \{x, y, t + 1\}, \dots, \{x, y, t + t_c\}$ cannot be used in the routing. Thus, the proposed technique will assign a very large weight to these grids to avoid these grids. Refer to Figure 4.9 for an example. In this example, the contaminated grid can not be used in routing for at least 2 time units, and the defective grid can not be used in routing at all.

4.3.2 Physical-Level Synthesis Considering Variation

Recall that multi-scale optimization technique including grid coarsening and fine-scale tuning in [5] only considers placement. In fine-scale tuning for placement, the modules below the current front line are fixed, and the modules above the current front line can

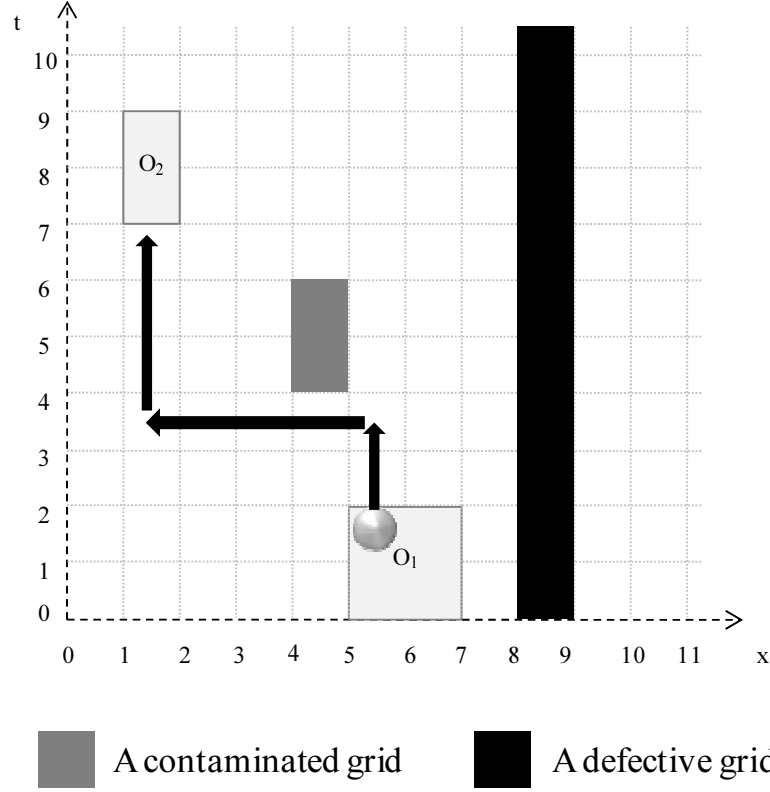


Figure 4.9: An example to illustrate the defective grid and contaminated grid in 2-D, i.e., x and t coordinates.

explore better placement in their local regions. The above procedure neglects the routing, and it may result in the negative effect on physical-level synthesis. That is, it may have no space for routing, which makes the synthesis fail. Based on the initial placement solution using [5], our synthesis technique seamlessly integrates routing to the multi-scale optimization technique for placement. Considering variation, it largely enhances the design technique to be aware of routing yield during routing and placing.

At a high level, our technique for physical-level synthesis performs routing in each front line based iteration of the fine-scale tuning while considering the constraints. The route ordering for each pair of modules is according to the topological order in the precedence

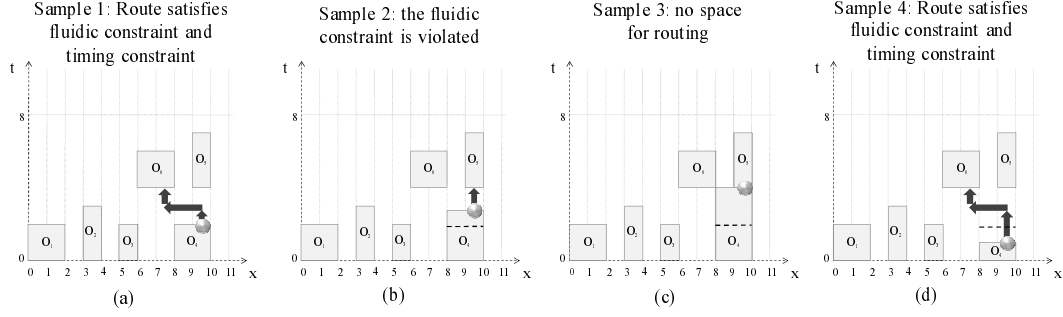


Figure 4.10: Different module variational heights result in different physical-level synthesis samples of the fine-tuned solution based on initial solution in Figure 4.6(a). (a) A fine-scale tuning solution for Figure 4.6(a), and sample 1 of routing and placement. It satisfies the timing constraint and fluidic constraint. (b) Sample 2 violates the fluidic constraint. (c) Sample 3 has no space for routing. (d) Sample 4 satisfies the timing constraint and fluidic constraint.

graph. If the fine-scale tuned design for i -th front line is routable such that all the constraints are satisfied, the fine-scale tuned solution is taken instead of the original design. Otherwise, this solution will not be taken. Due to that [5] does not consider routing, it could happen that even before fine-scale tuning, the design is not routable. In this case, some perturbation of the modules will be performed to make it routable. For example, move up some modules, or, switch two of the modules as long as all the constraints are satisfied. In this fashion, it guarantees that the design using our technique has space to route and satisfies all constraints. After the last front line, the last feasible solution is returned as the solution.

In detail, the initial coarsened placement is utilized as the input. The basic routing check is first performed to see whether it is routable. If not, some perturbation of the modules will be performed to improve it to be. In our simulation, it moves up some adjacent modules, or moves some adjacent modules parallel to other unoccupied space with the

same t -dimension, by which those modules are not adjacent to others. After that, it starts with the 1st front line. The modules whose lower left corner below the 1st front line are fixed as in the initial coarsened solution, and the others will search a better placement in their local region. The routing will be performed after the search. If the resulting routing can satisfy the fluidic constraint, timing constraint as well as the routing yield constraint, this solution will be kept. Otherwise, this solution will be discarded. It will then proceed to next front line, i.e., 2nd front line. The above procedures are iterated until all the front lines are processed. A synthesized solution with the routing and tuned placement can be obtained, which guarantees that all the constraints including the routing yield constraints are satisfied. Figure 4.11 shows the detailed flow.

The routing yield constraint is checked during fine-scale tuning to consider variation as follows. To evaluate routing yield, the technique of Latin Hypercube sampling is utilized. Refer to Section 4.3.3 for the details. Given a certain physical-level synthesis solution, a number of values with the module variational heights are generated according to Gaussian distribution. Some variational heights would lead to the failure of the design due to that a grid initially assigned for the routing may be occupied by the module. Refer to Figure 4.3 for an example. Thus, the routing yield is the ratio of the number of samples with successful routing over the total number of samples. When a new solution is obtained after proceeding to i -th front line, the routing yield of this design will be computed. The *routing yield constraint* is that, if the routing yield is greater than the user defined number, e.g., 99%, this solution is taken, and the algorithm proceeds to next front line. Otherwise, this solution will

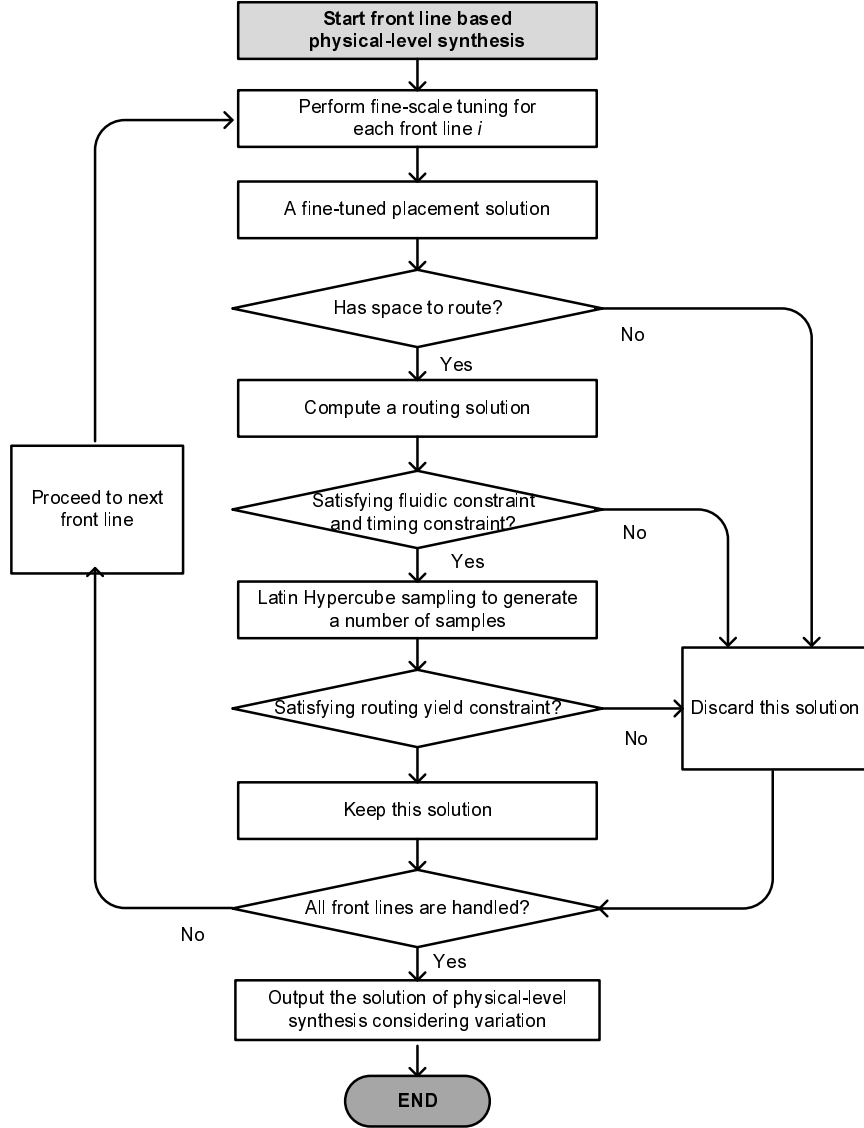


Figure 4.11: The detailed flow chart of our front line based physical-level synthesis.

be discarded and the previous solution is kept. The algorithm will proceed to $(i + 1)$ -th front line similarly. Consequently, this technique guarantees that after each round of fine-scale tuning, the design always satisfies the routing yield constraint. Refer to Figure 4.10 for an illustration. A fine-scale tuning solution is computed from a grid coarsening solution in Figure 4.6(a). Suppose that there are totally 4 physical-level synthesis samples due to the

different module variational heights. 2 samples violates the constraint. Thus, the routing yield will be only 50%, which is less than 99%. Thus, the routing yield constraint is not satisfied and this fine-tuned solution will be discarded.

