Title
Net Weighting Methods and Other Novel Approaches in Variation-Aware Placement and Sizing

Author
Radke, Eric M

Publication Date
2015

Peer reviewed|Thesis/dissertation

University of California
Los Angeles

Net Weighting Methods and Other Novel Approaches in Variation-Aware Placement and Sizing

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

by

Eric Matthew Radke

2015

Abstract of the Dissertation

Net Weighting Methods and Other Novel Approaches in Variation-Aware Placement and Sizing

by

Eric Matthew Radke

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2015

Professor Tony F. Chan, Chair

The research is divided into three primary sections. In the first section, a rigorous mathematical framework is developed under which *net weighting methods*, common in VLSI physical design, may be proven to converge, given that they satisfy several natural properties defined in the work. Several net weighting techniques from industry are analyzed within this framework and adapted so to form a class of schemes that satisfy the convergent properties. These are implemented and experimentally tested within the analytical placer *mPL* [13]. In the second section, the net weighting framework is extended to statistical timing-aware placement in a novel algorithm, termed *STAP*, with promising experimental results. In the third section, a new approach to statistical timing-aware gate sizing, termed *ProSize*, is developed from current research in convex optimization.

The dissertation of Eric Matthew Radke is approved.

Joseph Teran

Luminita Vese

Jason Cong

Tony F. Chan, Committee Chair

University of California, Los Angeles

2015

To my mother and father
who always believed in me.

Table of Contents

List of Figures

List of Tables

Acknowledgments

The author gratefully acknowledges the dissertation committee, and in particular Prof. Jason Cong, Guojie Luo, Kirill Minkovich, and Prof. Joe Shinnerl of the UCLA Computer Science Department, Prof. Tony F. Chan and Prof. Luminita Vese of the UCLA Mathematics Department, and John Lee and Prof. Lieven Vandenberghe of the UCLA Electrical Engineering Department for their numerous thoughtful and illuminating discussions and assistance.

Further, the author acknowledges the National Science Foundation and Semiconductor Research Corporation for their support from SRC Grant 1460.001 and NSF Grants CCF-0430077 and CCF-0528583, as well as the NSF VIGRE Fellowship.

Vita

2000-2004	BS, Mathematics, Case Western Reserve University, Cleveland, Ohio. Full tuition recipient; Summa Cum Laude.
2003	Research Assistant, National Science Foundation VIGRE REU Program on Elliptic Curves, University of Utah, Salt Lake City, Utah. Advisor: Aaron Bertram, University of Utah Mathematics Professor.
2003	Research Assistant, National Science Foundation VIGRE REU Program on Geometric Group Theory, University of Illinois - Urbana-Champaign. Advisor: Kim Whittlesey, University of Illinois Mathematics Professor.
2004	Research Assistant, Electron and Optical Devices Branch, NASA Glenn Research Center, Cleveland, Ohio. Advisor: Karl Vaden, senior researcher.
2004-2006	MA, Mathematics, University of California - Los Angeles. National Science Foundation VIGRE Fellowship recipient.
2005	Research Assistant, UAV Path Planning Project, Raytheon Corp., Los Angeles, California. Advisor: Andrea Bertozzi, UCLA Mathematics Professor, Director of Applied Mathematics.
2005-present	Research Assistant, VLSI Architecture, Synthesis, and Technology Laboratory, University of California - Los Angeles. Advisors: Tony Chan, President, Hong Kong University of Science and Technology, and Jason Cong, Director, Center for Customizable Domain-Specific Computing.
2005-present	Teaching Assistant and Course Instructor, Mathematics Department, University of California - Los Angeles. Robert Sorgenfrey Distinguished Teaching Assistant Award recipient.
2006-present	Ph.D. Candidate, Mathematics, University of California - Los Angeles. National Science Foundation VIGRE Fellowship recipient.

Publications

T. F. Chan, J. Cong, and E. M. Radke, "Convergent Net Weighting Schemes in Hypergraph-based Optimization." UCLA Computational and Applied Math Reports, 2011.

T. F. Chan, J. Cong, and E. M. Radke, "A Rigorous Framework for Convergent Net Weighting Schemes in Timing-Driven Placement." Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, November 2009.

J. Cong, G. Luo, and E. M. Radke, "Highly Efficient Gradient Computation for Density-Constrained Analytical Placement," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 27, Number 12, pp. 2133-2144, December 2008.

CHAPTER 1

Introduction

1.1 The VLSI Placement Problem

1.1.1 Modeling the Circuit

The base problem under consideration in this work is the *VLSI Placement Problem*. In it, we are given a *hypergraph* – that is, a graph in which an edge may connect an arbitrary number of nodes, referred to as *nets*. The problem is to determine a position for each node. This corresponds to the physical location of circuit elements on a chip surface. The nets correspond to connections that need to be wired between the circuit elements. Thus, we seek a vector of locations $\mathbf{x}$ as a solution to the problem, minimizing some objective typically associated with the estimated wire length of the resulting circuit.

One common estimate of the routing wire length of a circuit is the *half-perimeter wire length*, which is half the perimeter of the smallest bounding box of a net [41]. Frequently, this function is smoothed so the objective can be differentiated, for example using the *log-sum-exp* approximation [27] as shown in 1.1.1 below. Other approximations use a quadratic function to estimate routed wire length.

$$h_i(\mathbf{x}) = \eta \log \sum_{j=1}^{m} \exp(x_{i_j}/\eta) + \eta \log \sum_{j=1}^{m} \exp(-x_{i_j}/\eta) \tag{1.1.1}$$

Typically, we seek to use an objective approximation that satisfies three criteria: it satisfactorily approximates the routed wire length, it is not too computationally expensive, and for analytical placers, it is sufficiently smooth. The VLSI placement problem is NP-hard with millions of variables, and studies indicate that recent placement solutions are still

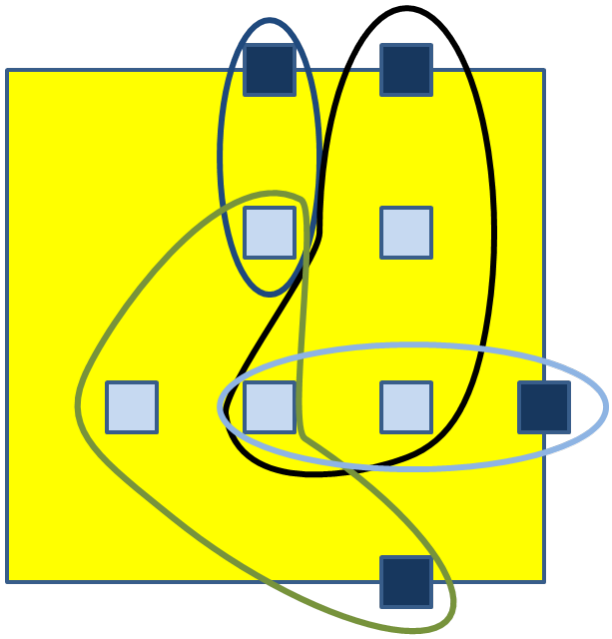

Figure 1.1.1: A simplified circuit (hypergraph).

between 20% and 150% from optimal, despite its study for decades [14, 17].

1.1.2 Density Constraints and Placement Phases

Further, constraints need to be imposed to handle the non-overlap restriction of the nodes. This makes the problem highly nonconvex. Due to the large scale of the problem, overlap is not restricted explicitly in the constraints but is typically handled in two phases: *global placement* and *detailed placement* [16].

Global placement is the primary placement phase in which some overlap among the cells is allowed. As opposed to direct enforcement of non-overlap which does not scale for problems of this size, typically cell spreading is enforced spread using *density constraints*. These divide the chip region into a grid of bins, each of which contains a density constraint [13]. In this way, the cells are spread while some overlap still occurs. The purpose of global placement is to get a good general placement of the circuit, to be refined locally in the second phase.

Detailed placement is the second placement phase in which cell overlap is eliminated and each cell is moved to a standard row in a physically viable location on the chip. This is

termed *legalization*. The detailed placement phase is typically much more local in nature, relying on global placement to assign each cell's neighborhood.

1.1.3 Formulation

This research is primarily concerned with the global placement phase, but our results shall include the circuit characteristics after detailed placement for comparison. We shall give this problem a more specific formulation in the following chapters, but the global problem can be fundamentally described as the following *general VLSI placement problem*:

Minimize: Estimated total wire length
subject to:
Density constraints

1.2 The Timing-Constrained VLSI Placement Problem

It is often of interest to minimize the timing of the circuit as well. The maximum amount of time a signal takes to traverse any path in the circuit length in a circuit will directly affect the clock speed.

In order to study the timing-constrained VLSI placement problem, we require that the given hypergraph be *directed* and *acyclic*. As such, the *general timing-constrained VLSI placement problem* is given as the following:

Minimize: Estimated total wire length
subject to:
Timing constraints
Density constraints

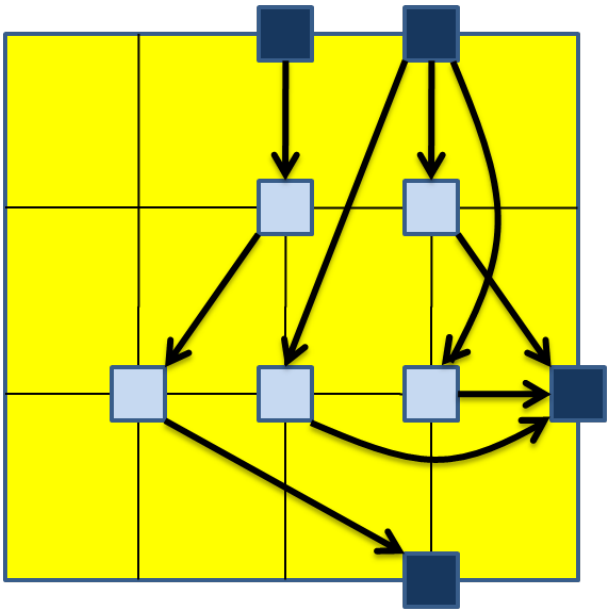

Figure 1.2.1: Simplified directed acyclic hypergraph for timing-constrained placement.

1.2.1 Common Methods

A number of approaches exist in the literature for handling timing-driven placement. These may be broadly defined under two categories: *path-based* and *net-based* schemes.

Path-based schemes are broadly characterized by a consideration of the paths of the circuit from the input nodes to the output nodes. These schemes can include mathematical programming and iterative critical path minimization methods. For example, [26] used recursive partitioning with a min cut objective. [22] enforced timing constraints through the use of Lagrange multipliers, while [47] used simulated annealing [28, 9] to minimize a set of timing-critical paths. Path-based schemes have the benefit that they can accurately handle timing constraints. However, the consideration of full path information is computationally expensive and as such generally does not scale well for modern circuits.

On the other hand, net-based schemes are characterized by a direct emphasis on nets. Examples include the use of net length constraints, as in [44], which used a star model to decompose nets into edges and imposed net constraints on a small number of critical nets. [49] employed delay bounds on nets, using the KKT conditions [40] to determine changes from iteration to iteration.

1.2.2 Net Weighting

Of particular interest to this research is a class of net-based methods known as *net weighting* [42, 29, 27, 34]. These employ the use of weights to prioritize nets in critical paths.

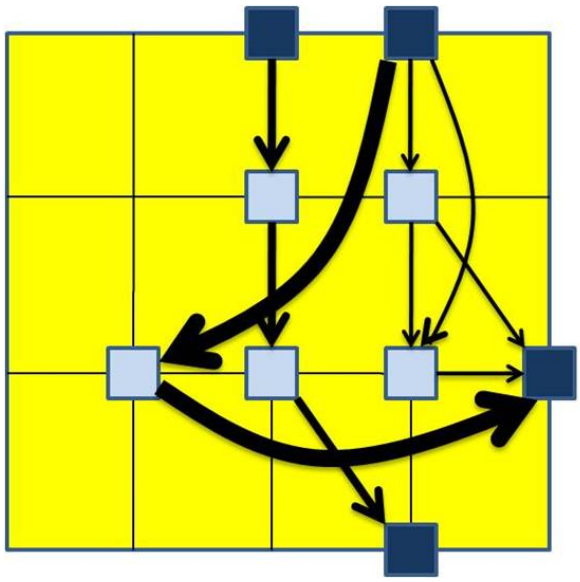

Figure 1.2.2: Simplified directed acyclic hypergraph placement using net weighting.

Net weighting methods enjoy a number of benefits, including ease of implementation within already existing placement tools, low computational complexity and high flexibility. However, previous to this research, they have been used in an ad-hoc manner and lack theoretical justification. One of the main contributions of this work is the theoretical examination of these methods.

Among commonly used schemes is the *VPR weighting* [34], a *polynomial* scheme defined on the edges by

$$w_e = \left(1 - \frac{\sigma(e)}{T_p}\right)^{\alpha}$$

where $\sigma(e)$ is the slack of edge e, T_p is the max path delay of the previous iterate, and α is a user-defined constant. Another scheme is the *PATH weighting* [29], an *exponential* scheme that considers all paths in a circuit efficiently:

$$w_e = \sum_{\pi \ni e} \alpha^{-\frac{\sigma(\pi)}{T}}$$

where $\sigma(\pi)$ is the slack of path π, T is a desired max path delay, and α is again a user-defined constant. A third scheme is the *APlace weighting* [27], a *piecewise polynomial* scheme given by

$$w_e = \sum_{\pi \ni e} f\big(delay(\pi), T_u\big),$$

$$\text{where } f(d, T_u) = \begin{cases} \left(\frac{d}{T_u}\right)^{\alpha} - 1 & \text{if } d > T_u \\ 0 & \text{otherwise} \end{cases}$$

Here $T_u = (1-u)T_p$, where u is a constant selected to be 0.1, 0.2, or 0.3, and α is also a user-defined constant. These schemes will be discussed in greater depth in the following chapters.

1.3 Sizing

In Chapter 4, we examine a different problem from the placement problem in VLSI physical design – that of the VLSI gate sizing problem. Typically after placement, each circuit gate must be given a size. Gate sizes affect the timing of a circuit as well as other important aspects of chip performance, including power and area consumption. Although gate sizes are discrete in practice, the gate size constraint is often relaxed to allow for continuous gate sizes. We thus have the *general VLSI gate sizing problem*:

Minimize: Maximum circuit timing

subject to:

Power constraints

Area constraints

Gate size constraints

Power and area are typically modeled as convex functions so that the overall problem is convex. This problem has been studied extensively and is efficiently solved via geometric programming [5]. As such, the sizing problem is simpler than the placement problem and thus lends itself to analytic study much more naturally.

1.4 Considering Process Variations

Also of central interest is the effect of process variations upon chip performance. As transistor sizes have shrunk according to Moore's law, the impact of process variations has grown in causing failed chips [53]. One reason is that manufacturing process control is not improving at the same rate as scaling [43]. Another cause is the increasing impact of atomic-scale randomness, such as variation in the number of dopants in the transistor channel [48].

As a result, we introduce a new wrinkle of sophistication in the problem of interest. We model circuit element delays as *distributions*, usually normal, as opposed to fixed numbers. Instead of enforcing a constant timing constraint as in the general timing-constrained VLSI placement problem above, we use a *timing yield constraint*. This constraint requires that the probability of a chip being less than some time T to be greater than some user-defined yield *eta*. The timing yield constraint corresponds to an upper bound T for the timing of a chip to be acceptable, above which the chip will be discarded. We require a yield of more than $\eta \cdot 100\%$ acceptable chips in each batch produced.

1.4.1 Previous Approaches

The effect of process variation has been considered much more commonly in the area of sizing than that of placement. The existing attempts [6, 31, 30, 46] to consider process variation in the timing-constrained placement problem employ simple heuristics, are conducted on FPGAs, and are all based on modifications to the placer VPR [34]. As such, this work represents the first attempt, to our knowledge, to conduct statistical placement on ASICs, with promising results.

In the area of sizing, however, a number of methods have been developed. One popular group of approaches falls under the category of *padded methods* [32]. These methods use a heuristic "padding" term added to each delay to account for the statistical delay, and then proceed with deterministic methods. These are often inexpensive but also inaccurate – using timing corners may be overly pessimistic, as they represent a number of worst-case scenarios that are unlikely to occur simultaneously. However, avoiding using corners on other timing

elements may ignore statistical issues that later arise in fabrication [54].

Related to padded methods is *guardbanding* [7]. This involves sweeping a padding term on overall deterministic circuit timing and adjusting according to a more accurate statistical timing. Guardbanding benefits from low computational complexity as well as ease of integration into existing deterministic tools. We shall discuss guardbanding in more depth in Chapter 3.

Another group of approaches is termed *block-based methods* [54]. They use an analytical approach termed *block-based statistical timing analysis* to conduct timing analysis and propagate timing distributions through the circuit. This shall be covered in greater depth in Chapter 3.

1.5 Organization of the Dissertation

In Chapter 2, our focus is the timing-constrained VLSI placement problem. In particular, net weighting methods are studied and a theoretical rigorous framework is developed. A class of net weighting schemes is developed which are proven to have desired convergence properties. Several existing net weighting schemes are then adapted to satisfy the required properties for convergence and tested experimentally.

In Chapter 3, we make a large step in the little-studied area of the statistically timing-constrained VLSI placement problem. Building upon the work in Chapter 2, a scheme is developed to handle statistical timing-driven placement. This is the first known work of statistical timing-constrained VLSI placement on ASIC circuits.

In Chapter 4, we focus attention on the statistically timing-constrained gate sizing problem. We build upon previous research and implement a fast proximal gradient method on a reformulation of the problem.

Finally, Chapter 5 briefly summarizes the contributions and results of this work.

CHAPTER 2

Convergent Net Weighting Schemes in Hypergraph-based Optimization

2.1 Introduction

Approaches for solving the timing-driven placement problem have traditionally been either net-based or path-based; see, e.g., [16] for an overview. *Net weighting* methods, which fall in the latter category, have been a popular tool in analytical placers [49, 21, 27] for handling timing-driven placement. They enjoy a number of advantages, including very low computational complexity, high flexibility, and ease of implementation – weighting algorithms can be implemented within the framework of an existing placement tool by a simple modification of the objective function. However, net weighting methods suffer the disadvantage that they are largely *ad-hoc*; to date there has been very little theoretical justification for their use [29]. As a result, a number of very different weighting schemes have been proposed, of which some have been shown to be effective in reducing delay. Our goal is to distill the essential properties of a robust and effective net weighting method.

Among commonly used schemes is the *VPR weighting* [34], a *polynomial* scheme defined on the edges by

$$w_e = \left(1 - \frac{\sigma(e)}{T_p}\right)^{\alpha} \tag{2.1.1}$$

where $\sigma(e)$ is the slack of edge e, T_p is the max path delay of the previous iterate, and α is a user-defined constant. Another scheme is the *PATH weighting* [29], an *exponential* scheme that considers all paths in a circuit efficiently:

$$w_e = \sum_{\pi \ni e} \alpha^{-\frac{\sigma(\pi)}{T}} \tag{2.1.2}$$

where $\sigma(\pi)$ is the slack of path π, T is a desired max path delay, and α is again a user-defined constant. A third scheme is the *APlace weighting* [27], a *piecewise polynomial* scheme given by

$$w_e = \sum_{\pi \ni e} f\big(delay(\pi), T_u\big),$$
$$\text{where } f(d, T_u) = \begin{cases} \left(\frac{d}{T_u}\right)^{\alpha} - 1 & \text{if } d > T_u \\ 0 & \text{otherwise} \end{cases} \tag{2.1.3}$$

Here $T_u = (1-u)T_p$, where u is a constant selected to be 0.1, 0.2, or 0.3, and α is also a user-defined constant.

It has remained an open question under what conditions a net weighting scheme will converge to a timing feasible placement, and what quality can be expected of such a placement. These are addressed in the contributions of this work:

1. We develop a rigorous, generalized framework under which certain net weighting schemes are *guaranteed* to converge to the optimum of the original timing-constrained placement problem, provided the net weighted objective is minimized to a satisfactory degree. We then identify particular net weighting schemes that adhere to this framework.

2. In the case we are able to find global minimizers of unconstrained net weighted objectives, convergence is guaranteed to the global minimizer of the original timing-constrained problem.

3. However, most placers in practice cannot find a global minimizer and instead search for *approximate* local minimizers of the net weighted objectives. In this case, convergence is still guaranteed to a local minimum candidate point of the original timing-constrained problem.

4. We implement convergent weighting schemes in the state-of-the-art placer mPL [12, 35]. When we compare one scheme, a modification of the VPR weighting, to the original method, we find an average 4% delay improvement on the MCNC benchmarks [56]. In

every one of the benchmarks tested, delay improvement was superior for the modified weighting.

These results have also been published in [11] and [10].

2.2 Preliminaries

2.2.1 Definitions and Problem Formulation

We use the timing-driven placement problem as the most immediate application of this framework, and refer to it throughout the remainder of this chapter. However, the framework need not be restricted to placement, as it can be considered in a general hypergraph-based optimization problem in which net weights are applied to handle timing constraints.

