LogSumExp Objectives for VLSI Placement

Karl Stefan Welzel

Geboren am 18. Dezember 1998 in Bonn

23. Juli 2020

Bachelorarbeit Mathematik

Betreuer: Prof. Dr. Stephan Held

Zweitgutachter: Dr. Ulrich Brenner

Forschungsinstitut für Diskrete Mathematik

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

1 Introduction

This bachelor thesis is concerned with a subproblem of the global placement problem originating in computer chip design. Chip design is a field of research that tries to develop methods to automate the design of very-large scale integrated (VLSI) chipsets because purely manual design would not be able to handle the ever increasing complexity of modern day computer chips. Many of the challenging problems that arise in this field are only solvable efficiently enough with heuristic algorithms. Chip design remains an active research area as new heuristics are explored and some problems change with further progress in chip manufacturing technologies.

At some point during the chip design process it is fixed which prebuilt components will make up the chip and how they are connected. At this stage the rectangular components called cells need to be placed on the rectangular chip area without overlaps such that the estimated length of the wires needed to connect the cells is minimal given that some of the cells need to be connected to previously placed contact points on the chip. As part of a classical heuristic approach to this problem one relaxes the constraint that the cells may not overlap in the very first step. This leads to an unconstrained minimization problem that mainly depends on how the wirelength is estimated. Some wirelength estimations may be minimized very efficiently while others come closer to estimating the real wirelength. In this bachelor thesis we investigate the use of two wirelength estimations based on the maximum approximations LogSumExp and Weighted-Average, which proved to be successful in other contexts, during this unconstrained minimization. These two wirelength estimations are differentiable approximations of the canonical (and not differentiable) wirelength estimation HPWL and allow for the use of convex optimization methods such as gradient descent and Nesterov's accelerated gradient method.

This thesis consists of two main parts: In the first part the problem and the optimization methods are described in detail and the theoretical properties of the two wirelength estimations are summarized. Here, the properties that are relevant either to their use as wirelength estimations or to their use as objective functions that are minimized with convex optimization methods are presented. The second part is concerned with some pratical considerations and the evaluation of the performance of these wirelength estimations in the unconstrained minmization problem outlined above.

2 Global Placement

This chapter will give an overview of the global placement problem, its origin and its mathematical formalization. Afterwards, the netlength estimations that play a role in this thesis are presented.

2.1 Global Placement and Chip Design

Global placement is one of the automated steps in the chip design process. In order to explain the role of global placement, the following section will be a rough depiction of the chip design process. For more detailed information about the chip design process we refer to [AMS08] and [Lav+16].

At the start of the chip design process the behavior and performance details of the chip need to be specified. Then these requirements are translated into a set of prebuilt rectangular components and connections between these components. The components are called cells, their in- and outputs are called pins and the connections are specified in terms of sets of pins that are electrically equivalent and need to be connected. In one of the last steps prior to production this representation is converted into a geometric model of the final chip satisfying performance and manufacturing requirements. This step is called *physical design* and while it is already largely automated physical design automation remains an active area of research that aims at further improving the existing algorithms and solving new problems that come up with new advancements in chip technology.

Physical design is further split up into multiple stages. These stages are not only executed one after the other but often the results of later stages are used to adapt design decisions which then also influence the earlier stages. The two major steps that are necessary to understand the core optimization problem of this thesis are *placement* and *routing*. Placement tries to find valid positions for the components and routing defines the paths of the conductors that connect the pins as specified afterwards. The placement results heavily influence the results of the routing step and are important for the final performance of the chip.

This leads to the question of how the performance of a chip is measured. What is the objective function of placement and routing? While there are multiple desired traits of a chip such as short cycle times, low power consumption, and low manufacturing cost, it is hard to put them in a single objective function and it would be even harder to optimize this objective function. Therefore, an objective function is used that is easier to handle and correlates with these design goals: Total weighted wirelength, i.e. the weighted sum of the lengths of the conductor paths

that connect the pins. Lower wirelength leads to less resistance, less heat and less power consumption and also shorter cycle times. Weights are introduced to put emphasis on certain connections, for example those that are critical for faster clock cycles.

When choosing wirelength as the objective for placement and routing, a new problem arises. To compute the total weighted wirelength during placement, we would need to execute a complete routing step and evaluate it afterwards. This is not feasible since the runtime would be too high. Instead, the total weighted wirelength is estimated with a simple function that is fast to evaluate. Taking all those considerations into account, we might formulate the placement problem this way: Find positions for all cells such that the estimated total weighted wirelength is minimal while the cells do not overlap and meet some additional requirements.

A standard requirement is that the cells need to fit into the power grid that is already laid out on the chip. The chip area is partitioned into rows that lie between power lines alternating between current supply and ground. The majority of the cells have a height of exactly one row, these are called standard cells, and some have a height spanning multiple rows, these are called macros. To correctly fit within the power grid now means that all cell positions have to be aligned with these vertical rows. Further requirements often include that some cells are only allowed to occupy certain areas of the chip (some ares might be blocked by previously placed large chip components) and that the cells need to leave enough free space for the conductor paths. This shows how complex the placement task can become once all practical demands are formulated.

However, since the placement problem is still too hard to be tackled at once, it is instead split up into three steps: The first step called *global placement* tries to determine well-spread cell position with little overlap and low estimated wirelength. The result is then taken as the input to the *legalization* step which tries to legalize it with as little movement of the cells as possible. Here, a legal placement refers to a placement without any overlaps that also fulfills the additional requirements mentioned above. Often, an additional optimization step called *detailed placement* is applied to the legal placement to find local improvements not violating any constraints. In this thesis we will consider a simplification of the global placement task: Minimization of wirelength estimations without any constraints, in particular without considering cell overlaps.

Solving this problem can be combined with an iterative geometric partitioning approach (see [BV08, pp. 12-16]) that assigns the cells to smaller and smaller regions of the chip to spread out the cells evenly and reduce overlap. The idea is to minimize the wirelength estimation for all cells at once first, then use this placement to assign the cells to regions (for example the four quadrants of the chip) such that the cell density in each region is reasonable, minimize wirelength estimations in each region, subdivide the regions and continue alternating between minimization and partitioning until the regions are small enough for the legalization step. This means that even though minimizing the netlength estimation for all cells at once will most likely not produce a placement that is even close to an optimal overlap-

free placement, it should provide useful insights into the relative placements of the cells, i.e. if a cell should rather be placed on the left half of the chip or on the right half and if cell A should rather be placed above or below cell B.

2.2 Mathematical Description

At this point, we will introduce the mathematical formalism necessary to rigorously state the global placement problem. Note that this description of the global placement problem is a simplification of the what needs to be done in the global placement step as discussed above. We will mostly follow the notation in [BV08]. There are four main ingredients: The set of movable cells C, the set of pins P, the set of nets $\mathcal{N}$ and the rectangular chip area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$.

The cells are rectangular components to be placed on the chip area, but since there are no disjointness constraints we do not care about their shape. We distinguish between movable and fixed cells: Fixed cells are already placed beforehand and cannot be moved while movable cells are those that we are given the task to determine the position of. The position of (an anchor point of) a movable cell $c \in C$ is denoted by $(x(c), y(c)) \in \mathbb{R}^2$. We also use $\mathbf{x}^C, \mathbf{y}^C \in \mathbb{R}^C$ as vectors containing all cell positions of movable cells.

Every cell has one or more pins whose relative positions are specified by some offset to the anchor point of the cell. For fixed cells we may specify the offsets relative to a global anchor point instead of multiple cell-specific anchor points. Therefore, we assume that there is exactly one fixed cell with coordinates $(0, 0)$ which we will call $\square$. To denote that the pin $p \in P$ belongs to the cell $c \in C \cup \{\square\}$, we will write $\alpha(p) = c$ and the offset of the pin p relative to the anchor point of $\alpha(p)$ is denoted by $(x_{\text{offs}}(p), y_{\text{offs}}(p)) \in \mathbb{R}^2$.

Lastly, $\mathcal{N}$ is a partition of P into sets containing at least two elements. Each net $N \in \mathcal{N}$ represents a set of pins that need to be connected on the final chip. Because not all nets are equally important, a weight $w(N) \in \mathbb{R}_{\geq 0}$ is assigned to every net N. The information about cells, pins, pin offsets and nets combined is called the *netlist*.

Let $L \colon \{\, V \subseteq \mathbb{R}^2 \mid 2 \leq |V| < \infty \,\} \to \mathbb{R}_{\geq 0}$ be a function that estimates the length of the wires required to connect all pins in a net given their positions (the *netlength*). Now, the the global placement problem is to find values $x(c) \in [x_{\min}, x_{\max}]$ and $y(c) \in [y_{\min}, y_{\max}]$ for each movable cell $c \in C$ such that the weighted sum of netlengths

$$\sum_{N \in \mathcal{N}} w(N) L(\{\, (x(\alpha(p)) + x_{\text{offs}}(p),\ y(\alpha(p)) + y_{\text{offs}}(p)) \mid p \in N \,\})$$

is minimized.

All netlength estimations L have the property that a translation of all pins in a net leaves the value unchanged. If there is a connected component in the hypergraph $(P, \mathcal{N})$ without a fixed pin, there is no unique optimal placement and the choice of an

optimal placement of (a part of) the cells will be arbitrary because all cell positions in this connected component can be shifted simultaneously without changing the value of the objective function. To avoid this problem, we will only consider those instances of the problem in which each connected component in the hypergraph $(P, \mathcal{N})$ contains at least one fixed pin. This is typically the case for all practical instances.

Additionally, we assume that $x_{\text{offs}}(p) \in [x_{\min}, x_{\max}]$ and $y_{\text{offs}}(p) \in [y_{\min}, y_{\max}]$ for every fixed pin p. In practice, all pin offsets of pins of movable cells are small and any selection of cell positions that minimizes the objective function will meet the condition that $x(c) \in [x_{\min} - \varepsilon, x_{\max} + \varepsilon]$ and $y(c) \in [y_{\min} - \varepsilon, y_{\max} + \varepsilon]$ for all $c \in C$ and some small $\varepsilon > 0$. (Moving a cell away from the chip area only increases the distance to every fixed pin.) Thus, the objective function changes very little if one just moves every cell that lies outside of the chip area in an optimal solution to the closest position inside the chip area afterwards and the solution stays close to optimal. In this sense it suffices to solve the unconstrained minimization problem and this property is necessary for the convex optimization methods used in this thesis.

For brevity we introduce some additional notation: Let $\mathbf{x}^P_{\text{offs}}(N), \mathbf{y}^P_{\text{offs}}(N) \in \mathbb{R}^N$ be the vectors containing the pin offsets of the pins in a net N and $A(N) \in \mathbb{R}^{N \times C}$ be the matrix that maps cells to pins of N by setting $(A(N))_{p,c} = 1$ if $\alpha(p) = c$ and 0 otherwise. Multiplying a vector with entry 1 at position $c \in C$ with this matrix gives a vector with entry 1 at each pin of N belonging to cell c. These definitions allow us to express the pin positions $\mathbf{x}^P(N), \mathbf{y}^P(N) \in \mathbb{R}^N$ in N as an affine linear transformation of the cell positions:

$$\mathbf{x}^P(N) = A(N)\mathbf{x}^C + \mathbf{x}^P_{\text{offs}}(N) \quad \text{and} \quad \mathbf{y}^P(N) = A(N)\mathbf{y}^C + \mathbf{y}^P_{\text{offs}}(N)$$

By defining $L(\mathbf{x}, \mathbf{y})$ as a shorthand for $L(\{\, (x_i, y_i) \mid 1 \leq i \leq n \,\})$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we can write the objective function of the global placement problem as

$$T_L(\mathbf{x}, \mathbf{y}) := \sum_{N \in \mathcal{N}} w(N) L\big(A(N)\mathbf{x} + \mathbf{x}^P_{\text{offs}}(N), A(N)\mathbf{y} + \mathbf{y}^P_{\text{offs}}(N)\big)$$

At this point, it is clear that there is a difference between a netlength estimation L and its corresponding single net objective function. While L estimates the netlength based on the pin positions, the corresponding objective function estimates the netlength based on the cell positions (using L). Whenever we mean the objective function for a net N, we will use

$$L^N(\mathbf{x}, \mathbf{y}) = L\big(A(N)\mathbf{x} + \mathbf{x}^P_{\text{offs}}(N), A(N)\mathbf{y} + \mathbf{y}^P_{\text{offs}}(N)\big)$$

as an abbreviation.

2.3 Netlength Estimation

An important factor for the performance of any algorithm that aims to solve the global placement problem is the choice of the function that estimates the netlength.