4.3.3 Computation of Routing Yield Using Latin Hypercube Sampling

Latin Hypercube sampling technique is first proposed in [100] and has been often utilized in analysis with uncertain data. Take two-dimensional Latin Hypercube sampling as an example. The range of each variable is divided to bins with equal probability. For each bin, a sample value for this variable is generated. As shown in the example in Figure 4.12, it has two variables, and there are five bins for each variable. It only needs five samples to well cover the whole space. The advantage of Latin Hypercube sampling is that it only needs a small number of samples while maintaining the same accuracy. The Latin Hypercube sampling is used for routing yield estimation. With a certain physical-level synthesis solution, a set of sample values for the module variational heights are generated based on Latin Hypercube sampling according to Gaussian distribution. These variational heights result in different physical-level synthesis samples and are utilized to compute the routing yield.

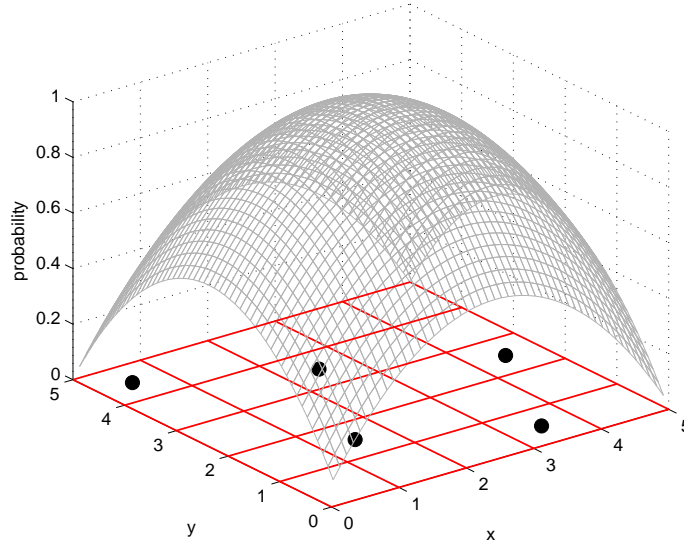


Figure 4.12: An example to illustrate Latin Hypercube sampling. Green grids represent the probability density, blue grids represent the intervals with equal probability, and red points are Latin Hypercube samples.

4.4 Simulation Results

In the simulation, the proposed technique considering variation is implemented in C++, and tested on a Pentium IV machine with $1.86GHz$ CPU and $3GB$ main memory. We conduct the simulation on a set of standard testcases used in [5, 3]. We would like to compare our simulation results with the previous works. However, since this is the first work for lab-on-a-chip physical-level synthesis considering variation, contamination and defect, there is no previous work for comparison. Thus, a set of simulations of the design without considering variation are also conducted. Recall that the routing constraints include fluidic, timing and routing yield constraints, and the placement constraint includes the

non-overlapping, resource and scheduling constraints. Denote the proposed technique considering variation by ROUTE with V. Denote the technique without routing yield constraint by ROUTE without V. A set of simulations considering contamination and defect are performed to show that the proposed technique can well handle them. Denote our technique considering contamination and defect by ROUTE with V, D/C.

In the simulation, Gaussian distribution is utilized for variation distribution. Note that we could also use other distributions. The σ in Gaussian distribution is set to $1/30$. Thus, 3σ is 10% and the variational range \hat{h} is set to 10% of h . It coarsens the time dimension by a factor of $\kappa = 12$ on t for the first 4 testcases and $\kappa = 4$ on t for the last 6 testcases, since this setting obtains the best tradeoff between runtime and solution quality. The simulation results of the comparison between ROUTE without V and ROUTE with V are summarized in Table 4.1. Table 4.2 shows the worst case design. Figure 4.13 illustrates one routing solution for vitro3-1 using ROUTE without V and one using ROUTE with V. The following observations can be obtained.

† Without considering variation, the optimization solutions using both of ROUTE without V and ROUTE with V are routable. That is, all the droplets can be transported from their source to destination while satisfying the timing constraint and fluidic constraint.

† With considering variation, ROUTE with V is always able to satisfy all design constraints, including routing yield constraint. For example, for the design vitro1-1,

the routing yield is 100.0%. In contrast, ROUTE without V can not satisfy the routing yield constraint. For example, for the design vitro1-1, the routing yield is only 15.0%. Thus, the routing yield improvement of vitro1-1 is as high as 85.0%.

† The average routing yield improvement of all the testcases is 51.2%. The designs using ROUTE with V satisfying routing yield constraint demonstrates that they are insensitive to the variation.

† In terms of overall completion time, compared to ROUTE without V, ROUTE with V computes the solutions with smaller completion time for two testcases, i.e., vitro1-3 and vitro1-4, while ROUTE with V computes the solution with larger completion time for the other testcases.

† It can be seen that there is a tradeoff between overall completion time and routing yield. ROUTE without V saves some completion time in the design. However, the routing yield using it is much worse. Although ROUTE with V may need more completion time, the average completion time increase is only 3.5%, while the average routing yield improvement can be 51.2%. Since it is critical to consider the impact of variation to the biochemical operations, it is worth while to satisfy routing yield constraint through sacrificing some completion time.

† The routing length for the droplet transportation of ROUTE with V is longer than that of ROUTE without V. This is due to that considering the variation, the modules might have larger distance between each other.

† The runtime of ROUTE with V is similar to ROUTE without V. This demonstrates

that our design with considering variation does not necessarily take more runtime during the optimization compared to the design without considering variation.

† The result in [5] shows a shorter overall completion time compared to ROUTE with V. For example, the overall completion time of vitro3-1 using ROUTE with V is 52 while that in [5] is only 45. However, [5] does not consider routing at all, and its solution may have no space for routing. Refer to the example for testcase vitro3-1 shown in Figure 4.4. The placement yield in [5] is only for the placement. Although the placement yield is 100% for vitro3-1, its routing yield is close to 0%. We have the similar observations on other testcases. Thus, the small overall completion time and high placement yield in [5] are not useful for the physical-level synthesis.

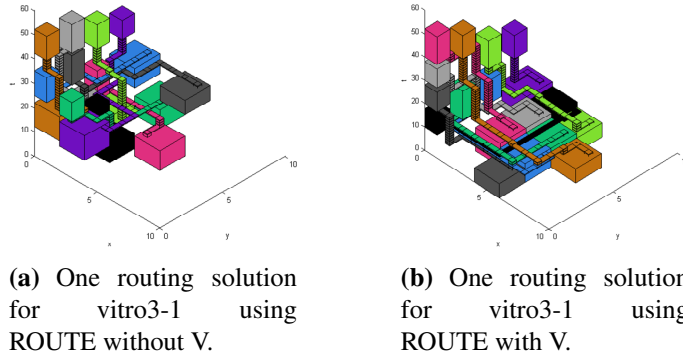


Figure 4.13: Routing solution for vitro3-1. The droplets are transported from the sources to the destinations. A pair of source and destination, and the corresponding routing are painted by the same color.

To demonstrate that ROUTE with V can handle contamination and defect in the lab-on-a-chip, we also conduct a set of simulations considering contamination and defect,

Table 4.1

The comparison of ROUTE without V and ROUTE with V. Compl. time refers to the overall completion time of the last module in the solution of the lab-on-a-chip placement and routing. R. length is the total routing length which is the total number of grids needed for the transportation of droplets in this design. R. yield is routing yield. CPU refers to the runtime in seconds. Compl. time increase is computed through comparing the completion time of ROUTE without V and ROUTE with V. R. yield impr. is the routing yield improvement computed through comparing the routing yield of ROUTE without V and ROUTE with V.

Testcase	ROUTE without V				ROUTE with V					
	Compl. time	R. length	R. yield	CPU(s)	Compl. time	R. length	R. yield	CPU(s)	Compl. time increase	R. yield impr.
vitro1-1	62	310	15.0%	2666.4	76	316	100.0%	2226.0	0%	85.0%
vitro1-2	62	348	15.0%	2772.9	76	352	100.0%	2355.3	0%	85.0%
vitro1-3	74	340	63.0%	1421.5	73	350	100.0%	1112.8	-1.4%	37.0%
vitro1-4	75	390	64.0%	2625.7	74	401	100.0%	1012.9	-1.3%	36.0%
vitro2-1	49	190	41.5%	3271.8	53	209	100.0%	2499.0	8.2%	58.5%
vitro2-2	48	206	62.5%	2839.4	53	263	100.0%	1663.5	10.4%	37.5%
vitro2-3	48	162	56.0%	1666.8	51	185	100.0%	1801.0	6.3%	44.0%
vitro3-1	52	158	46.0%	653.6	52	182	100.0%	728.2	0%	54.0%
vitro3-2	48	142	69.0%	758.2	53	142	100.0%	716.2	10.4%	31.0%
vitro3-3	48	120	56.0%	648.3	49	122	100.0%	641.3	2.1%	44.0%
Average									3.5%	51.2%

i.e., ROUTE with V, D/C. It arbitrarily selects a set of defective grids and contaminated grids among all the grids before placement. Subsequently, both of placement and routing can not use these grids. Table 4.3 summarizes the simulation results. The number of defective grid is 5 and that of contaminated grid is also 5. Note that other numbers could also be handled. The following observations can be made.

† The contamination and defect have some impact on the overall completion time of all the testcases. For example, for the design vitro1-1, the completion time increases to 77, while the previous completion time without considering contamination and defect is 76. This is due to that the defective and contaminated grids are arbitrarily

Table 4.2

The worst case design of ROUTE with V. R. length is the total routing length which is the total number of grids needed for the transportation of droplets in this design. R. yield is routing yield.

Testcase	Compl. time	R. length	R. yield
vitro1-1	83	584	100%
vitro1-2	83	584	100%
vitro1-3	102	616	100%
vitro1-4	74	570	100%
vitro2-1	54	261	100%
vitro2-2	67	240	100%
vitro2-3	61	269	100%
vitro3-1	62	222	100%
vitro3-2	64	205	100%
vitro3-3	61	164	100%

generated, and the placement of modules and routing may need more space to avoid these grids. However, some increase of completion time is reasonable in order to keep all the modules/operations effective.

† The average violation percentage of ROUTE with V, D/C is 0%, while that of ROUTE with V is 17%. It shows that most of the routings in ROUTE with V would fail since the defective/contaminated grids are used. In contrast, ROUTE with V, D/C avoids using those grids and can obtain the successful solution.