Suppose we wish to minimize total wire length of a circuit by determining the pin locations $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, subject to the constraint that the total delay along any path should not exceed an upper bound $T > 0$. The objective we wish to minimize is given by

$$G(\mathbf{x}) = \sum_{\text{nets } i \in \mathcal{N}} h_i(\mathbf{x})$$

where $\mathcal{N}$ is the (finite) set of all nets in the circuit, and $h_i(\mathbf{x})$ is a *continuous, nonnegative* function measuring net $i = \{i_1, i_2, \ldots, i_m\}$ with the property

$$h_i(\mathbf{x}) = 0 \implies x_{i_1} = x_{i_2} = \cdots = x_{i_m} \tag{2.2.1}$$

For h_i, we could use for example half-perimeter wire length or its log-sum-exp approximation [41, 27],

$$h_i^{\text{lse}}(\mathbf{x}) = \eta \log \sum_{j=1}^{m} \exp(x_{i_j}/\eta) + \eta \log \sum_{j=1}^{m} \exp(-x_{i_j}/\eta)$$

for $\eta > 0$ small. The quadratic wire length approximation, as in e.g. [21], given by

$$h_i^{\text{quad}}(\mathbf{x}) = \sum_{j=1}^{m} \sum_{k=j+1}^{m} h_{i_j,i_k} |x_{i_j} - x_{i_k}|^2$$

is also suitable in this framework.

To measure delay along each edge e in the circuit, we use the *delay function* $d_e(\mathbf{x})$. (For each source-sink pair of pins connected by a net, we consider an edge connecting them. We may also consider internal delay of a node by considering an edge between an input and output pin of the node.) The delay function $d_e(\mathbf{x})$ measuring the delay of edge $e = (s, t)$ is a function of the overall placement $\mathbf{x}$, and is *continuous, nonnegative, convex*, and has the property

$$x_s = x_t \implies d_e(\mathbf{x}) = 0 \tag{2.2.2}$$

For example, the Euclidean and quadratic delay functions, given by $d_e(\mathbf{x}) = \gamma_e|x_t - x_s|$, and $d_e(\mathbf{x}) = \gamma_e(x_t - x_s)^2$, respectively, are suitable in this framework. For simplicity we have assumed that the placement area is one-dimensional. The analogous formulation for a two- or three-dimensional placement follows in a straightforward manner. Note in particular that the Euclidean distance delay function is suitable in higher dimensions, e.g.

$$d_e(\mathbf{x}, \mathbf{y}) = \gamma_e\sqrt{(x_t - x_s)^2 + (y_t - y_s)^2}$$

In the interest of brevity, we often use the *edge delay vector* $\mathbf{d}(\mathbf{x})$, which is defined simply as the vector of delays $d_e(\mathbf{x})$ along each edge e in the circuit, i.e. $\mathbf{d}(\mathbf{x}) = [d_1(\mathbf{x}) \ \ldots \ d_M(\mathbf{x})]^T$, where M is the number of edges in the circuit.

Further, we consider the *slacks* of the circuit, which are functions of the edge delay vector. For each path π we have the *path slack* σ_π:

$$\sigma_\pi(\mathbf{d}(\mathbf{x})) = \ T - \sum_{\text{edges } e \in \pi} d_e(\mathbf{x}) \tag{2.2.3}$$

We require the path slack to be nonnegative in our problem formulation.

Also useful in this framework is the notion of *edge slack*. First define the *arrival time* of pin t, $A_t(\mathbf{d}(\mathbf{x}))$, recursively as follows:

$$A_t(\mathbf{d}(\mathbf{x})) = \begin{cases} 0 & \text{if } t \in \mathcal{P}_\mathcal{I} \\ \max_{s \in \text{fanin}(t)}\{A_s(\mathbf{d}(\mathbf{x})) + d_{(s,t)}(\mathbf{x})\} & \text{otherwise} \end{cases} \tag{2.2.4}$$

Similarly we define the *required arrival time* of pin s, $R_s(\mathbf{d}(\mathbf{x}))$, as follows:

$$R_s(\mathbf{d}(\mathbf{x})) = \begin{cases} T & \text{if } s \in \mathcal{P}_{\mathcal{O}} \\ \min_{t \in \text{fanout}(s)} \{R_t(\mathbf{d}(\mathbf{x})) - d_{(s,t)}(\mathbf{x})\} & \text{otherwise} \end{cases} \tag{2.2.5}$$

Then for the edge $e = (s, t)$, we define its *slack* as:

$$\sigma_e(\mathbf{d}(\mathbf{x})) = R_t(\mathbf{d}(\mathbf{x})) - A_s(\mathbf{d}(\mathbf{x})) - d_e(\mathbf{x}) \tag{2.2.6}$$

For any net i, we define its slack as $\sigma_i(\mathbf{d}(\mathbf{x})) = \min_{e \in i} \sigma_e(\mathbf{d}(\mathbf{x}))$.

Now let $\mathcal{P}$, $\mathcal{P}_{\mathcal{I}}$, and $\mathcal{P}_{\mathcal{O}}$ denote the (finite) set of all pins, input pins, and output pins in the circuit, respectively. All paths $\pi = (\pi_1, \pi_2, \ldots, \pi_p)$ begin with an input pin $\pi_1 \in \mathcal{P}_{\mathcal{I}}$, end with an output pin $\pi_p \in \mathcal{P}_{\mathcal{O}}$, and adjacent pins in the path are connected by an edge. We require that $\mathcal{P}_{\mathcal{I}} \cap \mathcal{P}_{\mathcal{O}} = \emptyset$, i.e. there are no single-pin paths. Let Π denote the set of all paths in the circuit, and let $S = |\Pi|$. Note that we can also think of each net i and path π as a set and string of edges, respectively, and do so when appropriate.

Now we are ready to present the problem to be solved. For simplicity, to prevent overlaps we temporarily assume some pins are fixed; the more practical case involving density constraints is fully compatible in this framework and discussed in section 2.3.2 below.

$$\begin{aligned} &\min_{\mathbf{x}} \ G(\mathbf{x}) \\ &\textit{subject to:} \\ &\sigma_\pi(\mathbf{d}(\mathbf{x})) \geq 0 \quad \forall \text{ paths } \pi \in \Pi \\ &\text{(some pins fixed)} \end{aligned} \tag{2.2.7}$$

Finally, one last piece of notation. Let Π_e denote the set of all paths containing edge e, and let $\Pi_i = \bigcup_{e \in i} \Pi_e$ denote the set of all paths passing through net i. Denote by $\mathfrak{F}$ the set of feasible placements $\mathbf{x}$ in the problem (2.2.7). Analogously, denote by $\mathfrak{F}_0$ the set of placements strictly feasible for all path timing constraints, i.e. the set of all $\mathbf{x}$ for which $\sigma_\pi(\mathbf{d}(\mathbf{x})) > 0$ for all paths $\pi \in \Pi$. We make the mild assumption that $\mathfrak{F}_0 \neq \emptyset$.

2.2.2 Net Weighting Framework

Now let us introduce the sequence of *weighted objective functions* $N^k(\mathbf{x})$ corresponding to the index $k = 1, 2, \ldots$ as the following:

$$N^k(\mathbf{x}) = G(\mathbf{x}) + \Psi^k(\mathbf{x}) \tag{2.2.8}$$

where

$$\Psi^k(\mathbf{x}) = \sum_{\text{nets } i \in \mathcal{N}} w_i^k(\mathbf{d}(\mathbf{x}))h_i(\mathbf{x})$$

Here for each net i, $\{w_i^k\}_{k=1}^{\infty}$ is a sequence of *continuous, nonnegative* weighting functions which we require to satisfy the following property, termed the *asymptotic slack control*, for any fixed edge delay vector $\mathbf{d} = \mathbf{d}(\mathbf{x})$:

$$\lim_{k\to\infty} w_i^k(\mathbf{d}) = \begin{cases} 0, & \text{if } \sigma_i(\mathbf{d}) > 0. \\ c_i(\mathbf{d}), & \text{if } \sigma_i(\mathbf{d}) = 0, \text{ where } c_i(\mathbf{d}) \text{ is some finite constant.} \\ \infty, & \text{if } \sigma_i(\mathbf{d}) < 0. \end{cases}$$

(2.2.9) Asymptotic slack control.

Note that since $N^k(\mathbf{x}) = \sum_{\text{nets } i \in \mathcal{N}}[1+w_i^k(\mathbf{x})]h_i(\mathbf{x})$, implementing the new objective amounts to simply multiplying each net measure $h_i(\mathbf{x})$ by the *net weight* $1 + w_i^k(\mathbf{x})$ in the original objective $G(\mathbf{x})$.

Please refer to Tables 2.8.1 and 2.8.2 in the Appendix for a full summary of notation used throughout this work.

2.2.3 Weighting Examples

We now identify particular weighting schemes that satisfy the sufficient criteria for convergence.

We can modify each of the VPR, PATH, and APlace weighting schemes, as defined in (2.1.1), (2.1.2), and (2.1.3), respectively, to satisfy the asymptotic slack control. To do this, we create *sequences* of weighting functions using an increasing *weighting parameter* α_k in

place of the ad-hoc, user-defined parameter α. It should be noted that this change to satisfy the asymptotic slack control is necessary for convergence to the optimum: without such a modification, specific circuit examples can be found for which the original VPR, PATH, and APlace methods may fail to arrive at the optimum placement. Also, each of these three weighting schemes have been originally defined as *edge* weighting schemes; we may broaden their scope to *net* weighting schemes by summing over the edge weights, i.e. $w_i = \sum_{e \in i} w_e$.

We modify the VPR weights to the following, named *VPR-c*:

$$w_i^k(\mathbf{d}(\mathbf{x})) = \sum_{e \in i} \left(1 - \frac{\sigma_e(\mathbf{d}(\mathbf{x}))}{T}\right)^{\alpha_k} \tag{2.2.10}$$

Here $\{\alpha_k\}$ is a sequence of parameters approaching infinity, and the max delay of the previous placement, T_p, has been replaced with the desired max path delay T. This weighting scheme satisfies the asymptotic slack control.

We use the following modified formulation of the PATH weighting, which we name *PATH-c*:

$$w_i^k(\mathbf{d}(\mathbf{x})) = \sum_{\pi \in \Pi_i} \alpha_k^{-\frac{\sigma_\pi(\mathbf{d}(\mathbf{x}))}{T}} \tag{2.2.11}$$

Here, Π_i is the set of all paths passing through net i, and $\{\alpha_k\}$ is again a sequence of parameters approaching infinity, with $\alpha_k > 1$ for each k. This weighting scheme satisfies the asymptotic slack control. Figure 2.2.1 shows the shape of the PATH-c weighting function for increasing values of the net weighting parameter.

For the APlace weighting, we additionally replace the *desired timing improvement* T_u with the fixed desired max path delay T. The following modified weighting formulation is named *APlace-c*:

$$\begin{gathered} w_i^k(\mathbf{d}(\mathbf{x})) = \sum_{\pi \in \Pi_i} f^{\alpha_k}\big(d_\pi(\mathbf{x})\big), \\ \text{where } f^\alpha(d) = \begin{cases} \left(\frac{d}{T}\right)^\alpha - 1 & \text{if } d > T \\ 0 & \text{otherwise} \end{cases} \end{gathered} \tag{2.2.12}$$

Here, $d_\pi(\mathbf{x}) = \sum_{e \in \pi} d_e(\mathbf{x})$, and $\{\alpha_k\}$ is a sequence of parameters approaching infinity. This weighting scheme satisfies the asymptotic slack control.

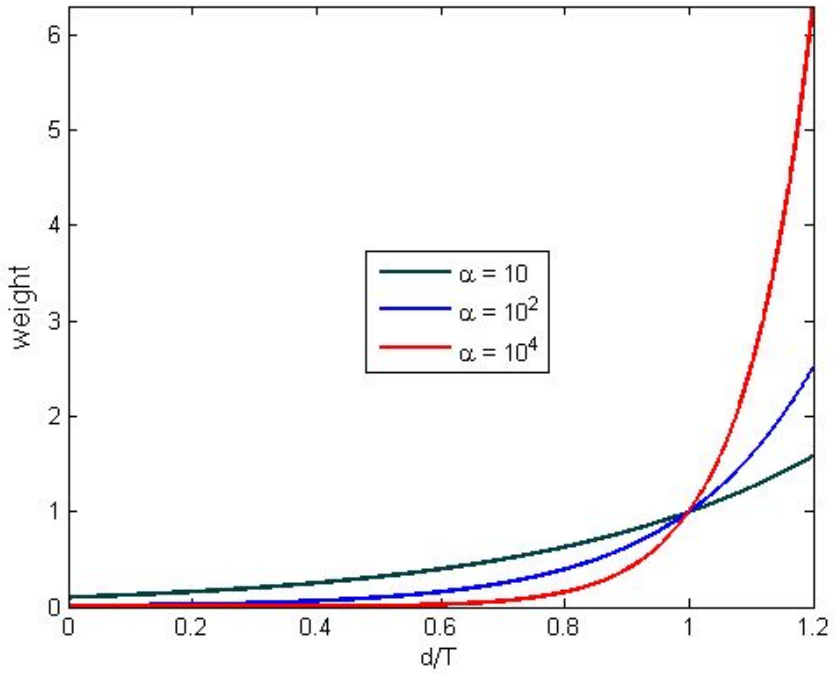

Figure 2.2.1: PATH-c weighting for increasing net weighting parameter α.

Note that the authors of the PATH and APlace weightings have devised efficient methods to calculate the weights, despite the exponential number paths through a net enumerated in their definitions. As described in [29], the PATH algorithm computes all weights in time linear to the number of pins plus the number of edges.

2.2.4 Preliminary Results

Before stating the main results of this chapter, we present a necessary lemma and corollary to be used later.

Lemma 2.2.1. *For any edge delay vector* $\mathbf{d} = [d_1 \ \ldots \ d_M]^T$*, the edge slack is related to the path slacks by*

$$\sigma_e(\mathbf{d}) = \min_{\pi \in \Pi_e} \{\sigma_\pi(\mathbf{d})\}$$

Proof. First we show that $\sigma_e(\mathbf{d}) \leq \sigma_\pi(\mathbf{d})$ for any $\pi \in \Pi_e$. Given $e = (s_1, t_1)$, take any

$\pi = (s_p, \ldots, s_2, s_1, t_1, t_2, \ldots, t_q) \in \Pi_e$. Then by the definitions above we have

$$\begin{aligned}
\sigma_e(\mathbf{d}) &= R_{t_1}(\mathbf{d}) - A_{s_1}(\mathbf{d}) - d_{(s_1,t_1)} \\
&\leq R_{t_2}(\mathbf{d}) - d_{(t_1,t_2)} - A_{s_2}(\mathbf{d}) - d_{(s_2,s_1)} - d_{(s_1,t_1)} \\
&\vdots \\
&\leq R_{t_q}(\mathbf{d}) - A_{s_p}(\mathbf{d}) - \sum_{e_0 \in \pi} d_{e_0} \\
&= T - \sum_{e_0 \in \pi} d_{e_0} = \sigma_\pi(\mathbf{d})
\end{aligned} \tag{2.2.13}$$

Now, using a similar line of reasoning, given $e = (s_1, t_1)$ we show there exists a path $\pi^* \in \Pi_e$ such that $\sigma_e(\mathbf{d}) = \sigma_{\pi^*}(\mathbf{d})$. We do so by construction:

Start with $p = 1$, $q = 1$.

While $s_p \notin \mathcal{P}_\mathcal{I}$

Choose $s_{p+1} \in \text{fanin}(s_p)$ such that $A_{s_{p+1}}(\mathbf{d}) + d_{(s_{p+1},s_p)} = A_{s_p}(\mathbf{d})$.

Set $p \leftarrow p + 1$.

End while

While $t_q \notin \mathcal{P}_\mathcal{O}$

Choose $t_{q+1} \in \text{fanout}(t_q)$ such that $R_{t_{q+1}}(\mathbf{d}) - d_{(t_q,t_{q+1})} = R_{t_q}(\mathbf{d})$.

Set $q \leftarrow q + 1$.

End while

Set $\pi^* = (s_p, \ldots, s_2, s_1, t_1, t_2, \ldots, t_q)$.

It follows from construction of π^* that each inequality in (2.2.13) becomes an equality, so that $\sigma_e(\mathbf{d}) = \sigma_{\pi^*}(\mathbf{d})$. Combining this with the above statement that $\sigma_e(\mathbf{d}) \leq \sigma_\pi(\mathbf{d})$ for any $\pi \in \Pi_e$, we conclude that $\sigma_e(\mathbf{d}) = \min_{\pi \in \Pi_e}\{\sigma_\pi(\mathbf{d})\}$. □

Corollary 2.2.2. *The condition*

$$\textit{If } \sigma_\pi(\mathbf{d}) > 0 \quad \forall \pi \in \Pi_i, \quad \textit{then } \lim_{k\to\infty} w_i^k(\mathbf{d}) = 0. \tag{2.2.14a}$$

$$\textit{If } \sigma_\pi(\mathbf{d}) \geq 0 \quad \forall \pi \in \Pi_i, \quad \textit{then } \lim_{k\to\infty} w_i^k(\mathbf{d}) = c_i(\mathbf{d}). \tag{2.2.14b}$$

$$\textit{If } \sigma_\pi(\mathbf{d}) < 0 \quad \textit{for some } \pi \in \Pi_i, \quad \textit{then } \lim_{k\to\infty} w_i^k(\mathbf{d}) = \infty. \tag{2.2.14c}$$

where $c_i(\mathbf{d})$ is some finite constant, is equivalent to the asymptotic slack control (2.2.9).

This corollary will be useful in the following discussion.

2.3 Global Convergence

2.3.1 Proof of Convergence

Given this framework and net weightings defined by (2.2.8), we can guarantee convergence of a subsequence of the global minimizers of the weighted objectives $N^k(\mathbf{x})$ to a global minimizer of the original constrained problem.

Theorem 2.3.1. *Suppose that $\mathbf{x}^k$ is a global minimizer of the weighted objective $N^k(\mathbf{x})$ for each $k = 1, 2, \ldots$. Then every limit point of the sequence $\{\mathbf{x}^k\}$ is a global minimizer of the problem (2.2.7).*

Proof. Let $\mathbf{z}$ be a global minimizer of the problem (2.2.7). First note that since each $\mathbf{x}^k$ minimizes $N^k(\mathbf{x})$, we have that

$$N^k(\mathbf{x}^k) \leq N^k(\mathbf{z}), \text{ for all } k$$

i.e.,

$$G(\mathbf{x}^k) + \Psi^k(\mathbf{x}^k) \leq G(\mathbf{z}) + \Psi^k(\mathbf{z}), \text{ for all } k \tag{2.3.1}$$

Suppose $\mathbf{x}^*$ is a limit point of $\{\mathbf{x}^k\}$, so there exists some subsequence K such that $\lim_{k\in K} \mathbf{x}^k = \mathbf{x}^*$.

We first show that $\mathbf{x}^* \in \mathfrak{F}$. We have by continuity that

$$\sigma_\pi(\mathbf{d}(\mathbf{x}^*)) = \lim_{k\in K} \sigma_\pi(\mathbf{d}(\mathbf{x}^k))$$

for each path π. Suppose that $\sigma_\pi(\mathbf{d}(\mathbf{x}^*)) < 0$ for some π. Then there there exists some Z such that for all $k \in K$ with $k > Z$,

$$\sigma_\pi(\mathbf{d}(\mathbf{x}^k)) < -\epsilon$$

for some $\epsilon > 0$. By property (2.2.14) it follows that for each net i through which π passes,

$$w_i^k(\mathbf{d}(\mathbf{x}^k)) \to \infty$$

Furthermore, since $\sigma_\pi(\mathbf{d}(\mathbf{x}^*)) < 0$ it follows by definition that $\sum_{e\in\pi} d_e(\mathbf{x}^*) > T > 0$, so there exists at least one $\hat{e} = (\hat{s}, \hat{t}) \in \pi$ such that $d_{\hat{e}}(\mathbf{x}^*) > 0$. Thus by (2.2.2), $x^*_{\hat{s}} \neq x^*_{\hat{t}}$ so that for the net $\hat{i} \ni \hat{e}$, it follows from (2.2.1) and nonnegativity that $h_{\hat{i}}(\mathbf{x}^*) > 0$. Then again by continuity we can choose $k \in K$ sufficiently large so that for some $R > 0$,

$$h_{\hat{i}}(\mathbf{x}^k) > R$$

Thus we get:

$$\begin{aligned} \Psi^k(\mathbf{x}^k) &= \sum_{\text{nets } i\in\mathcal{N}} w_i^k(\mathbf{d}(\mathbf{x}^k))h_i(\mathbf{x}^k) \\ &\geq w_{\hat{i}}^k(\mathbf{d}(\mathbf{x}^k))h_{\hat{i}}(\mathbf{x}^k) \\ &\geq w_{\hat{i}}^k(\mathbf{d}(\mathbf{x}^k))R \to \infty \text{ as } k \to \infty \end{aligned} \tag{2.3.2}$$

Now, since $\mathbf{z} \in \mathfrak{F}$, it follows from (2.2.14) that $\lim_{k\in K} \Psi^k(\mathbf{z}) = C$ for some constant $C \geq 0$. This fact, combined with (2.3.2), contradicts (2.3.1) for sufficiently large $k \in K$. We thus conclude that $\sigma_\pi(\mathbf{d}(\mathbf{x}^*)) \geq 0$ for all paths π, so that $\mathbf{x}^* \in \mathfrak{F}$.