In this thesis, our goal is to compare different netlength estimations and their optimization methods. This section should serve as an overview of the estimations in this thesis as well as some important ones in practice and theory.

When routing a chip, i.e. laying out all the conductor paths on the chip, these paths can only go horizontally or vertically, never diagonally. In other words, the rectilinear distance is used to measure the length of a conductor path between two pins on the chip. Because of this property, some functions that are estimating the netlength are a sum of the estimated netlength in x direction and the estimated netlength in y direction. Of course, there is no substantial difference between those directions so these functions (most often) are of the form: $L(\mathbf{x}, \mathbf{y}) = \overline{L}(\mathbf{x}) + \overline{L}(\mathbf{y})$. This property allows us to optimize x- and y-values independently from one another allowing for parellel execution of the optimization processes.

We will also use property of the netlength approximations to shorten the notation in this thesis. Whenever there is a bar above a function L estimating the netlength it refers to the one-dimensional part of L and the corresponding function for two dimensions is $\overline{L}(\mathbf{x}) + \overline{L}(\mathbf{y})$. To be well-defined, we set $\overline{L}^N$ to be the one-dimensional single net objective function in x-direction but this notation will only be used in a general setting in which this decision does not matter.

2.3.1 Rectilinear Steiner Trees

As already mentioned, conductor paths can only go horizontally or vertically. So when routing a single net, the lowest total wirelength can be achieved by routing with a shortest rectilinear Steiner tree where the terminals are the pin positions. This makes the length of a shortest rectilinear Steiner tree an obvious starting point for netlength estimation. We will denote this length as $\mathrm{STEINER}(\mathbf{x}, \mathbf{y})$. Unfortunately, finding the length of a shortest rectilinear Steiner tree for a given set of points in the plane is NP-hard as shown in [GJ77]. Because algorithms to determine a shortest (or even just sufficiently short) rectilinear Steiner tree are often very complex and time-consuming, this netlength estimation is not often used in practice. Also, this function will not be discussed in this thesis except for some theoretical comparisons.

2.3.2 Half-Perimeter Wirelength

The *half-perimeter wirelength* (HPWL) or *bounding box* heuristic estimates the netlength as half the perimeter of the bounding box of the pin positions:

$$\begin{aligned}\mathrm{HPWL}(\mathbf{x}, \mathbf{y}) &:= \max_{i,j \in \{1,\dots,n\}} |x_i - x_j| + \max_{i,j \in \{1,\dots,n\}} |y_i - y_j| \\ &= \max(\mathbf{x}) - \min(\mathbf{x}) + \max(\mathbf{y}) - \min(\mathbf{y})\end{aligned}$$

The advantage of this expression is that it is easy to write and fast to evaluate. For nets with two or three pins the HPWL coincides with the length of a rectilinear Steiner tree. The HPWL is the standard metric to compare the wirelength of

placements in benchmarks suites like the ISPD 2005 [Nam+05] and ISPD 2006 [Nam06]. This function is not differentiable thus excluding analytical methods as optimization methods but the minimization of weighted wirelength with HPWL as netlength estimation can be formulated as an LP or a MinCostFlow problem as shown in [BV08, pp. 5-7].

2.3.3 Quadratic Wirelength

The idea of quadratic wirelength is to replace the rectilinear distance function

$$d_L((x_p, y_p), (x_q, y_q)) = |x_p - x_q| + |y_p - y_q|$$

which is not differentiable by a quadratic distance function

$$d_Q((x_p, y_p), (x_q, y_q)) = (x_p - x_q)^2 + (y_p - y_q)^2.$$

Because a distance function can not handle more than two points a netmodel is used to break down a net with more than two pins into two pin nets. A classic example of such a netmodel is the CLIQUE net model. It transforms a net N into $\binom{|N|}{2}$ two pin nets by creating a new net for each pair of pins. To keep large nets from dominating the objective function, each new net is weighted by $\frac{1}{|N|-1}$. The resulting netlength estimation is

$$\text{QCLIQUE}(\mathbf{x}, \mathbf{y}) := \frac{1}{|N| - 1} \sum_{i,j \in \{1,\dots,n\}} (x_i - x_j)^2 + (y_i - y_j)^2.$$

For an overview of different net models and quadratic wirelength we refer to [BV08]. Minimizing quadratic wirelength is, for example, used in GORDIAN [Kle+91] and BonnPlace [BS05]. Replacing the distance function comes at the cost of a skewed objective function which might lead to suboptimal results when evaluated with HPWL.

2.3.4 LogSumExp

The usage of the *LogSumExp* (LSE) function in placement was proposed in [NDS01] and was used in multiple nonlinear placers such as mPL6 [Cha+06], NTUplace3 [Che+08] and APlace [KW06]. The LSE function itself is defined as

$$\text{LSE}_\gamma(\mathbf{x}) = \gamma \log\left(\sum_{i=1}^{n} \exp\left(\frac{x_i}{\gamma}\right)\right), \quad \gamma > 0$$

for all $\mathbf{x} \in \mathbb{R}^n$ and approximates the maximum function. Because the HPWL can be written solely in terms of the maximum function, we may use the LSE function to get a differentiable approximation of HPWL which we will call *Netlength-LogSumExp* (NLSE):

$$\overline{\text{NLSE}}_\gamma(\mathbf{x}) := \text{LSE}_\gamma(\mathbf{x}) + \text{LSE}_\gamma(-\mathbf{x}) \approx \max(\mathbf{x}) + \max(-\mathbf{x}) = \overline{\text{HPWL}}(\mathbf{x}).$$

2.3.5 Weighted-Average

A close relative to the LSE function is the *Weighted-Average* (WA) function defined as

$$\mathrm{WA}_\gamma(\mathbf{x}) = \frac{\sum_{i=1}^n x_i \exp(x_i/\gamma)}{\sum_{i=1}^n \exp(x_i/\gamma)}, \quad \gamma > 0$$

which has been proposed in [HCB11] and used in ePlace [Lu+15]. It approximates the maximum function even better than LSE in terms of absolute error and can also be used to get a differentiable approximation of HPWL which we will call *Netlength-Weighted-Average* (NWA) in the same way as above:

$$\overline{\mathrm{NWA}}_\gamma(\mathbf{x}) := \mathrm{WA}_\gamma(\mathbf{x}) + \mathrm{WA}_\gamma(-\mathbf{x}) \approx \max(\mathbf{x}) + \max(-\mathbf{x}) = \overline{\mathrm{HPWL}}(\mathbf{x}).$$

2.3.6 A Visual Comparison

The previous description mostly consists of formulas. To get a direct visual comparison, Figure 2.1 graphs all of the netlength estimations above (except for STEINER) for a net with one fixed pin at coordinates $(0, 0)$ and one movable pin at coordinate $(x, 0)$. The STEINER netlength estimation is omitted because it coincides with the HPWL netlength estimation for a two-pin net.

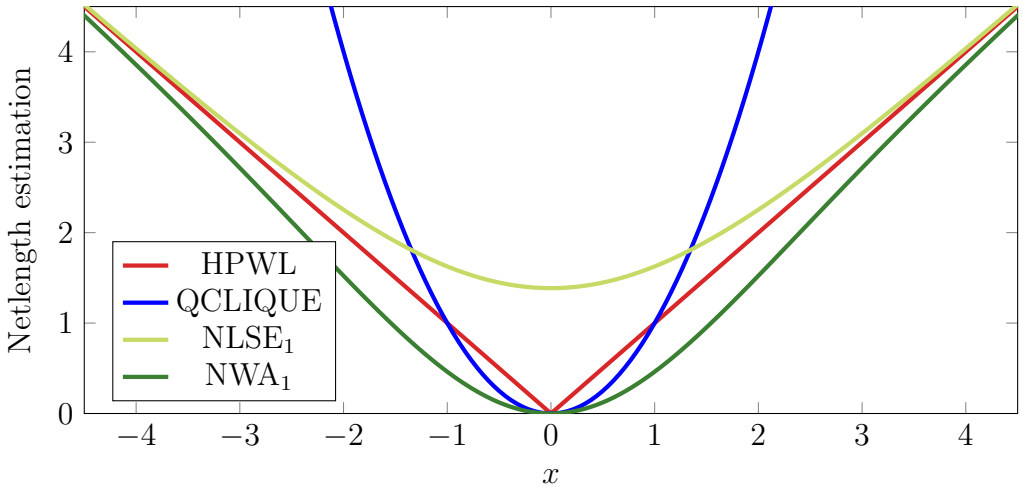


Figure 2.1: One-dimensional netlength estimations for a net with one fixed and one movable pin

3 Convex Optimization

In this chapter, we will introduce the two methods of convex optimization used in this thesis. Both methods and varying parameters will be used to optimize NLSE and NWA. The optimization problem solved by these methods is the unconstrained minimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

for a function $f\colon \mathbb{R}^n \to \mathbb{R}$. As discussed in Section 2.2, it suffices to solve an unconstrained minimization problem in the case of minimizing weighted total wirelength.

3.1 Gradient Descent

The idea of *gradient descent* (GD) is simple: Start with some vector $\mathbf{x}^{(0)}$ and iterate the following process

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - h_k \nabla f\big(\mathbf{x}^{(k)}\big)$$

until we are satisfied. The variable h_k is called the *step size* and needs to be a positive number that may depend on the vectors of previous iterations.

Let us first inspect the theoretical convergence properties of this method:

Theorem 3.1 (Corollary 2.1.2 in [Nes04])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex differentiable function with a Lipschitz continuous gradient:*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2 \quad \forall\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

Let the step size be $h_k = 1/L$ for all $k \in \mathbb{N}$ and $(\mathbf{x}^{(k)})_{k\in\mathbb{N}}$ be the sequence generated by the gradient descent method. Then

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq \frac{2L\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2}{k+4}$$

where $\mathbf{x}^*$ is any point at which f attains its minimum.

This theorem guarantees convergence to a point with a value that is close to optimal under three conditions: The objective function is convex, differentiable and has a Lipschitz continuous gradient. To check whether these conditions are fulfilled by our objective functions, will be discussed in the next chapter.

In this chapter, we want to discuss the parameter choices of the gradient descent method:

1. What should $\mathbf{x}^{(0)}$ be?
2. How should we choose h_k?
3. When do we stop the process?

3.1.1 The Start Vector

The question of which start vector to use obviously depends on the use case and thus cannot be answered in this general description of the gradient descent method. The theorem above suggests that the start vector should be close to the solution vector. Since very little is known about the solution vector in advance one has to start with a best guess. Note that even though a lower value of the objective function of a start vector can serve as an indicator of the vector being close to the optimum the guaranteed convergence in the above theorem only contains the distance between the start vector and the closest optimal vector $\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2$ and not the difference $f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)$.

3.1.2 The Step Size Strategy

The step size strategy is the most important aspect of the gradient descent method for its convergence properties. If the step size is too small, gradient descent will converge extremely slowly, if it is too large the values may oscillate and can even diverge. Theorem 3.1 shows that a step size close to $1/L^*$ (where L^* is the optimal Lipschitz constant of the gradient) would yield the fastest method in the case in which the theorem proves convergence. The best Lipschitz constant is often unknown and other step size strategies have been proposed. In this thesis, we will consider the following:

- `constant`: $h_k = h_0$
- `scaled-by-sqrt`: $h_k = \frac{h_0}{\sqrt{k+1}}$
- `backtracking-line-search`: $h_k = 2^{-n} h_{k-1}$ with minimal n such that

$$f(\mathbf{x}^{(k)} - h_k \nabla f(\mathbf{x}^{(k)})) \leq f(\mathbf{x}^{(k)}) - \frac{1}{2} h_k \left\| \nabla f(\mathbf{x}^{(k)}) \right\|_2^2$$

- `observed-lipschitz`: $h_k = \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2}{\|\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)})\|_2}$
- `min-observed-lipschitz`: $h_k = \min\left\{ h_{k-1}, \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2}{\|\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)})\|_2} \right\}$
- `barzilai-borwein`: $h_k = \frac{\left| \left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)^T \left(\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)})\right) \right|}{\|\nabla f(\mathbf{x}^{(k-1)}) - \nabla f(\mathbf{x}^{(k)})\|_2^2}$

`constant` is the simplest method and by the theorem above it is also effective if a small Lipschitz constant is known. But even if a Lipschitz constant can be derived by analytical means, it might be a suboptimal value with respect to the search space and can lead to very slow convergence.

`scaled-by-sqrt` is a variant of the constant step size. The step size is gradually lowered such that it will be low enough for convergence eventually. Because it depends on a start step size h_0, which has to be guessed, it has the same problems as a constant step size. Both of the previous step size strategies are mentioned in [Nes04].