† The routing yield is as good as the previous designs using ROUTE with V which does not consider the contamination and defect. This demonstrates that our design considering variation is able to well handle the contamination and defect.

† Since a few grids can not be utilized for routing, they result in the increase of some routing length.

Table 4.3

The comparison of ROUTE with V and ROUTE with V, D/C. # of D, C refers to the number of defective grid and contaminated grid. Compl. time refers to the overall completion time of the last module in the solution of the lab-on-a-chip placement and routing. R. length is the total routing length which is the total number of grids needed for the transportation of droplets in this design. R. yield is routing yield. Violation percentage refers to the number of defective/contaminated grids used in the routing over the total number of defective/contaminated grids.

Testcase Testcase	# of D, C	ROUTE with V				ROUTE with V, D/C			
		Compl. time	R. length	R. yield	Violation percentage	Compl. time	R. length	R. yield	Violation percentage
vitro1-1	5,5	76	316	100.0%	20.0%	77	513	100.0%	0.0%
vitro1-2	5,5	76	352	100.0%	30.0%	78	547	100.0%	0.0%
vitro1-3	5,5	73	350	100.0%	30.0%	89	535	100.0%	0.0%
vitro1-4	5,5	74	401	100.0%	20.0%	86	552	99.0%	0.0%
vitro2-1	5,5	53	209	100.0%	10.0%	73	321	100.0%	0.0%
vitro2-2	5,5	53	263	100.0%	0.0%	54	257	99.0%	0.0%
vitro2-3	5,5	51	185	100.0%	20.0%	54	286	100.0%	0.0%
vitro3-1	5,5	52	182	100.0%	10.0%	53	215	99.0%	0.0%
vitro3-2	5,5	53	142	100.0%	20.0%	56	174	100.0%	0.0%
vitro3-3	5,5	49	122	100.0%	10.0%	50	154	99.0%	0.0%
average					17.0%				0.0%

4.5 Summary

This is the first physical-level synthesis work for lab-on-a-chip which considers variation, contamination and defect. It proposes the maze routing based, variation, contamination and defect aware droplet routing technique, and it seamlessly integrates the routing to the placement technique in [5]. The simulation results demonstrate that the proposed technique can route the droplets using a small number of grids considering contamination and defect. There is no violation of using the defective/contaminated grids through the proposed technique, while the technique without considering contamination and defect

uses 17% of the defective/contaminated grids on average. On the other hand, without considering variation, the routing yield is very small which may result in the failure of the design. With considering variation, our routing technique can successfully route the droplets for a set of standard testcases while satisfying the routing yield constraint with only a little increase of completion time. Compared to that without considering variation, the average routing yield improvement is as high as 51.2% while the average completion time increase is only 3.5%.

Chapter 5

Approximation Scheme For Restricted Discrete Gate Sizing Targeting Delay Minimization¹

5.1 Introduction

The increasing chip density leads to the extensive use of gate sizing optimization in the combinational circuit design [39, 40, 41, 42, 43]. Effective algorithms for gate sizing are

¹©Springer and the original publisher, Journal of Combinatorial Optimization, vol. 21, issue 4, 2009, pp. 497 - 510, “*Approximation Scheme For Restricted Discrete Gate Sizing Targeting Delay Minimization*”, Chen Liao and Shiyang Hu, original copyright notice is given to the publication in which the material was originally published, by adding; with kind permission from Springer Science and Business Media.

highly desirable to improve the design quality especially in terms of delay minimization and power saving. A large multitude of previous works with different objectives have been designed. The standard gate sizing techniques for exploring delay and power tradeoff are proposed in [39, 40, 41, 43, 44, 45, 46, 47]. As the extensions to them, gate sizing techniques considering process variations are designed in [48, 49, 50], gate sizing techniques for cross-talk noise reduction are proposed in [51, 52], a reliability driven gate sizing technique is proposed in [54], and a security aware gate sizing technique is proposed in [55].

Unfortunately, most of the existing techniques such as a Lagrangian relaxation based technique in [40] and a posynomial programming based approach in [44] can only handle the continuous gate sizing problem which assumes that gate sizes can be any values within certain range [56]. This assumption is not realistic since it is difficult and not practical to manufacture gates with continuous sizes. In practice, only a small set of gate sizes are available, which imposes a pressing need for the techniques to handle discrete gate sizes. Precisely, the discrete gate sizing problem is to assign a size to each gate from a given set of available gate sizes such that the circuit delay is minimized while the cost target is satisfied. This problem is known as strongly NP-hard [57]. To obtain a discrete gate sizing solution, rounding the sizes of a continuous solution to discrete sizes is fast and intuitive. However, it will result in the significant degradation of circuit delay compared to the obtained continuous gate sizing solution [56, 58]. This motivates some recent works to design combinatorial algorithms which directly handle discrete gate size, such

as a continuous solution guided dynamic programming technique in [56], a network-flow based approach in [59], a parallelization and randomization based technique in [60], and a multi-dimensional gradient descent based algorithm in [61]. These algorithms are effective, however, they are all heuristics without any theoretical guarantee on the quality of their discrete gate sizing solutions. This limits the understanding of the discrete gate sizing problem in theory.

This chapter aims to deepen the understanding of the discrete gate sizing problem from the theoretical point of view. Recall that given a minimization problem, an algorithm is said to approximate the optimal solution within a factor α if this algorithm can always produce a solution whose objective function value is at most α times the value of the optimal solution. The problem admits a fully polynomial time approximation scheme (FPTAS) if there is an algorithm which approximates the optimal solution within a factor of $(1 + \varepsilon)$ for any $\varepsilon > 0$ and runs in time polynomial in both of the input size and $1/\varepsilon$.

In this chapter, the first fully polynomial time approximation scheme is designed for the delay driven discrete gate sizing problem. The algorithm works under the scaling and rounding framework of [112, 113, 114]. The proposed approximation scheme involves a level based dynamic programming algorithm which handles the specific structures in gate sizing problem and adopts an efficient oracle query procedure. It can approximate the optimal gate sizing solution within a factor of $(1 + \varepsilon)$ in $O(n^{1+c}m^{2c}/\varepsilon^c)$ time for $0 < \varepsilon < 1$ and in $O(n^{1+c}m^{2c})$ time for $\varepsilon \geq 1$, where n is the number of gates, m is the maximum

number of gate sizes for any gate, and the constant c is the maximum number of gates per level. The technique needs the assumption that c is a constant, which is why our algorithm is said to approximate the restricted discrete gate sizing. Due to the fact that the discrete gate sizing problem is strongly NP-hard [57], making such an assumption is reasonable.

The rest of the chapter is organized as follows: Section 5.2 presents the notations and the problem formulation of the discrete gate sizing problem. Section 5.3 proposes our approximation scheme to solve the discrete gate sizing problem and analyzes the time complexity and approximation ratio. A summary of work is given in Section 5.4.

5.2 Preliminaries

5.2.1 Notations and Definitions

A combinational circuit can be represented by a directed acyclic graph (DAG). Given a DAG $G = (V, E)$ with $n = |V|$ nodes, each node corresponds to a gate. Following the convention of the gate sizing literature, let *the primary input gates*, denoted by PI , specify the nodes with zero in-degree, and *the primary output gates*, denoted by PO , specify the nodes with zero out-degree. In the gate sizing problem, the arrival times at all primary input gates are 0. The arrival time of a gate is defined as the time when the current arrives at the input of the gate, and the gate delay in this chapter is computed according to Elmore delay

model, i.e., it refers to the product of its resistance and the total capacitance of all its fan-out gates. The delay of a circuit refers to the maximum arrival time at any primary output gate of the circuit. Discrete gate sizing is to minimize the circuit delay through appropriately assigning the gate size at each gate since different gate sizes lead to different delays. Refer to Figure 5.1. Think of pushing a flow into the primary input gates of G and it goes from g_1 to g_3 . We call that g_1 is upstream to g_3 , and g_3 is downstream to g_1 . Denote by $fin(g)$ the set of fan-in (input) gates of gate g and by $fout(g)$ the set of fan-out (output) gates of gate g . For example, $fin(g_2) = \{g_1\}$ and $fout(g_1) = \{g_2\}$. For each gate g , denote by $s(g)$ the assigned gate size. It needs to be in a set of available gate sizes for g , denoted by $S(g)$. Denote by m the maximum number of gate sizes for any gate, i.e., $|S(g)| \leq m, \forall g$.

The circuit will be partitioned by levels. Initially, we would let *level-1 gates* specify all the primary input gates and let *level-2 gates* specify the gates immediately downstream to level 1 gates. However, since a gate g can be reached through different paths from primary input gates, it can belong to different levels according to the above definition. Thus, for each gate, its level is defined as the minimum possible level reachable from any primary input gate. In addition, for a gate g , if there exists a path from g to any level- i gate g' , the level of g will be no greater than i , the level of g' . This means that whenever a gate g' is included into a level i , we also include all the gates along any path to g' to level i provided that they are not yet included in any of the previous levels (level $1, 2, \dots, i-1$). Denote by c the maximum number of gates in any level. Our FPTAS needs the assumption that there are constant gates in each level, i.e., $c = O(1)$. Thus, it is said to approximate the restricted

discrete gate sizing.

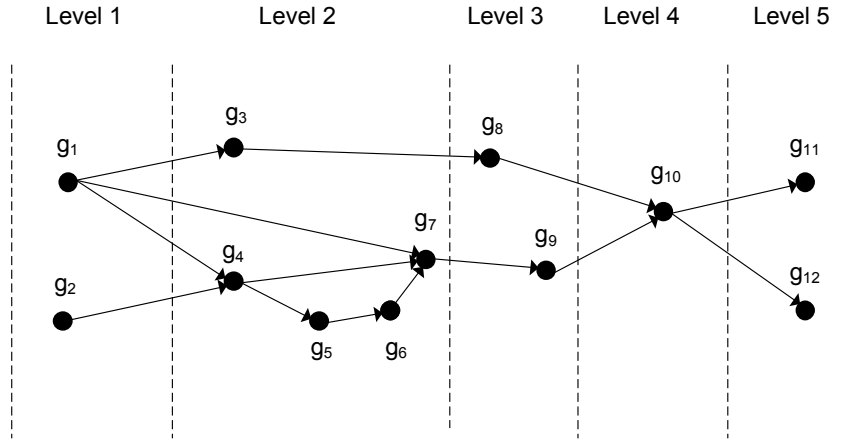


Figure 5.1: Illustration of levels.