Now suppose that $\mathbf{x}^*$ is *not* a global minimizer of (2.2.7), i.e. $G(\mathbf{x}^*) > G(\mathbf{z})$. We will show that this implies that there exists a point $\mathbf{y} \in \mathfrak{F}_0$ such that $G(\mathbf{x}^*) > G(\mathbf{y})$.

If $\mathbf{z} \in \mathfrak{F}_0$, take $\mathbf{y} = \mathbf{z}$. Otherwise, choose any point $\tilde{\mathbf{x}} \in \mathfrak{F}_0$, and assume $G(\mathbf{x}^*) \leq G(\tilde{\mathbf{x}})$ (otherwise, take $\mathbf{y} = \tilde{\mathbf{x}}$). Now let $\mathbf{x}_\beta = \beta\tilde{\mathbf{x}} + (1-\beta)\mathbf{z}$ for $\beta \in (0,1)$. For each path π, we know that $\sigma_\pi(\mathbf{d}(\tilde{\mathbf{x}})) > 0$ and $\sigma_\pi(\mathbf{d}(\mathbf{z})) \geq 0$. It follows by convexity of the delay function that the pathwise slack function is concave; hence $\sigma_\pi(\mathbf{d}(\mathbf{x}_\beta)) \geq \beta\big(\sigma_\pi(\mathbf{d}(\tilde{\mathbf{x}}))\big) > 0$, so that $\mathbf{x}_\beta \in \mathfrak{F}_0$ for all $\beta \in (0,1)$.

Now it follows by continuity that we can choose $\beta = \beta_1$ sufficiently small so that $G(\mathbf{x}_{\beta_1}) < G(\mathbf{x}^*)$. We have found our desired point $\mathbf{y} = \mathbf{x}_{\beta_1}$.

Now recall that by definition of $\mathbf{x}^k$,

$$G(\mathbf{x}^k) + \Psi^k(\mathbf{x}^k) \leq G(\mathbf{y}) + \Psi^k(\mathbf{y}), \text{ for all } k$$

We can take the limit of both sides of this inequality to obtain

$$G(\mathbf{x}^*) + \lim_{k \in K} \Psi^k(\mathbf{x}^k) \leq G(\mathbf{y})$$

and so by nonnegativity of each w_i^k and h_i,

$$G(\mathbf{x}^*) \leq G(\mathbf{y}) \tag{2.3.3}$$

which contradicts our previous assertion that $G(\mathbf{x}^*) > G(\mathbf{y})$. We conclude that $\mathbf{x}^*$ is a global minimizer of (2.2.7). □

2.3.2 Extension to Generalized Density Constraints

Up to this point, we have made the simplifying assumption that some pins are fixed and there are no density or other additional constraints in the problem. We show in this section that the extension of problem (2.2.7) to include more general equality constraints does not interfere with the convergence of the net weighting scheme. The treatment of the density constraint as an equality constraint was conducted effectively in [13]. We choose to include a penalty term on the weighted objective and use standard techniques, as in e.g. [36], to show convergence.

Suppose we wish to solve the modified problem

$$\begin{aligned} &\min_{\mathbf{x}} \ G(\mathbf{x}) \\ &\textit{subject to:} \\ &\sigma_\pi(\mathbf{d}(\mathbf{x})) \geq 0 \quad \forall \text{ paths } \pi \in \Pi \\ &D_r(\mathbf{x}) = 0 \quad \forall r = 1, \ldots, R \end{aligned} \tag{2.3.4}$$

where the constraints $D_r(\mathbf{x}) = 0$ represent generalized density constraints, which could be used to account for overlap, routability, temperature, and so on. Traditionally, analytical placers (e.g., [41]) divide the placement area into a grid of "bins" and discourage overlapping cells by bounding the average density in each bin using an inequality constraint. In [13], filler "dummy" cells were introduced to convert the inequality constraints into equality constraints and were shown to be effective.

Let us redefine the weighted objective (2.2.8) to include additional penalty terms for the generalized density constraints:

$$\hat{N}^k(\mathbf{x}) = G(\mathbf{x}) + \Psi^k(\mathbf{x}) + \sum_{r=1}^{R} p_r^k\left(D_r(\mathbf{x})\right) \tag{2.3.5}$$

Here each p_r^k is a nonnegative function such that

$$\lim_{k\to\infty} p_r^k(D) = \begin{cases} 0 & \text{if } D = 0 \\ \infty & \text{otherwise} \end{cases}$$

Let $\mathfrak{D} = \{\mathbf{x} : D_r(\mathbf{x}) = 0,\ r = 1, \ldots, R\}$. We make the assumptions that $\mathfrak{F}_0 \cap \mathfrak{D} \neq \emptyset$, and that $\mathfrak{F} \cap \mathfrak{D}$ is in the closure of $\mathfrak{F}_0 \cap \mathfrak{D}$. It follows readily that we can modify Theorem 2.3.1 to include the penalty term:

Corollary 2.3.2. *Suppose that $\mathbf{x}^k$ is a global minimizer of the objective $\hat{N}^k(\mathbf{x})$ for each $k = 1, 2, \ldots$. Then every limit point of the sequence $\{\mathbf{x}^k\}$ is a global minimizer of the problem* (2.3.4).

Proof. We use the proof of Theorem 2.3.1 with a few modifications. First, note that the limit point $\mathbf{x}^* \in \mathfrak{D}$ in addition to $\mathbf{x}^* \in \mathfrak{F}$ as shown above, since otherwise $\sum_{r=1}^{R} p_r^k\left(D_r(\mathbf{x}^k)\right) \to \infty$, contradicting the fact that $\mathbf{x}^k$ minimizes $\hat{N}^k(\mathbf{x})$ for sufficiently large $k \in K$. Then, assuming that $\mathbf{x}^*$ is not a global minimum, we can find $\mathbf{y} \in \mathfrak{F}_0 \cap \mathfrak{D}$ such that $G(\mathbf{x}^*) > G(\mathbf{y})$, following from the fact that we can select points in $\mathfrak{F}_0 \cap \mathfrak{D}$ arbitrarily close to the global minimum $\mathbf{z}$, and use it to obtain a contradiction as in (2.3.3). □

2.4 Local Convergence

In many cases, given highly non-convex density constraints, finding global minimizers of the weighted objectives N^k is not practical. With this in mind, we now consider the case where we can only find a local minimizer for each subproblem. We show that doing so will still converge to a Karush-Kuhn-Tucker (KKT) point [36], i.e. a candidate point for a local minimizer, of an equivalent formulation of the original problem (2.2.7).

2.4.1 Preliminaries

2.4.1.1 Assumptions

We may retain all original assumptions about the functions d_e and h_i; however, the essential properties for local convergence are that for each edge e and net i, each h_i and d_e is *nonnegative*.

We also make the additional assumptions that the functions d_e and h_i are *continuously differentiable*, and that h_i is *Lipschitz continuous* in the feasible region $\mathfrak{F}$ and bounded below by some $\kappa > 0$. Note that Lipschitz continuity follows from continuous differentiability of h_i if $\mathfrak{F}$ is compact. Also, note that the lower bound κ on h_i holds for the log-sum-exp wire length definition; for the quadratic and half-perimeter wire lengths, it can be satisfied simply by making the adjustment $h_i^{\text{new}} = h_i + \kappa$.

2.4.1.2 Net Weighting Framework

As in Section 2.4.1.1, we may retain all original requirements on the net weighting functions w_i^k, but the essential properties for local convergence are that each w_i^k is *nonnegative*, and that the asymptotic slack control (2.2.9) is satisfied.

Further, we require that each w_i^k is *continuously differentiable* and make some additional requirements on their structure. First, we require that each w_i^k is *nondecreasing*:

$$\boxed{\nabla_{\mathbf{d}} w_i^k(\mathbf{d}(\mathbf{x})) \geq \mathbf{0}}$$

(2.4.1) Nondecreasing property.

Thus, increasing any edge delay while holding all other delays constant cannot have the effect of *reducing* a weight.

Secondly, we require that the weighting functions are *non-critically indifferent*: in the

limit, changes in non-critical edge delays do not have an impact on the weights.

$$\boxed{\text{If } \sigma_e(\mathbf{d}(\mathbf{x})) > 0, \text{ then } \lim_{k\to\infty} \frac{\partial w_i^k(\mathbf{d}(\mathbf{x}))}{\partial d_e} = 0 \ \ \forall \text{ nets } i}$$

(2.4.2) Non-critical indifference.

Also, we require that each w_i^k is *path consistent*: it can be written as some function of the path delays in the circuit.

$$\boxed{w_i^k(\mathbf{d}(\mathbf{x})) = f_i^k\big(d_{\pi_1}(\mathbf{x}), \ldots, d_{\pi_S}(\mathbf{x})\big)}$$

(2.4.3) Path consistency.

Finally, we make the mild expectation that the net weighting objectives $N^k(\mathbf{x})$ do not become arbitrarily "jagged" around local minimizers in $\mathfrak{F}$; that is, there exists a $\xi > 0$ independent of k such that if $\mathbf{y} \in \mathfrak{F}$ is a local minimizer for $N^k(\mathbf{x})$, then $N^k(\mathbf{x})$ is convex in $B_{\mathfrak{F}}(\mathbf{y}, \xi) = \{\mathbf{x} : \mathbf{x} \in \mathfrak{F}, \ ||\mathbf{x} - \mathbf{y}|| < \xi\}$.

We can verify that each of the identified convergent weighting schemes VPR-c (2.2.10), PATH-c (2.2.11), and APlace-c (2.2.12) are indeed *nondecreasing*, *non-critically indifferent*, and *path consistent*.

The differentiability requirement is satisfied by the PATH-c weights, and may be satisfied by a slight modification in the VPR-c and APlace-c weights to smooth around non-differentiable points. The non-differentiability is due to the use of max and min functions in computing the slacks in the VPR-c weights, and due to the piecewise nature of the weighting function at the point $d = T$ in the APlace-c weights. The non-differentiable points in the VPR-c weightings can be removed if the log-sum-exp smooth approximations of the max and min functions [41] are used in computing the slacks. The non-differentiable points in the APlace-c weightings can be removed by a simple smoothing of f^a around the point $d = T$.

2.4.1.3 Problem Formulation

In an equivalent formulation to (2.2.7), we introduce the new variable $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, where a_j is an upper bound on the arrival time of pin j. The problem then becomes:

$$\begin{aligned}
&\min_{\mathbf{x},\mathbf{a}} \; G(\mathbf{x}) \\
&\textit{subject to:} \\
&a_s + d_e(\mathbf{x}) \leq a_t \quad \forall \text{ edges } e \in \mathcal{E}, \text{ where } e = (s,t) \\
&0 \leq a_j \quad \forall \text{ pins } j \in \mathcal{P}_{\mathcal{I}} \\
&a_\ell \leq T \quad \forall \text{ pins } \ell \in \mathcal{P}_{\mathcal{O}} \\
&\text{(some pins fixed)}
\end{aligned} \tag{2.4.4}$$

Here we denote the set of all edges by $\mathcal{E}$. Again we assume for simplicity that some pins are fixed; see Section 2.4.3 for inclusion of generalized density constraints. For this problem, the Lagrangian is:

$$\begin{aligned}
\mathcal{L}(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{v}, \boldsymbol{\lambda}) = \sum_{\text{nets } i \in \mathcal{N}} h_i(\mathbf{x}) &- \sum_{\text{pins } j \in \mathcal{P}_{\mathcal{I}}} u_j a_j - \sum_{\text{pins } \ell \in \mathcal{P}_{\mathcal{O}}} v_\ell (T - a_\ell) \\
&- \sum_{\text{edges } e=(s,t) \in \mathcal{E}} \lambda_e (a_t - a_s - d_e(\mathbf{x}))
\end{aligned}$$

Then for any local minimum $(\bar{\mathbf{x}}, \bar{\mathbf{a}})$ for the problem (2.4.4) at which the linear independence constraint qualification [36] holds, by the KKT conditions there are Lagrange multipliers

$(\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\boldsymbol{\lambda}})$ such that the following conditions are satisfied:

$$\sum_{e\in\mathcal{E}} \bar{\lambda}_e \nabla d_e(\bar{\mathbf{x}}) = -\sum_{i\in\mathcal{N}} \nabla h_i(\bar{\mathbf{x}}) \tag{2.4.5a}$$

$$\bar{u}_j = \sum_{e=(j,t)\in\mathcal{E}} \bar{\lambda}_e \qquad \forall j \in \mathcal{P}_{\mathcal{I}} \tag{2.4.5b}$$

$$\bar{a}_j \geq 0 \text{ and } \bar{u}_j \geq 0, \text{ with at least one a strict equality} \qquad \forall j \in \mathcal{P}_{\mathcal{I}} \tag{2.4.5c}$$

$$\bar{v}_\ell = \sum_{e=(s,\ell)\in\mathcal{E}} \bar{\lambda}_e \qquad \forall \ell \in \mathcal{P}_{\mathcal{O}} \tag{2.4.5d}$$

$$\bar{a}_\ell \leq T \qquad \forall \ell \in \mathcal{P}_{\mathcal{O}} \tag{2.4.5e}$$

$$\bar{v}_\ell \geq 0 \qquad \forall \ell \in \mathcal{P}_{\mathcal{O}} \tag{2.4.5f}$$

$$\bar{v}_\ell = 0 \text{ or } \bar{a}_\ell = T \qquad \forall \ell \in \mathcal{P}_{\mathcal{O}} \tag{2.4.5g}$$

$$\sum_{e_1=(s,m)\in\mathcal{E}} \bar{\lambda}_{e_1} = \sum_{e_2=(m,t)\in\mathcal{E}} \bar{\lambda}_{e_2} \qquad \forall m \in \mathcal{P} \setminus (\mathcal{P}_{\mathcal{I}} \cup \mathcal{P}_{\mathcal{O}}) \tag{2.4.5h}$$

$$\bar{a}_s + d_e(\mathbf{x}) \leq \bar{a}_t \qquad \forall e = (s,t) \in \mathcal{E} \tag{2.4.5i}$$

$$\bar{\lambda}_e \geq 0 \qquad \forall e \in \mathcal{E} \tag{2.4.5j}$$

$$\bar{\lambda}_e = 0 \text{ or } \bar{a}_s + d_e(\mathbf{x}) = \bar{a}_t \qquad \forall e = (s,t) \in \mathcal{E} \tag{2.4.5k}$$

2.4.2 Local Convergence Proof

Given a placement $\mathbf{x}$ and an index k, we shall define the variable $\tilde{\mathbf{a}}(\mathbf{d}(\mathbf{x}))$ and the *Lagrange multiplier estimates* $\tilde{\mathbf{u}}^k(\mathbf{x})$, $\tilde{\mathbf{v}}^k(\mathbf{x})$, and $\tilde{\boldsymbol{\lambda}}^k(\mathbf{x})$ as follows:

$$\tilde{a}_m(\mathbf{d}(\mathbf{x})) = A_m(\mathbf{d}(\mathbf{x})) \qquad \forall m \in \mathcal{P} \tag{2.4.6a}$$

$$\tilde{\lambda}_e^k(\mathbf{x}) = \sum_{i\in\mathcal{N}} \frac{\partial w_i^k(\mathbf{d}(\mathbf{x}))}{\partial d_e} h_i(\mathbf{x}) \qquad \forall e \in \mathcal{E} \tag{2.4.6b}$$

$$\tilde{u}_j^k(\mathbf{x}) = \sum_{e=(j,t)\in\mathcal{E}} \tilde{\lambda}_e^k(\mathbf{x}) \qquad \forall j \in \mathcal{P}_{\mathcal{I}} \tag{2.4.6c}$$

$$\tilde{v}_\ell^k(\mathbf{x}) = \sum_{e=(s,\ell)\in\mathcal{E}} \tilde{\lambda}_e^k(\mathbf{x}) \qquad \forall \ell \in \mathcal{P}_{\mathcal{O}} \tag{2.4.6d}$$

Theorem 2.4.1. *Suppose that $\mathbf{x}^k$ is a local minimizer of the weighted objective $N^k(\mathbf{x})$ for each $k = 1, 2, \ldots,$ and that $\mathbf{x}^k \in \mathfrak{F}_0$ for all k sufficiently large. Then every limit point of $\{\mathbf{x}^k\}$ is a KKT point of the problem* (2.4.4).

Proof. As in the proof of Theorem 2.3.1, let $\mathbf{x}^*$ be a limit point of $\{\mathbf{x}^k\}$, so there exists some subsequence K such that $\lim_{k\in K} \mathbf{x}^k = \mathbf{x}^*$. The proof will proceed by considering the variable $\tilde{\mathbf{a}}(\mathbf{d}(\mathbf{x}^k))$ and the estimates $\tilde{\boldsymbol{\lambda}}^k(\mathbf{x}^k)$, $\tilde{\mathbf{u}}^k(\mathbf{x}^k)$, and $\tilde{\mathbf{v}}^k(\mathbf{x}^k)$ given by (2.4.6) and showing that these satisfy the KKT conditions (2.4.5) in the limit.

Properties (2.4.5b), (2.4.5d), and (2.4.5i) are clearly satisfied at every iteration via construction. Property (2.4.5j) is also satisfied at every iteration, following from the nondecreasing property (2.4.1). Property (2.4.5c) is satisfied at every iteration as well: this follows from the aforementioned fact that $\tilde{\lambda}_e^k(\mathbf{x}) \geq 0$ for all k and $e \in \mathcal{E}$, and from the fact that $\tilde{a}_m(\mathbf{d}(\mathbf{x})) = A_m(\mathbf{d}(\mathbf{x})) = 0$ for all $m \in \mathcal{P}_{\mathcal{I}}$. Similarly, Property (2.4.5f) is satisfied at every iteration.

For Property (2.4.5h), first note that for any $m \in \mathcal{P} \setminus (\mathcal{P}_{\mathcal{I}} \cup \mathcal{P}_{\mathcal{O}})$,

$$\bigcup_{e_1=(s,m)\in\mathcal{E}} \Pi_{e_1} = \bigcup_{e_2=(m,t)\in\mathcal{E}} \Pi_{e_2}$$

Also, from path consistency (2.4.3) we have

$$\frac{\partial w_i^k(\mathbf{d})}{\partial d_e} = \sum_{\pi\in\Pi_e} \frac{\partial f_i^k\big(d_{\pi_1}, \ldots, d_{\pi_S}\big)}{\partial d_\pi}$$

Combining these, we get

$$\begin{aligned}
\sum_{e_1=(s,m)\in\mathcal{E}} \tilde{\lambda}_{e_1}^k(\mathbf{x}) &= \sum_{e_1=(s,m)\in\mathcal{E}} \left[\sum_{i\in\mathcal{N}} \frac{\partial w_i^k(\mathbf{d}(\mathbf{x}))}{\partial d_{e_1}} h_i(\mathbf{x})\right] \\
&= \sum_{e_1=(s,m)\in\mathcal{E}} \left[\sum_{i\in\mathcal{N}} \left(\sum_{\pi\in\Pi_{e_1}} \frac{\partial f_i^k\big(d_{\pi_1}(\mathbf{x}), \ldots, d_{\pi_S}(\mathbf{x})\big)}{\partial d_\pi} h_i(\mathbf{x})\right)\right] \\
&= \sum_{e_2=(m,t)\in\mathcal{E}} \left[\sum_{i\in\mathcal{N}} \left(\sum_{\pi\in\Pi_{e_2}} \frac{\partial f_i^k\big(d_{\pi_1}(\mathbf{x}), \ldots, d_{\pi_S}(\mathbf{x})\big)}{\partial d_\pi} h_i(\mathbf{x})\right)\right] \\
&= \sum_{e_2=(m,t)\in\mathcal{E}} \tilde{\lambda}_{e_2}^k(\mathbf{x})
\end{aligned}$$

Thus Property (2.4.5h) is satisfied at every iteration.