`backtracking-line-search` is a strategy that gradually lowers the step size to guarantee that the objective function of the minimization sequence $\mathbf{x}^{(k)}$ is decreasing enough. If $h_k \leq 1/L^*$, the condition is guaranteed to hold so the search process terminates and $h_k \geq 1/(2L^*)$ for all k. The condition is called *Armijo rule* and was proposed in [Arm66]. Armijo even showed that under slightly stronger requirements than in previous theorem convergence is guaranteed.

`observed-lipschitz` and `min-observed-lipschitz` estimate the reciprocal of the Lipschitz constant by the reciprocal of the "observed Lipschitz constant". While `observed-lipschitz` always uses the current observed Lipschitz constant, `min-observed-lipschitz` tries to get closer to the reciprocal of the actual Lipschitz constant by only ever decreasing the step size. The `observed-lipschitz` method is used in [Lu+15] as "Lipschitz constant prediction" to avoid the additional runtime of `backtracking-line-search`.

Finally, `barzilai-borwein` is named after its inventors Barzilai and Borwein which proposed it in [BB88]. Although convergence is only guaranteed for a more sophisticated adaptation of this strategy (see [Ray97]), the good pratical results justify a direct use for our purposes.

Note that for all of theses step size strategies the very first step size h_0 needs to be specified because there are no previous values and previous gradients to calculate the step size. This means that there are two independent parameters of the gradient descent method here: The first step size h_0 and the step size strategy.

3.1.3 The Stopping Criterion

Because the gradient descent method is an iterative process, it has no obvious ending. We have to specify a criterion that estimates if further iterations are likely to decrease the objective function by a considerable amount. A standard choice is

$$\|\nabla f(\mathbf{x}^{(k)})\|_2 < \varepsilon$$

(see [BV04, p. 466]). A small norm of the gradient indicates that $\mathbf{x}^{(k)}$ is close to a point $\mathbf{x}$ satisfying $\nabla f(\mathbf{x}) = 0$ and if f is convex any such point is a global minimum. In fact, if the objective function is strongly convex, then a vector with a small norm of the gradient is provably close to optimum depending on the strong convexity parameter.

3.2 Nesterov's Accelerated Gradient

Nesterov's accelerated gradient (NAG) method aims to improve the gradient descent method. It was introduced by Yurii Nesterov in [Nes83] and later reformulated as a momentum method by Sutskever et al. in the appendix of [Sut+13]. We will use the easier momentum formulation.

The NAG method has two steps in each iteration:

$$\begin{aligned}\widetilde{\mathbf{x}}^{(k+1)} &= \mathbf{x}^{(k)} + \mu_k\big(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\big)\\ \mathbf{x}^{(k+1)} &= \widetilde{\mathbf{x}}^{(k+1)} - h_k \nabla f\big(\widetilde{\mathbf{x}}^{(k+1)}\big)\end{aligned}$$

We will refer to the first as the *momentum step* and to the second as the *gradient step*. The positive coefficient μ_k is called the decay factor and Nesterov specifies that it should be $\mu_k = (a_k - 1)/a_{k+1}$ where

$$a_0 = 1 \quad \text{and} \quad a_{k+1} = \frac{1 + \sqrt{4a_k^2 + 1}}{2}.$$

This way, μ_k starts at 0 and tends to 1 as k increases.

The momentum step is added to go further in the direction that was determined in the last iteration. Note that this is not the same as increasing the step size h_k because the direction $\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$ is not only determined by the previous iteration's gradient step but also influenced by the momentum step of the previous iteration.

Again, we look at the theoretical convergence properties:

Theorem 3.2 (Theorem 1 from [Nes83])**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex differentiable function with a Lipschitz continuous gradient:*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2 \quad \forall\ \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

Let the step size be $h_k = 1/L$ for all $k \in \mathbb{N}$ and $(\mathbf{x}^{(k)})_{k\in\mathbb{N}}$ be the sequence generated by the NAG method. Then

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^*) \leq \frac{2L\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2}{(k+2)^2}$$

where $\mathbf{x}^*$ is any point at which f attains its minimum.

The important part of this theorem is that the theoretical convergence of this method is faster than the convergence of the gradient descent method. As for the gradient descent method, we must choose $\mathbf{x}^{(0)}$, the step size strategy and the stopping criterion. The step size strategy is the only aspect we need to consider again to address a subtle change.

3.2.1 The Step Size Strategy

In the method of gradient descent there was only one sequence $(\mathbf{x}^{(k)})_{k\in\mathbb{N}}$. With the introduction of the momentum step there are two sequences $(\mathbf{x}^{(k)})_{k\in\mathbb{N}}$ and $(\widetilde{\mathbf{x}}^{(k)})_{k\in\mathbb{N}}$ and we need to decide which of them shall be used for the calculation of the step size strategies mentioned in Section 3.1.2. In this thesis, $(\widetilde{\mathbf{x}}^{(k)})_{k\in\mathbb{N}}$ will be used because hopefully, the computed step sizes are more relevant to the computed gradient (which also depends on $\widetilde{\mathbf{x}}^{(k)}$) and because this choice avoids the computation of the gradients $\nabla f(\mathbf{x}^{(k)})$. Also, in the special case of `backtracking-line-search` it allows us to use that Nesterov showed in [Nes83] that choosing the step size as $h_k = 2^{-n}h_{k-1}$ with minimal n such that

$$f(\widetilde{\mathbf{x}}^{(k)} - h_k\nabla f(\widetilde{\mathbf{x}}^{(k)})) \leq f(\widetilde{\mathbf{x}}^{(k)}) - \frac{1}{2}h_k\left\|\nabla f(\widetilde{\mathbf{x}}^{(k)})\right\|_2^2$$

leads to a similar guaranteed convergence as the one presented in Theorem 3.2.

4 Theoretical Properties of LogSumExp and Weighted-Average

To justify the use of NLSE and NWA as a netlength estimations and also the use of convex optimization techniques when minimizing the corresponding objective functions, this chapter will explore the properties of both functions. First, the results for the functions LSE and WA are presented and then those results are used to show properties of the single net objective functions $\overline{\text{NLSE}}^N$ and $\overline{\text{NWA}}^N$. It suffices to consider the one-dimensional functions because x- and y-coordinates are optimized independently. The most relevant facts are

- LSE and WA approximate the maximum function
- LSE and WA are (at least) twice differentiable
- LSE is convex, WA is bounded by two convex functions
- LSE and WA have a Lipschitz continuous gradient
- NLSE and NWA approximate the HPWL function
- $\overline{\text{NLSE}}^N$ and $\overline{\text{NWA}}^N$ are (at least) twice differentiable
- $\overline{\text{NLSE}}^N$ is convex, $\overline{\text{NWA}}^N$ is bounded by two convex functions
- $\overline{\text{NLSE}}^N$ and $\overline{\text{NWA}}^N$ have a Lipschitz continuous gradient

All of the results stated above are already known except for the Lipschitz continuity of the gradient of WA. The results about NLSE and NWA, although they seem to not have been explicitly stated, are easy corollaries.

Prior to this, let us introduce some notation. Throughout this thesis bold lowercase letters are used to denote vectors. For example $\mathbf{x} \in \mathbb{R}^n$ is a vector and its entries are $x_1, x_2, \ldots, x_n$. $\mathbf{0}$ is the n-dimensional zero vector, $\mathbf{1}$ is the n-dimensional vector with entry 1 at each position and $\mathbf{e}_i$ is the ith canonical basis vector of $\mathbb{R}^n$. Regarding matrices, $\text{diag}(\mathbf{y})$ denotes the matrix whose entries on the diagonal are $y_1, \ldots, y_n$ and whose other entries are zero and $\odot$ denotes elementwise matrix multiplication or elementwise vector multiplication. Lastly, $\|.\|_p$ for $1 \leq p \leq \infty$ refers to the p-norm for vectors and its induced norm for matrices.

4.1 LogSumExp

Many of the properties of the LogSumExp function are well known and are covered most comprehensively in [GP17] and [BV04].

4.1.1 Definition

Definition 4.1. Let $n \in \mathbb{N}_{\geq 1}$. The LogSumExp function $\mathrm{LSE}_\gamma \colon \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\mathrm{LSE}_\gamma(\mathbf{x}) = \gamma \log\left(\sum_{k=1}^{n} \exp(x_k/\gamma)\right)$$

where $\gamma > 0$ is referred to as the smoothing parameter.

Definition 4.2. Let $n \in \mathbb{N}_{\geq 1}$. The softmax function $\sigma \colon \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\sigma(\mathbf{x}) := \left(\frac{\exp(x_1)}{\sum_{k=1}^{n} \exp(x_k)}, \ldots, \frac{\exp(x_n)}{\sum_{i=k}^{n} \exp(x_k)}\right)^T$$

Remark. Note that σ maps into the set of n dimensional stochastic vectors, which is also called the $n-1$ dimensional unit simplex:

$$\Delta^{n-1} = \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \mathbf{1}^T\mathbf{x} = 1,\ \mathbf{x} \geq \mathbf{0} \right\}$$

4.1.2 Approximation Properties

Theorem 4.1 ([BV04, p. 72])**.** *The LSE function is an approximation of the maximum function in the following sense*

$$\max(\mathbf{x}) \leq \mathrm{LSE}_\gamma(\mathbf{x}) \leq \max(\mathbf{x}) + \gamma \log(n) \quad \forall \mathbf{x} \in \mathbb{R}^n$$

Proof. Because the exponential function is monotonically increasing, we get

$$\exp(\max(x_1/\gamma, \ldots, x_n/\gamma)) \leq \sum_{k=1}^{n} \exp(x_k/\gamma) \leq n \exp(\max(x_1/\gamma, \ldots, x_n/\gamma)).$$

Applying the natural logarithm on both sides and multiplying by γ proves the claim. □

4.1.3 Derivatives

Theorem 4.2 ([GP17], Proposition 1 and 2)**.** *The LSE function is twice continuously differentiable and the gradient and Hessian are*

$$\begin{aligned} \nabla \mathrm{LSE}_\gamma(\mathbf{x}) &= \sigma(\mathbf{x}/\gamma) \\ \nabla^2 \mathrm{LSE}_\gamma(\mathbf{x}) &= \frac{1}{\gamma}\big(\mathrm{diag}(\sigma(\mathbf{x}/\gamma)) - \sigma(\mathbf{x}/\gamma)\sigma(\mathbf{x}/\gamma)^T\big) \end{aligned}$$

for all $\mathbf{x} \in \mathbb{R}^n$.

Proof. Let us first consider the gradient $\nabla \operatorname{LSE}_\gamma$. For all $\mathbf{x} \in \mathbb{R}^n$ and $1 \leq i \leq n$ we get

$$\frac{\partial \operatorname{LSE}_\gamma}{\partial x_i}(\mathbf{x}) = \gamma \frac{1}{\sum_{k=1}^n \exp(x_k/\gamma)} \exp(x_i/\gamma)\frac{1}{\gamma} = \frac{\exp(x_i/\gamma)}{\sum_{k=1}^n \exp(x_k/\gamma)} = \sigma_i(\mathbf{x}/\gamma)$$

as the first partial derivative.

Now we will calculate all the second partial derivatives to get the Hessian $\nabla^2 \operatorname{LSE}_\gamma$. Let $1 \leq i, j \leq n$. The elements on the diagonal ($i = j$) are

$$\begin{aligned}\frac{\partial^2 \operatorname{LSE}_\gamma}{\partial x_i^2}(\mathbf{x}) &= \frac{\exp(x_i/\gamma)\frac{1}{\gamma}(\sum_{k=1}^n \exp(x_k/\gamma)) - \exp(x_i/\gamma)\exp(x_i/\gamma)\frac{1}{\gamma}}{\left(\sum_{k=1}^n \exp(x_k/\gamma)\right)^2} \\ &= \frac{1}{\gamma}\sigma_i(\mathbf{x}/\gamma) - \frac{1}{\gamma}\sigma_i(\mathbf{x}/\gamma)\sigma_i(\mathbf{x}/\gamma)\end{aligned}$$

and the off-diagonal entries ($i \neq j$) are

$$\frac{\partial^2 \operatorname{LSE}_\gamma}{\partial x_i \partial x_j}(\mathbf{x}) = \frac{0 \cdot (\sum_{k=1}^n \exp(x_k/\gamma)) - \exp(x_i/\gamma)\exp(x_j/\gamma)\frac{1}{\gamma}}{\left(\sum_{k=1}^n \exp(x_k/\gamma)\right)^2} = -\frac{1}{\gamma}\sigma_i(\mathbf{x}/\gamma)\sigma_j(\mathbf{x}/\gamma)$$

This shows that the Hessian has the structure claimed above. □

4.1.4 Convexity

Theorem 4.3 ([GP17], Lemma 4)**.** *The LSE function is convex but not strictly convex.*

Proof. We will first show convexity. For a twice differentiable function f with a convex domain being convex is equivalent to the Hessian $\nabla^2 f(\mathbf{x})$ being positive semidefinite (see [BV04, p. 71]). Therefore, we will now show that the Hessian of the LSE function is positive semidefinite analogous to the proof in [BV04, p. 74].