It is helpful to look at an example to illustrate the concept of level. Refer to Figure 5.1 where the levels for gates are shown. After g_4 is classified as a level-2 gate, g_5 would be classified as a level-3 gate. However, g_7 is also classified as a level-2 gate since it connects to g_1 which is a level-1 gate. Backtracking the graph from g_7 , g_6 and g_5 can be reached and they are not yet classified. Thus, they are also included in level 2. The backtracking stops when a gate with classified level is encountered. The total backtracking time (summing up backtracking time over all nodes) takes $O(|E|) = O(n^2)$ time which will be bounded by the FPTAS time. In summary, in Figure 5.1, $L_1 = \{g_1, g_2\}$, $L_2 = \{g_3, g_4, g_5, g_6, g_7\}$, $L_3 = \{g_8, g_9\}$, $L_4 = \{g_{10}\}$, and $L_5 = \{g_{11}, g_{12}\}$. In addition, $c = 5$ which is due to the level-2 gates.

Recall that the circuit delay is defined as the arrival time at any circuit primary output gate.

Since the delay along a path is the sum of delay of each gate along the path, equivalently,

discrete gate sizing problem is to minimize the longest path delay. In this chapter, the gate delay is computed using *Elmore delay* model, which is widely used in VLSI design [115]. The delay of a gate is equal to the product of its resistance and the capacitance of all its immediate downstream gates. Note that the gate size determines the resistance and capacitance of a gate, and gate delay is related to the capacitance of its fan-out gates rather than the capacitance of itself.

Refer to the example in Figure 5.2 for better understanding of the arrival time and gate delay. Suppose that in the example, g_1 is assigned with gate size 2, g_2 is assigned with gate size 5, g_3 is assigned with gate size 1, g_4 is assigned with gate size 1, and g_5 is assigned with gate size 7. Denote by $t(g)$ the arrival time at the input of a gate g . Recall that the arrival time of a gate is defined as the time when the current arrives at the input of the gate. For example, the arrival time at g_2 is shown as $t(g_2)$ in the figure. Let $d(g, s(g))$ denote the gate delay of g when g is assigned with gate size $s(g)$. This incorporates the fact that different gate size assignment will lead to different gate delay. Similarly, let $C(g, s(g))$ and $R(g, s(g))$ denote the capacitance and resistance of gate g when it is assigned with gate size $s(g)$, respectively. In Figure 5.2, g_1 is the primary input gate which has arrival time 0. The arrival time at g_2 , i.e., $t(g_2)$, is equal to the delay of g_1 , i.e., $t(g_2) = d(g_1, 2) = R(g_1, 2)C(g_2, 5)$. $t(g_3)$ and $t(g_4)$ are both equal to $t(g_2)$ plus the delay of g_2 , i.e., $t(g_3) = t(g_4) = t(g_2) + d(g_2, 5) = t(g_2) + R(g_2, 5)(C(g_3, 1) + C(g_4, 1))$. For the gate with in-degree greater than 1, the maximum arrival time needs to be taken to guarantee the worst-case circuit performance. For example, $t(g_5) = \max\{t(g_3) + d(g_3, 1), t(g_4) +$

$$d(g_4, 1)\} = \max\{t(g_3) + R(g_3, 1)C(g_5, 7), t(g_4) + R(g_4, 1)C(g_5, 7)\}.$$

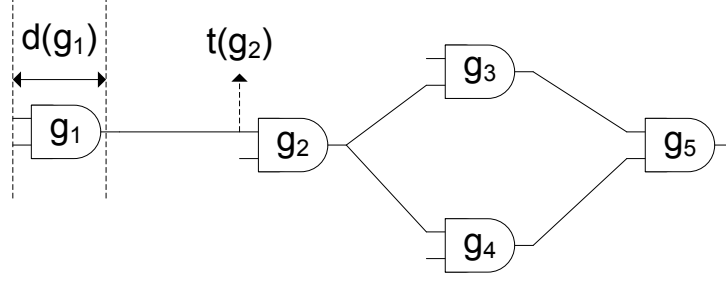


Figure 5.2: Illustration of gate delay d and arrival time t .

On the other hand, the circuit cost will be also impacted by gate size assignment. Each gate g at size s is associated with a cost $w(s(g))$. The cost of a set of gates is defined as the sum of costs of all gates. The cost constraint W says that the sum of costs of all gates in G needs to be less than or equal to W .

The following notations will be used in our FPTAS design. Let T^* denote the delay of the optimal gate sizing solution. Let \bar{T} and \underline{T} denote certain upper bound and lower bound on T^* , respectively. Various techniques can be used to obtain \bar{T} and \underline{T} . For example, \bar{T} can be obtained from always using the largest capacitance and the largest resistance at any gate (even if they belong to different sizes), ignoring the cost constraint. Similarly, \underline{T} can be obtained from always using the smallest capacitance and the smallest resistance at any gate, ignoring the cost constraint. \bar{T} and \underline{T} can be certainly computed in linear time through evaluating the circuit delay (by a traversal) in each case. Note that although the real lower bound and upper bound may be different, the bounds make sense due to that the optimal solution must be in this range and can be searched out. On the other hand, as our FPTAS is

independent of \overline{T} and \underline{T} , it is not important to design techniques which can generate tighter upper or lower bounds.

Some notations frequently used in this chapter are summarized as follows.

- † n : the total number of gates.
- † m : the maximum number of gate sizes for any gate.
- † c : the maximum number of gates in any level.
- † T^* : delay of the optimal discrete gate sizing solution.
- † \underline{T} : a lower bound on T^* .
- † \overline{T} : an upper bound on T^* .
- † $t(g)$: the arrival time at the input of a gate g .
- † L_i : the set of gates in level i .

5.2.2 Problem Formulation

Discrete Gate Sizing Problem: Given a DAG $G = (V, E)$, and a set $S(g)$ of available gate sizes for gate g , to compute a gate size assignment at each gate in G from the available gate sizes such that the arrival time at any primary output gate (circuit delay) is minimized while the cost constraint W is satisfied.

Instead of enumerating all paths to obtain the circuit delay, the discrete gate sizing problem is usually formulated to a mathematical programming problem as follows [56].

$$\begin{aligned}
\min \quad & T \\
s.t. \quad & \sum_{\forall g_i} w(s(g_i)) \leq W, \\
& t(g_i) \leq T, \forall g_i \in PO \\
& t(g_i) + d(g_i, s(g_i)) \leq t(g_j), \forall g_i \in fin(g_j), \\
& d(g_i, s(g_i)) = R(g_i, s(g_i)) \sum_{\forall g_j \in fout(g_i)} C(g_j, s(g_j)), \\
& 0 \leq t(g_i), \forall g_i \in PI \\
& s(g_i) \in S(g_i), \forall g_i.
\end{aligned} \tag{5.1}$$

It is clear that the delay of a gate depends on the gate size assignment on its immediate downstream (fan-out) gates and the arrival time at a gate depends on the gate assignments of all the gates along any path from primary input gates to the gate.

Note that as the same as many previous works, the objective of our discrete gate sizing problem is to minimize the longest path delay. With more than one paths in a combinational circuit, the path with the largest delay is defined as the critical path, and its corresponding delay is equivalent to the circuit delay T .

5.3 The Discrete Gate Sizing Algorithm

5.3.1 Overview of the Algorithm

Our FPTAS is motivated from [116, 117] and it works under the framework of the classic scaling and rounding based FPTAS design [112, 113, 114]. In contrast to discrete gate sizing problem, [116, 117] consider different problems, namely, the minimum cost delay driven buffering problem and layer assignment problem. Although the new technique shares some common features with [116, 117] in appearance, there are underlying difference between their technique and our technique. In particular, [116, 117] can only handle tree topology while our discrete gate sizing technique needs to handle DAG which is a larger class of graphs. Due to this, a level based dynamic programming technique is proposed in this work.

At a high level, the FPTAS works in the framework of [112, 113, 114]. Recall that T^* denote the circuit delay of the optimal gate sizing solution. The algorithm first makes a guess on T^* . Denote the guessed value by T . Subsequently, check whether the guessed value T is a good guess, namely, whether it is sufficiently close to (at most ε away from) the optimal cost T^* . If T is a good guess, the corresponding discrete gate sizing solution will be returned as an approximate solution which is at most ε away from the optimal

solution. Otherwise, the other guess is made. This process is repeated until a good guess is obtained.

There are two major algorithmic design issues with the above flow. First, since the optimal solution is not known, how can we tell whether a solution is sufficiently close to the optimal solution? Second, how can we effectively make the new guess to reduce the total number guesses if the current guess is not good? Certainly, one should utilize the information from the previous guesses in generating a possibly good new guess.

For the first issue, a procedure called *oracle* is used to check whether a solution is good. That is, the oracle can approximately decide whether $T^* \geq T$ for any positive number T efficiently. Once we have the oracle in hand, the second issue can be handled by efficient search using oracle. For example, one could perform a binary search between the upper and lower bounds of T^* using the oracle. However, this kind of technique cannot be used in designing the FPTAS since the number of iterations for binary search depends on the initial bounds. If the range between them is unbounded, the FPTAS will run in unbounded time. Thus, an efficient bound independent oracle based search technique proposed in [113] is used in this chapter.

5.3.2 Oracle Construction

5.3.2.1 Level Based Dynamic Programming

We begin with describing how to construct the oracle. The key part of the oracle is a level based dynamic programming algorithm which can efficiently tell either $(1 + \epsilon)T \geq T^*$ or $T < T^*$ for any $T > 0$. The efficiency is achieved by effectively pruning inferior solutions to make the number of solutions polynomially bounded during dynamic programming. By the proposed level based dynamic programming, the following lemma can be reached.