The complementary slackness condition (2.4.5k) is satisfied in the limit: since each $\mathbf{x}^k$ is timing feasible for k sufficiently large, it follows that $A_t(\mathbf{d}(\mathbf{x}^k)) \leq R_t(\mathbf{d}(\mathbf{x}^k))$ for all k sufficiently large and all pins $t \in \mathcal{P}$. Consider any edge $e = (s,t) \in \mathcal{E}$ such that $\tilde{a}_s(\mathbf{d}(\mathbf{x}^*)) +$

$d_e(\mathbf{x}^*) < \tilde{a}_t(\mathbf{d}(\mathbf{x}^*))$; thus, $\tilde{a}_s(\mathbf{d}(\mathbf{x}^k)) + d_e(\mathbf{x}^k) < \tilde{a}_t(\mathbf{d}(\mathbf{x}^k)) - \epsilon$ for some $\epsilon > 0$ for all k sufficiently large. Then

$$\sigma_e(\mathbf{d}(\mathbf{x}^k)) = R_t(\mathbf{d}(\mathbf{x}^k)) - A_s(\mathbf{d}(\mathbf{x}^k)) - d_e(\mathbf{x}^k) \geq A_t(\mathbf{d}(\mathbf{x}^k)) - A_s(\mathbf{d}(\mathbf{x}^k)) - d_e(\mathbf{x}^k) > \epsilon$$

Thus $\sigma_e(\mathbf{d}(\mathbf{x}^*)) > \epsilon$ so it follows from non-critical indifference (2.4.2) and continuity that $\lim_{k\in K} \tilde{\lambda}_e^k(\mathbf{x}^k) = 0$.

Properties (2.4.5e) and (2.4.5g) are shown in a similar manner to Property (2.4.5k). It follows from timing feasibility for k sufficiently large that Property (2.4.5e) is satisfied in the limit. For Property (2.4.5g), consider any $\ell \in \mathcal{P}_{\mathcal{O}}$ such that $\tilde{a}_\ell(\mathbf{d}(\mathbf{x}^*)) < T$, so that $\tilde{a}_\ell(\mathbf{d}(\mathbf{x}^k)) < T - \epsilon$ for some $\epsilon > 0$ for all k sufficiently large. Now note that for any edge $(s, \ell) \in \mathcal{E}$, we have that $A_\ell(\mathbf{d}(\mathbf{x}^k)) \geq A_s(\mathbf{d}(\mathbf{x}^k)) + d_{(s,\ell)}(\mathbf{x}^k)$, and since $\ell \in \mathcal{P}_{\mathcal{O}}$, we have $R_\ell(\mathbf{d}(\mathbf{x}^k)) = T$. Thus,

$$\sigma_{(s,\ell)}(\mathbf{d}(\mathbf{x}^k)) = T - A_s(\mathbf{d}(\mathbf{x}^k)) - d_{(s,\ell)}(\mathbf{x}^k) \geq T - A_\ell(\mathbf{d}(\mathbf{x}^k)) > \epsilon$$

Thus again by non-critical indifference (2.4.2), it follows that $\lim_{k\in K} \tilde{v}_\ell^k(\mathbf{x}^k) = 0$.

Finally we prove Property (2.4.5a) in the limit. First we show that

$$w_i^k(\mathbf{d}(\mathbf{x}^k)) \to 0 \text{ for all } i \in \mathcal{N}$$

As a means to show this, denote by $\mathcal{N}_+$ the set of all nets $i \in \mathcal{N}$ such that there exists some $\epsilon > 0$ such that $w_i^k(\mathbf{d}(\mathbf{x}^k)) > \epsilon$ for all $k \in K$ sufficiently large. Further, let $\mathcal{N}_0 = \mathcal{N} \setminus \mathcal{N}_+$.

Suppose temporarily that $\mathcal{N}_+ \neq \emptyset$. Then given an arbitrarily small $0 < \delta_1 < \epsilon$ and letting $\delta = \min\{\delta_1, \xi\}$, we can find an index q and a point $\mathbf{y} \in \mathfrak{F}_0$ such that

$$||\mathbf{y} - \mathbf{x}^q|| < \delta$$
$$w_i^q(\mathbf{d}(\mathbf{y})) < \delta \quad \forall i \in \mathcal{N}$$

q and $\mathbf{y}$ can be found by first selecting an M sufficiently large and $\mathbf{y} \in \mathfrak{F}_0$ sufficiently close to $\mathbf{x}^*$ so that $\mathbf{x}^k \in \mathfrak{F}_0$ and $\left|\left|\mathbf{x}^k - \mathbf{y}\right|\right| < \delta$ for all $k > M$, then fixing $\mathbf{y}$ and increasing $q > M$

until $w_i^q(\mathbf{d}(\mathbf{y})) < \delta$ for all $i \in \mathcal{N}$. Then we have

$$\begin{aligned} N^q(\mathbf{x}^q) &> \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \epsilon \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) \\ &= \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \delta \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) + (\epsilon-\delta) \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) \\ &\geq \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \delta \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) + (\epsilon-\delta)\,|\mathcal{N}_+|\,\kappa \end{aligned} \tag{2.4.7}$$

Now by Lipschitz continuity, there exists some constant L such that

$$|h_i(\mathbf{x}) - h_i(\mathbf{z})| \leq L\,||\mathbf{x}-\mathbf{z}||$$

for all $i \in \mathcal{N}$ and $\mathbf{x}, \mathbf{z} \in \mathfrak{F}$. Thus $|h_i(\mathbf{y}) - h_i(\mathbf{x}^q)| \leq L\delta$, so

$$h_i(\mathbf{y}) \leq h_i(\mathbf{x}^q) + L\delta$$

for all $i \in \mathcal{N}$. Then we have:

$$\begin{aligned} N^q(\mathbf{y}) &< \sum_{i\in\mathcal{N}} \big[h_i(\mathbf{x}^q) + L\delta\big] + \delta \sum_{j\in\mathcal{N}_+} \big[h_j(\mathbf{x}^q) + L\delta\big] + \delta \sum_{j\in\mathcal{N}_0} H \\ &= \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \delta \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) + \delta\,|\mathcal{N}|\,L + \delta^2\,|\mathcal{N}_+|\,L + \delta\,|\mathcal{N}_0|\,H \end{aligned} \tag{2.4.8}$$

where $H = \max\{h_i(\mathbf{x}) : i \in \mathcal{N}, \mathbf{x} \in \mathfrak{F}\}$ is a positive real number, independent of q or $\mathbf{y}$. Thus it follows from (2.4.7) and (2.4.8) that we can choose δ_1 sufficiently small to obtain q and $\mathbf{y}$ such that $N^q(\mathbf{y}) < N^q(\mathbf{x}^q)$ for $||\mathbf{y}-\mathbf{x}^q|| < \xi$, a contradiction of the fact that $\mathbf{x}^q$ is a local minimizer of $N^q(\mathbf{x})$. We conclude that $\mathcal{N}_+ = \emptyset$ so that $w_i^k(\mathbf{d}(\mathbf{x}^k)) \to 0$ for all $i \in \mathcal{N}$.

Next, let us restate our definition of the weighted objective $N^k(\mathbf{x})$ in a slightly modified form:

$$N^k(\mathbf{x}) = \sum_{\text{nets } i\in\mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{x}))\Big) h_i(\mathbf{x})$$

Taking the gradient of this, we get

$$\begin{aligned} \nabla N^k(\mathbf{x}) &= \sum_{i\in\mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{x}))\Big) \nabla h_i(\mathbf{x}) + \sum_{i\in\mathcal{N}} \left[\sum_{e\in\mathcal{E}} \tfrac{\partial w_i^k(\mathbf{d}(\mathbf{x}))}{\partial d_e} \nabla d_e(\mathbf{x})\right] h_i(\mathbf{x}) \\ &= \sum_{i\in\mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{x}))\Big) \nabla h_i(\mathbf{x}) + \sum_{e\in\mathcal{E}} \left[\sum_{i\in\mathcal{N}} \tfrac{\partial w_i^k(\mathbf{d}(\mathbf{x}))}{\partial d_e} h_i(\mathbf{x})\right] \nabla d_e(\mathbf{x}) \\ &= \sum_{i\in\mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{x}))\Big) \nabla h_i(\mathbf{x}) + \sum_{e\in\mathcal{E}} \tilde{\lambda}_e^k(\mathbf{x}) \nabla d_e(\mathbf{x}) \end{aligned} \tag{2.4.9}$$

Now, since $w_i^k(\mathbf{d}(\mathbf{x}^k)) \to 0$ for all $i \in \mathcal{N}$ and since $\nabla N^k(\mathbf{x}) = 0$, it follows from (2.4.9) that Property (2.4.5a) is satisfied in the limit. As all KKT conditions are satisfied, we conclude that $\mathbf{x}^*$ is a KKT point of (2.4.4). □

The following corollary shows that we need not find local minimizers of the weighted objectives $N^k(\mathbf{x})$ *exactly*, but may instead use approximations.

Corollary 2.4.2. *Suppose that $\mathbf{x}^k$ approximates a local minimizer $\mathbf{x}_{min}^k$ of the weighted objective $N^k(\mathbf{x})$ for each $k = 1, 2, \ldots$, in that $\left|\left|\mathbf{x}^k - \mathbf{x}_{min}^k\right|\right| \to 0$ and $\left|\left|\nabla N^k(\mathbf{x}^k)\right|\right| \to 0$ as $k \to \infty$. If $\mathbf{x}^k \in \mathfrak{F}_0$ and $\mathbf{x}_{min}^k \in \mathfrak{F}$ for all k sufficiently large, then every limit point of $\{\mathbf{x}^k\}$ is a KKT point of the problem* (2.4.4).

Proof. The proof proceeds as that for Theorem 2.4.1 with a few modifications. We discuss Property (2.4.5a) fully.

First we show that $w_i^k(\mathbf{d}(\mathbf{x}^k)) \to 0$ for all $i \in \mathcal{N}$. Suppose temporarily that $\mathcal{N}_+ \neq \emptyset$. Then given an arbitrarily small $0 < \delta_1 < \epsilon$ and letting $\delta = \min\{\delta_1, \frac{\xi}{2}\}$, we can find an index q and a point $\mathbf{y} \in \mathfrak{F}_0$ such that

$$||\mathbf{y} - \mathbf{x}^q|| < \delta$$
$$||\nabla N^q(\mathbf{x}^q)|| < 1$$
$$||\mathbf{x}^q - \mathbf{x}_{\min}^q|| < \delta$$
$$w_i^q(\mathbf{d}(\mathbf{y})) < \delta \quad \forall i \in \mathcal{N}$$

$\mathbf{y}$ and q can be found by first selecting an M sufficiently large and $\mathbf{y}$ sufficiently close to $\mathbf{x}^*$ so that $\mathbf{x}^k \in \mathfrak{F}_0$, $\mathbf{x}_{\min}^k \in \mathfrak{F}$, $\left|\left|\mathbf{y} - \mathbf{x}^k\right|\right| < \delta$, $\left|\left|\nabla N^k(\mathbf{x}^k)\right|\right| < 1$, and $\left|\left|\mathbf{x}^k - \mathbf{x}_{\min}^k\right|\right| < \delta$ for all $f \in K$ such that $k > M$, then fixing $\mathbf{y}$ and increasing $q > M$ until $w_i^q(\mathbf{d}(\mathbf{y})) < \delta$ for all $i \in \mathcal{N}$.

Now note that it follows from Taylor's theorem and the Cauchy-Schwarz inequality that

$$|N^q(\mathbf{x}^q) - N^q(\mathbf{x}_{\min}^q)| = \left|(\mathbf{x}^q - \mathbf{x}_{\min}^q)^T \nabla N^q(\boldsymbol{\chi}^q)\right| \leq ||\mathbf{x}^q - \mathbf{x}_{\min}^q||\,||\nabla N^q(\boldsymbol{\chi}^q)|| \tag{2.4.10}$$

for some $\boldsymbol{\chi}^q = \psi\mathbf{x}^q + (1-\psi)\mathbf{x}_{\min}^q$, where $\psi \in (0,1)$. Then we have that $\boldsymbol{\chi}^q \in \mathfrak{F}_0$ by convexity of the delay function. Thus since $\boldsymbol{\chi}^q \in B_{\mathfrak{F}}(\mathbf{x}_{\min}^q, \xi)$, it follows from non-jaggedness and from

$||\nabla N^q(\mathbf{x}^q)|| < 1$ and $||\nabla N^q(\mathbf{x}^q_{\min})|| = 0$ that $||\nabla N^q(\boldsymbol{\chi}^q)|| < 1$. Thus, using (2.4.10) and $||\mathbf{x}^q - \mathbf{x}^q_{\min}|| < \delta$, we get

$$|N^q(\mathbf{x}^q) - N^q(\mathbf{x}^q_{\min})| < \delta$$

so that, in particular, $N^q(\mathbf{x}^q_{\min}) > N^q(\mathbf{x}^q) - \delta$.

Then we have

$$\begin{aligned} N^q(\mathbf{x}^q_{\min}) &> N^q(\mathbf{x}^q) - \delta \\ &> \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \epsilon \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) - \delta \\ &= \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \delta \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) + (\epsilon - \delta) \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) - \delta \\ &\geq \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \delta \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) + (\epsilon - \delta)\,|\mathcal{N}_+|\,\kappa - \delta \end{aligned} \tag{2.4.11}$$

Now by Lipschitz continuity, there exists some constant L such that

$$|h_i(\mathbf{x}) - h_i(\mathbf{z})| \leq L\,||\mathbf{x} - \mathbf{z}||$$

for all $i \in \mathcal{N}$ and $\mathbf{x}, \mathbf{z} \in \mathfrak{F}$. Thus $|h_i(\mathbf{y}) - h_i(\mathbf{x}^q)| \leq L\delta$, so

$$h_i(\mathbf{y}) \leq h_i(\mathbf{x}^q) + L\delta$$

for all $i \in \mathcal{N}$. Then we have:

$$\begin{aligned} N^q(\mathbf{y}) &< \sum_{i\in\mathcal{N}} \big[h_i(\mathbf{x}^q) + L\delta\big] + \delta \sum_{j\in\mathcal{N}_+} \big[h_j(\mathbf{x}^q) + L\delta\big] + \delta \sum_{j\in\mathcal{N}_0} H \\ &= \sum_{i\in\mathcal{N}} h_i(\mathbf{x}^q) + \delta \sum_{j\in\mathcal{N}_+} h_j(\mathbf{x}^q) + \delta\,|\mathcal{N}|\,L + \delta^2\,|\mathcal{N}_+|\,L + \delta\,|\mathcal{N}_0|\,H \end{aligned} \tag{2.4.12}$$

where $H = \max\{h_i(\mathbf{x}) : i \in \mathcal{N}, \mathbf{x} \in \mathfrak{F}\}$ is a positive real number, independent of $\mathbf{y}$ and q. Thus it follows from (2.4.11) and (2.4.12) that we can choose δ_1 sufficiently small to obtain $\mathbf{y}$ and q such that $N^q(\mathbf{y}) < N^q(\mathbf{x}^q_{\min})$ for $||\mathbf{y} - \mathbf{x}^q_{\min}|| \leq ||\mathbf{y} - \mathbf{x}^q|| + ||\mathbf{x}^q - \mathbf{x}^q_{\min}|| < \xi$, a contradiction of the fact that $\mathbf{x}^q_{\min}$ is a local minimizer of $N^q(\mathbf{x})$. We conclude that $\mathcal{N}_+ = \emptyset$ so that $w_i^k(\mathbf{d}(\mathbf{x}^k)) \to 0$ for all $i \in \mathcal{N}$.

Next, taking the gradient of the weighted objective $N^k(\mathbf{x})$ exactly as above in (2.4.9), we obtain

$$\nabla N^k(\mathbf{x}) = \sum_{i\in\mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{x}))\Big) \nabla h_i(\mathbf{x}) + \sum_{e\in\mathcal{E}} \tilde{\lambda}_e^k(\mathbf{x}) \nabla d_e(\mathbf{x}) \tag{2.4.13}$$

Now, since $w_i^k(\mathbf{d}(\mathbf{x}^k)) \to 0$ for all $i \in \mathcal{N}$ and since $\nabla N^k(\mathbf{x}) \to 0$, it follows from (2.4.13) that Property (2.4.5a) is satisfied in the limit.

□

2.4.3 Extension to Generalized Density Equality Constraints

As in the global convergence proof, we will show extensibility of Theorem 2.4.1 to generalized density equality constraints. We find that in handling these constraints by a standard penalty function, the framework does not change in a fundamental manner.

With the introduction of generalized density equality constraints the problem (2.4.4) becomes:

$$\begin{aligned}
&\min_{\mathbf{x},\mathbf{a}} \; G(\mathbf{x}) \\
&\textit{subject to:} \\
&a_s + d_e(\mathbf{x}) \le a_t \quad \forall \text{ edges } e \in \mathcal{E}, \text{ where } e = (s,t) \\
&0 \le a_j \quad \forall \text{ pins } j \in \mathcal{P}_{\mathcal{I}} \\
&a_\ell \le T \quad \forall \text{ pins } \ell \in \mathcal{P}_{\mathcal{O}} \\
&D_r(\mathbf{x}) = 0 \quad \forall r = 1, \ldots, R
\end{aligned} \tag{2.4.14}$$

We introduce the new Lagrange multiplier $\boldsymbol{\tau}$, and the Lagrangian becomes

$$\begin{aligned}
\breve{\mathcal{L}}(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\tau}) = &\sum_{\text{nets } i \in \mathcal{N}} h_i(\mathbf{x}) - \sum_{\text{pins } j \in \mathcal{P}_{\mathcal{I}}} u_j a_j - \sum_{\text{pins } \ell \in \mathcal{P}_{\mathcal{O}}} v_\ell (T - a_\ell) \\
&- \sum_{\text{edges } e=(s,t) \in \mathcal{E}} \lambda_e (a_t - a_s - d_e(\mathbf{x})) - \sum_{r=1}^{R} \tau_r D_r(\mathbf{x})
\end{aligned}$$

The KKT conditions (2.4.5) remain unchanged, with the exception that condition (2.4.5a) becomes

$$\sum_{e \in \mathcal{E}} \bar{\lambda}_e \nabla d_e(\bar{\mathbf{x}}) = -\sum_{i \in \mathcal{N}} \nabla h_i(\bar{\mathbf{x}}) + \sum_{r=1}^{R} \bar{\tau}_r \nabla D_r(\bar{\mathbf{x}}) \tag{2.4.15}$$

and we have the additional condition of feasibility with respect to the density constraints:

$$D_r(\mathbf{x}) = 0, \qquad r = 1, \ldots, R \tag{2.4.16}$$

We introduce a modified form of the penalty function as compared to (2.3.5):

$$\breve{N}^k(\mathbf{x}) = G(\mathbf{x}) + \Psi^k(\mathbf{x}) + \sum_{r=1}^{R} \mu^k \breve{p}_r \left(D_r(\mathbf{x})\right)$$

Here the penalty terms have the form $\breve{p}_r^k = \mu^k \breve{p}_r$, where μ^k is a sequence of positive real numbers such that $\mu^k \nearrow \infty$. The functions $\breve{p}_r$ are *nonnegative, continuously differentiable*, and have the properties

$$\breve{p}_r(D) = 0 \iff D = 0 \tag{2.4.17}$$

$$(\breve{p}_r)'(D) = 0 \implies D = 0 \tag{2.4.18}$$

Corollary 2.4.3. *Suppose that $\mathbf{x}^k$ is a local minimizer of the function $\breve{N}^k(\mathbf{x})$ for each $k = 1, 2, \ldots,$ and that $\mathbf{x}^k \in \mathfrak{F}_0$ for all k sufficiently large. Then every limit point of the sequence $\{\mathbf{x}^k\}$ at which the linear independence constraint qualification (LICQ) [40] holds is a KKT point of the problem* (2.4.14).

Proof. First we show Property (2.4.16). Corresponding to (2.4.9) we have

$$\nabla \breve{N}^k(\mathbf{x}) = \sum_{i \in \mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{x}))\Big) \nabla h_i(\mathbf{x}) + \sum_{e \in \mathcal{E}} \tilde{\lambda}_e^k(\mathbf{x}) \nabla d_e(\mathbf{x}) + \mu^k \sum_{r=1}^{R} (\breve{p}_r)' \left(D_r(\mathbf{x})\right) \nabla D_r(\mathbf{x})$$

Since each $\mathbf{x}^k$ is a local minimizer of $\breve{N}^k(\mathbf{x})$, we have that $\nabla \breve{N}^k(\mathbf{x}^k) = 0$. Thus for each $k \in K$,

$$\sum_{r=1}^{R} (\breve{p}_r)' \left(D_r(\mathbf{x}^k)\right) \nabla D_r(\mathbf{x}^k) = \frac{1}{\mu^k} \left[- \sum_{i \in \mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{x}^k))\Big) \nabla h_i(\mathbf{x}^k) - \sum_{e \in \mathcal{E}} \tilde{\lambda}_e^k(\mathbf{x}^k) \nabla d_e(\mathbf{x}^k) \right]$$

Taking the limit of both sides, we get

$$\sum_{r=1}^{R} (\breve{p}_r)' \left(D_r(\mathbf{x}^*)\right) \nabla D_r(\mathbf{x}^*) = 0$$

so that, provided the LICQ holds at $\mathbf{x}^*$, we have that $(\breve{p}_r)' \left(D_r(\mathbf{x}^*)\right) = 0$ for all $r = 1, \ldots, R$. Thus it follows from (2.4.18) that $D_r(\mathbf{x}^*) = 0$ for all $r = 1, \ldots, R$, so that Property (2.4.16) is satisfied in the limit.