Note that $\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x}) = \frac{1}{\gamma}\nabla^2 \operatorname{LSE}_1(\mathbf{x}/\gamma)$. Scaling the matrix by $\frac{1}{\gamma} > 0$ does not change whether the matrix is positive semidefinite or not and scaling the arguments by $\frac{1}{\gamma}$ does not matter because we look at all $\mathbf{x} \in \mathbb{R}^n$. Therefore, we may assume without loss of generality that $\gamma = 1$. Let $\mathbf{v} \in \mathbb{R}^n$, then

$$\begin{aligned}\mathbf{v}^T\big(\nabla^2 \operatorname{LSE}_1(\mathbf{x})\big)\mathbf{v} &= \mathbf{v}^T\big(\operatorname{diag}(\sigma(\mathbf{x})) - \sigma(\mathbf{x})\sigma(\mathbf{x})^T\big)\mathbf{v} \\ &= \mathbf{v}^T \operatorname{diag}(\sigma(\mathbf{x}))\mathbf{v} - \mathbf{v}^T\sigma(\mathbf{x})\sigma(\mathbf{x})^T\mathbf{v} \\ &= \mathbf{v}^T \operatorname{diag}(\sigma(\mathbf{x}))\mathbf{v} - \big(\sigma(\mathbf{x})^T\mathbf{v}\big)^2 \\ &= \sum_{k=1}^n v_k\sigma_k(\mathbf{x})v_k - \left(\sum_{k=1}^n \sigma_k(\mathbf{x})v_k\right)^2 \\ &= \sum_{k=1}^n \sigma_k(\mathbf{x})v_k^2 - \left(\sum_{k=1}^n \sqrt{\sigma_k(\mathbf{x})}\sqrt{\sigma_k(\mathbf{x})}v_k\right)^2\end{aligned}$$

$$\begin{aligned}
&\overset{\text{CS}}{\geq} \sum_{k=1}^{n} \sigma_k(\mathbf{x}) v_k^2 - \left(\sum_{k=1}^{n} \sigma_k(\mathbf{x}) v_k^2\right) \underbrace{\left(\sum_{k=1}^{n} \sigma_k(\mathbf{x})\right)}_{=1} \\
&= \sum_{k=1}^{n} \sigma_k(\mathbf{x}) v_k^2 - \sum_{k=1}^{n} \sigma_k(\mathbf{x}) v_k^2 \\
&= 0
\end{aligned}$$

for all $\mathbf{x} \in \mathbb{R}^n$, so $\nabla^2 \operatorname{LSE}_1$ is positive semidefinite. We used the Cauchy-Schwarz inequality to see that

$$\left(\sum_{k=1}^{n} \sqrt{\sigma_k(\mathbf{x})}\sqrt{\sigma_k(\mathbf{x})} v_k\right)^2 \leq \left(\sum_{k=1}^{n} \sigma_k(\mathbf{x}) v_k^2\right)\left(\sum_{k=1}^{n} \sigma_k(\mathbf{x})\right).$$

We still need to show that the function is not strictly convex. Consider the line $\mathbf{x} = t\mathbf{1}$ for all $t \in \mathbb{R}$. We get

$$\operatorname{LSE}(\mathbf{x}) = \operatorname{LSE}_\gamma(t\mathbf{1}) = t + \gamma \log(n)$$

so LSE is a linear function on this line and not strictly convex. □

Remark. With a similar line of reasoning via properties of the Hessian it is also possible to show two stronger results in special cases. If one fixes at least one of the variables (the variable is no longer considered an argument to the function but fixed beforehand) the LSE function will be strictly convex and if one additionally restricts the domain of the function and of the fixed variables to be a hypercube of the form $[a, b]^n$ then the function will be strongly convex, although with a very small convexity parameter.

4.1.5 Lipschitz Continuous Gradient

Let $\|\cdot\|_2$ denote the Euclidean norm for vectors and the matrix norm induced by the Euclidean norm for matrices. We want to show that the gradient $\nabla \operatorname{LSE}_\gamma$ is Lipschitz continuous with respect to the Euclidean norm

$$\|\nabla \operatorname{LSE}_\gamma(\mathbf{x}) - \nabla \operatorname{LSE}_\gamma(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

and get tight bounds on the best (smallest) Lipschitz constant L^*. We will see that

$$L^* \overset{(1)}{=} \sup_{\mathbf{x} \in \mathbb{R}^n} \|\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x})\|_2 \overset{(2)}{\leq} \sup_{\mathbf{x} \in \mathbb{R}^n} \|\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x})\|_F \overset{(3)}{\leq} \frac{1}{2\gamma}$$

where $\|\cdot\|_F$ is the Frobenius norm which is defined as $\|A\|_F = (\sum_{i=1}^{n} \sum_{j=1}^{n} |A_{ij}|^2)^{\frac{1}{2}}$. Equation (1) follows directly from Lemma 1.2.2 in [Nes04, p. 21]. Inequality (2) comes from the fact that the matrix norm $\|\cdot\|_2$ is bounded by $\|\cdot\|_F$, see [GL13],

while the last inequality will be shown in Theorem 4.4. Afterwards, we will prove that this bound is indeed tight for $n \geq 2$. It was already known that LSE has a Lipschitz continuous gradient but it seems like the best previously known Lipschitz constant was $\frac{1}{\gamma}$ (see for example Proposition 4 in [GP17]).

Theorem 4.4. *$\nabla \operatorname{LSE}_\gamma$ is Lipschitz continuous (with respect to the Euclidean norm) and its best global Lipschitz constant is at most $\frac{1}{2\gamma}$.*

Proof. Fix $\mathbf{x} \in \mathbb{R}^n$. Let $A = \nabla^2 \operatorname{LSE}_1(\mathbf{x})$, then

$$|A_{ij}|^2 = \begin{cases} \left(\sigma_i(\mathbf{x}) - \sigma_i(\mathbf{x})^2\right)^2 = \sigma_i(\mathbf{x})^2 - 2\sigma_i(\mathbf{x})^3 + \sigma_i(\mathbf{x})^4 & \text{if } i = j \\ \left(-\sigma_i(\mathbf{x})\sigma_j(\mathbf{x})\right)^2 = \sigma_i(\mathbf{x})^2\sigma_j(\mathbf{x})^2 & \text{if } i \neq j \end{cases}$$

Therefore, we have

$$\begin{aligned} \left\|\nabla^2 \operatorname{LSE}_1(\mathbf{x})\right\|_F^2 = \|A\|_F^2 &= \sum_{i=1}^n \sum_{j=1}^n |A_{ij}|^2 \\ &= \sum_{i=1}^n \sigma_i(\mathbf{x})^2 - 2\sum_{i=1}^n \sigma_i(\mathbf{x})^3 + \sum_{i=1}^n \sum_{j=1}^n \sigma_i(\mathbf{x})^2\sigma_j(\mathbf{x})^2 \\ &= \sum_{i=1}^n \sigma_i(\mathbf{x})^2 - 2\sum_{i=1}^n \sigma_i(\mathbf{x})^3 + \left(\sum_{i=1}^n \sigma_i(\mathbf{x})^2\right)^2 \\ &\overset{\text{CS}}{\leq} \sum_{i=1}^n \sigma_i(\mathbf{x})^2 - 2\left(\sum_{i=1}^n \sigma_i(\mathbf{x})^2\right)^2 + \left(\sum_{i=1}^n \sigma_i(\mathbf{x})^2\right)^2 \\ &= \sum_{i=1}^n \sigma_i(\mathbf{x})^2 - \left(\sum_{i=1}^n \sigma_i(\mathbf{x})^2\right)^2 \leq \frac{1}{4} \end{aligned}$$

The first inequality follows from the Cauchy-Schwarz inequality

$$\left(\sum_{i=1}^n \sigma_i(\mathbf{x})^2\right)^2 = \left(\sum_{i=1}^n \sigma_i(\mathbf{x})^{\frac{3}{2}}\sigma_i(\mathbf{x})^{\frac{1}{2}}\right)^2 \leq \left(\sum_{i=1}^n \sigma_i(\mathbf{x})^3\right)\left(\sum_{i=1}^n \sigma_i(\mathbf{x})^1\right) = \sum_{i=1}^n \sigma_i(\mathbf{x})^3$$

and the second one from the fact that $a - a^2 \leq \frac{1}{4}$ for all $a \in \mathbb{R}$. We have shown that

$$\sup_{\mathbf{x}\in\mathbb{R}^n} \left\|\nabla^2 \operatorname{LSE}_1(\mathbf{x})\right\|_2 \leq \sup_{\mathbf{x}\in\mathbb{R}^n} \left\|\nabla^2 \operatorname{LSE}_1(\mathbf{x})\right\|_F \leq \sqrt{\frac{1}{4}} = \frac{1}{2}$$

which implies that

$$\sup_{\mathbf{x}\in\mathbb{R}^n} \left\|\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x})\right\|_2 = \sup_{\mathbf{x}\in\mathbb{R}^n} \left\|\frac{1}{\gamma}\nabla^2 \operatorname{LSE}_1(\mathbf{x}/\gamma)\right\|_2 = \frac{1}{\gamma} \sup_{\mathbf{x}\in\mathbb{R}^n} \left\|\nabla^2 \operatorname{LSE}_1(\mathbf{x})\right\|_2 \leq \frac{1}{2\gamma}$$

thus proving the statement. □

Theorem 4.5. *Let $n \geq 2$. Any global Lipschitz constant of $\nabla \operatorname{LSE}_\gamma$ is at least $\frac{1}{2\gamma}$.*

Proof. Fix arbitrary real numbers $(x_i)_{3 \leq i \leq n}$. We will show that there is a sequence of $\mathbf{x} \in \mathbb{R}^n$ such that $\|\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x})\|_2$ converges to a value that is at least $\frac{1}{2\gamma}$ by setting the first two entries of $\mathbf{x}$ to a and letting a tend to infinity. With the help of Theorem 4.2 we can easily calculate the limit of the gradient

$$\mathbf{x}_a := \begin{pmatrix} a \\ a \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \implies \lim_{a \to \infty} \sigma(\mathbf{x}_a/\gamma) = \frac{1}{2} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

and subsequently the limit of the Hessian:

$$\lim_{a \to \infty} \nabla^2 \operatorname{LSE}_\gamma(\mathbf{x}_a) = \frac{1}{4\gamma} \left(\begin{array}{cc|c} 1 & -1 & 0 \\ -1 & 1 & \vdots \\ \hline 0 & \dots & 0 \end{array}\right)$$

With $\mathbf{v} := \mathbf{e}_1 - \mathbf{e}_2 \in \mathbb{R}^n$ we have

$$\begin{aligned} \sup_{\mathbf{x} \in \mathbb{R}^n} \left\|\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x})\right\|_2 &\geq \lim_{a \to \infty} \left\|\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x}_a)\right\|_2 \\ &\geq \lim_{a \to \infty} \frac{\|\nabla^2 \operatorname{LSE}_\gamma(\mathbf{x}_a)\mathbf{v}\|_2}{\|\mathbf{v}\|_2} \\ &= \frac{\sqrt{\left(\frac{1}{2\gamma}\right)^2 + \left(-\frac{1}{2\gamma}\right)^2}}{\sqrt{1^2 + (-1)^2}} = \frac{\sqrt{\frac{1}{2}\frac{1}{\gamma^2}}}{\sqrt{2}} = \frac{1}{2\gamma} \end{aligned}$$

and this proves the statement. □

Corollary 4.6. *Let $n \geq 2$. Then $\nabla \operatorname{LSE}_\gamma$ is Lipschitz continuous and its best global Lipschitz constant is $\frac{1}{2\gamma}$.*

Remark. For the case $n = 1$ we have that $\nabla \operatorname{LSE}_\gamma(x) = \frac{\exp(x/\gamma)}{\exp(x/\gamma)} = 1$ (the gradient is constant) and therefore the Lipschitz constant of the gradient is zero.

4.2 Weighted-Average

The Weighted-Average function was proposed to be used in placement in [HCB11] but the analysis of the function presented in this paper lacks detailed proofs for most of the claims. It is shown that in terms of absolute error the WA function approximates the maximum function better than the LSE function. Furthermore, strict convexity and continuous differentiability are claimed without proof. we will thoroughly investigate the properties of this function and conclude that the function is indeed continuously differentiable but not convex.