Lemma 1: *Given any $T, \epsilon > 0$, the level based dynamic programming algorithm can compute a solution with the delay at most $(1 + \epsilon)T$, or report that there is no solution which can have delay no greater than T , in $O(nm^{2c} \cdot (n/\epsilon)^c)$ time, where n is the number of gates, m is the maximum number of gate sizes for any gate and the constant c is the maximum number of gates per level.*

Proof: Let V_i denote the set of all the gates up to level i in G . A gate sizing solution on V_i refers to a gate size assignment on gates in V_i . We call such a solution a level- i solution, denoted by $\gamma(V_i)$. Given a level- i solution $\gamma(V_i)$, to model the impact to the next level gates,

it is characterized by

$$(V_i, t(g_1), s(g_1), t(g_2), s(g_2), \dots, \dots, t(g_k), s(g_k), w(V_i)), \quad (5.2)$$

where $L_i = \{g_1, g_2, \dots, g_k\}$. This means that for any gate $g \in L_i$, the solution is characterized by the arrival time $t(g)$ at g when it is assigned to the size of $s(g)$. Further, the solution characterization also includes the cumulative cost so far, denoted by $w(V_i)$. It is computed as the sum of costs for all the gates in V_i , i.e., $w(V_i) = \sum w(s(g)), \forall g \in V_i$. It can be easily seen that a solution is uniquely characterized as in Eqn. (5.2). Further, if there are multiple solutions having the same characterization on $V_i, t(g_1), s(g_1), t(g_2), s(g_2), \dots, \dots, t(g_k), s(g_k)$, only one of them (with smallest cumulative cost $w(V_i)$) needs to be maintained in the dynamic programming and all others are called *redundant* which can be pruned for acceleration. Note that the cumulative cost $w(V_i)$ is the total cost of the set of all gates up to level i , and $\{g_1, g_2, \dots, g_k\}$ are the set of gates in level i . The level based dynamic programming begins with the primary input gates, i.e., L_1 . For each gate g in L_1 , for any size, the arrival time at the input of g is always 0 since it is a primary input gate. For any solution, its cumulative cost can be easily computed by summing up the costs of all gates in L_1 (under the gate size assignment of the solution). Since there are at most m gate sizes for any gate and at most c gates per level, there are at most $O(m^c)$ solutions for V_1 ($V_1 = L_1$).

The level based dynamic programming then proceeds to the second level gates, i.e., L_2 . For

a solution $\gamma_1(V_1)$, we grow it to incorporate the gate size assignment on all level-2 gates. For this, all the possible sizes of level-2 gates are enumerated and a solution is generated from $\gamma_1(V_1)$ per combination. Since there are at most m gate sizes for any gate and there are at most c gates per level, there will be at most m^c new solutions generated from a solution $\gamma_1(V_1)$. This is also the case for other level-1 solutions $\gamma_2(V_1), \gamma_3(V_1), \dots$. Totally, there would be $O(m^c \cdot m^c) = O(m^{2c})$ level-2 solutions.

If this process is continued, the number of solutions would be exponential in terms of the number of levels which is $O(n)$. On the other hand, there are at most $O(m^c)$ possibilities on gate size assignment for the gates in L_2 . If we are able to polynomially bound the number of possibilities on arrival times for all gates in L_2 , the number of solutions can be polynomially bounded. This is due to the following fact. For two solutions with the same gate size assignment and the same arrival time at every gate in L_2 , we only need to pick the solution with smaller cumulative cost (in case of tie, arbitrarily pick one) to propagate in dynamic programming since both solutions have the same impact to the downstream gates except the cumulative cost. This motivates us to use the classic rounding technique to bound the number of possibilities on the arrival time for all gates in L_2 .

Given a solution, for each gate $g_i \in L_2$, round down its arrival time $t(g_i)$ to be the nearest multiple of $T\varepsilon/n$, i.e., $t(g_i) = \lfloor t(g_i)/(T\varepsilon/n) \rfloor \cdot T\varepsilon/n$. Recall that T is the guessed circuit delay and we are only interested in whether there is a solution with circuit delay (approximately) upper bounded by T . Thus, if there is any solution with the arrival time at

any gate larger than T , the solution will be pruned. Due to this, at any gate, there can be at most n/ε distinct arrival times after rounding. Thus, after processing the level-2 gates and the rounding procedure, there can be at most $O(m^c \cdot (n/\varepsilon)^c)$ solutions (m^c distinct gate sizes for L_2 and $(n/\varepsilon)^c$ distinct arrival times for L_2).

It is helpful to see an example to illustrate the above process. Refer to Figure 5.3 where two solutions are presented. In Figure 5.3, $[a, b]$ to the left of a gate g means that its rounded arrival time is a and its gate size is b . The two solutions are level-3 solutions and their gate size assignments at every level-3 gate are the same. Suppose that after rounding, both solutions have the same arrival time at every level-3 gate. Since the second solution has larger cumulative cost $w(V_3) = 12$, it will be pruned.

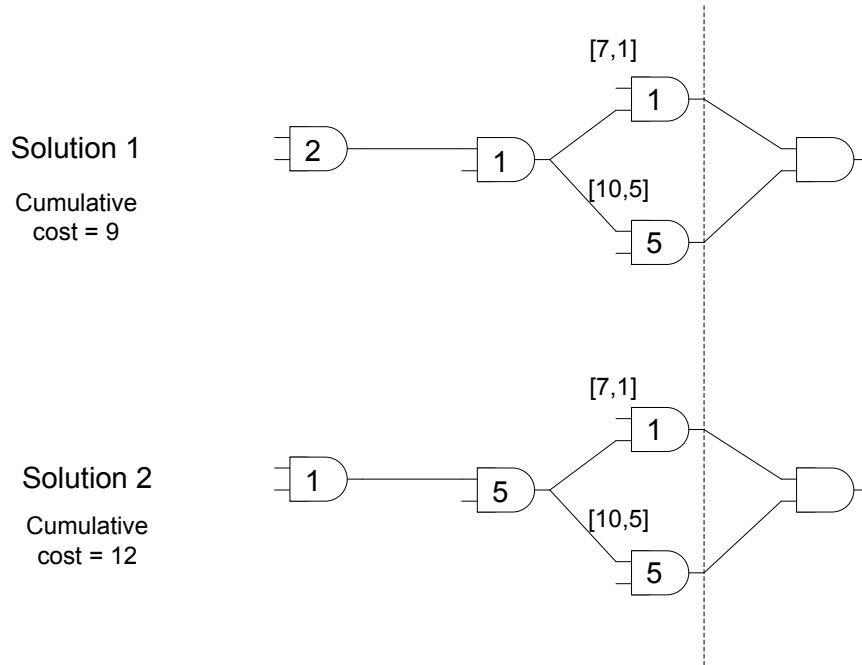


Figure 5.3: An example of pruning where solution 2 is inferior to solution 1.

Suppose that both level-1 gates and level-2 gates have been processed in the above fashion, the level based dynamic programming proceeds to level-3 gates. For each of the $O(m^c \cdot (n/\epsilon)^c)$ level-2 solutions, it can generate at most $O(m^c)$ level-3 solutions. Totally, there would be $O(m^c \cdot (n/\epsilon)^c \cdot m^c)$ solutions. Subsequently, the rounding procedure is performed and at most $O(m^c \cdot (n/\epsilon)^c)$ solutions will be kept after rounding. In general, one can see that there are at most $O(m^c)$ distinct gate sizes and $O((n/\epsilon)^c)$ distinct arrival times for any level. Consequently, the number of solutions at any level is always bounded by $O(m^c \cdot (n/\epsilon)^c)$ after rounding.

In this fashion, the level based dynamic programming proceeds level by level until the last level is handled. At each level, at most $O(m^c \cdot (n/\epsilon)^c \cdot m^c)$ solutions will be generated where only $O(m^c \cdot (n/\epsilon)^c)$ of them are not redundant according to rounding. Note that updating cumulative cost can be easily performed in $O(c)$ time for each solution. The pruning can be implemented using a multi-dimensional array where each entry links to a solution with different gate sizes and rounded arrival times for a level. Locating an entry takes $O(c)$ time. When a new level- i solution is generated, find and compare its cost to the level- i solution with the same gate sizes and arrival times at all level- i gates. If there is no such solution, link the solution to the entry. Otherwise, compare the cost to the cost of the existing solution. If its value is larger, prune the new solution. Otherwise, replace the solution with the new solution. Thus, the time complexity can be bounded as $O(c \cdot m^c \cdot (n/\epsilon)^c \cdot m^c)$ per level and $O(nm^{2c} \cdot (n/\epsilon)^c)$ for all levels, assuming that $O(c) = O(1)$. Note that if T is too small, it is possible that no solution can be generated at any primary output gate (e.g., at a level, all

the solutions have delay greater than T and are pruned).

Recall that the delay of a circuit refers to the maximum arrival time at any primary output gate. Due to the rounding procedure, the obtained circuit delay is not accurate. First, it is a lower bound on the actual circuit delay since only down-rounding is performed. Thus, if the dynamic programming technique cannot find a solution which has delay no greater than T , there is no solution which can have the delay no greater than T with unrounded delay at each gate. Second, since the rounding error in delay for each gate is bounded by $T\varepsilon/n$ which is our rounding factor, together with the fact that there are at most n gates along any path between a primary input gate and a primary output gate, the rounding error for the whole circuit is bounded by $T\varepsilon$. This means that if a solution has the circuit delay of T , its actual circuit delay without rounding is bounded by $(1 + \varepsilon)T$.

5.3.2.2 Oracle Construction

The oracle will decide whether either $(1 + \varepsilon)T \geq T^*$ or $T < T^*$ for any $T > 0$. For this, the oracle calls the level based dynamic programming with the input T and ε . If there is a solution returned, it means that a solution with the circuit delay at most $(1 + \varepsilon)T$ has been obtained. Thus, $T^* \leq (1 + \varepsilon)T$. Otherwise, it means that there is no solution which can have the circuit delay T . In the first case, the oracle will return TRUE. In the second case, the oracle will return FALSE.

5.3.3 The FPTAS

FPTAS works as searching for the best delay T within the lower bound \underline{T} and the upper bound \overline{T} in an iterative manner. None of binary search and logarithmic scale binary search within these bounds can terminate in time independent of the bounds [113, 116]. If the range between them is unbounded, the time complexity of FPTAS will be unbounded. Thus, a technique proposed in [113], which is also used in [116, 117], is adopted to tackle this difficulty. By the proposed FPTAS, we can reach the following theorem.

Theorem 1: *The discrete gate sizing problem can be approximated within a factor of $(1 + \epsilon)$ in $O(n^{1+c}m^{2c}/\epsilon^c)$ time for any $0 < \epsilon < 1$ and in $O(n^{1+c}m^{2c})$ time for any $\epsilon \geq 1$, where n is the number of gates, m is the maximum number of gate sizes for any gate, and the constant c is the maximum number of gates per level.*

Proof: The idea of the searching technique is that instead of sticking to ϵ during optimization, adapting ϵ can significantly improve the runtime. This is due to the fact that the time complexity of the dynamic programming algorithm is inversely proportional to ϵ . If one sets ϵ as a geometrically decreasing sequence leading to ϵ , the total runtime will be bounded by the last run, independent of the number of iterations and the initial bounds.