To show Property (2.4.15), we introduce the Lagrange multiplier estimate $\tilde{\boldsymbol{\tau}}^k(\mathbf{x})$, in addition to those in (2.4.6):

$$\tilde{\tau}_j^k(\mathbf{x}) = -\mu^k(\breve{p}_r)'\left(D_r(\mathbf{x})\right) \qquad \forall r = 1, \ldots, R \tag{2.4.19}$$

It readily follows from the proof of Theorem 2.4.1 and the fact that $\nabla \breve{N}^k(\mathbf{x}^k) = 0$ for every k that Property (2.4.15) is satisfied. Note in particular that we may choose $\mathbf{y} \in \mathfrak{D}$ in addition to the properties outlined in the proof of Theorem 2.4.1 as a consequence of the fact that $\mathfrak{F} \cap \mathfrak{D}$ is in the closure of $\mathfrak{F}_0 \cap \mathfrak{D}$.

The remaining KKT conditions can be shown to be satisfied as a straightforward extension of the proof of Theorem 2.4.1. We conclude that $\mathbf{x}^*$ is a KKT point of (2.4.14). □

We may also extend Corollary 2.4.2 to account for the generalized density constraints.

Corollary 2.4.4. *Suppose that $\mathbf{x}^k$ approximates a local minimizer $\mathbf{x}_{min}^k$ of the weighted objective $\breve{N}^k(\mathbf{x})$ for each $k = 1, 2, \ldots$, in that $\left|\left|\mathbf{x}^k - \mathbf{x}_{min}^k\right|\right| \to 0$ and $\left|\left|\nabla N^k(\mathbf{x}^k)\right|\right| \to 0$ as $k \to \infty$. If $\mathbf{x}^k \in \mathfrak{F}_0$ and $\mathbf{x}_{min}^k \in \mathfrak{F}$ for all k sufficiently large, then every limit point of $\{\mathbf{x}^k\}$ at which the LICQ holds is a KKT point of the problem* (2.4.14).

Proof. The extension of Corollary 2.4.2 to the above is nearly identical to that in the proof of Corollary 2.4.3. We choose the new Lagrange multiplier estimate (2.4.19) as above and use it to show that Property (2.4.15) is satisfied. Similarly, we may use the same limit argument as above using the LICQ to show that Property (2.4.16) is satisfied in the limit. It is a straightforward extension of the proofs of Theorem 2.4.1 and Corollary 2.4.2 to show that the remaining KKT conditions are satisfied. □

2.4.4 Non-instantaneous Weighting Updates

We present one final result that relates to the practical implementation of this net weighting framework. Until now, we have made the assumption that the net weights are *continuously* updated according to the current placement $\mathbf{x}$. Normally in practice, we rely on a previous placement $\mathbf{y}$ to compute a set of net weights, then minimize using those fixed weights.

Symbolically, we minimize the modified objective function

$$\breve{N}^k_{\mathbf{d}(\mathbf{y})}(\mathbf{x}) = \sum_{i\in\mathcal{N}} \Big(1 + w_i^k(\mathbf{d}(\mathbf{y}))\Big) h_i(\mathbf{x}) + \sum_{r=1}^{R} \mu^k \breve{p}_r\left(D_r(\mathbf{x})\right)$$

over $\mathbf{x}$. Using the previous iterate $\mathbf{y} = \mathbf{x}^{k-1}$ to set the net weights, we get the following result.

Corollary 2.4.5. *Given any $\mathbf{x}^0$, suppose that $\mathbf{x}^k$ is a local minimizer of the weighted objective $\breve{N}^k_{\mathbf{d}(\mathbf{x}^{k-1})}(\mathbf{x})$ for each $k = 1, 2, \ldots$ and that $\left|\left|\mathbf{x}^k - \mathbf{x}^{k-1}\right|\right| \to 0$. Then $\{\mathbf{x}^k\}$ converges to a KKT point of the problem (2.4.14), provided that point is in $\mathfrak{F}_0$ and satisfies the LICQ.*

Proof. It follows from the asymptotic slack control (2.2.9), from $\left|\left|\mathbf{x}^k - \mathbf{x}^{k-1}\right|\right| \to 0$, and from the fact that x^{k-1} must eventually become strictly timing feasible that we can write $\breve{N}^k_{\mathbf{d}(\mathbf{x}^{k-1})}(\mathbf{x})$ as $\sum_i (1 + \theta_i^k) h_i(\mathbf{x}) + \sum_r \mu^k \breve{p}_r\left(D_r(\mathbf{x})\right)$, where $\theta_i^k \leq \theta^k$ for all nets i and $\theta^k \to 0$. Now using analogous variables and Lagrange multiplier estimates to those given above, we get $\tilde{a}_m(\mathbf{d}(\mathbf{x}))$ and $\tilde{\tau}_j^k(\mathbf{x})$ as defined in in (2.4.6) and (2.4.19), respectively, and $\tilde{\lambda}_e^k(\mathbf{x}) = 0$, $\tilde{u}_j^k(\mathbf{x}) = 0$, and $\tilde{v}_\ell^k(\mathbf{x}) = 0$. Using these Lagrange multiplier estimates, it is straightforward to verify that all KKT conditions are satisfied in the limit. □

The practical implication of Theorem 2.4.5 is that, given a placer that can find approximate local minimizers for the unconstrained subproblems $\min_{\mathbf{x}} N^k(\mathbf{x})$, for $k = 1, 2, \ldots$, we can use the previous placement $\mathbf{x}^{k-1}$ as initial guess for the next unconstrained local minimization. Then the placements $\mathbf{x}^k$ will converge to a very likely local minimum of the constrained problem (2.4.14). Note that we must ensure that the placements eventually become strictly timing feasible.

2.5 Implementation

Motivated by the results in Sections 2.3 and 2.4, to solve the problem (2.4.14), we can iteratively perform unconstrained minimization of $N^k_{\mathbf{d}(\mathbf{x}^{k-1})}(\mathbf{x})$ using the previous solution $\mathbf{x} \leftarrow \mathbf{x}^{k-1}$ as a starting point. In each outer iteration, we set $\mathbf{x}^k \leftarrow \mathbf{x}$ if $\mathbf{x}$ approximates a

local minimizer within some tolerance τ^k, where $\tau^k \searrow 0$. This framework is described in Algorithm 2.5.1.

Algorithm 2.5.1 Example, iterative placement with net weighting.

Choose initial placement $\mathbf{x}^0$, tolerances $\{\tau^k\}$ such that $\tau^k \searrow 0$, and convergence tolerance $\rho > 0$.

$k \leftarrow 0$

while $k \leq 1$ or $\left|\left|\mathbf{x}^k - \mathbf{x}^{k-1}\right|\right| > \rho$ **do**

  $k \leftarrow k + 1$

  $\mathbf{x} \leftarrow \mathbf{x}^{k-1}$

  while $\left|\left|\nabla N^k_{\mathbf{d}(\mathbf{x}^{k-1})}(\mathbf{x})\right|\right| > \tau^k$ **do**

    iterate $\mathbf{x}$ in unconstrained minimization of $N^k_{\mathbf{d}(\mathbf{x}^{k-1})}(\mathbf{x})$

  end while

  $\mathbf{x}^k \leftarrow \mathbf{x}$

end while

We implement each of the weighted objectives described in Section 2.2.3 in the state-of-the-art placer mPL [13]. Although significantly more sophisticated than the example given in Algorithm 2.5.1, similar principles are followed in mPL. We update the weighting parameter once every outer loop iteration, using the previous iterate to calculate net weights, then solve the inner loop subproblems using the Uzawa algorithm [1] for smoothed density equality constraints. A decreasing schedule of tolerances for cell overlap is used to determine sufficient convergence of the iterates. We use the log-sum-exp wire length approximation [41, 27] and a quadratic delay model, which satisfy all conditions for h_i and d_e required in this framework.

2.6 Experimental Results

In Table 2.6.1, we measure the efficacy of each method implemented in mPL on selected MCNC benchmarks [56]. These circuits were used because they contained the necessary timing information; the ISPD '05 and '06 contest examples could not be used due to lack

of such information. The Cadence RTL compiler was used to synthesize the circuits with the Nangate 45nm open cell library. The results represent the best result obtained for each scheme in 8 runs, measured by shortest max delay after detailed placement; statistics are given for the placement after both global and detailed placement. The net weights were updated once every outer iteration in the mPL placer, which occurred approximately 50-70 times before sufficient convergence was attained. Note that net weightings were only applied during the global placement phase, so some degradation in the delays can be observed after the global placement phase. Column "HPWL" gives the half-perimeter wire length, "Delay" gives the max path delay of the circuit, and "CPU" is a measure of the computation time necessary to complete the placement. The values are scaled against those obtained using the regular, non-weighted mPL placer. On average, the VPR-c scheme yields the best improvement in delay, the smallest increase in CPU time compared to non-weighted mPL, and matches the APlace-c weights in smallest increase in wire length compared to non-weighted mPL.

In Table 2.6.2, we compare the VPR-c weighting method against the original method in [34], which we term "VPR." The difference between the two schemes is that the VPR scheme is flat, without increasing net weighting parameter, and the max delay is set dynamically to be the current max delay at every static timing analysis (VPR). The entries in the table are arranged as those in Table 2.6.1, with the exception that computation time has been omitted, as it did not vary significantly between the two methods (static timing analysis and re-calculation of net weights are carried out at every outer iteration in both methods). As in Table 2.6.1, the best result over 8 runs for each method is shown, measured by the shortest max delay after detailed placement. After detailed placement, the VPR-c scheme nearly matched the VPR scheme in wire length while outperforming it in delay on all 10 benchmarks, 4% on average. Thus, the modifications necessary for theoretical convergence yield an improvement over the method in [34].

Table 2.6.1: Comparison of weighting schemes in mPL.

		Global Placement		Detailed Placement		
Circuit	Weight	HPWL	Delay	HPWL	Delay	CPU
ex5p	None	1 (1.61E+07)*	1 (27.87)	1 (1.65E+07)	1 (28.41)	1 (5.9)
	VPR-c	1.14	0.86	1.04	0.88	1.61
	PATH-c	1.07	0.95	1.01	0.93	2.16
	APlace-c	1.13	0.88	1.03	0.89	3.81
alu4	None	1 (1.98E+07)	1 (29.55)	1 (2.00E+07)	1 (29.63)	1 (7.3)
	VPR-c	1.02	0.82	1.02	0.85	1.31
	PATH-c	1.19	0.92	1.10	0.93	2.16
	APlace-c	1.02	0.95	1.01	0.95	1.93
apex2	None	1 (2.57E+07)	1 (29.91)	1 (2.61E+07)	1 (30.13)	1 (9.3)
	VPR-c	1.05	0.85	1.02	0.89	1.49
	PATH-c	1.17	0.98	1.12	0.96	1.77
	APlace-c	1.12	0.83	1.03	0.91	2.26
pdc	None	1 (1.13E+08)	1 (54.56)	1 (1.19E+08)	1 (55.35)	1 (27.1)
	VPR-c	1.09	0.80	1.03	0.82	1.48
	PATH-c	1.11	0.89	1.06	0.88	1.81
	APlace-c	1.08	0.78	1.03	0.82	7.22
apex4	None	1 (2.60E+07)	1 (29.80)	1 (2.64E+07)	1 (28.58)	1 (7.4)
	VPR-c	1.04	0.87	1.01	0.95	1.42
	PATH-c	1.10	0.88	1.06	0.91	1.72
	APlace-c	1.02	0.88	1.00	0.97	2.47
des	None	1 (5.55E+07)	1 (32.59)	1 (5.60E+07)	1 (32.52)	1 (17.7)
	VPR-c	1.02	0.91	1.00	0.93	1.29
	PATH-c	1.05	0.95	1.03	0.98	1.48
	APlace-c	1.01	0.92	1.00	0.93	2.64
ex1010	None	1 (2.79E+07)	1 (32.25)	1 (2.83E+07)	1 (32.27)	1 (8.6)
	VPR-c	1.07	0.82	1.03	0.87	1.34
	PATH-c	1.08	0.84	1.06	0.85	1.72
	APlace-c	1.05	0.86	1.02	0.89	2.52
misex3	None	1 (1.74E+07)	1 (27.36)	1 (1.78E+07)	1 (26.22)	1 (6.9)
	VPR-c	1.07	0.85	1.02	0.95	1.42
	PATH-c	1.05	0.92	1.03	0.96	1.63
	APlace-c	1.08	0.82	1.03	0.93	2.09
seq	None	1 (3.25E+07)	1 (30.34)	1 (3.28E+07)	1 (30.53)	1 (10.2)
	VPR-c	1.04	0.82	1.03	0.84	1.32
	PATH-c	1.11	0.84	1.08	0.87	1.55
	APlace-c	1.05	0.81	1.03	0.86	1.83
spla	None	1 (1.11E+08)	1 (53.84)	1 (1.11E+08)	1 (53.33)	1 (23.6)
	VPR-c	1.01	0.84	1.01	0.90	1.32
	PATH-c	1.16	0.84	1.13	0.85	1.70
	APlace-c	1.04	0.85	1.01	0.91	5.17
Average	None	1.00	1.00	1.00	1.00	1.00
	VPR-c	1.05	0.85	1.02	0.89	1.40
	PATH-c	1.11	0.90	1.07	0.91	1.77
	APlace-c	1.06	0.86	1.02	0.91	3.19

*units are in microns

Table 2.6.2: Comparison of VPR-like methods in mPL.

		Global Placement		Detailed Placement	
Circuit	Weight	HPWL	Delay	HPWL	Delay
ex5p	VPR-c	1.04	0.90	1.01	0.90
	VPR	1.01	0.93	1.01	0.93
alu4	VPR-c	1.02	0.82	1.02	0.85
	VPR	1.01	0.91	1.02	0.92
apex2	VPR-c	1.05	0.85	1.02	0.89
	VPR	1.01	0.95	1.02	0.95
pdc	VPR-c	1.09	0.80	1.03	0.82
	VPR	1.04	0.87	1.01	0.87
apex4	VPR-c	1.04	0.87	1.01	0.95
	VPR	1.01	0.87	1.01	0.96
des	VPR-c	1.02	0.91	1.00	0.93
	VPR	1.03	0.91	1.02	0.94
ex1010	VPR-c	1.07	0.82	1.03	0.87
	VPR	1.01	0.88	1.01	0.91
misex3	VPR-c	1.07	0.85	1.02	0.95
	VPR	1.01	0.89	1.01	0.97
seq	VPR-c	1.04	0.82	1.03	0.84
	VPR	1.01	0.89	1.01	0.89
spla	VPR-c	1.01	0.84	1.01	0.90
	VPR	1.00	0.92	1.00	0.91
Average	VPR-c	1.05	0.85	1.02	0.89
	VPR	1.01	0.90	1.01	0.93

2.7 Conclusions and Future Work

In this work, we determined a set of criteria defining a class of net weighting schemes that were shown to converge to optimal placements for the original timing-constrained problem. When a *global* minimizer to the unconstrained weighted objective can be found, the essential property for convergence of a weighting algorithm is the *asymptotic slack control*. In the more practical case when only an *approximate local* minimizer to the unconstrained weighted objective can be found, a weighting must also be *nondecreasing, non-critically indifferent,* and *path consistent*. Several schemes satisfying these properties were identified and implemented in the mPL placer on the MCNC benchmarks. The *VPR-c* scheme outperformed all others, improving delay by an average of 11% while increasing wire length by 2% and increasing

computation time by 40%, as compared to unweighted placement. Further, VPR-c outperformed the original VPR weighting scheme with improved delay in all MCNC benchmarks tested, an average of 4%; additionally, minimal increase in wire length and no change in computation time was observed.

In the future, we plan to implement Lagrangian schemes for comparison with our net weighting methodology. It can be shown that, using a scheme with projection of the multipliers as in [15], the method can be viewed as a net weighting that satisfies many of the required properties in the framework. We would like to further investigate this and the potential insights it can provide.

2.8 Notation

A summary of the notation used throughout this chapter is provided for reference in Tables 2.8.1 and 2.8.2.

Table 2.8.1: Notation, Part I

Symbol	Definition
$\mathbf{x}$	The vector of pin locations of the circuit
T	The desired upper timing bound for the circuit
$\mathcal{N}$	The set of all nets in the circuit
$\mathcal{P}$	The (finite) set of all pins in the circuit
$\mathcal{P}_{\mathcal{I}}$	The set of all input pins in the circuit
$\mathcal{P}_{\mathcal{O}}$	The set of all output pins in the circuit
$\mathcal{P}_{\mathcal{M}}$	$\mathcal{P} \setminus (\mathcal{P}_{\mathcal{I}} \cup \mathcal{P}_{\mathcal{O}})$
Π	The set of all paths in the circuit
Π_e	The set of all paths containing edge e
Π_i	The set of all paths passing through net i
$\mathcal{E}$	The set of all edges in the circuit
S	$\lvert\Pi\rvert$, i.e. the number of paths in the circuit
M	$\lvert\mathcal{E}\rvert$, i.e. the number of edges in the circuit
$h_i(\mathbf{x})$	Function measuring estimated wire length of net i (see Sec. 2.2.1)
$d_e(\mathbf{x})$	Function measuring delay of edge e (see Sec. 2.2.1)
$\mathbf{d}(\mathbf{x})$	$[d_1(\mathbf{x}) \ \ldots \ d_M(\mathbf{x})]^T$
$w_i^k(\mathbf{d}(\mathbf{x}))$	Weighting function for net i and index k (see Sec. 2.2.2)
$G(\mathbf{x})$	$\sum_{i \in \mathcal{N}} h_i(\mathbf{x})$
$\Psi^k(\mathbf{x})$	$\sum_{i \in \mathcal{N}} w_i^k(\mathbf{d}(\mathbf{x}))h_i(\mathbf{x})$
$N^k(\mathbf{x})$	$G(\mathbf{x}) + \Psi^k(\mathbf{x})$
$\sigma_\pi(\mathbf{d}(\mathbf{x}))$	The slack of path π (see (2.2.3))
$A_t(\mathbf{d}(\mathbf{x}))$	Arrival time of pin t (see (2.2.4))
$R_s(\mathbf{d}(\mathbf{x}))$	Required arrival time of pin s (see (2.2.5))
$\sigma_e(\mathbf{d}(\mathbf{x}))$	The slack of edge e (see (2.2.6))
$\sigma_i(\mathbf{d}(\mathbf{x}))$	The slack of net i, defined as $\min_{e \in i} \sigma_e(\mathbf{d}(\mathbf{x}))$

Table 2.8.2: Notation, Part II

Symbol	Definition
$\mathfrak{F}$	$\{\mathbf{x}\mid \sigma_\pi(\mathbf{d}(\mathbf{x})) \geq 0 \ \forall \text{ paths } \pi \in \Pi\}$
$\mathfrak{F}_0$	$\{\mathbf{x}\mid \sigma_\pi(\mathbf{d}(\mathbf{x})) > 0 \ \forall \text{ paths } \pi \in \Pi\}$
$D_r(\mathbf{x})$	Generalized density constraint (see Sec. 2.3.2)
R	Number of generalized density constraints
$p_r^k\left(D_r(\mathbf{x})\right)$	Penalty function for generalized density constraint (see Sec. 2.3.2)
$\hat{N}^k(\mathbf{x})$	$G(\mathbf{x}) + \Psi^k(\mathbf{x}) + \sum_{r=1}^R p_r^k\left(D_r(\mathbf{x})\right)$
$\mathfrak{D}$	$\{\mathbf{x} : D_r(\mathbf{x}) = 0,\ r = 1, \ldots, R\}$
κ	Lower bound for $h_i(\mathbf{x})$ (see Sec. 2.4.1.1)
ξ	"Jaggedness" bounding constant for $N^k(\mathbf{x})$ (see Sec. 2.4.1.2)
a_j	Variable in (2.4.4) and (2.4.14); upper bound on the arrival time of pin j
$\mathbf{a}$	$(a_1, \ldots, a_n)$
$\mathcal{L}(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{v}, \boldsymbol{\lambda})$	Lagrangian for the problem (2.4.4)
$\mathbf{u}$, $\mathbf{v}$, $\boldsymbol{\lambda}$	Lagrange multipliers corresponding to timing constraints
$\tilde{\mathbf{a}}(\mathbf{d}(\mathbf{x}))$	Variable $\mathbf{a}$ based on a placement $\mathbf{x}$ (see (2.4.6))
$\tilde{\mathbf{u}}^k(\mathbf{x})$, $\tilde{\mathbf{v}}^k(\mathbf{x})$, $\tilde{\boldsymbol{\lambda}}^k(\mathbf{x})$	Lagrange multiplier estimates based on a placement $\mathbf{x}$ (see (2.4.6))
$\breve{\mathcal{L}}(\mathbf{x}, \mathbf{a}, \mathbf{u}, \mathbf{v}, \boldsymbol{\lambda}, \boldsymbol{\tau})$	Lagrangian for the problem (2.4.14)
$\boldsymbol{\tau}$	Lagrange multipliers corresponding to density constraints
$\tilde{\tau}_j^k(\mathbf{x})$	Lagrange multiplier estimate for τ_j (See 2.4.19)
$\breve{N}^k(\mathbf{x})$	$G(\mathbf{x}) + \Psi^k(\mathbf{x}) + \sum_{r=1}^R \mu^k \breve{p}_r\left(D_r(\mathbf{x})\right)$
$\breve{p}_r^k$	Penalty function for generalized density constraint (see Sec. 2.4.3)
μ^k	Penalty coefficient in $\breve{p}_r^k$, i.e. $\breve{p}_r^k = \mu^k \breve{p}_r$ (see Sec. 2.4.3)
$\breve{p}_r$	Index-independent component of penalty function $\breve{p}_r^k$ (see Sec. 2.4.3)
$\breve{N}^k_{\mathbf{d}(\mathbf{y})}(\mathbf{x})$	$G(\mathbf{x}) + \sum_{i\in\mathcal{N}} w_i^k(\mathbf{d}(\mathbf{y}))h_i(\mathbf{x}) + \sum_{r=1}^R \mu^k \breve{p}_r\left(D_r(\mathbf{x})\right)$

CHAPTER 3

A Novel Method to Conduct Statistical Timing-aware Placement

3.1 The Jump from Deterministic to Statistical Timing

As transistors become smaller, the impact of process variations has become a considerable issue leading to failed chips [53]. One reason is that manufacturing process control is not improving at the same rate as scaling [43]. Another cause is the increasing impact of atomic-scale randomness, such as variation in the number of dopants in the transistor channel [48].