4.2.1 Definition

Definition 4.3. Let $n \in \mathbb{N}_{\geq 1}$. The Weighted-Average function $\mathrm{WA}_\gamma \colon \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\mathrm{WA}_\gamma(\mathbf{x}) = \frac{\sum_{k=1}^n x_k \exp(x_k/\gamma)}{\sum_{k=1}^n \exp(x_k/\gamma)} = \sigma(\mathbf{x}/\gamma)^T \mathbf{x}$$

where $\gamma > 0$ is referred to as the smoothing parameter.

Remark. The name originates from the last equality: $\mathrm{WA}_\gamma(\mathbf{x})$ is the weighted average of $\mathbf{x}$ where the weights are determined by the softmax function. This way, larger values of an entry in $\mathbf{x}$ will result in a higher weight and we will show that this leads to a better approximation of the maximum function compared to the LSE function.

4.2.2 Approximation Properties

Theorem 4.7 ([HCB11], Theorem 1)**.** *The WA function is an approximation of the maximum function in the following sense:*

$$\max(\mathbf{x}) - \gamma W(\tfrac{n-1}{e}) \leq \max(\mathbf{x}) - \frac{\Delta\mathbf{x}}{1 + \frac{\exp(\Delta\mathbf{x}/\gamma)}{n-1}} \leq \mathrm{WA}_\gamma(\mathbf{x}) \leq \max(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^n$$

where $\Delta\mathbf{x} = \max(\mathbf{x}) - \min(\mathbf{x})$ *and* W *is the Lambert W function.*

Proof. $\mathrm{WA}_\gamma(\mathbf{x})$ is the weighted average of the elements of $\mathbf{x}$ and the weights sum up to one, so $\min(\mathbf{x}) \leq \mathrm{WA}_\gamma(\mathbf{x}) \leq \max(\mathbf{x})$.

To show the first lower bound from the claim, let us assume without loss of generality that $x_1 \geq \ldots \geq x_n$. This implies that $\Delta\mathbf{x} = x_1 - x_n$. Now

$$\begin{aligned}
\max(\mathbf{x}) - \mathrm{WA}_\gamma(\mathbf{x}) &= x_1 - \sum_{k=1}^n \sigma_k(\mathbf{x}/\gamma) x_k \\
&= (1 - \sigma_1(\mathbf{x}/\gamma)) x_1 - \sum_{k=2}^n \sigma_k(\mathbf{x}/\gamma) x_k \\
&\leq (1 - \sigma_1(\mathbf{x}/\gamma)) x_1 - \sum_{k=2}^n \sigma_k(\mathbf{x}/\gamma) x_n \\
&= \left(\sum_{k=2}^n \sigma_k(\mathbf{x}/\gamma)\right) x_1 - \left(\sum_{k=2}^n \sigma_k(\mathbf{x}/\gamma)\right) x_n \\
&= \frac{\sum_{k=2}^n \exp(x_k/\gamma)}{\sum_{k=1}^n \exp(x_k/\gamma)} (x_1 - x_n) = \frac{\Delta\mathbf{x}}{1 + \frac{\exp(x_1/\gamma)}{\sum_{k=2}^n \exp(x_k/\gamma)}} \\
&\leq \frac{\Delta\mathbf{x}}{1 + \frac{\exp(x_1/\gamma)}{\sum_{k=2}^n \exp(x_n/\gamma)}} = \frac{\Delta\mathbf{x}}{1 + \frac{\exp(x_1/\gamma - x_n/\gamma)}{n-1}}
\end{aligned}$$

$$= \frac{\Delta \mathbf{x}}{1 + \frac{\exp(\Delta \mathbf{x}/\gamma)}{n-1}}$$

For the case $x_2 = x_3 = \ldots = x_n$ all inequalities are tight.

In the last part, we will give a bound on $\max(\mathbf{x}) - \mathrm{WA}_\gamma(\mathbf{x})$ that is independent of $\mathbf{x}$. Let $f_\gamma \colon \mathbb{R}_{\geq 0} \to \mathbb{R}$ be defined by

$$f_\gamma(z) = \frac{z}{1 + \frac{\exp(z/\gamma)}{n-1}}$$

then $\sup_{\mathbf{x} \in \mathbb{R}^n}(\max(\mathbf{x}) - \mathrm{WA}_\gamma(\mathbf{x})) = \sup_{z \in \mathbb{R}_{\geq 0}} f_\gamma(z)$. We can easily see that $\lim_{z \to 0} f_\gamma(z) = \lim_{z \to \infty} f_\gamma(z) = 0$, $f_\gamma(z) \geq 0$ for all $z \in \mathbb{R}$ and $f_\gamma \neq 0$. These observations let us conclude that there is a global maximum in the interior of the domain. The function is differentiable and its derivative is

$$f'_\gamma(z) = \frac{1 + \frac{\exp(z/\gamma)}{n-1} - z\frac{\exp(z/\gamma)}{\gamma(n-1)}}{\left(1 + \frac{\exp(z/\gamma)}{n-1}\right)^2}$$

The global maximum must occur at a critical point z^* with $f'_\gamma(z^*) = 0$. This implies

$$n - 1 = (z^*/\gamma - 1)\exp(z^*/\gamma)$$

Substituting $z^*/\gamma - 1$ by w we get $\frac{n-1}{e} = w\exp(w)$ and equations of this type are only solvable with the help of the Lambert W function. For our purposes we will define the function $W \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ as the inverse of $x \mapsto x\exp(x)$ for $x \in \mathbb{R}_{\geq 0}$. This function is well defined because $x\exp(x)$ is monotonically increasing on the positive real numbers. It follows that $w = W(\frac{n-1}{e})$ and $z^* = \gamma(W(\frac{n-1}{e}) + 1)$. To compute the global maximum, let us evaluate f_γ at z^*:

$$f_\gamma(z^*) = \frac{\gamma(W(\frac{n-1}{e}) + 1)}{1 + \frac{\exp(W(\frac{n-1}{e})+1)}{n-1}} = \gamma\frac{W(\frac{n-1}{e}) + 1}{1 + e\frac{n-1}{eW(\frac{n-1}{e})}\frac{1}{n-1}} = \gamma\frac{W(\frac{n-1}{e}) + 1}{1 + \frac{1}{W(\frac{n-1}{e})}} = \gamma W(\tfrac{n-1}{e}).$$

Thus, we may conlude that

$$\sup_{\mathbf{x} \in \mathbb{R}^n}(\max(\mathbf{x}) - \mathrm{WA}_\gamma(\mathbf{x})) = \sup_{z \in \mathbb{R}_{\geq 0}} f_\gamma(z) = \gamma W(\tfrac{n-1}{e})$$

which proves the claim. □

Theorem 4.8 ([HCB11], Theorem 2)**.** *The WA function approximates max better than the LSE function in the sense that the maximum absolute error is lower, i.e.*

$$\sup_{\mathbf{x} \in \mathbb{R}^n} |\mathrm{WA}_\gamma(\mathbf{x}) - \max(\mathbf{x})| < \sup_{\mathbf{x} \in \mathbb{R}^n} |\mathrm{LSE}_\gamma(\mathbf{x}) - \max(\mathbf{x})|$$

for all $\gamma > 0$ and $n \in \mathbb{N}_{\geq 2}$.

Proof. Assume $n \geq 2$. Then $\log(n) \geq \log(2) > \frac{1}{e}$ and

$$W(\tfrac{n-1}{e}) \exp(W(\tfrac{n-1}{e})) = \frac{n-1}{e} < \frac{1}{e} \cdot n < \log(n) n = \log(n) \exp(\log(n))$$

which implies $W(\frac{n-1}{e}) < \log(n)$ because $x \exp(x)$ is monotonically increasing on the positive real numbers. With the results from Theorem 4.1 and Theorem 4.7 this yields

$$\sup_{\mathbf{x} \in \mathbb{R}^n} |\mathrm{WA}_\gamma(\mathbf{x}) - \max(\mathbf{x})| = \gamma W(\tfrac{n-1}{e}) < \gamma \log(n) = \sup_{\mathbf{x} \in \mathbb{R}^n} |\mathrm{LSE}_\gamma(\mathbf{x}) - \max(\mathbf{x})|$$

and we are done. □

Remark. Although $\frac{W(\frac{n-1}{e})}{\log(n)} \to 1$ for $n \to \infty$, the ratio is much better for small $n \geq 2$. The expression is monotonically increasing and here are some values at selected points:

$$\frac{W(\frac{2-1}{e})}{\log(2)} \approx 0.402 \quad \frac{W(\frac{20-1}{e})}{\log(20)} \approx 0.509 \quad \frac{W(\frac{200-1}{e})}{\log(200)} \approx 0.594$$

4.2.3 Derivatives

The gradient of the WA function is already stated in [Lan+14, Eq. 14] but they did not give the formula for the Hessian.

Theorem 4.9. *The WA function is twice continuously differentiable and the gradient and Hessian are*

$$\begin{aligned}
\nabla \mathrm{WA}_\gamma(\mathbf{x}) &= \sigma(\mathbf{x}/\gamma) \odot \left(\mathbf{1} + \mathbf{x}/\gamma - \frac{1}{\gamma} \mathrm{WA}_\gamma(\mathbf{x}) \cdot \mathbf{1} \right) \\
\nabla^2 \mathrm{WA}_\gamma(\mathbf{x}) &= \frac{1}{\gamma} \operatorname{diag}\left(\sigma(\mathbf{x}/\gamma) \odot \left(2 \cdot \mathbf{1} + \mathbf{x}/\gamma - \frac{1}{\gamma} \mathrm{WA}_\gamma(\mathbf{x}) \cdot \mathbf{1} \right) \right) \\
&\quad - \frac{1}{\gamma} \sigma(\mathbf{x}/\gamma) \sigma(\mathbf{x}/\gamma)^T \odot \left(2 \cdot \mathbf{1}\mathbf{1}^T + \mathbf{1}\mathbf{x}^T/\gamma + \mathbf{x}\mathbf{1}^T/\gamma - \frac{2}{\gamma} \mathrm{WA}_\gamma(\mathbf{x}) \cdot \mathbf{1}\mathbf{1}^T \right)
\end{aligned}$$

for all $\mathbf{x} \in \mathbb{R}^n$ where $\odot$ denotes the element-wise matrix multiplication.

Proof. We will only consider the case $\gamma = 1$. The general formulas follow directly from the identity $\mathrm{WA}_\gamma(\mathbf{x}) = \gamma \, \mathrm{WA}_1(\mathbf{x}/\gamma)$ and the chain rule.

Let us begin by calculating the gradient $\nabla \mathrm{WA}_1$. For all $\mathbf{x} \in \mathbb{R}^n$ and $1 \leq i \leq n$ we have

$$\begin{aligned}
\frac{\partial \, \mathrm{WA}_1}{\partial x_i}(\mathbf{x}) &= \frac{\partial}{\partial x_i} \left(\sum_{k=1}^n \sigma_k(\mathbf{x}) x_k \right) \\
&= \sigma_i(\mathbf{x}) + \sigma_i(\mathbf{x}) x_i - \sum_{k=1}^n \sigma_i(\mathbf{x}) \sigma_k(\mathbf{x}) x_k
\end{aligned}$$

$$= \sigma_i(\mathbf{x})(1 + x_i - \mathrm{WA}_1(\mathbf{x}))$$

as the first partial derivatives. The derivative of σ_k was already obtained in Theorem 4.2.

Now we will calculate all the second partial derivatives to get the Hessian $\nabla^2 \mathrm{WA}_1$. Let $1 \leq i, j \leq n$. The elements on the diagonal ($i = j$) are

$$\begin{aligned}
\frac{\partial^2 \mathrm{WA}_1}{\partial x_i^2}(\mathbf{x}) &= \frac{\partial}{\partial x_i}(\sigma_i(\mathbf{x})(1 + x_i - \mathrm{WA}_1(\mathbf{x}))) \\
&= (\sigma_i(\mathbf{x}) - \sigma_i(\mathbf{x})\sigma_i(\mathbf{x}))(1 + x_i - \mathrm{WA}_1(\mathbf{x})) \\
&\quad + \sigma_i(\mathbf{x})(1 - \sigma_i(\mathbf{x})(1 + x_i - \mathrm{WA}_1(\mathbf{x}))) \\
&= \sigma_i(\mathbf{x})(2 + x_i - \mathrm{WA}_1(\mathbf{x})) - \sigma_i(\mathbf{x})\sigma_i(\mathbf{x})(2 + 2x_i - 2\,\mathrm{WA}_1(\mathbf{x}))
\end{aligned}$$

and the off-diagonal entries ($i \neq j$) are

$$\begin{aligned}
\frac{\partial^2 \mathrm{WA}_1}{\partial x_i \partial x_j}(\mathbf{x}) &= \frac{\partial}{\partial x_j}(\sigma_i(\mathbf{x})(1 + x_i - \mathrm{WA}_1(\mathbf{x}))) \\
&= -\sigma_i(\mathbf{x})\sigma_j(\mathbf{x})(1 + x_i - \mathrm{WA}_1(\mathbf{x})) + \sigma_i(\mathbf{x})(-\sigma_j(\mathbf{x})(1 + x_j - \mathrm{WA}_1(\mathbf{x}))) \\
&= -\sigma_i(\mathbf{x})\sigma_j(\mathbf{x})(2 + x_i + x_j - 2\,\mathrm{WA}_1(\mathbf{x}))
\end{aligned}$$

This shows that Hessian has the structure claimed above. □

4.2.4 Convexity

As already mentioned, Hsu et al. claimed in [HCB11] that the WA function is strictly convex but the following theorem shows that it is not.