[113] shows that this is possible. The oracle is called iteratively. In i -th iteration, set

$\varepsilon_i = \sqrt{\overline{T}_i/\underline{T}_i} - 1$ and $T_i = \sqrt{\overline{T}_i \underline{T}_i / (1 + \varepsilon_i)}$. Use T_i and ε_i as the input to the oracle. Depending on the binary result of the oracle, either \overline{T}_{i+1} will be updated to $(1 + \varepsilon)T_i$ (when the oracle returns TRUE) or \underline{T}_{i+1} will be updated to T_i (when the oracle returns FALSE). This process is iterated until $\overline{T}_i/\underline{T}_i < 2$. During iterations, the ratio $\overline{T}_i/\underline{T}_i$ will be progressively reduced as $\overline{T}_{i+1}/\underline{T}_{i+1} = (\overline{T}_i/\underline{T}_i)^{3/4}$ [113]. Note that the total runtime for dynamic programming is in the form of $O(\sum_i nm^{2c} \cdot (n/\varepsilon_i)^c) = O(nm^{2c}n^c \cdot \sum_i (1/\varepsilon_i)^c)$. It is shown in [113] that $1/\varepsilon_i < (2 + \sqrt{2})\sqrt{\overline{T}_i/\underline{T}_i}$. As a result, the runtime bound becomes $O(nm^{2c}n^c \cdot \sum_i (\sqrt{\overline{T}_i/\underline{T}_i})^c)$. Since $c \geq 1$, $O(nm^{2c}n^c \cdot \sum_i (\sqrt{\overline{T}_i/\underline{T}_i})^c) = O(nm^{2c}n^c \cdot (\sum_i \sqrt{\overline{T}_i/\underline{T}_i})^c)$. Together with the fact that $\sum_i \sqrt{\overline{T}_i/\underline{T}_i} = O(1)$ when setting T_i and ε_i as above [113], the total time will be bounded by $O(nm^{2c}n^c \cdot O(1)^c)$.

At some point, $\overline{T}_i/\underline{T}_i < 2$. The above iterative procedure terminates. Similar to [112], the level based dynamic programming is applied using the following setting. For each gate, its arrival time will be down rounded to the nearest multiple of $\underline{T}\varepsilon/n$ where ε is the target approximation ratio. If its arrival time is greater than \overline{T}_i , it will be pruned. Thus, at any gate, there will be at most $2n/\varepsilon$ distinct arrival times. In any level, there are at most $(2n/\varepsilon)^c$ possibilities of arrival times. After the whole circuit is processed, pick the smallest delay T_s with the cost no greater than the cost constraint W to be our solution. Since arrival time is only down-rounded, $T_s \leq T^*$. Rounding the arrival time of this gate sizing solution back, the delay is at most $(1 + \varepsilon)T_s \leq (1 + \varepsilon)T^*$. This single run of dynamic programming takes $O(nm^{2c} \cdot (2n/\varepsilon)^c)$ time since the only difference from Lemma 1 is that we can have $O((2n/\varepsilon)^c)$ possible arrival time combinations at any level instead of

$O((n/\varepsilon)^c)$ possibilities. Together with the runtime for the iterative oracle calls, the total runtime is bounded by $O(nm^{2c}n^c \cdot O(1)^c + nm^{2c} \cdot (2n/\varepsilon)^c)$. This gives an FPTAS when c is a constant.

The pseudo-code for the algorithm is shown in Algorithm 1, Algorithm 2 and Algorithm 3.

Algorithm 1 Level based dynamic programming.

```

DP( $T_r, T, \varepsilon$ )
// there are  $l$  levels in the combinational circuit
 $i \leftarrow 1$ 
while  $i \leq l$  do
  for each level- $(i-1)$  solution do
    generate level- $i$  solutions by enumerating all gate size
    assignment of level- $i$  gates
  end for
  for each level- $i$  solution  $\gamma$  do
    if the cost of  $\gamma$  is  $> W$  then
      remove  $\gamma$ 
    else
      for each level- $i$  gate  $g$  do
        if the arrival time at  $g$  is  $> T$  then
          remove  $\gamma$ 
        else
          round it down to the nearest multiple of  $T_r \varepsilon / n$ 
        end if
      end for
    end if
    if cost of  $\gamma$  is larger than cost of the level- $i$  solution
    with the same arrival time at every level- $i$  gate then
      remove  $\gamma$ 
    else
      replace the solution with  $\gamma$ 
    end if
  end for
   $i \leftarrow i + 1$ 
end while
return the best delay solution or no feasible solution

```

Algorithm 2 The oracle.

ORACLE(T, ε)
if $\text{DP}(T, T, \varepsilon)$ returns a solution **then**
 return TRUE
else
 return FALSE
end if

Algorithm 3 The fully polynomial time approximation scheme for discrete gate sizing problem.

FPTAS($\bar{T}, \underline{T}, \varepsilon$)
while $\bar{T}/\underline{T} > 2$ **do**
 $\varepsilon' \leftarrow \sqrt{\bar{T}/\underline{T}} - 1$
 $T \leftarrow \sqrt{\bar{T}\underline{T}}/(1 + \varepsilon')$
 if $\text{ORACLE}(T, \varepsilon') = \text{TRUE}$ **then**
 $\bar{T} \leftarrow T(1 + \varepsilon')$
 else
 $\underline{T} \leftarrow T$
 end if
end while
return $\text{DP}(\underline{T}, \bar{T}, \varepsilon)$

5.4 Summary

Discrete gate sizing is a critical optimization due to its effectiveness in obtaining various delay and power trade-off in a combinational circuit. However, all of the existing works are heuristics without any theoretical guarantee on the quality of their gate sizing solutions. This chapter designs the first FPTAS for the discrete gate sizing problem. Our algorithm can obtain an approximation within a factor of $(1 + \varepsilon)$ in $O(n^{1+c}m^{2c}/\varepsilon^c)$ time for any $0 < \varepsilon < 1$ and in $O(n^{1+c}m^{2c})$ time for any $\varepsilon \geq 1$, where n is the number of gates, the constant c is the maximum number of gates per level, and m is the maximum number of gate sizes for

any gate. The FPTAS needs the assumption that c is a constant. An interesting future work would be to design an FPTAS with a relaxed assumption.

Chapter 6

Conclusions

This dissertation considers FRTU installation and VLSI physical design. It proposes optimization techniques all of them are based on computer aided design. The first part is feeder remote terminal unit (FRTU) installation considering the security of secondary distribution network in smart grid. The second part consists of two problems. They are physical-level synthesis for microfluidic lab-on-a-chip, including lab-on-a-chip placement and routing, and discrete gate sizing in VLSI circuit.

For the smart grid, a cross entropy based algorithm is proposed to deploy FRTUs in the primary network considering cybersecurity in secondary distribution network. Compared to the greedy algorithm which may always violate the constraint, the proposed algorithm can minimize the cost of the FRTU deployment while satisfying the constraint, i.e.,

whenever a load is tampered, with the probability of λ , one can locate it within a range of η loads, where λ and η are user defined parameters.

For the microfluidic lab-on-a-chips, it considers physical-level synthesis. Physical-level synthesis of lab-on-a-chip includes the lab-on-a-chip placement and routing. Placement is to determine the physical location and the starting time of each operation such that the overall completion time is minimized while satisfying the precedence constraint, non-overlapping constraint and resource constraint. Routing transports the droplets from the source to the destination such that the timing constraint and fluidic constraint are satisfied. In this procedure, variation, contamination and defect need to be considered. This thesis proposes the first optimization technique based on CAD for physical-level synthesis work which considers variation, contamination and defect of the lab-on-a-chip design. It also proposes maze routing based, variation, contamination and defect aware droplet routing technique, and it seamlessly integrates the routing to the placement technique in [5]. The proposed technique improves the placement solution for routing, which may initially have no space for routing, and achieves the placement and routing co-optimization while resisting the negative impact of variation, contamination and defect.

For discrete gate sizing problem, this thesis aims to deepen the understanding of the discrete gate sizing problem from the theoretical point of view. It designs the first fully polynomial time approximation scheme (FPTAS) for the delay driven discrete gate sizing problem. The proposed algorithm can obtain an approximation within a factor of $(1 + \epsilon)$

in $O(n^{1+c}m^{2c}/\epsilon^c)$ time for any $0 < \epsilon < 1$ and in $O(n^{1+c}m^{2c})$ time for any $\epsilon \geq 1$, where n is the number of gates, the constant c is the maximum number of gates per level, and m is the maximum number of gate sizes for any gate. The FPTAS needs the assumption that c is a constant. An interesting future work would be to design an FPTAS with a relaxed assumption.

References

- [1] F. Su, K. Chakrabarty, and R. Fair, “Microfluidics-based biochips: technology issues, implementation platforms, and design automation challenges,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, no. 2, pp. 211–223, 2006.
- [2] K. Bohringer, “Modelling and controlling parallel tasks in droplet-based microfluidic systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 25, no. 2, pp. 329–339, 2006.
- [3] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, “Placement of digital microfluidic biochips using the t-tree formulation,” in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, (San Francisco, United States), pp. 931–934, July 2006.
- [4] F. Su and K. Chakrabarty, “Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips,” in *Proceedings of the Design Automation Conference (DAC)*, (San Diego, United States), pp. 825–830, June 2005.

- [5] C. Liao and S. Hu, “Multi-scale variation-aware techniques for high performance digital microfluidic lab-on-a-chip component placement,” *IEEE Transactions on Nanobioscience*, vol. 10, no. 1, pp. 51–58, 2011.
- [6] H. Khurana, M. Hadley, L. Ning, and D. Frincke, “Smart-grid security issues,” *IEEE Security and Privacy*, vol. 8, no. 1, pp. 81–85, 2010.
- [7] P. McDaniel and S. McLaughlin, “Security and privacy challenges in the smart grid,” *IEEE Security and Privacy*, vol. 7, no. 3, pp. 75–77, 2009.
- [8] R. Zhang, Y. Du, and Y. Liu, “New challenges to power system planning and operation of smart grid development in China,” in *Proceedings of International Conference on Power System Technology*, (Hangzhou, China), pp. 1–8, Oct. 2010.
- [9] H. Farhangi, “The path of the smart grid,” *IEEE Power and Energy Magazine*, vol. 8, no. 1, pp. 18–29, 2010.
- [10] D. Owens, “Time to speak up! get involved in developing smart grid standards,” *IEEE Power and Energy Magazine*, vol. 8, no. 2, pp. 88–89, 2010.
- [11] D. Wei, Y. Lu, M. Jafari, P. Skare, and K. Rohde, “Protecting smart grid automation systems against cyberattacks,” *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 782–795, 2011.