As a result, it becomes increasingly pertinent to model circuit element delays as *distributions*, usually normal, as opposed to fixed numbers. In this case, the deterministic timing constraint from the previous chapter becomes a *timing yield constraint*:

$$P(\check{T}(\mathbf{x}) < T) > \eta$$

Here, we let $\check{T}(\mathbf{x})$ represent the timing of the circuit, as a function of the placement $\mathbf{x}$ and subject to random variation. (We will further clarify this concept in subsequent sections.) The timing yield constraint corresponds to an upper bound T for the timing of a chip to be acceptable, above which the chip will be discarded. We require a yield of more than $\eta \cdot 100\%$ acceptable chips in each batch produced.

If we assume a normal distribution, this constraint often becomes

$$\mu(\mathbf{x}) + 3\sigma(\mathbf{x}) < T$$

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are the mean and standard deviation of the circuit, respectively. In this case, the implied desired yield η is that all chips up to 3 standard deviations shall be accepted: $\eta \approx 0.997$.

Thus the *statistical timing-driven placement problem* will be given as follows. As before, we wish to minimize the total wirelength of the circuit under the timing yield constraint and a standard density constraint.

$$\begin{aligned} &\min_{\mathbf{x}} \ G(\mathbf{x}) \\ &\textit{subject to:} \\ &P(\check{T}(\mathbf{x}) < T) > \eta \\ &D_r(\mathbf{x}) = 0 \quad \forall r = 1, \ldots, R \end{aligned} \tag{3.1.1}$$

For simplicity, we treat the total wirelength quantity $G(\mathbf{x})$ and the bin density functions $D_r(\mathbf{x})$ as deterministic functions of the placement $\mathbf{x}$. The precise manner in which the timing quantity $P(\check{T}(\mathbf{x}) < T)$ will be measured is the subject of discussion in the following sections. Specifically, see Sections 3.2.1 and 3.3.1.

3.2 Background

The statistical timing-driven sizing problem has been studied much more extensively than the statistical timing-driven placement problem, due to the convexity of the sizing problem. However, many approaches can be extended to statistical timing-driven placement.

One popular group of approaches falls under the category of *padded methods* [32]. These methods use a heuristic "padding" term added to each delay to account for the statistical delay, and then proceed with deterministic methods. These are often inexpensive but also inaccurate – using timing corners may be overly pessimistic, as they represent a number of worst-case scenarios that are unlikely to occur simultaneously. However, avoiding using corners on other timing elements may ignore statistical issues that later arise in fabrication [54].

Related to padded methods is *guardbanding* [7]. This involves sweeping a padding term on overall deterministic circuit timing and adjusting according to a more accurate statistical timing. We shall discuss guardbanding in more depth in Section 3.2.2.

Another group of approaches is termed *block-based methods* [54]. They use an analyti-

cal approach termed *block-based statistical timing analysis* to conduct timing analysis and propagate timing distributions through the circuit. This shall be covered in greater depth in Section 3.2.1.

While this problem has been studied extensively in the arena of sizing, previous approaches to statistical timing-driven *placement* are quite limited in the literature. The existing attempts [6, 31, 30, 46] employ simple heuristics, are conducted on FPGAs, and are all based on modifications to T-VPlace in VPR [34]. As such, this work represents the first attempt, to our knowledge, to conduct statistical placement on ASICs, with promising results.

3.2.1 Block-based statistical timing analysis

In block-based statistical timing analysis [54], each circuit element – the gate delay, arrival times, and slacks – are modeled as normal distributions. We assume Q global sources of variation as well as an independent source of variation for each element. These are modeled in *canonical form*, which is given by

$$g_0 + \sum_{j=1}^{Q} g_j \Delta X_j + g_{Q+1} \Delta R_g$$

(3.2.1) Canonical form.

Here, g_0 represents the *nominal* or *mean* value. Each ΔX_j represents the variation from a global source and is a $N(0,1)$ random variable. Similarly, ΔR_g represents variation from an independent source for the particular circuit element and is also a $N(0,1)$ random variable. Each g_j is the sensitivity of the circuit element to each corresponding source of variation. We may assume the random variables X_j and R_g are unit normal $N(0,1)$ by scaling the sensitivities g_j accordingly.

Adding and subtracting two timing quantities in canonical form is a straightforward and exact procedure; for min and max operations, however, the resulting distribution is not normal. In order to account for this, an approximation is made in canonical form which

matches the first and second moments of the distribution. Details may be found in [54].

An advantage of canonical form is that we may consider analogous timing quantities in statistical form to those previously studied in deterministic form. These include, most importantly, arrival times, backward arrival times, and slacks. The manner in which these are computed are straightforward once we have the tools of addition, subtraction, maximum and minimum of quantities in canonical form.

3.2.2 Global Guardbanding

The idea behind guardbanding [7] is straightforward. Apply a guardbanding variable δ and conduct deterministic timing-driven circuit placement with a timing constraint of $T - \delta$. Then conduct a statistical timing analysis, such as block-based statistical timing analysis, to measure the resulting placement. Continue the process, sweeping over values of δ, until the desired timing yield constraint has been achieved. See Figure 3.2.1 for an illustration of this concept.

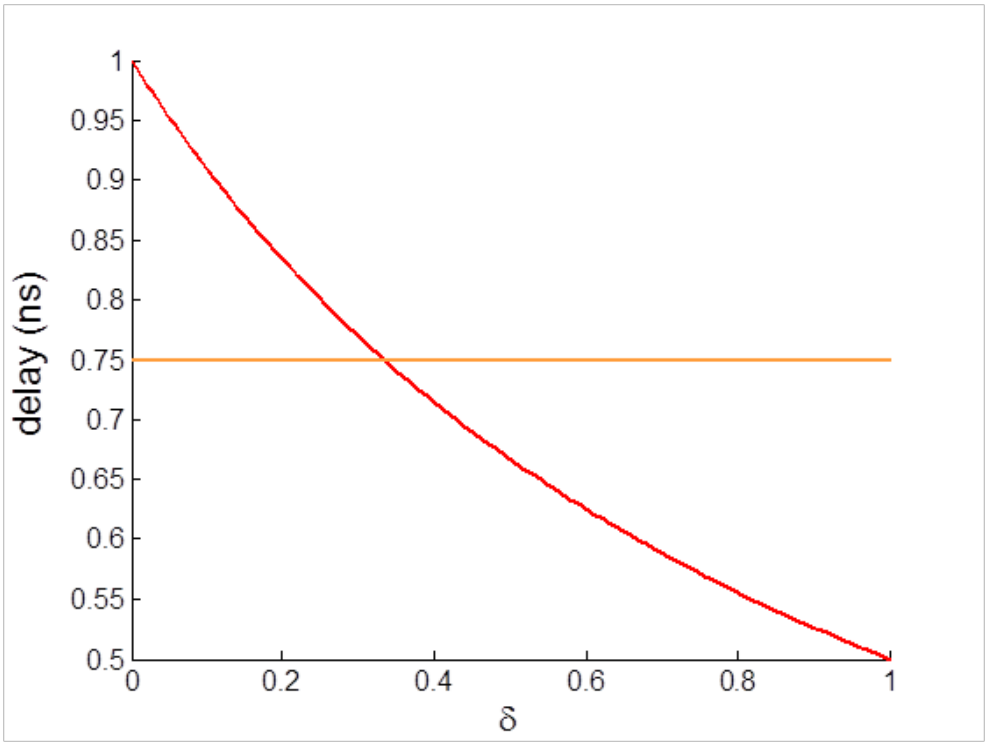

Figure 3.2.1: Outcomes of sweeping δ via global guardbanding.

Global guardbanding benefits from low computational complexity as well as ease of integration into existing deterministic tools. Further, it was found in [7] that when 3σ of process variation is 15%, guardbanding was outperformed by a direct statistical method by at most approximately 2%. Additionally, under 3σ process variations of 30% and 45%, the global guardband was outperformed by at most 4% and 6%, respectively. Thus, when there is a reasonable level of variation in the circuit, the guardbanding method offers a viable option to reach good quality placements with low cost overhead.

3.3 A novel method for statistical timing-aware placement

Our novel method for statistical timing-aware placement, which we abbreviate using the initials STAP, is herein described. We implemented this placer as a modification of the existing high-quality analytical placer mPL [13] while taking advantage of the convergent net weighting methods developed in the previous chapter.

3.3.1 Statistical Timing Model

Our statistical timing model follows the canonical form of the block-based model [54]. For each gate k, its delay is given according to the canonical form:

$$\check{d}_k = g_{0,k} + \sum_{j=1}^{Q} g_{j,k}\Delta X_j + g_{Q+1,k}\Delta R_k \tag{3.3.1}$$

Edge delays are modeled as a linear function of the wirelength. For each edge m, its delay is given as

$$\check{d}_m(\mathbf{x}) = h_{e_m}(\mathbf{x}) \left(g_{0,m} + \sum_{j=1}^{Q} g_{j,m}\Delta X_j + g_{Q+1,m}\Delta R_m \right) \tag{3.3.2}$$

For clarity, we shall denote timing quantities in block-based statistical format with the check symbol $\check{}$, such as statistical arrival time of pin t, $\check{A}_t(\mathbf{x}) \equiv A_t(\check{\mathbf{d}}(\mathbf{x}))$, statistical required arrival time of pin s, $\check{R}_s(\mathbf{x}) \equiv R_s(\check{\mathbf{d}}(\mathbf{x}))$, statistical timing of the circuit $\check{T}(\mathbf{x}) \equiv \max_t\{\check{A}_t(\mathbf{x})\}$, and statistical timing slack of edge e, $\check{\sigma}_e(\mathbf{x}) \equiv \sigma_e(\check{\mathbf{d}}(\mathbf{x}))$. These quantities are calculated in a straightforward manner analogously to their counterparts in Section 2.2.1.

Sometimes we wish to work with the 3σ confidence level of a timing distribution. For these purposes, the $\tilde{}$ symbol is used. For example, we are interested in constraining the statistical timing of the circuit up to 3σ, so if $\check{T} = t_0 + \sum_{j=1}^{Q} t_j \Delta X_j + t_{Q+1} \Delta R_k$, then

$$\tilde{T} = t_0 + 3\sqrt{\sum_{j=1}^{Q+1} {t_j}^2}$$

3.3.2 Key Methods

As previously described, we build upon the existing mPL framework for STAP. As mPL is an analytic placer, STAP builds upon it using analytic methods. Many of these methods are derived from existing methods proven useful in the literature and are described herein.

3.3.2.1 Analytic Group Move and Ripple: Motivating Existing Approaches

One key feature of STAP, termed the *analytic group move and ripple*, is derived from several previous successful techniques and adapted to our analytic, statistical framework.

In the incremental timing-driven placer ITOP [52], local improvements are made by *incremental critical path smoothing*. Critical paths are identified and then minimized by "smoothing," which seeks to straighten the critical paths and distribute the modules evenly along those paths. Locations of non-critical modules remain unchanged as the algorithm iteratively searches for the best configurations of the critical cells. For each critical cell, eight movement scores are computed that correspond to tentatively moving the cell to eight neighboring bins. A score for each move is calculated based on a reduction in weighted wire length for all nets connected to the critical cell. The highest positive reduction is accepted; otherwise, if no positive reductions exist, the cell remains in its current bin. Legalization then follows by first legalizing critical cells, then legalizing non-critical cells, treating the already-legalized critical cells as placement blockages.

Ripple Move is a technique developed for the placer Mongrel [23]. Mongrel uses a *relaxation-based local search* [24] in which, given a current global placement, a subset of

cells are selected and designated as mobile cells. The optimal placement of these cells is found, ignoring density constraints and keeping all other cells fixed. Ripple move is then used as a legalization procedure to relocate overlapping cells as a result of the relaxation-based local search. The ripple move procedure starts at overfull bins of cells, searches for nearby bins which are under their capacity constraints and "ripples" the cells toward the underfull bins. This is a discrete procedure of moving cells to adjacent bins.

In mPL [13], the *generalized force-directed algorithm* is used to enforce cell spread in an analytic manner. It can be viewed as a generalization of the force directed algorithm introduced in the Kraftwerk placer [21].

3.3.2.2 Analytic Group Move and Ripple: The Approach

In the analytic group move and ripple, we use the idea of the relaxation-based local search and ripple move procedure but apply it in an analytic fashion. We further enhance the procedure by using a *group move*. The group move idea makes sense in our framework for several reasons.

First, a single cell move procedure can easily fall into sinks and oscillation problems when adjusting critical paths. See Figure 3.3.1 below for an illustration of this pitfall. Here, we wish to move the critical path which contains nodes B and C and the fixed nodes at top and bottom. As a first iteration, a single move algorithm starts with Node B and moves it along the red arrow to a point along the line segment between Node C and the top fixed node. In a second iteration, Node C is then subsequently moved along the red arrow to a location between the new location for Node B and the bottom fixed node. This procedure continues for several iterations until, eventually, we arrive at the desired configuration with Node B and C in line vertically with the top and bottom fixed nodes.

Now consider a group move in which Node B and C may be moved in conjunction. Both Node B and C are moved along the blue dotted line in a single iteration. Only this single iteration is needed to move both nodes entirely to the optimal position.

A second major advantage of the group move is that, under a single cell move procedure,

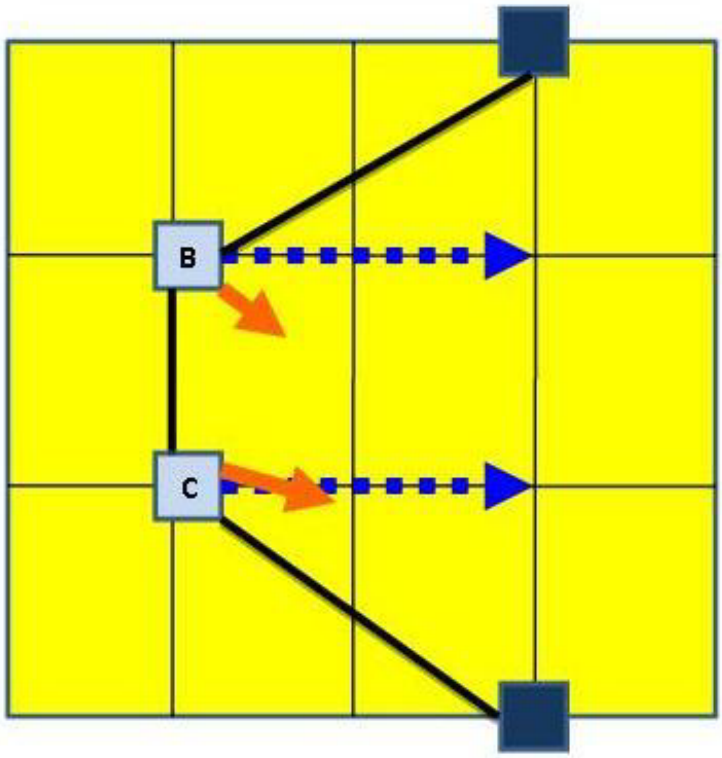

Figure 3.3.1: Illustration of an advantage of group move. The red arrows represent a single iteration under single cell moves, while the dotted blue arrows represent a single iteration under a group cell move.

useful cell move candidates may be disregarded due to the nature of the half-perimeter wire length measure. Due to the nature of the measure, single cell moves will not change the half-perimeter wire length of a net unless the cell lies on or is moved to the boundary of the net. If a single cell move does not change the half-perimeter wire length of a net, it very possibly will be disregarded as giving no improvement to the current configuration. A group move of cells in unison, however, has a greatly improved ability to decrease the half perimeter wire length and thus is likely to be accepted if it does indeed improve the configuration.

In its implementation in STAP, clustering is restricted to groups of two cells at a time. Statistically critical paths, computed by the usual statistical criticality measure using the block-based statistical model, are identified, then cells along each critical path are visited pairwise to determine whether they are good candidates to cluster. It should be clarified

that in the use of the term "clustering," we mean that cells are moved as a single unit – not that they are moved relative to each other or placed in an overlapping configuration.

The negative gradient of the net weighted objective is calculated for each cell in order to determine its "sameness" in direction of improvement with an adjacent cell in a critical path. If the negative gradient directions of two adjacent cells differs by less than 90 degrees, a procedure similar to that in [52] is used to arrive at a candidate pair of directions that improve the timing in the critical path by the best amount. For each negative gradient direction, 3 of the 8 standard 45 degree-spaced directions which lie closest to it are given as candidate directions. Then each combination of directions is tested to measure level of improvement. For 2-cell groups, this translates to $3^2 = 9$ combinations to test. Improvement is measured by improvement in timing along the critical path when each cell is moved a small distance ϵ along its candidate direction.

See Figure 3.3.2 below for an illustration of how this process works and why it is effective. The red arrows represent the directions given by the negative gradient of the net weighted objective for cells B and C. The group move algorithm then selects the three standard directions most similar to the negative gradient for each cell. These standard directions can be seen as the blue and green arrows in the figure. Then each possible combination of standard directions for the two cells is tested – 9 in all. The blue arrows represent the standard directions selected by the algorithm as those that most effectively reduce criticality in the path. It should be noted that these blue-colored directions are *not* the standard directions nearest to the negative gradient of either cell, illustrating why it is necessary to test the neighborhood of directions nearest to the negative gradient direction. Simply moving cells simultaneously along negative gradient directions will result in another iterative sink, as it will require a number of iterations before both cells are moved to the optimal position (given in Figure 3.3.1).

In order to legalize possible clustering of cells that result from the group move, The discrete Ripple Move algorithm is replaced in a rather natural manner by mPL's native generalized force-directed algorithm for cell spread. Weight is added to the critical cells to encourage diffusion in non-critical cells.

Algorithm 3.3.1 Analytic Group Move algorithm.

Begin with a given current placement $\mathbf{x}$ and an ordered set of most critical nodes $\mathcal{P}_\mathcal{C}$.

while $\mathcal{P}_\mathcal{C} \neq \emptyset$ **do**

Select first node $i \in \mathcal{P}_\mathcal{C}$.

for each neighbor $j \in \mathcal{P}_\mathcal{C}$, in order, **do**

Compare net weighted objective gradient directions for nodes i and j:

if $|\Theta\left(\nabla_{x_i} N^k(\mathbf{x}, T^k)\right) - \Theta\left(\nabla_{x_j} N^k(\mathbf{x}, T^k)\right)| < 90^\circ$ **then**

Group cells i and j and conduct group move search: locate 3 closest standard directions, $\hat{\Theta}_k$ and $\hat{\Theta}_\ell$, $k, \ell = 1, 2, 3$, to $-\Theta\left(\nabla_{x_i} N^k(\mathbf{x}, T^k)\right)$ and $-\Theta\left(\nabla_{x_j} N^k(\mathbf{x}, T^k)\right)$, respectively.

for each pair (k, ℓ) **do**

Test if move of (x_i, x_j) in direction $(\hat{\Theta}_k, \hat{\Theta}_\ell)$ improves $T_\pi(\mathbf{x})$. Save as (k^*, ℓ^*) if improvement is greater than any previous pair of directions.

end for

if successful group move indices (k^*, ℓ^*) found **then**

Move nodes (x_i, x_j) in directions $(\hat{\Theta}_{k^*}, \hat{\Theta}_{\ell^*})$, using line search in deterministic timing to determine step length.