Theorem 4.10. *The WA function is not convex for any $n \in \mathbb{N}_{\geq 2}$.*

Proof. Again, for a twice differentiable function f with a convex domain being convex is equivalent to the Hessian $\nabla^2 f(\mathbf{x})$ being positive semidefinite everywhere (see [BV04, p. 71]). To show that $\nabla^2 \mathrm{WA}_\gamma(\mathbf{x})$ is not positive semidefinite for all $\mathbf{x} \in \mathbb{R}^n$, we pick $\mathbf{x} = (\gamma a, 0, \ldots, 0)^T \in \mathbb{R}^n$. We define $\tilde{\sigma} : \mathbb{R} \to (0, 1)$ by

$$\tilde{\sigma}(a) := \sigma_1(\mathbf{x}/\gamma) = \frac{e^a}{e^a + (n-1)e^0} = \frac{e^a}{e^a + n - 1}$$

and notice that

$$\frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x}) = \sum_{k=1}^{n} \sigma_k(\mathbf{x}/\gamma)(x_k/\gamma) = \sigma_1(\mathbf{x}/\gamma)(x_1/\gamma) = \tilde{\sigma}(a)a.$$

Now we are ready to evaluate $\mathbf{e}_1^T(\nabla^2 \mathrm{WA}_\gamma(\mathbf{x}))\mathbf{e}_1$ with the help of Theorem 4.9:

$$\mathbf{e}_1^T\big(\nabla^2 \mathrm{WA}_\gamma(\mathbf{x})\big)\mathbf{e}_1 = \frac{1}{\gamma}\tilde{\sigma}(a)(2 + a - \tilde{\sigma}(a)a) - \frac{1}{\gamma}\tilde{\sigma}(a)^2(2 + 2a - 2\tilde{\sigma}(a)a)$$

$$= \frac{1}{\gamma}\tilde{\sigma}(a)\big((2a)\tilde{\sigma}(a)^2 - (3a+2)\tilde{\sigma}(a) + (2+a)\big)$$

Because $\frac{1}{\gamma}\tilde{\sigma}(a)$ is positive for all $a \in \mathbb{R}$, it suffices to show that the last factor can be negative. Consider the quadratic function $b \mapsto (2a)b^2 - (3a+2)b + (2+a)$. For all $a \notin \{0, 2\}$ it has exactly two roots: $b = 1$ and $b = \frac{1}{a} + \frac{1}{2}$.

Assume that $a > 2$, then

$$\tilde{\sigma}(a) = \frac{e^a}{e^a + n - 1} > \frac{1}{a} + \frac{1}{2} \iff 1 + \frac{n-1}{e^a} < \frac{2a}{2+a}$$

and analogously for $a < -2$

$$\tilde{\sigma}(a) = \frac{e^a}{e^a + n - 1} < \frac{1}{a} + \frac{1}{2} \iff 1 + \frac{n-1}{e^a} > \frac{2a}{2+a}$$

Because $\lim_{a\to\infty} 1 + \frac{n-1}{e^a} = 1$, $\lim_{a\to-\infty} 1 + \frac{n-1}{e^a} = \infty$ and $\lim_{a\pm\infty} \frac{2a}{2+a} = 2$, there is an $M > 2$ such that the first inequality is true for all $a > M$ and the second inequality is true for all $a < -M$.

Back to the quadratic function $b \mapsto (2a)b^2 - (3a+2)b + (2+a)$. For $a > M > 0$ the function is negative within the interval $[\frac{1}{a} + \frac{1}{2}, 1]$ and for $a < -M < 0$ it is negative outside of the interval $[\frac{1}{a} + \frac{1}{2}, 1]$. Our considerations from above and the fact that $\tilde{\sigma}(a) < 1$ imply that in the first case $\tilde{\sigma}(a)$ is inside the interval and in the second case $\tilde{\sigma}(a)$ is outside of the interval. Therefore for $|a| > M$ we have

$$(2a)\tilde{\sigma}(a)^2 - (3a+2)\tilde{\sigma}(a) + (2+a) < 0$$

and the Hessian $\nabla^2 \mathrm{WA}_\gamma(\mathbf{x})$ is not positive semidefinite. □

Remark. Although the WA function is not convex, it violates the convexity conditions only slightly. However, by Theorem 4.7 the following inequality holds:

$$\max(\mathbf{x}) - \gamma W(\tfrac{n-1}{e}) \leq \mathrm{WA}_\gamma(\mathbf{x}) \leq \max(\mathbf{x})$$

Because the maximum function is convex, this shows that WA_γ is bounded by two convex functions that are only $\gamma W(\frac{n-1}{e})$ apart.

Alternatively, Figure 2.1 already shows that the NWA function (and therefore the WA function) is not convex: The graph converges towards HPWL for $x \to \infty$ forcing it to curve slightly to the right thus breaking the convexity condition.

4.2.5 Lipschitz Continuous Gradient

Although the WA function is used in conjunction with convex optimization methods (for example in [Lu+15]), there does not seem to be a rigorous proof prior to the following that the gradient of the function is Lipschitz continuous. We will use the same notation and results that were introduced in Section 4.1.5. Before we get to the main result of this section, we need a small lemma:

Lemma 4.11. *For all $1 \leq i \leq n$ we have*

$$\left|\sigma_i(\mathbf{x}/\gamma)\left(x_i/\gamma - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x})\right)\right| \leq W(\tfrac{n-1}{e})/\gamma.$$

Proof. Note that

$$\sum_{i=1}^{n} \sigma_i(\mathbf{x}/\gamma)\left(x_i/\gamma - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x})\right) = \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x}) - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x}) = 0$$

and by Theorem 4.7

$$\sigma_i(\mathbf{x}/\gamma)\left(x_i/\gamma - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x})\right) \leq \sigma_i(\mathbf{x}/\gamma)W(\tfrac{n-1}{e})/\gamma \leq W(\tfrac{n-1}{e})/\gamma$$

for every $1 \leq i \leq n$. These observations let us conclude that

$$\sigma_i(\mathbf{x}/\gamma)\left(x_i/\gamma - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x})\right) \geq -\sum_{k=1}^{n} \sigma_k(\mathbf{x}/\gamma)W(\tfrac{n-1}{e})/\gamma = -W(\tfrac{n-1}{e})/\gamma$$

Combining the last two inequalities proves the claim. □

Theorem 4.12. *$\nabla\,\mathrm{WA}_\gamma$ is Lipschitz continuous (with respect to the Euclidean norm).*

Proof. If we write the expression for the Hessian of WA_γ from Theorem 4.9 in a different way, we have

$$\begin{aligned}\nabla^2\,\mathrm{WA}_\gamma(\mathbf{x}) = \frac{2}{\gamma}\big(\mathrm{diag}(\sigma(\mathbf{x}/\gamma)) - \sigma(\mathbf{x}/\gamma)\sigma(\mathbf{x}/\gamma)^T\big) \\ + \frac{1}{\gamma}\,\mathrm{diag}\left(\sigma(\mathbf{x}/\gamma) \odot \left(\mathbf{x}/\gamma - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x}) \cdot \mathbf{1}\right)\right) \\ - \frac{1}{\gamma}\sigma(\mathbf{x}/\gamma)\sigma(\mathbf{x}/\gamma)^T \odot \left(\mathbf{1}\mathbf{x}^T/\gamma + \mathbf{x}\mathbf{1}^T/\gamma - \frac{2}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x}) \cdot \mathbf{1}\mathbf{1}^T\right)\end{aligned}$$

We will find an upper bound for the Frobenius norm of each of these terms.

Let us start with the first one. By Theorem 4.2 we know that it is twice the Hessian of LSE_γ and by Theorem 4.4 we know a bound for the Frobenius norm of $\nabla^2\,\mathrm{LSE}_\gamma(\mathbf{x})$. Combining both gives us

$$\left\|\frac{2}{\gamma}\big(\mathrm{diag}(\sigma(\mathbf{x}/\gamma)) - \sigma(\mathbf{x}/\gamma)\sigma(\mathbf{x}/\gamma)^T\big)\right\|_F = 2\|\nabla^2\,\mathrm{LSE}_\gamma(\mathbf{x})\|_F \leq \frac{1}{\gamma}$$

For the second and third term we use the previous lemma to see that

$$\left\|\frac{1}{\gamma}\,\mathrm{diag}\left(\sigma(\mathbf{x}/\gamma) \odot \left(\mathbf{x}/\gamma - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x}) \cdot \mathbf{1}\right)\right)\right\|_F$$

$$= \frac{1}{\gamma}\sqrt{\sum_{i=1}^{n}\left(\sigma_i(\mathbf{x}/\gamma)\left(x_i/\gamma - \frac{1}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\right)\right)^2}$$
$$\leq \frac{1}{\gamma}\sqrt{nW(\tfrac{n-1}{e})^2/\gamma^2} = \frac{1}{\gamma^2}\sqrt{n}\cdot W(\tfrac{n-1}{e})$$

and

$$\left\|\frac{1}{\gamma}\sigma(\mathbf{x}/\gamma)\sigma(\mathbf{x}/\gamma)^T \odot \left(\mathbf{1}\mathbf{x}^T/\gamma + \mathbf{x}\mathbf{1}^T/\gamma - \frac{2}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\cdot\mathbf{1}\mathbf{1}^T\right)\right\|_F$$
$$= \frac{1}{\gamma}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\sigma_i(\mathbf{x}/\gamma)\sigma_j(\mathbf{x}/\gamma)\left(x_i/\gamma + x_j/\gamma - \frac{2}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\right)\right)^2}$$
$$= \frac{1}{\gamma}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_i(\mathbf{x}/\gamma)^2\sigma_j(\mathbf{x}/\gamma)^2\left(\left|x_i/\gamma - \frac{1}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\right| + \left|x_j/\gamma - \frac{1}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\right|\right)^2}$$
$$\leq \frac{1}{\gamma}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_i(\mathbf{x}/\gamma)^2\sigma_j(\mathbf{x}/\gamma)^2\left(\left(x_i/\gamma - \frac{1}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\right)^2 + \left(x_j/\gamma - \frac{1}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\right)^2\right)}$$
$$= \frac{1}{\gamma}\sqrt{2\left(\sum_{i=1}^{n}\sigma_i(\mathbf{x}/\gamma)^2\right)\left(\sum_{j=1}^{n}\sigma_j(\mathbf{x}/\gamma)^2\left(x_j/\gamma - \frac{1}{\gamma}\operatorname{WA}_\gamma(\mathbf{x})\right)^2\right)}$$
$$\leq \frac{1}{\gamma}\sqrt{2\cdot 1\cdot nW(\tfrac{n-1}{e})^2/\gamma^2} = \frac{\sqrt{2}}{\gamma^2}\sqrt{n}\cdot W(\tfrac{n-1}{e})$$

Adding up all the upper bounds of the Frobenius norms gives us

$$\sup_{\mathbf{x}\in\mathbb{R}^n}\left\|\nabla^2\operatorname{WA}_\gamma(\mathbf{x})\right\|_2 \leq \sup_{\mathbf{x}\in\mathbb{R}^n}\left\|\nabla^2\operatorname{WA}_\gamma(\mathbf{x})\right\|_F \leq \frac{1}{\gamma} + \frac{1+\sqrt{2}}{\gamma^2}\sqrt{n}\cdot W(\tfrac{n-1}{e})$$

and this is an upper bound on the best global Lipschitz constant. □

4.3 Netlength-LogSumExp

Most of the results in this section are corollaries from the results in Section 4.1 intended to complete the picture and explicitly show the relevant properties of the objective function $\overline{\operatorname{NLSE}}_\gamma^N(\mathbf{x})$ used later on. The results about approximation ratios between NLSE and HPWL/STEINER aim to extend the results in [BV01] and were not previously considered in the literature.