- [12] G. Ericsson, "Cybersecurity and power system communication – essential parts of a smart grid infrastructure," *IEEE Transactions on Power Delivery*, vol. 25, no. 3, pp. 1501–1507, 2010.
- [13] C.-W. Ten, M. Govindarasu, and C.-C. Liu, "Cybersecurity for critical infrastructures: Attack and defense modeling," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 40, no. 4, pp. 853–865, 2010.
- [14] C.-W. Ten, C.-C. Liu, and M. Govindarasu, "Vulnerability assessment of cybersecurity for SCADA systems," *IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1836–1846, 2008.
- [15] T. Mander, F. Nabhani, L. Wang, and R. Cheung, "Integrated network security protocol layer for open-access power distribution systems," in *Proceedings of IEEE Power Engineering Society General Meeting*, (Tampa, Florida, United States), pp. 1–8, June 2007.
- [16] I.-H. Lim, S. Hong, M.-S. Choi, S.-J. Lee, T.-W. Kim, S.-W. Lee, and B.-N. Ha, "Security protocols against cyber attacks in the distribution automation system," *IEEE Transactions on Power Delivery*, vol. 25, no. 1, pp. 448–455, 2010.
- [17] H. Mohsenian-Rad and A. Leon-Garcia, "Distributed internet-based load altering attacks against smart power grids," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 667–674, 2011.

- [18] I.-K. Yang, N.-J. Jung, and Y.-I. Kim, "Status of advanced metering infrastructure development in korea," in *Proceedings of IEEE Transmission & Distribution Conference & Exposition: Asia and Pacific*, (Seoul, Korea), pp. 1–3, Oct. 2009.
- [19] A. Vojdani, "Smart integration," *IEEE Power and Energy Magazine*, vol. 6, pp. 71–79, Nov. 2008.
- [20] E. Liu, M. Chan, C. Huang, N. Wang, and C. Lu, "Electricity grid operation and planning related benefits of advanced metering infrastructure," in *Proceedings of 5th International Conference on Critical Infrastructure (CRIS)*, (Beijing, China), pp. 1–5, Sept. 2010.
- [21] H. Sui, H. Wang, M. Lu, and W. Lee, "An ami system for the deregulated electricity markets," *IEEE Transactions on Industrial Electronics*, vol. 45, pp. 2104–2108, Nov. 2009.
- [22] E. Valigi and E. di Marino, "Networks optimization with advanced meter infrastructure and smart meters," in *Proceedings of 20th International Conference and Exhibition on Electricity Distribution-Part 1*, (Prague, Czech), pp. 1–4, June 2009.
- [23] R. Berthier, W. Sanders, and H. Khurana, "Intrusion detection for advanced metering infrastructures: Requirements and architectural directions," in *Proceedings of First IEEE International Conference on Smart Grid Communications*, (Gaithersburg, Maryland, USA), pp. 350–355, Oct. 2010.

- [24] R. Shein, "Security measures for advanced metering infrastructure components," in *Proceedings of Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, (Chengdu, China), pp. 1–3, Mar. 2010.
- [25] F. M. Cleveland, "Cybersecurity issues for advanced metering infrastructure (AMI)," in *Proceedings of IEEE Power and Energy Society General Meeting—Conversion and Delivery of Electrical Energy in the 21st Century*, (Pittsburgh, United States), pp. 1–5, July 2008.
- [26] Electric Power Research Institute, "Advanced metering infrastructure," <http://www.ferc.gov/eventcalendar/Files/20070423091846-EPRI-AdvancedMetering.pdf>, Apr. 2007.
- [27] T. Mukherjee, "Design automation issues for biofluidic microchips," *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 463–470, Nov. 2005.
- [28] Infineon Electronic DNA Chip, "<http://www.infineon.com>,"
- [29] Y. Wong, T. Cheung, K. Lo, V. Wong, C. Chan, T. Ng, T. Chung, and S. Mok, "Protein profiling of cervical cancer by protein-biochips: proteomic scoring to discriminate cervical cancer from normal cervix," *Cancer Letters*, vol. 211, no. 2, pp. 227–234, 2004.
- [30] V. Srinivasan, V. Pamula, M. Pollack, and R. Fair, "Clinical diagnostics on human whole blood, plasma, serum, urine, saliva, sweat, and tears on a digital microfluidic

- platform,” in *Proceedings of Micro Total Analysis System (μ TAS)*, (Squaw Valley, United States), pp. 1287 – 1290, October 2003.
- [31] “Advances in lab-on-chip and microfluidics technologies for drug discovery and clinical diagnostics,” *Frost & Sullivan*, 2007.
- [32] V. Srinivasan, V. Pamula, and R. Fair, “An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids,” *Lab on a chip*, pp. 310–315, 2004.
- [33] F. Su and K. Chakrabarty, “Module placement for fault-tolerant microfluidics-based biochips,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 3, pp. 682–710, 2006.
- [34] P.-H. Yu, C.-L. Yang, and Y.-W. Chang, “Bioroute: A network-flow-based routing algorithm for the synthesis of digital microfluidic biochips,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 11, pp. 1928–1941, 2008.
- [35] T.-W. Huang and T.-Y. Ho, “A two-stage integer linear programming-based droplet routing algorithm for pin-constrained digital microfluidic biochips,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 2, pp. 215–228, 2011.
- [36] K. Chakrabarty, R. Fair, and J. Zeng, “Design tools for digital microfluidic biochips: Towards functional diversification and more than moore,” *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems, vol. 29, pp. 1001 – 1017, 2010.

- [37] M. Alistar, E. Maftai, P. Pop, and J. Madsen, “Synthesis of biochemical applications on digital microfluidic biochips with operation variability,” in *Proceedings of Symposium on Design Test Integration and Packaging of MEMS/MOEMS (DTIP)*, (Seville, Spain), pp. 350–357, May 2010.
- [38] V. Joshiand, G. Li, S. Wang, and S. Sun, “Biochemical stability of components for use in a DNA detection system,” *IEEE Transactions on Magnetics*, vol. 40, no. 4, pp. 3012–3014, 2004.
- [39] S. S. W. Chuang and I. Hajj, “Timing and area optimization for standard-cell vlsi circuit design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 3, pp. 308–320, 1995.
- [40] C.-P. Chen, C. Chu, and D. Wong, “Fast and exact simultaneous gate and wire sizing by lagrangian relaxation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 1014–1025, 1999.
- [41] H. Tennakoon and C. Sechen, “Gate sizing using lagrangian relaxation combined with a fast gradient-based pre-processing step,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (San Jose, United States), pp. 395–402, November 2002.

- [42] V. Sundararajan, S. Sapatnekar, and K. Parhi, “Fast and exact transistor sizing based on iterative relaxation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 5, pp. 568–581, 2002.
- [43] J. Wang, D. Das, and H. Zhou, “Gate sizing by lagrangian relaxation revisited,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (San Jose, United States), pp. 111–118, November 2007.
- [44] J. Fishburn and A. Dunlop, “Tilos: a posynomial programming approach to transistor sizing,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (Santa Clara, United States), pp. 326–328, November 1985.
- [45] S. Sapatnekar, V. Rao, and P. Vaidya, “An exact solution to the transistor sizing problem for cmos circuits using convex optimization,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 11, pp. 1621–1634, 1993.
- [46] M. Berkelaar and J. Jess, “Gate sizing in MOS digital circuits with linear programming,” in *Proceedings of the European Conference on Design Automation*, (Los Alamitos, United States), pp. 217–221, March 1990.
- [47] A. Murugavel and N. Ranganathan, “Gate sizing and buffer insertion using economic models for power optimization,” in *Proceedings of IEEE International Conference on VLSI Design*, (Mumbai, India), pp. 195–200, January 2004.

- [48] M. Mani and M. Orshansky, “A new statistical optimization algorithm for gate sizing,” in *Proceedings of International Conference on Computer Design*, (San Jose, CA), pp. 272–277, October 2004.
- [49] J. Singh, V. Nookala, Z. Luo, and S. Sapatnekar, “Robust gate sizing by geometric programming,” in *Proceedings of ACM/IEEE Design Automation Conference*, (San Diego, United States), pp. 315–320, June 2005.
- [50] V. Mahalingam, N. Ranganathan, and J. H. III, “A novel approach for variation aware power minimization during gate sizing,” in *Proceedings of International Symposium on Low Power Electronics and Design*, (Tegernsee, Germany), pp. 174–179, October 2006.
- [51] D. Sinha and H. Zhou, “Gate sizing for crosstalk reduction under timing constraints by lagrangian relaxation,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (San Jose, United States), pp. 14–19, November 2004.
- [52] N. Hanchate and N. Ranganathan, “Simultaneous interconnect delay and crosstalk noise optimization through gate sizing using game theory,” *IEEE Transactions on Computers*, vol. 55, no. 8, pp. 1011–1023, 2006.
- [53] M. Becer, D. Blaauw, I. Algor, R. P. C. Oh, V. Zolotov, and I. Hajj, “Postroute gate sizing for crosstalk noise reduction,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 12, pp. 1670–1677, 2004.

- [54] Q. Zhou and K. Mohanram, “Gate sizing to radiation harden combinational logic,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 1, pp. 155–166, 2006.
- [55] K. Bhattacharya and N. Ranganathan, “A linear programming formulation for security-aware gate sizing,” in *Proceedings of ACM Great Lakes Symposium on VLSI*, (Orlando, United States), pp. 273–278, May 2008.
- [56] S. Hu, M. Ketkar, and J. Hu, “Gate sizing for cell library-based designs,” in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, (San Diego, United States), pp. 847–852, June 2007.
- [57] W. Ning, “Strongly np-hard discrete gate-sizing problems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 8, pp. 1045–1051, 1994.
- [58] F. Beeftink, P. Kudva, D. Kung, and L. Stok, “Gate size selection for standard cell libraries,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (San Jose, United States), pp. 545–550, November 1998.
- [59] H. Ren and S. Dutt, “A network-flow based cell sizing algorithm,” in *Proceedings of International Workshop on Logic Synthesis*, (Lake Tahoe, United States), pp. 7–14, June 2008.

- [60] T.-H. Wu, L. Xie, and A. Davoodi, "A parallel and randomized algorithm for large-scale discrete dual-vt assignment and continuous gate sizing," in *Proceedings of International Symposium on Low Power Electronics and Design*, (Bangalore, India), pp. 45–50, August 2008.
- [61] O. Coudert, "Gate sizing for constrained delay/power/area," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 4, pp. 465–472, 1997.
- [62] M. El-Kady, "Computer-aided planning of distribution substation and primary feeders," *IEEE Transactions on Power Apparatus and Systems*, vol. 103, no. 6, pp. 1183–1189, 1984.
- [63] I. Ramirez-Rosado and T. Gonen, "Pseudodynamic planning for expansion of power distribution systems," *IEEE Transactions on Power Systems*, vol. 6, no. 1, pp. 245–254, 1991.
- [64] S. Khator and L. Leung, "Power distribution planning: a review of models and issues," *IEEE Transactions on Power Systems*, vol. 12, no. 3, pp. 1151–1159, 1997.
- [65] P. Paiva, H. Khodr, J. Dominguez-Navarro, J. Yusta, and A. Urdaneta, "Integral planning of primary-secondary distribution systems using mixed integer linear programming," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1134–1143, 2005.