$\mathcal{P}_\mathcal{C} \leftarrow \mathcal{P}_\mathcal{C} \setminus j$

Break.

end if

end if

end for

If no successful groupings, move node i as a group of size 1 (details omitted).

$\mathcal{P}_\mathcal{C} \leftarrow \mathcal{P}_\mathcal{C} \setminus i$

end while

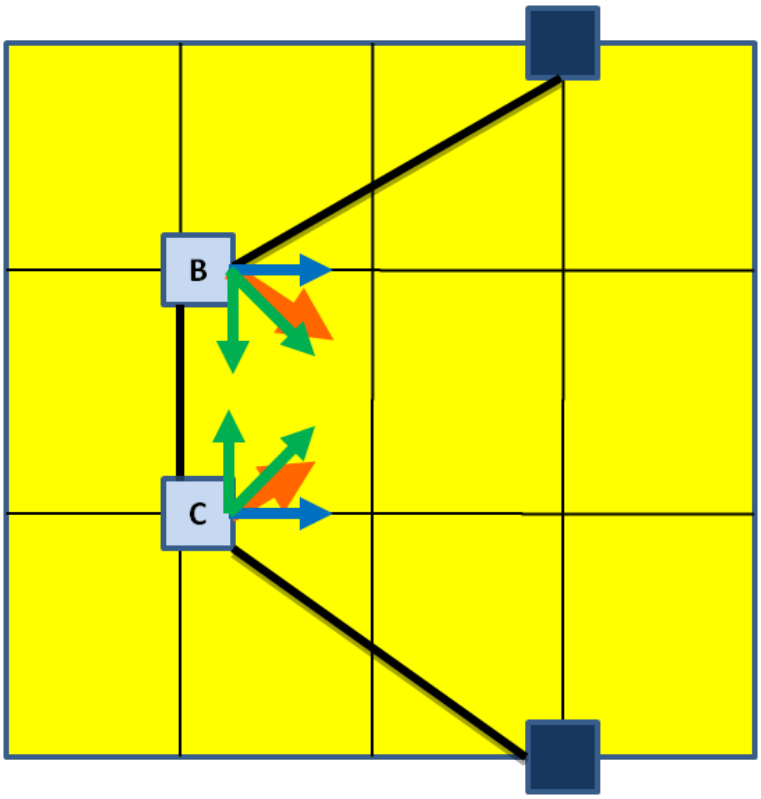

Figure 3.3.2: Illustration of the group move standard direction candidate and selection process. The red arrows represent the negative gradient directions of the net weighted objective for cells B and C; the green and blue are the three standard direction candidates for the group move algorithm. The blue arrow directions give the greatest reduction in critical path length. Note that these blue-colored directions are *not* the standard directions nearest to the negative gradient of either cell.

3.3.3 Flow

The STAP method is divided into two main phases. The first is an analytic global placement using net weights with adaptive guardband. The second is a gradient-based analytic group move and ripple. The main flow of the algorithm is given below in Algorithm 3.3.2; the remaining key components of this algorithm are then fleshed out in the remaining sections.

Phase I of STAP is somewhat similar to a standard placement with net weighted objective as described in the previous chapter. We introduce a new variable T^k, as opposed to the fixed constant T used in the previous chapter for the desired timing of the circuit, so that at

each iterate k the net weighted objective is also a function of T^k and is denoted by $N^k(\mathbf{x}, T^k)$.

Algorithm 3.3.2 STAP Algorithm flow.

Phase I: Analytic global placement using net weights with adaptive guardband

Set initial guardband level T^0 and change parameter δ^0. Set $k = 0$.

repeat

Conduct an iteration of net weighted placement with net weighted objective $N^k(\mathbf{x}, T^k)$ under deterministic timing to get a placement $\mathbf{x}^k$.

Conduct statistical timing analysis of the current placement to find $\check{T}(\mathbf{x})$.

$k \leftarrow k + 1$

Update guardband to get new step δ^k and upper bound T^k using Guardband Update Procedure (see Algorithm 3.3.3)

Recalculate net weighted objective based on new iterate k and guardband T^k.

until sufficient cell spread achieved: $|D_r(\mathbf{x})| \leq \rho$, $r = 1, \ldots, R$, and statistical timing achieved to satisfactory base level T

Phase II: Analytic Group Move and Ripple

repeat

Recalculate net weighted objective $N^k(\mathbf{x}, T^k)$ based on new iterate k, guardband T^k, and statistically critical nets. Modify weight of statistically critical nets by 3.3.3. Create a list $\mathcal{P}_\mathcal{C}$ of most critical nodes, ordered by criticality.

Conduct Analytic Group Move on $\mathcal{P}_\mathcal{C}$ by Algorithm 3.3.1

Conduct Analytic Ripple Move by Generalized Force-Directed algorithm [13].

Conduct statistical timing analysis of the current placement to find $\check{T}(\mathbf{x})$.

Update guardband to get new step δ^k and upper bound T^k using Guardband Update Procedure (see Algorithm 3.3.3).

until no improvement in $\tilde{T}(\mathbf{x})$ in m iterations

Comparing Phase I of STAP to our net weighting framework devised in the previous chapter, it can be noted that, in addition to the net weighting parameter k discussed in the previous chapter, the aforementioned parameter T^k associated with the guardband comes into play in place of the fixed constant T. These two parameters can actually be viewed as

two sides of the same coin, with similar effects when timing needs to be tightened: increasing the net weights according to the parameterized schedule and increasing net weights according to a tightening of the guard band amount to a similar effect on the net weighted objective. However, if, on the other hand, the STAP algorithm finds that a relaxation of the guard band is necessary, it has an effect of easing the net weighting on the current objective.

As such, if we simply keep the net weights at a mild increase in the objectives N^k and instead rely on the update the guardband T^k by the guardband update procedure in Algorithm 3.3.3, we have in effect a more sophisticated procedure for updating the net weighted objective in which the weights are generally increasing but are allowed to decrease temporarily from iteration to iteration. This process bears some resemblance to simulated annealing [28, 9] and even more closely to simulated tempering [33, 18]. In simulated annealing, worse configurations may be randomly selected with a gradually lowering probability, whereas in simulated tempering, the probability of randomly selecting worse configurations fluctuates according to a random variable, generally decreasing over time. In STAP, the process is governed by the timing performance of the previous configuration, as opposed to a random variable. If the previous configuration does not have acceptable statistical timing, we make an adjustment to become more likely to accept a configuration with worse wire length in trade for one with a better timing profile. If the previous configuration is more timing acceptable then necessary, we relax the timing constraints in favor of improving the chances of arriving at a placement with wirelength reduction. The design space is thus searched in a more directed manner, but allows for acceptance of worse configurations with respect to timing or wirelength in order to fully explore the space for the best configuration according to the timing needs of the circuit.

In Phase II, we include an additional heuristic on the most critical nets to accelerate improvements. This is given by

$$\hat{w}_i^k(\mathbf{x}) = w_i^k(\mathbf{d}(\mathbf{x})) + \gamma\left(\sigma_e(\mathbf{x}) - \tilde{\sigma}_e(\mathbf{x})\right) \tag{3.3.3}$$

where γ is a user-defined constant. Phase II is dominated by the Analytic Group Move algorithm. See Section 3.3.2.2 for further details.

3.3.3.1 Guardbanding the Net Weighted Objective

Both Phase I and II employ guardbanding as a quick and efficient control over the statistical timing profile. The guardbanding procedure is relatively straightforward and described in Algorithm 3.3.3. If the statistical timing oscillates between infeasibility and feasibility, the guardband step size decreases in magnitude, allowing for finer adjustments. If we have multiple iterates of statistical timing satisfied, greater iterative relaxations of the guardband are permitted.

Algorithm 3.3.3 Guardband update procedure in STAP.

Given: a current placement $\mathbf{x}^k$, current guardband level T^k and a previous guardband update δ^k.

if Statistical timing is satisfied: $\tilde{T}(\mathbf{x}^k) < T$ **then**

  if Statistical timing also satisfied in previous iterate: $\tilde{T}(\mathbf{x}^{k-1}) < T$ **then**

    $\delta^{k+1} = 1.5|\delta^k|$

  else

    $\delta^{k+1} = 0.7|\delta^k|$

  end if

else

  if Statistical timing satisfied in previous iterate **then**

    $\delta^{k+1} = -0.7|\delta^k|$

  else

    $\delta^{k+1} = -|\delta^k|$

  end if

end if

$T^{k+1} = T^k + \delta^{k+1}$

3.4 Results

for the implementation, the VPR-c weighting scheme, based upon VPR [34] and developed in the previous chapter, was used, as it was shown to be the most effective of the convergent net weighting schemes in the previous study. For our testing, we used 3 global sources of variation and one independent source, which amounted to a 3σ of approximately 30% of the nominal delay. The procedure was implemented on 10 MCNC [56] benchmarks. The results in each category were chosen according to the placement that yielded the best $\mu + 3\sigma$ after detailed placement in 5 trials. The detailed results are shown in Table 3.4.1 and summarized in Figure 3.4.1 below. Each placement's statistics were normalized by the unweighted mPL placement for each circuit. Unweighted mPL, the VPR-c scheme, and STAP schemes were compared. STAP showed an average of 5% improved statistical delay over VPR-c after detailed placement, with comparable nominal delay and a 1% additional wirelength overhead. As such, STAP showed promising results in improving the statistical timing of the MCNC benchmarks.

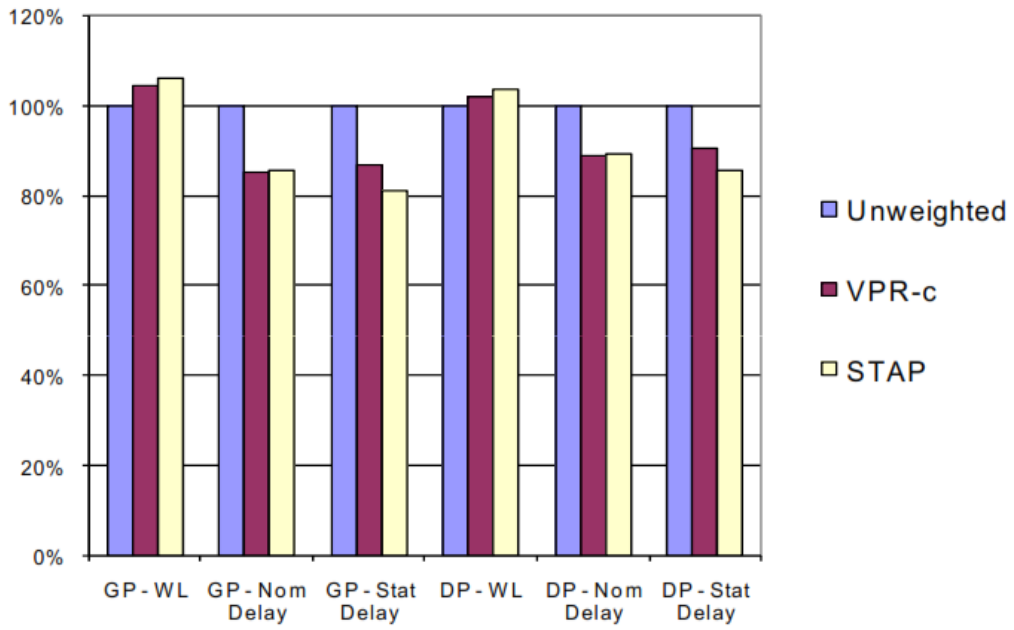

Figure 3.4.1: A summarized comparison among unweighted mPL, VPR-c, and STAP methods on 10 MCNC benchmarks.

Table 3.4.1: Comparison of statistical placement schemes.

		Global Placement			Detailed Placement			
Circuit	Method	HPWL	Nom. Delay	Stat. Delay	HPWL	Nom. Delay	Stat. Delay	CPU
ex5p	mPL-orig	1 (1.60E+07)*	1 (28.22)	1 (42.60)	1 (1.65E+07)	1 (29.12)	1 (44.67)	1 (6.2)
	VPR-c	1.04	0.90	0.91	1.02	0.90	0.93	1.58
	STAP	1.05	0.87	0.82	1.02	0.90	0.84	7.01
alu4	mPL-orig	1 (2.00E+07)	1 (29.42)	1 (44.12)	1 (2.01E+07)	1 (30.21)	1 (45.23)	1 (7.5)
	VPR-c	1.03	0.82	0.84	1.02	0.86	0.88	1.42
	STAP	1.06	0.84	0.78	1.04	0.86	0.83	9.59
apex2	mPL-orig	1 (2.56E+07)	1 (30.11)	1 (50.12)	1 (2.59E+07)	1 (30.55)	1 (51.85)	1 (9)
	VPR-c	1.05	0.87	0.89	1.02	0.89	0.9	1.49
	STAP	1.04	0.88	0.83	1.03	0.94	0.86	8.15
pdc	mPL-orig	1 (1.13E+08)	1 (55.25)	1 (83.02)	1 (1.20E+08)	1 (55.69)	1 (84.69)	1 (26.9)
	VPR-c	1.08	0.82	0.83	1.03	0.82	0.84	1.43
	STAP	1.1	0.85	0.79	1.07	0.83	0.82	7.54
apex4	mPL-orig	1 (2.61E+07)	1 (29.06)	1 (40.05)	1 (2.65E+07)	1 (29.13)	1 (41.16)	1 (7.6)
	VPR-c	1.04	0.87	0.9	1.01	0.94	0.95	1.45
	STAP	1.09	0.89	0.78	1.03	0.93	0.86	7.6
des	mPL-orig	1 (5.53E+07)	1 (32.52)	1 (52.23)	1 (5.60E+07)	1 (32.78)	1 (53.36)	1 (17.6)
	VPR-c	1.02	0.91	0.92	1	0.93	0.94	1.31
	STAP	1.03	0.89	0.84	1.04	0.88	0.89	6.84
ex1010	mPL-orig	1 (2.80E+07)	1 (32.59)	1 (53.69)	1 (2.84E+07)	1 (32.79)	1 (54.93)	1 (8.6)
	VPR-c	1.07	0.83	0.85	1.03	0.86	0.89	1.3
	STAP	1.07	0.84	0.85	1.03	0.93	0.9	7.69
misex3	mPL-orig	1 (1.74E+07)	1 (27.45)	1 (38.59)	1 (1.78E+07)	1 (27.31)	1 (38.89)	1 (7.1)
	VPR-c	1.06	0.85	0.87	1.02	0.94	0.97	1.53
	STAP	1.07	0.85	0.82	1.04	0.91	0.88	8.23
seq	mPL-orig	1 (3.22E+07)	1 (30.12)	1 (54.61)	1 (3.27E+07)	1 (30.43)	1 (55.72)	1 (10.4)
	VPR-c	1.04	0.81	0.83	1.03	0.82	0.85	1.39
	STAP	1.06	0.82	0.78	1.05	0.84	0.8	6.52
spla	mPL-orig	1 (1.11E+08)	1 (52.67)	1 (81.52)	1 (1.12E+08)	1 (52.15)	1 (82.07)	1 (23.5)
	VPR-c	1.02	0.84	0.86	1.01	0.91	0.92	1.25
	STAP	1.03	0.85	0.82	1.02	0.92	0.87	6.23
Average	mPL-orig	1 (4.45E+07)	1 (34.741)	1 (54.055)	1 (4.56E+07)	1 (35.016)	1 (55.257)	1 (12.44)
	VPR-c	1.045	0.852	0.87	1.019	0.887	0.907	1.415
	STAP	1.06	0.858	0.811	1.037	0.894	0.855	7.54

*units are in microns

3.5 Notation

A summary of the notation used throughout this chapter is provided for reference in Table 3.5.1 below.

Table 3.5.1: Notation

Symbol	Definition
$\mathbf{x}$	The vector of pin locations of the circuit
$D_r(\mathbf{x})$	Generalized density constraint (see Sec. 2.3.2)
R	Number of generalized density constraints
$\check{d}_e(\mathbf{x})$	Function measuring statistical delay of edge e (see Sec. 3.3.1)
$p_r^k\left(D_r(\mathbf{x})\right)$	Penalty function for generalized density constraint
$\mathfrak{D}$	$\{\mathbf{x} : D_r(\mathbf{x}) = 0,\ r = 1, \ldots, R\}$
$\check{N}^k(\mathbf{x})$	$G(\mathbf{x}) + \Psi^k(\mathbf{x}) + \sum_{r=1}^{R} \mu^k \check{p}_r\left(D_r(\mathbf{x})\right)$
$\check{A}_t(\mathbf{x})$	Statistical arrival time of pin t (see Sec. 3.3.1)
$\check{R}_s(\mathbf{x})$	Statistical required arrival time of pin s (see Sec. 3.3.1)
$\check{\sigma}_e(\mathbf{x})$	Statistical slack of edge e (see Sec. 3.3.1)
$\check{T}(\mathbf{x})$	Statistical timing of the circuit (see Sec. 3.3.1)
$\hat{\Theta}$	A standard direction (north, east, south, west, and four diagonals).
$N^k(\mathbf{x}, T^k)$	Guardbanded net weighted objective.
γ	Critical net weight multiplier. (see 3.3.3)
$\mathcal{P}_{\mathcal{C}}$	Set of statistically most critical nodes.

CHAPTER 4

Gate Sizing via Mean Excess Delay using Fast Iterative Methods

4.1 Deterministic Gate Sizing

We now visit the problem where the circuit has a predetermined fixed placement and each circuit gate i must be given a single size variable, s_i. Traditionally this problem has been studied under a deterministic timing model, so that each gate delay is a direct function of the vector $\mathbf{s}$ of sizes in the circuit, $d_i(\mathbf{s})$. The gate sizes also affect other important aspects of chip performance, including power and area consumption. As in previous chapters, we may examine delays along the circuit paths, i.e. $d_\pi(\mathbf{s}) = \sum_{i \in \pi} d_i(\mathbf{s})$, and we consider the vector of gate delays $\mathbf{d}(\mathbf{s}) = [d_1(\mathbf{s}) \ \ldots \ d_J(\mathbf{s})]^T$, where J is the total number of gates in the circuit. For this problem, we may treat the max circuit delay as a direct function of the vector of gate delays in the circuit: $T(\mathbf{s}) = \max_{\pi \in \Pi} d_\pi(\mathbf{s})$. In this problem, we wish to minimize the max circuit delay subject to some power and size constraints, which will be denoted as $P(\mathbf{s})$ and $A(\mathbf{s})$, respectively. We thus have the *deterministic gate sizing problem*:

$$\begin{aligned} &\min_{\mathbf{s}} \ T(\mathbf{s}) \\ &\textit{subject to:} \\ &P(\mathbf{s}) \leq P_0 \\ &A(\mathbf{s}) \leq A_0 \\ &\mathbf{s} \geq 0 \end{aligned} \tag{4.1.1}$$

Note that $P(\mathbf{s})$ and $A(\mathbf{s})$ are typically modeled as convex functions so that the overall

problem is convex. This problem has been studied extensively and is efficiently solved via geometric programming [5].

4.2 Statistical Considerations

As transistors become smaller, the impact of process variations has become a considerable issue leading to failed chips [53]. This has led to the study of the *statistical* circuit sizing problem, in which each gate delay is subject to a zero-mean random variable v_i. The effect of the random variable on the delay is often modeled, as in [20], as:

$$d_i(\mathbf{s}, \mathbf{v}) = (1 + v_i)d_i(\mathbf{s}) \tag{4.2.1}$$

Correspondingly, we denote the max delay subject to a set of gate sizes $\mathbf{s}$ and a set of random variables $\mathbf{v}$ as $T(\mathbf{s}, \mathbf{v}) = \max_{\pi \in \Pi} d_\pi(\mathbf{s}, \mathbf{v})$. This problem has also been studied using a number of techniques; see [4] for details.

4.3 Motivation and Contribution

[19] introduced a method using the *mean excess delay*, described below, which originated in the finance industry [45], and applied it to the circuit sizing problem under process variations. The key insight of the η-mean excess delay is that it provides an upper bound on the η-quantile and thus indirectly minimizes the η-quantile, while having the benefits that it is straightforward to calculate and preserves the convexity properties of the underlying display distribution. This implementation in [19] showed promise, but was slow in practice.