4.3.1 Definition

Definition 4.4. Let $n \in \mathbb{N}_{\geq 1}$. The Netlength-LogSumExp function $\operatorname{NLSE}_\gamma\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\operatorname{NLSE}_\gamma(\mathbf{x},\mathbf{y}) := \operatorname{LSE}_\gamma(\mathbf{x}) + \operatorname{LSE}_\gamma(-\mathbf{x}) + \operatorname{LSE}_\gamma(\mathbf{y}) + \operatorname{LSE}_\gamma(-\mathbf{y})$$

where $\gamma > 0$ is a smoothing parameter as in Definition 4.1.

Remark. Correspondingly, $\overline{\mathrm{NLSE}}_\gamma(\mathbf{x}) = \mathrm{LSE}_\gamma(\mathbf{x}) + \mathrm{LSE}_\gamma(-\mathbf{x})$ and

$$\overline{\mathrm{NLSE}}_\gamma^N(\mathbf{x}) = \mathrm{LSE}_\gamma(A(N)\mathbf{x} + \mathbf{x}^P_{\mathrm{offs}}(N)).$$

4.3.2 Approximation Properties

Corollary 4.13. *The NLSE function is an approximation of the HPWL function in the following sense:*

$$\mathrm{HPWL}(\mathbf{x}, \mathbf{y}) \leq \mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y}) \leq \mathrm{HPWL}(\mathbf{x}, \mathbf{y}) + 4\gamma \log(n) \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

Proof. This follows directly from the definition of NLSE and Theorem 4.1. □

Theorem 4.14. *The approximation ratios between the NLSE and HPWL netlength estimations are*

$$\inf_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y})}{\mathrm{HPWL}(\mathbf{x}, \mathbf{y})} = 1 \quad \textit{and} \quad \sup_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y})}{\mathrm{HPWL}(\mathbf{x}, \mathbf{y})} = \infty$$

where $S = (\mathbb{R}^n \times \mathbb{R}^n) \setminus \{\, t \cdot (\mathbf{1}, \mathbf{1}) \mid t \in \mathbb{R} \,\}$ *is the space of all pin placements such that the netlength is not zero.*

Proof. Let $(\mathbf{x}, \mathbf{y}) \in S$. By Corollary 4.13, $\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y}) \geq \mathrm{HPWL}(\mathbf{x}, \mathbf{y})$ so the infimum is at least 1. To see that the infimum is exactly 1, consider

$$\sup_{\delta\to\infty} \frac{\mathrm{NLSE}_\gamma(\delta\mathbf{x}, \delta\mathbf{y})}{\mathrm{HPWL}(\delta\mathbf{x}, \delta\mathbf{y})} = \sup_{\delta\to\infty} \frac{\delta\, \mathrm{NLSE}_{\gamma/\delta}(\mathbf{x}, \mathbf{y})}{\delta\, \mathrm{HPWL}(\mathbf{x}, \mathbf{y})} = \sup_{\delta\to\infty} \frac{\mathrm{NLSE}_{\gamma/\delta}(\mathbf{x}, \mathbf{y})}{\mathrm{HPWL}(\mathbf{x}, \mathbf{y})} = 1$$

where $\mathrm{NLSE}_{\gamma/\delta}(\mathbf{x}, \mathbf{y}) \to \mathrm{HPWL}(\mathbf{x}, \mathbf{y})$ for $\delta \to \infty$ by Corollary 4.13.

To see that the ratio is unbounded from above, let $\mathbf{x}$ and $\mathbf{y}$ converge to zero. In this case, $\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y}) \to 4\gamma \log(n)$ and $\mathrm{HPWL}(\mathbf{x}, \mathbf{y}) \to 0$ so the ratio converges to infinity. □

Theorem 4.15. *The approximation ratios between the NLSE and STEINER netlength estimations are*

$$\inf_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y})}{\mathrm{STEINER}(\mathbf{x}, \mathbf{y})} = \inf_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{HPWL}(\mathbf{x}, \mathbf{y})}{\mathrm{STEINER}(\mathbf{x}, \mathbf{y})} \quad \textit{and}$$
$$\sup_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y})}{\mathrm{STEINER}(\mathbf{x}, \mathbf{y})} = \infty$$

where $S = (\mathbb{R}^n \times \mathbb{R}^n) \setminus \{\, t \cdot (\mathbf{1}, \mathbf{1}) \mid t \in \mathbb{R} \,\}$ *is the space of all pin placements such that the netlength is not zero.*

Proof. Let $(\mathbf{x}, \mathbf{y}) \in S$. By Corollary 4.13, $\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y}) \geq \mathrm{HPWL}(\mathbf{x}, \mathbf{y})$ and similarly to the proof above

$$\inf_{\delta \to \infty} \frac{\mathrm{NLSE}_\gamma(\delta\mathbf{x}, \delta\mathbf{y})}{\mathrm{STEINER}(\delta\mathbf{x}, \delta\mathbf{y})} = \inf_{\delta \to \infty} \frac{\mathrm{NLSE}_{\gamma/\delta}(\mathbf{x}, \mathbf{y})}{\mathrm{STEINER}(\mathbf{x}, \mathbf{y})} = \frac{\mathrm{HPWL}(\mathbf{x}, \mathbf{y})}{\mathrm{STEINER}(\mathbf{x}, \mathbf{y})}$$

for all $(\mathbf{x}, \mathbf{y}) \in S$.

The proof for the supremum is the same as before: Let $\mathbf{x}$ and $\mathbf{y}$ converge to zero. In this case $\mathrm{NLSE}_\gamma(\mathbf{x}, \mathbf{y}) \to 4\gamma \log(n)$ and $\mathrm{STEINER}(\mathbf{x}, \mathbf{y}) \to 0$ so the ratio converges to infinity. □

Remark. It was shown in [BV01] that

$$\inf_{(\mathbf{x},\mathbf{y}) \in S} \frac{\mathrm{HPWL}(\mathbf{x}, \mathbf{y})}{\mathrm{STEINER}(\mathbf{x}, \mathbf{y})} \geq \frac{4}{2\lceil \sqrt{n-2} \rceil + 3}$$

4.3.3 Derivatives

Lemma 4.16. *Let $f \colon \mathbb{R}^m \to \mathbb{R}$ be a twice differentiable function with gradient ∇f and Hessian $\nabla^2 f$, $A \in \mathbb{R}^{m \times n}$ a matrix and $\mathbf{b} \in \mathbb{R}^m$ a vector. Then the function $\tilde{f} \colon \mathbb{R}^n \to \mathbb{R}$ defined by $\tilde{f}(\mathbf{x}) = f(A\mathbf{x} + \mathbf{b})$ is twice differentiable and its derivatives are*

$$\nabla \tilde{f}(\mathbf{x}) = A^T(\nabla f(A\mathbf{x} + \mathbf{b}))$$
$$\nabla^2 \tilde{f}(\mathbf{x}) = A^T\big(\nabla^2 f(A\mathbf{x} + \mathbf{b})\big)A$$

Proof. This follows directly from the chain rule. □

Corollary 4.17. *$\overline{\mathrm{NLSE}}_\gamma^N$ is twice continuously differentiable and the gradient and Hessian are*

$$\nabla \overline{\mathrm{NLSE}}_\gamma^N(\mathbf{x}) = A(N)^T\big(\nabla \mathrm{LSE}_\gamma(\mathbf{x}^P(N)) - \nabla \mathrm{LSE}_\gamma(-\mathbf{x}^P(N))\big)$$
$$\nabla^2 \overline{\mathrm{NLSE}}_\gamma^N(\mathbf{x}) = A(N)^T\big(\nabla^2 \mathrm{LSE}_\gamma(\mathbf{x}^P(N)) + \nabla^2 \mathrm{LSE}_\gamma(-\mathbf{x}^P(N))\big)A(N)$$

for all $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{x}^P(N) = A(N)\mathbf{x} + \mathbf{x}^P_{\text{offs}}(N)$.

Proof. By the chain rule we have that

$$\nabla \overline{\mathrm{NLSE}}_\gamma(\mathbf{x}) = \nabla \mathrm{LSE}_\gamma(\mathbf{x}) - \nabla \mathrm{LSE}_\gamma(-\mathbf{x})$$
$$\nabla^2 \overline{\mathrm{NLSE}}_\gamma(\mathbf{x}) = \nabla^2 \mathrm{LSE}_\gamma(\mathbf{x}) + \nabla^2 \mathrm{LSE}_\gamma(-\mathbf{x}).$$

Combining this with Lemma 4.16 gives the claim. □

4.3.4 Convexity

Lemma 4.18 ([BV04, p. 79])**.** *Composition with an affine linear transformation preserves convexity: Let $f\colon \mathbb{R}^m \to \mathbb{R}$ be convex, $A \in \mathbb{R}^{m\times n}$ a matrix and $\mathbf{b} \in \mathbb{R}^m$ a vector. Then the function $\tilde{f}\colon \mathbb{R}^n \to \mathbb{R}$ defined by $\tilde{f}(\mathbf{x}) = f(A\mathbf{x}+\mathbf{b})$ is also convex.*

Proof. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ and $0 \le t \le 1$ then

$$\begin{aligned}\tilde{f}((1-t)\mathbf{x}_1 + t\mathbf{x}_2) &= f(A((1-t)\mathbf{x}_1 + t\mathbf{x}_2) + \mathbf{b}) \\ &= f((1-t)(A\mathbf{x}_1+\mathbf{b}) + t(A\mathbf{x}_2+\mathbf{b})) \\ &\le (1-t)f(A\mathbf{x}_1+\mathbf{b}) + tf(A\mathbf{x}_2+\mathbf{b}) \\ &= (1-t)\tilde{f}(\mathbf{x}_1) + t\tilde{f}(\mathbf{x}_2)\end{aligned}$$

which shows that $\tilde{f}$ is convex. □

Corollary 4.19. *$\overline{\mathrm{NLSE}}_\gamma^N$ is convex.*

Proof. By the chain rule and Theorem 4.3 we have that

$$\nabla^2 \overline{\mathrm{NLSE}}_\gamma(\mathbf{x}) = \nabla^2 \operatorname{LSE}_\gamma(\mathbf{x}) + \nabla^2 \operatorname{LSE}_\gamma(-\mathbf{x})$$

is positive semidefinite. This shows that $\overline{\mathrm{NLSE}}_\gamma$ is convex and by Lemma 4.18 $\overline{\mathrm{NLSE}}_\gamma^N$ is also convex for every net N. □

4.3.5 Lipschitz Continuous Gradient

Lemma 4.20. *Let $f\colon \mathbb{R}^m \to \mathbb{R}$ be a twice differentiable function with gradient ∇f and Hessian $\nabla^2 f$ such that the gradient is Lipschitz continuous. Let $A \in \mathbb{R}^{m\times n}$ be a matrix and $\mathbf{b} \in \mathbb{R}^m$ a vector. Then the function $\tilde{f}\colon \mathbb{R}^n \to \mathbb{R}$ defined by $\tilde{f}(\mathbf{x}) = f(A\mathbf{x}+\mathbf{b})$ also has a Lipschitz continuous gradient.*

Proof. As stated in Section 4.1.5, having a Lipschitz continuous gradient is equivalent to $\sup_{\mathbf{x}\in\mathbb{R}^n} \|\nabla^2 f(\mathbf{x})\|_2 \le L$ for some $L \in \mathbb{R}_{\ge 0}$. With the help of Lemma 4.16, it is easy to see that

$$\begin{aligned}\sup_{\mathbf{x}\in\mathbb{R}^n} \|\nabla^2 \tilde{f}(\mathbf{x})\|_2 &= \sup_{\mathbf{x}\in\mathbb{R}^n} \|A^T \nabla^2 f(A\mathbf{x}+\mathbf{b}) A\|_2 \\ &\le \sup_{\mathbf{x}\in\mathbb{R}^n} \|A^T\|_2 \|\nabla^2 f(A\mathbf{x}+\mathbf{b})\|_2 \|A\|_2 \\ &\le L\|A\|_2\|A^T\|_2 < \infty\end{aligned}$$

and this proves that $\nabla\tilde{f}$ is Lipschitz continuous. □

Corollary 4.21. *$\nabla\overline{\mathrm{NLSE}}_\gamma^N$ is Lipschitz continuous.*