- [66] S. Haffner, L. Pereira, L. Pereira, and L. Barreto, “Multistage model for distribution expansion planning with distributed generation, part i: Problem formulation,” *IEEE Transactions on Power Delivery*, vol. 23, no. 2, pp. 915–923, 2008.
- [67] P. Zhang and W. Li, “Boundary analysis of distribution reliability and economic assessment,” *IEEE Transactions on Power Systems*, vol. 25, no. 2, pp. 714–721, 2010.
- [68] “Introduction to ISO 27002.”
- [69] North American Electric Reliability Corporation (NERC), “Cybersecurity (permanent).”
- [70] R. Evans. Control Systems Cybersecurity Standards Support Activities, Jan. 2009.
- [71] B. Krebs. FBI: Smart Meter Hacks Likely to Spread, KrebsOnSecurity daily blog on computer security and cybercrime, Apr. 2012.
- [72] Y.-H. Chang, P. Jirutitijaroen, and C.-W. Ten, “A simulation model of cyber threats for energy metering devices in a secondary distribution network,” in *Proc. 5th Intl. CRIS on Critical Infrastructures*, (Beijing, China), Sep. 2010.
- [73] Y.-S. Ko, T.-K. Kang, H.-Y. Park, H.-Y. Kim, and H.-S. Nam, “The FRTU-based fault-zone isolation method in the distribution systems,” *IEEE Transactions on Power Delivery*, vol. 25, no. 2, pp. 1001–1009, 2010.

- [74] K. Aoki, K. Nara, H. Itoh, T. Satoh, and H. Kuwabara, "A new algorithm for service restoration in distribution systems,"
- [75] Y.-S. Ko, "A self-isolation method for the HIF zone under the network-based distribution system,"
- [76] I. Lim, Y. Kim, H. Lim, M. Choi, S. Hong, S. Lee, S. Lim, S. Lee, and B. Ha, "Distributed restoration system applying multi-agent in distribution automation system," in *Proceedings of IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, (Pittsburgh, United States), pp. 1–7, July 2008.
- [77] A. Meliopoulos, G. Cokkinides, R. Huang, E. Farantatos, S. Choi, Y. Lee, and X. Yu, "Smart grid technologies for autonomous operation and control," *IEEE Transactions on Smart Grid*, vol. 2, no. 1, pp. 1–10, 2011.
- [78] T. Hamilton, "Smart-meter energy data now online," <http://www.mysanantonio.com/news/energy/article/Smart-meter-energy-data-now-online-2133522.php>, Aug. 2011.
- [79] Electric Power Research Institute, "EPRI comment: sage report on radio-frequency (RF) exposures from smart meters," http://www.marbleheadelectric.com/EPRI_SageReport.pdf, Feb. 2001.

- [80] H.-P. Kriegel, P. Kroger, and A. Zimek, “Outlier detection techniques (tutorial),” in *Proceedings of 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, (Bankok, Thailand), June 2009.
- [81] IEEE Task Force on Load Representation for Dynamic Performance, “Load representation for dynamic performance analysis,” *IEEE Transactions on Power Systems*, vol. 8, no. 2, pp. 472–482, 1993.
- [82] S. Al-Alawi and S. Islam, “Principles of electricity demand forecasting. i. methodologies,” *Power Engineering Journal*, vol. 10, no. 3, pp. 139–143, 1996.
- [83] H. Smolleck and K. Kim, “An interactive distribution load forecasting methodology for minicomputer use based upon a markov-type process,” *IEEE Transactions on Power Systems*, vol. 3, no. 1, pp. 52–58, 1988.
- [84] C. Sutton and A. McCallum, “An introduction to conditional random fields for relational learning,” *Introduction to Statistical Relational Learning*, MIT Press, 2006.
- [85] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of 18th International Conference on Machine Learning*, (Williamstown, United States), pp. 282–289, June 2001.

- [86] R. Rubinstein and D. Kroese, “The cross-entropy method: A unified approach to combinatorial optimization, monte-carlo simulation, and machine learning,” *Springer-Verlag, New York*, July, 2004.
- [87] N. Li, A. Tourovskaia, and A. Folch, “Biology on a chip: microfabrication for studying the behaviour of cultured cells,” *Critical Reviews in Biomedical Engineering*, vol. 31, no. 5-6, pp. 423–488, 2003.
- [88] B. Weigl, R. Bardell, and C. Cabrera, “Lab-on-a-chip for drug development,” *Advanced Drug Delivery Reviews*, vol. 55, no. 3, pp. 349–377, 2003.
- [89] M. Rouse, “What is a biochip?,” <http://searchcio-midmarket.techtarget.com/definition/biochip>, Nov. 2006.
- [90] N. Rodrigues, Y. Sakai, and T. Fujii, “Cell-based microfluidic biochip for the electrochemical real-time monitoring of glucose and oxygen,” *Sensors and Actuators B: Chemical*, vol. 132, no. 2, pp. 608–613, 2008.
- [91] R. Baudoin, A. Corlu, L. Griscom, C. Legallais, and E. Leclerc, “Trends in the development of microfluidic cell biochips for in vitro hepatotoxicity,” *Toxicology in Vitro*, vol. 21, pp. 535–544, 2007.
- [92] M. Pollack, R. Fair, and A. Shenderov, “Electrowetting-based actuation of liquid droplets for microfluidic applications,” *Applied Physics Letters*, vol. 77, no. 11, pp. 1725–1726, 2000.

- [93] F. Su and K. Chakrabarty, “High-level synthesis of digital microfluidic biochips,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 3, no. 4, pp. 1–32, 2008.
- [94] N. Viswanathan and C. Chu, “Fastplace: Efficient analytical placement using cell spreading, iterative local refinement and a hybrid net model,” *IEEE Transactions on Computer-Aided Design (TCAD)*, vol. 24, no. 5, pp. 722–733, 2005.
- [95] P. Maidee, C. Ababei, and K. Bazargan, “Timing-driven partitioning-based placement for island style fpgas,” *IEEE Transactions on Computer-Aided Design (TCAD)*, vol. 24, no. 3, pp. 395–406, 2005.
- [96] J. Cong and M. Xie, “A robust mixed-size legalization and detailed placement algorithm,” *IEEE Transactions on Computer-Aided Design (TCAD)*, vol. 27, no. 8, pp. 1349–1362, 2008.
- [97] R. Hassin, “Approximation schemes for the restricted shortest path problem,” *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36–42, 1992.
- [98] S. Shah, A. Srivastava, D. Sharma, D. Sylvester, D. Blaauw, and V. Zolotov, “Discrete vt assignment and gate sizing using a self-snapping continuous formulation,” in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, (San Jose, United States), pp. 705 – 712, November 2005.

- [99] G. Fishman, “Monte carlo: Concepts, algorithms, and applications,” *Springer, New York*, 1995.
- [100] M. Mckay, R. Beckman, and W. Conover, “A comparison of three methods for selecting values of input variables in the analysis of output from a computer code,” *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.
- [101] S. Hu and J. Hu, “Unified adaptivity optimization of clock and logic signals,” in *Proceedings of IEEE/ACM international conference on Computer-aided design (ICCAD)*, (San Jose, United States), pp. 125–133, Nov. 2007.
- [102] D. Bertsimas and M. Sim, “The price of robustness,” *Operations Research*, vol. 52, no. 1, pp. 35–53, 2004.
- [103] Y. Zhao and K. Chakrabarty, “Cross-contamination avoidance for droplet routing in digital microfluidic biochips,” in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE)*, (Dresden, Germany), pp. 1290–1295, Mar. 2009.
- [104] T.-W. Huang, C.-H. Lin, and T.-Y. Ho, “A contamination aware droplet routing algorithm for the synthesis of digital microfluidic biochips,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1682–1695, 2010.
- [105] F. Su and K. Chakrabarty, “Defect tolerance based on graceful degradation and dynamic reconfiguration for digital microfluidics-based biochips,” *IEEE*

- Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2944–2953, 2006.
- [106] C.-Y. Lin and Y.-W. Chang, “Cross-contamination aware design methodology for pin-constrained digital microfluidic biochips,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 6, pp. 817–828, 2011.
- [107] T. Xu, K. Chakrabarty, and F. Su, “Defect-aware high-level synthesis and module placement for microfluidic biochips,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 2, no. 1, pp. 50–62, 2008.
- [108] P.-H. Yuh, S. Sapatnekar, C.-L. Yang, and Y.-W. Chang, “A progressive-ilp based routing algorithm for cross-referencing biochips,” in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, pp. 284–289, June 2008.
- [109] M. Cho and D. Pan, “A high performance droplet router for digital microfluidic biochips,” in *Proceedings of ACM International Symposium on Physical Design (ISPD)*, (Lake Tahoe, United States), pp. 200 – 206, March 2008.
- [110] F. Su, W. Hwang, and K. Chakrabarty, “Droplet routing in the synthesis of digital microfluidic biochips,” in *in proceedings of Design, Automation and Test in Europe Conference and Exposition (DATE)*, (Paris, France), pp. 323–328, February 2006.
- [111] C. Lee, “An algorithm for path connection and its application,” *IRE Transactions on Electronic Computers*, vol. EC-10, no. 2, pp. 346–365, 1961.

- [112] R. Hassin, “Approximation schemes for the restricted shortest path problem,” *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36–42, 1992.
- [113] F. Ergun, R. Sinha, and L. Zhang, “An improved fptas for restricted shortest path,” *Information Processing Letters*, vol. 83, no. 5, pp. 287–291, 2002.
- [114] V. Vazirani, “Approximation algorithms,” *Springer-Verlag, Berlin*, 2001.
- [115] W. Elmore, “The transient analysis of damped linear networks with particular regard to wideband amplifiers,” *Journal of Applied Physics*, vol. 19, no. 1, pp. 321–336, 1948.
- [116] S. Hu, Z. Li, and C. Alpert, “A fully polynomial time approximation scheme for timing driven minimum cost buffer insertion,” in *Proceedings of ACM/IEEE Design Automation Conference (DAC)*, (San Francisco, United States), pp. 424–429, July 2009.
- [117] S. Hu, Z. Li, and C. Alpert, “A faster approximation scheme for timing driven minimum cost layer assignment,” in *Proceedings of ACM International Symposium on Physical Design (ISPD)*, (San Diego, United States), pp. 167–174, March 2009.