Our motivation is to attempt to improve upon this work by reformulating the objective and using fast proximal methods, detailed below, as opposed to the analytic cutting plane method used in [19]. Fast proximal methods have shown great promise recently as a tool in large-scale nonsmooth convex optimization, with a $1/k^2$ convergence rate [2]. Example applications include compressed sensing, variable selection in regression, TV-regularized image denoising, and sensor network localization [50]. See below for further discussion. This implementation of fast proximal methods to the circuit sizing problem under process variation,

herein named *ProSize*, is the main contribution of this work.

4.4 Mean Excess Delay

Of particular interest, [19] used a technique in which a quantity termed the *mean excess delay* is minimized. The idea originated in the finance industry where it is termed the *conditional value-at-risk* (CVaR) [45].

The use of the CVaR is motivated as follows: ideally, we would like to minimize the η-quantile $q_\eta(\mathbf{s})$ of the problem, which is the minimum time for which a predefined percentage $\eta \in (0, 1)$ of the circuits produced will satisfy that timing constraint. Formally, it is defined as

$$q_\eta(\mathbf{s}) = \inf\{t : P[T(\mathbf{s}, \mathbf{v}) \leq t] \geq \eta\} \tag{4.4.1}$$

The problem with minimizing the η-quantile directly is that the problem formulation is generally nonconvex, and there is no closed form function for $T(\mathbf{s}, \mathbf{v})$.

The mean excess delay $m_\eta(\mathbf{s})$ is defined as the expected value of the tail of the delay distribution *past* the η-quantile $q_\eta(\mathbf{s})$:

$$m_\eta(\mathbf{s}) = E[T(\mathbf{s}, \mathbf{v}) : T(\mathbf{s}, \mathbf{v}) \geq q_\eta(\mathbf{s})] \tag{4.4.2}$$

The advantage of the mean excess delay is that it preserves convexity of the problem, and it serves as an upper bound on the η-quantile:

$$q_{\eta(\mathbf{s})} \leq m_\eta(\mathbf{s})$$

Thus, minimizing the mean excess delay may serve as an indirect minimization of the η-quantile.

This leads to the revised problem

$$\begin{aligned} &\min_{\mathbf{s}} \ m_\eta(\mathbf{s}) \\ &\textit{subject to:} \\ &P(\mathbf{s}) \leq P_0 \\ &A(\mathbf{s}) \leq A_0 \\ &\mathbf{s} \geq 0 \end{aligned} \tag{4.4.3}$$

Following [45] and [19], the MED may be minimized more accurately using the function

$$g_\eta(\mathbf{s}, t) = t + \frac{1}{1-\eta}\mathbf{E}_v[T(\mathbf{s}, \mathbf{v}) - t]^+ \tag{4.4.4}$$

This function is convex in $\mathbf{s}$ and t if $T(\mathbf{s}, \mathbf{v})$ is convex in $\mathbf{s}$ for fixed $\mathbf{v}$. Furthermore, if $\mathbf{v}$ is a continuous random vector, then the minimum value of $g_\eta(\mathbf{s}, t)$ over t is equal to the MED:

$$m_\eta(\mathbf{s}) = \inf_{t \in R} g_\eta(\mathbf{s}, t)$$

4.5 Discretization of the Random Variables

In order to handle the problem in a discrete manner, we may use scenario generation to generate a number of possible outcomes of the random variables $\mathbf{v^1}, \ldots, \mathbf{v^N}$.

Following from the formulation in [25] we may convert the problem to the scenario-based MED saddle point problem:

$$\begin{aligned} &\min \max_{q \in Q_N} \sum_{i=1}^{N} q_i T(\mathbf{s}, \mathbf{v^i}) \\ &\textit{subject to:} \\ &P(\mathbf{s}) \leq P_0 \\ &A(\mathbf{s}) \leq A_0 \\ &0 \leq \mathbf{s} \leq \mathbf{u} \end{aligned} \tag{4.5.1}$$

where the set of test probabilities Q_N are given by

$$Q_N = \{\mathbf{q} \in R^N : \mathbf{1}^T q = 1, 0 \leq \mathbf{q} \leq \frac{p}{1-\eta}\} \tag{4.5.2}$$

We may now use this formulation in the application of the methods in the following sections.

4.6 Proximal Gradient Method

We define the *proximal operator* of a convex function h as

$$\text{prox}_h(\mathbf{s}) = \text{argmin}_{\mathbf{u}} \left[h(\mathbf{u}) + \frac{1}{2} \|\mathbf{u} - \mathbf{s}\|_2^2 \right] \tag{4.6.1}$$

It can be shown that $\text{prox}_h(\mathbf{s})$ exists and is unique for all $\mathbf{s}$. The proximal operator can be thought of as a generalization of the standard Euclidean projection. If we define $I_C(\mathbf{u})$ to be the indicator function for some convex set C:

$$I_C(\mathbf{u}) = \begin{cases} 0 & \text{if } \mathbf{u} \in C \\ \infty & \text{otherwise} \end{cases} \tag{4.6.2}$$

then when $h(\mathbf{u}) = I_C(\mathbf{u})$, we have

$$\text{prox}_{I_C}(\mathbf{s}) = \text{argmin}_{\mathbf{u} \in C} \|\mathbf{u} - s\|_2^2 = \text{proj}_C(\mathbf{s}) \tag{4.6.3}$$

The *proximal gradient method* [51] is useful for convex problems with a composite objective:

$$f(\mathbf{s}) = g(\mathbf{s}) + h(\mathbf{s}) \tag{4.6.4}$$

where $g(\mathbf{s})$ is convex and smooth, and $h(\mathbf{s})$ is convex, possibly nonsmooth, and has an inexpensive proximal operator. We define the proximal gradient algorithm as:

$$\mathbf{s}^{k+1} = \text{prox}_{t_k h}(\mathbf{s}^k - t_k \nabla g(\mathbf{s}^k)) \tag{4.6.5}$$

where $t_k > 0$ is the step size, which can be kept constant or determined by line search. This can be interpreted as a minimization of $h(\mathbf{u})$ plus a quadratic approximation of $g(\mathbf{u})$

around $\mathbf{s}^k$. Also, analogously to the proximal operator, this method can be viewed as a generalization of the gradient projection method, to which it reduces when $h(\mathbf{u}) = I_C(\mathbf{u})$. This amounts to a minimization of $g(x)$ over C.

Assuming ∇g is Lipschitz continuous and an optimal solution $\mathbf{s}^*$ exists, it can be shown that $f(\mathbf{s}^k) - \mathbf{s}^*$ decreases at least as fast as $1/k$ [51].

4.7 Fast Proximal Gradient Methods

We now discuss a series of *fast proximal gradient methods* which converge as fast as $1/k^2$, as opposed to the $1/k$ convergence of the proximal gradient method in the previous section. Among the notable examples are Nesterov's first [37], second [39], and third [38] methods; the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [2], which is a proximal gradient version of Nesterov's first method; and TFOCS [3]. These methods enjoy rigorous analytical justification and have been successfully incorporated in a number of applications, including portfolio management, compressed sensing, variable selection in regression, TV-regularized image denoising, and sensor network localization [25, 50]. They are of interest to the current discussion due to their algorithmic rigor and promise for handling large nonsmooth convex optimization problems.

The FISTA algorithm is given as follows. Choose $\mathbf{s}^0 = \mathbf{v}^0 \in \text{dom}(h)$, and for $k \geq 0$,

$$\begin{aligned} \mathbf{s}^{k+1} &= \text{prox}_{t_k h}(\mathbf{v}^k - t_k \nabla g(\mathbf{v}^k)) \\ \mathbf{v}^{k+1} &= \mathbf{s}^{k+1} + \frac{k-1}{k+2}(\mathbf{s}^{k+1} - \mathbf{s}^k) \end{aligned} \tag{4.7.1}$$

The first iteration of this algorithm is a standard proximal gradient step at $\mathbf{s}^0$. All following iterations are proximal gradient steps at the extrapolated points $\mathbf{v}^k$. The sequence $\mathbf{s}^k$ remains feasible in $\text{dom}(h)$, but not necessarily $\mathbf{v}^k$. [2, 51]

It can be shown that if f has a finite optimal value at $\mathbf{s}^*$, $\text{dom}(g) = \mathbf{R}^n$, ∇g is Lipschitz continuous, and h is closed and convex, then $f(\mathbf{s}^k) - \mathbf{s}^*$ decreases at least as fast as $1/k^2$. This holds if a fixed step size $t^k = 1/L$ is used, or if a backtracking line search is used to determine each step size.

4.8 Application and Results

Focusing on $N = 1000$ scenarios for the problem 4.5.1 and using a standard deviation of 0.25 for each random variable, we apply the novel method using FISTA [2] with

$$f(\mathbf{s}) = \text{smoothed } T(\mathbf{s}, \mathbf{v^i}, \text{ using the log-sum-exp approximation} g(\mathbf{s}) = I_{\mathfrak{F}*}(\mathbf{s}) \qquad (4.8.1)$$

where $\mathfrak{F}* = \{P(\mathbf{s}) \leq P_0, A(\mathbf{s}) \leq A_0, 0 \leq \mathbf{s} \leq \mathbf{u}\}$. Using $\eta = 0.95$ this was compared in MATLAB to the cutting plane method in [19] on a set of 8 randomly generated circuits. We name the novel method *ProSize*. The results are summarized in Table 4.8.1 below:

Table 4.8.1: Comparison of the Novel method to cutting plane method on 8 circuits.

		ProSize		Cutting Plane [19]		
Circuit	Gates	$q_0.95$	d_{nom}	$q_0.95$	d_{nom}	CPU ratio
01	250	0.85	0.52	0.81	0.52	1.16
02	500	0.73	0.61	0.74	0.62	1.22
03	750	0.98	0.72	0.99	0.73	1.10
04	1000	1.1	0.82	1.1	0.84	1.08
05	1250	1.3	1.0	1.3	1.0	1.35
06	1500	1.2	0.95	1.2	0.93	1.44
07	2000	1.4	1.2	1.4	1.2	1.17
08	3000	1.7	1.5	1.7	1.5	1.29
Average		1.16	0.92	1.16	0.92	1.23

In the above table, d_nom represents the nominal delay of the resulting sizing from each method in nanoseconds, while $q_{0.95}$ represents the 95 percent quantile. CPU ratio represents the ratio of CPU time to solve for each circuit for the novel method compared to the cutting plane method. As we can see, the novel method took on average 23 percent longer to arrive at comparable solutions.

4.9 Conclusion

The main goal of this implementation was an attempt to extend fast proximal methods to solve the MED circuit sizing problem, with the intent of testing its computational feasibility

compared to the cutting plane method used in [19]. As can be seen in the results in Table 4.8.1 above, the ProSize scheme produced circuits of comparable performance but whose computational time was 23 percent greater on average. We may conclude that the method applied successfully in [25] to the portfolio management problem appears to not extend feasibly to the statistical sizing problem, due to the computational complexity involved in the objective and indicator functions, compared to the linear objective studied in the original problem. However, the results remain a relevant reference for potential future improvements to the method involving a reformulation of the objective.

CHAPTER 5

Conclusion

In this work, we determined a set of criteria defining a class of net weighting schemes that were shown to converge to optimal placements for the original timing-constrained problem. When a *global* minimizer to the unconstrained weighted objective can be found, the essential property for convergence of a weighting algorithm is the *asymptotic slack control*. In the more practical case when only an *approximate local* minimizer to the unconstrained weighted objective can be found, a weighting must also be *nondecreasing, non-critically indifferent,* and *path consistent*. Several schemes satisfying these properties were identified and implemented in the mPL placer on the MCNC benchmarks. The *VPR-c* scheme outperformed all others, improving delay by an average of 11% while increasing wire length by 2% and increasing computation time by 40%, as compared to unweighted placement. Further, VPR-c outperformed the original VPR weighting scheme with improved delay in all MCNC benchmarks tested, an average of 4%; additionally, minimal increase in wire length and no change in computation time was observed.

A new method for conducting statistically-timing driven placement was introduced, called STAP, which incorporated techniques including block-based statistical timing analysis [54], analytic group move and ripple, and the convergent net weighting schemes discovered in this work. This was the first placer to conduct statistically timing-driven placement on ASICs. STAP showed an average of 5% improved statistical delay over VPR-c after detailed placement, with comparable nominal delay and a 1% additional wirelength overhead. As such, STAP showed promising results in improving the statistical timing of ASIC benchmarks.

Finally, a new method was developed to extend fast proximal methods to solve the MED circuit sizing problem, with the intent of testing its computational feasibility compared to

the cutting plane method used in [19]. The scheme, called ProSize, produced circuits of comparable performance but whose computational time was 23 percent greater on average. We may conclude that the method applied successfully in [25] to the portfolio management problem appears to not extend feasibly to the statistical sizing problem, due to the computational complexity involved in the objective and indicator functions, compared to the linear objective studied in the original problem. However, the results remain a relevant reference for potential future improvements to the method.

References

[1] K. Arrow, L. Huriwcz and H. Uzawa, *Studies in Nonlinear Programming*, Stanford University Press, Stanford, CA, 1958.

[2] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," SIAM J. Imaging Sciences, vol. 2, nmber 1, pp. 183-202, 2009.

[3] S. R. Becker, E. J. Candes, and M. Grant, "Templates for Convex Cone Problems with Applications to Sparse Signal Recovery," Mathematical Programming Computation, vol. 3, number 3, pp. 165-218, 2011.

[4] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. "Statistical Timing Analysis: From Basic Principles to State of the Art," Trans. Computer-Aided Design 2008, pp. 589-607.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[6] Bsoul, Manjikian and Shang, "Reliability- and process variation-aware placement for FPGAs," Proc. DATE 2010.

[7] Burns et al., "Comparative analysis of conventional and statistical design techniques," Proc. Design Automation Conference 2007.

[8] M. Burstein and M. Youssef, "Timing Influenced Layout Design," Proc. Design Automation Conference 1985.

[9] V. Cern, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," Journal of Optimization Theory and Applications 45, pp. 4151, 1985.

[10] T. F. Chan, J. Cong, and E. M. Radke, "Convergent Net Weighting Schemes in Hypergraph-based Optimization," UCLA Computational and Applied Math Reports, 2011.

[11] T. F. Chan, J. Cong, and E. M. Radke, "A Rigorous Framework for Convergent Net Weighting Schemes in Timing-Driven Placement," *Proc. 2009 Int'l Conf. on Computer Aided Design*, November 2009.

[12] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: enhanced multilevel mixed-size placement," *Proc. 2006 Int'l Symp. on Phys. Design*, pp. 212-214, 2006.

[13] T. F. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," *Proc. 2005 Int'l Symp. on Phys. Design*, pp. 185-192, April 2005.

[14] C. C. Chang, J. Cong, and M. Xie. "Optimality and Scalability Study of Existing Placement Algorithms," Proc. Asia South Pacific Design Automation Conf. 2003, pp. 621-627.

[15] C.-P. Chen, C. C. N. Chu, and D. F. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 7, pp. 1014-1025, July 1999.

[16] J. Cong , T. Kong , J. Shinnerl , M. Xie, and X. Yuan, "Large-scale circuit placement," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 10, no. 2, pp. 389-430, Apr. 2005.

[17] J. Cong, T. Kong, J. Shinnerl, M. Xie, and X. Yuan. "Large-Scale Circuit Placement: Gap and Promise," Proc. Int'l Conf. on Computer Aided Design 2003, pp. 883-890.

[18] J. Cong, T. Kong, D. Xu, F. Liang, J.S. Liu, and W.H. Wong. "Relaxed Simulated Tempering for VLSI Floorplan Designs," Proc. Asia-South Pacific Design Automation Conference 1999.

[19] J. Cong, J. Lee, and L. Vandenberghe, "Robust gate sizing via mean excess delay minimization," *Proc. 2006 Int'l Symp. on Phys. Design*, pp. 10-14, 2008.

[20] A. Davoodi and A. Srivastava, "Variability driven gate sizing for binning yield optimization," *Proc. Design Automation Conf.*, pp. 959-964, 2006.

[21] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," *Proc. 35th ACM/IEEE Design Automation Conference*, pp. 269-274, 1998.

[22] T. Hamada et al., "Prime: A Timing-Driven Placement Tool using a Piecewise Linear Resistive Network Approach," Proc. Design Automation Conference 1993.

[23] S.-W. Hur and J. Lillis, "Mongrel - hybrid techniques for standard cell placement," Proc. Intl. Conf. Computer Aided Design 2000.

[24] S.-W. Hur and J. Lillis, "Relaxation and clustering in a local search framework: application to linear placement," Proc. Design Automation Conference 1999.

[25] G. Iyengar and A. Ma, "Fast gradient descent method for mean-CVaR optimization," Annals of Operations Research, vol. 205, no. 1, pp. 203-212, May 2013.

[26] M. Jackson and E. Kuh, "Performance-driven Placement of Cell Based IC's," Proc. Design Automation Conference 1989.

[27] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 5, pp. 1-14, May 2005.

[28] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," Science 220 (4598), pp. 671680, 1983.

[29] T. Kong, "A novel net weighting algorithm for timing-driven placement," *Proc. 2002 IEEE/ACM Int'l Conf. on Computer-aided Design*, pp. 172-176.

[30] Lin, Hutton and He, "Statistical placement for FPGAs considering process variation," IET Computers and Digital Technologies 2007.

[31] Lucas, Dong and Chen, "Variation-aware placement with multi-cycle statistical timing analysis for FPGAs," Trans. Computer Aided Design 2010.

[32] M. Mani and M. Orshansky, "A new statistical optimization algorithm for gate sizing," Int. Conf. Computer Aided Design 2004.

[33] E. Marinari and G. Parisi. "Simulated Tempering: A New Monte Carlo Scheme." *Europhys. Lett.* vol. 19, no. 6, pp. 451-455, 1992.

[34] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," *Proc. 2000 ACM/SIGDA Eighth Int'l Symp. on Field Programmable Gate Arrays*, pp. 203-213, February 2000.

[35] G.-J. Nam and J. Cong (eds.), *Modern Circuit Placement: Best Practices and Results*, Springer, 2007.

[36] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*, McGraw-Hill, 1996.

[37] Y. Nesterov, "A Method of Solving a Convex Programming Problem with Convergence Rate $O(1/k^2)$," Soviet Mathematics, vol. 27, no. 2, pp. 372-376, 1983.

[38] Y. Nesterov, "Smooth Minimization of Non-Smooth Functions," Journ. Mathematical Programming, vol. 103, no. 1, pp. 127-152, 2005.

[39] Y. Nesterov and A. Nemirovsky, "A General Approach to Polynomial-Time Algorithms Design for Convex Programming," Report, Central Economical and Mathematical Institute, USSR Academy of Sciences, 1988.

[40] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, New York, NY, 1999, pp. 490-525.

[41] W. Naylor, R. Donelly, and L. Sha, "Non-linear optimization system and method for wire length and delay optimization for an automatic electronic circuit placer," US Patent 6671859, 2003.

[42] Obermeier and Johannes, "Quadratic Placement Using an Improved Timing Model," Proc. Design Automation Conference 2004.

[43] M. Orshansky and A. Bandyopadhyay, "Fast Statistical Timing Analysis Handlingn Arbitrary Delay Correlations," Proc. Design Automation Conference 2004.

[44] K. Rajagopal et al., "Timing Driven Force Directed Placement with Physical Net Constraints," Proc. Int'l Symp. on Physical Design 2003, pp. 60-66.

[45] R. T. Rockafellar and S. Uryasev, "Optimization of Conditional Value-at-Risk," Journal of Risk 2000.

[46] Srinivasan and Narayanan, "Variation aware placement for FPGAs," ISVLSI 2006.

[47] W. Swartz and C. Sechen, "Timing Driven Placement for Large Standard Cell Circuits," Proc. Design Automation Conference 1995.

[48] Y. Taur et al., "CMOS Scaling into the Nanometer Regime," Proc. of the IEEE, vol. 85, issue 4, pp. 486-504, April 1997.

[49] R. S. Tsay and J. Koehl, "An analytic net weighting approach for performance optimization in circuit placement," *Proc. Design Automation Conf.*, pp. 620-625, 1991.

[50] P. Tseng, "Approximation Accuracy, Gradient Methods, and Error Bound for Structured Convex Optimization," Mathematical Programming, vol. 125, no. 2, October 2010, pp. 263-295.

[51] L. Vandenberghe, class notes, UCLA EE 236C: Large Scale Optimization, 2011.

[52] Viswanathan et al., "ITOP: Integrating Timing Optimization within Placement," ISPD 2010.

[53] C. Visweswariah, "Death, taxes and failing chips," *Proc. Design Automation Conference*, pp. 343-347, 2003.

[54] C. Visweswariah et al., "First-order incremental block-based statistical timing analysis," Trans. Computer Aided Design 2006.

[55] M. H. Wright, "Interior methods for constrained optimization," *Acta Numerica 1992*, pp. 341-407.

[56] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," tech. report, Microelectronics Center of North Carolina, Research Triangle Park, NC, 1991.