Proof. $\nabla\overline{\mathrm{NLSE}}_\gamma$ is Lipschitz continuous by Theorem 4.4:

$$\begin{aligned}\sup_{\mathbf{x}\in\mathbb{R}^n}\|\nabla^2\overline{\mathrm{NLSE}}_\gamma(\mathbf{x})\|_2 &= \sup_{\mathbf{x}\in\mathbb{R}^n}\|\nabla^2\,\mathrm{LSE}_\gamma(\mathbf{x})+\nabla^2\,\mathrm{LSE}_\gamma(-\mathbf{x})\|_2\\ &\le \sup_{\mathbf{x}\in\mathbb{R}^n}\|\nabla^2\,\mathrm{LSE}_\gamma(\mathbf{x})\|_2+\sup_{\mathbf{x}\in\mathbb{R}^n}\|\nabla^2\,\mathrm{LSE}_\gamma(-\mathbf{x})\|_2\\ &= \frac{1}{2\gamma}+\frac{1}{2\gamma}=\frac{1}{\gamma}\end{aligned}$$

Combining this with Lemma 4.20 gives the claim. □

Remark. By definition $A(N)$ contains exactly $|N|$ entries of value 1, so $\|A(N)\|_F = \sqrt{|N|}$ and with the previous calculations we get the following upper bound for the Lipschitz constant of $\nabla\overline{\mathrm{NLSE}}_\gamma^N$:

$$\sup_{\mathbf{x}\in\mathbb{R}^n}\|\nabla^2\overline{\mathrm{NLSE}}_\gamma^N(\mathbf{x})\|_2 \le \frac{1}{\gamma}\|A(N)\|_2\|A(N)^T\|_2 \le \frac{1}{\gamma}\|A(N)\|_F\|A(N)^T\|_F \le \frac{|N|}{\gamma}$$

4.4 Netlength-Weighted-Average

Most of the results in this section are corollaries from the results in Section 4.2 intended to complete the picture and explicitly show the relevant properties for the objective function $\overline{\mathrm{NWA}}_\gamma^N(\mathbf{x})$ used later on. The results about approximation ratios between NWA and HPWL/STEINER aim to extend the results in [BV01] and were not previously considered in the literature.

4.4.1 Definition

Definition 4.5. Let $n \in \mathbb{N}_{\ge 1}$. The Netlength-Weighted-Average function $\mathrm{NWA}_\gamma\colon \mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}$ is defined as

$$\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y}) := \mathrm{WA}_\gamma(\mathbf{x})+\mathrm{WA}_\gamma(-\mathbf{x})+\mathrm{WA}_\gamma(\mathbf{y})+\mathrm{WA}_\gamma(-\mathbf{y})$$

where $\gamma>0$ is a smoothing parameter as in Definition 4.3.

Remark. Correspondingly, $\overline{\mathrm{NWA}}_\gamma(\mathbf{x}) = \mathrm{WA}_\gamma(\mathbf{x})+\mathrm{WA}_\gamma(-\mathbf{x})$ and

$$\overline{\mathrm{NWA}}_\gamma^N(\mathbf{x}) = \mathrm{WA}_\gamma(A(N)\mathbf{x}+\mathbf{x}^P_{\mathrm{offs}}(N))$$

4.4.2 Approximation Properties

Corollary 4.22. *The NWA function is an approximation of the HPWL function in the following sense*

$$\mathrm{HPWL}(\mathbf{x},\mathbf{y})-4\gamma W(\tfrac{n-1}{e}) \le \mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y}) \le \mathrm{HPWL}(\mathbf{x},\mathbf{y}) \quad \forall\mathbf{x}\in\mathbb{R}^n$$

where W is the Lambert W function.

Proof. This follows directly from the definition of NWA and Theorem 4.7. □

Corollary 4.23. *The NWA function approximates HPWL better than the NLSE function in the sense that the maximum absolute error is lower, i.e.*

$$\sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^n} |\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y}) - \mathrm{HPWL}(\mathbf{x},\mathbf{y})| < \sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^n} |\mathrm{NLSE}_\gamma(\mathbf{x},\mathbf{y}) - \mathrm{HPWL}(\mathbf{x},\mathbf{y})|$$

for all $\gamma > 0$ *and* $n \in \mathbb{N}_{\geq 2}$.

Proof. By Corollary 4.13 and the fact that $\mathrm{NLSE}_\gamma(\mathbf{0},\mathbf{0}) = 4\gamma\log(n)$, we get that

$$\sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^n} |\mathrm{NLSE}_\gamma(\mathbf{x},\mathbf{y}) - \mathrm{HPWL}(\mathbf{x},\mathbf{y})| = 4\gamma\log(n)$$

and since $W(\frac{n-1}{e}) < \log(n)$ for $n \geq 2$ by Theorem 4.8 we have that

$$\sup_{\mathbf{x},\mathbf{y}\in\mathbb{R}^n} |\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y}) - \mathrm{HPWL}(\mathbf{x},\mathbf{y})| \leq 4\gamma W(\tfrac{n-1}{e}) < 4\gamma\log(n)$$

which proves the claim. □

Theorem 4.24. *The approximation ratios between the NWA and HPWL netlength estimations are*

$$\inf_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y})}{\mathrm{HPWL}(\mathbf{x},\mathbf{y})} = 0 \quad \textit{and} \quad \sup_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y})}{\mathrm{HPWL}(\mathbf{x},\mathbf{y})} = 1$$

where $S = (\mathbb{R}^n \times \mathbb{R}^n) \setminus \{\, t\cdot(\mathbf{1},\mathbf{1}) \mid t \in \mathbb{R} \,\}$ *is the space of all pin placements such that the netlength is not zero.*

Proof. Let $(\mathbf{x},\mathbf{y}) \in S$. By Corollary 4.22, $\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y}) \leq \mathrm{HPWL}(\mathbf{x},\mathbf{y})$ so that the supremum is at most 1. To see that the supremum is exactly 1, consider

$$\sup_{\delta\to\infty} \frac{\mathrm{NWA}_\gamma(\delta\mathbf{x},\delta\mathbf{y})}{\mathrm{HPWL}(\delta\mathbf{x},\delta\mathbf{y})} = \sup_{\delta\to\infty} \frac{\delta\,\mathrm{NWA}_{\gamma/\delta}(\mathbf{x},\mathbf{y})}{\delta\,\mathrm{HPWL}(\mathbf{x},\mathbf{y})} = \sup_{\delta\to\infty} \frac{\mathrm{NWA}_{\gamma/\delta}(\mathbf{x},\mathbf{y})}{\mathrm{HPWL}(\mathbf{x},\mathbf{y})} = 1$$

where $\mathrm{NWA}_{\gamma/\delta}(\mathbf{x},\mathbf{y}) \to \mathrm{HPWL}(\mathbf{x},\mathbf{y})$ for $\delta\to\infty$ by Corollary 4.22.

To see that the ratio can get arbitrarily small, let $\mathbf{x} = (a,0,\ldots,0)^T$ and $\mathbf{y} = \mathbf{0}$ for some $a > 0$. In this case,

$$\begin{aligned}\lim_{a\to 0} \frac{\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y})}{\mathrm{HPWL}(\mathbf{x},\mathbf{y})} &= \lim_{a\to 0} \frac{\frac{a\exp(a)}{\exp(a)+(n-1)} + \frac{-a\exp(-a)}{\exp(-a)+(n-1)}}{a} \\ &= \lim_{a\to 0} \frac{\exp(a)}{\exp(a)+(n-1)} - \frac{\exp(-a)}{\exp(-a)+(n-1)} = 0\end{aligned}$$

which shows that the infimum is zero. □

Theorem 4.25. *The approximation ratios between the NWA and STEINER netlength estimations are*

$$\inf_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y})}{\mathrm{STEINER}(\mathbf{x},\mathbf{y})} = 0 \quad \textit{and}$$
$$\sup_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y})}{\mathrm{STEINER}(\mathbf{x},\mathbf{y})} = \sup_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{HPWL}(\mathbf{x},\mathbf{y})}{\mathrm{STEINER}(\mathbf{x},\mathbf{y})}$$

where $S = (\mathbb{R}^n \times \mathbb{R}^n) \setminus \{\, t \cdot (\mathbf{1},\mathbf{1}) \mid t \in \mathbb{R} \,\}$ *is the space of all pin placements such that the netlength is not zero.*

Proof. Let $(\mathbf{x},\mathbf{y}) \in S$. By Corollary 4.22 $\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y}) \le \mathrm{HPWL}(\mathbf{x},\mathbf{y})$ and similarly to the proof above

$$\sup_{\delta\to\infty} \frac{\mathrm{NWA}_\gamma(\delta\mathbf{x},\delta\mathbf{y})}{\mathrm{STEINER}(\delta\mathbf{x},\delta\mathbf{y})} = \sup_{\delta\to\infty} \frac{\mathrm{NWA}_{\gamma/\delta}(\mathbf{x},\mathbf{y})}{\mathrm{STEINER}(\mathbf{x},\mathbf{y})} = \frac{\mathrm{HPWL}(\mathbf{x},\mathbf{y})}{\mathrm{STEINER}(\mathbf{x},\mathbf{y})}$$

for all $(\mathbf{x},\mathbf{y}) \in S$. This proofs the claim regarding the supremum.

The proof for the infimum is the same as before: Let $\mathbf{x} = (a,0,\ldots,0)^T$ and $\mathbf{y} = \mathbf{0}$ for some $a > 0$. In this case,

$$\begin{aligned}\lim_{a\to 0} \frac{\mathrm{NWA}_\gamma(\mathbf{x},\mathbf{y})}{\mathrm{STEINER}(\mathbf{x},\mathbf{y})} &= \lim_{a\to 0} \frac{\frac{a\exp(a)}{\exp(a)+(n-1)} + \frac{-a\exp(-a)}{\exp(-a)+(n-1)}}{a} \\ &= \lim_{a\to 0} \frac{\exp(a)}{\exp(a)+(n-1)} - \frac{\exp(-a)}{\exp(-a)+(n-1)} = 0.\end{aligned}$$

which shows that the infimum is zero. □

Remark. It was shown in [BV01] that

$$\sup_{(\mathbf{x},\mathbf{y})\in S} \frac{\mathrm{HPWL}(\mathbf{x},\mathbf{y})}{\mathrm{STEINER}(\mathbf{x},\mathbf{y})} = 1$$

4.4.3 Derivatives

Corollary 4.26. $\overline{\mathrm{NWA}}_\gamma^N$ *is twice continuously differentiable and the gradient and Hessian are*

$$\nabla\overline{\mathrm{NWA}}_\gamma^N(\mathbf{x}) = A(N)^T\big(\nabla\,\mathrm{WA}_\gamma(\mathbf{x}^P(N)) - \nabla\,\mathrm{WA}_\gamma(-\mathbf{x}^P(N))\big)$$
$$\nabla^2\overline{\mathrm{NWA}}_\gamma^N(\mathbf{x}) = A(N)^T\big(\nabla^2\,\mathrm{WA}_\gamma(\mathbf{x}^P(N)) + \nabla^2\,\mathrm{WA}_\gamma(-\mathbf{x}^P(N))\big)A(N)$$

for all $\mathbf{x} \in \mathbb{R}^n$ *where* $\mathbf{x}^P(N) = A(N)\mathbf{x} + \mathbf{x}^P_{\mathrm{offs}}(N)$.

Proof. By the chain rule,

$$\nabla\overline{\mathrm{NWA}}_\gamma(\mathbf{x}) = \nabla\,\mathrm{NWA}_\gamma(\mathbf{x}) - \nabla\,\mathrm{NWA}_\gamma(-\mathbf{x})$$
$$\nabla^2\overline{\mathrm{NWA}}_\gamma(\mathbf{x}) = \nabla^2\,\mathrm{NWA}_\gamma(\mathbf{x}) + \nabla^2\,\mathrm{NWA}_\gamma(-\mathbf{x}).$$

Combining this with Lemma 4.16 gives the claim. □

4.4.4 Convexity

The following convexity result shows that contrary to [HCB11] not only the WA function is not convex but NWA as well.

Corollary 4.27. *$\overline{\mathrm{NWA}}_\gamma$ is not convex for any $n \in \mathbb{N}_{\geq 2}$.*

Proof. Pick the same counterexample as in Theorem 4.10: $\mathbf{x} = (\gamma a, 0, \ldots, 0)^T \in \mathbb{R}^n$. From the proof of this theorem we know that there is an $M > 2$ such that

$$\mathbf{e}_1^T \big(\nabla^2 \mathrm{WA}_\gamma(\mathbf{x})\big)\mathbf{e}_1 < 0$$

for all $a \in \mathbb{R}$ with $|a| > M$. This implies that $\mathbf{e}_1^T(\nabla^2 \mathrm{WA}_\gamma(\mathbf{x}))\mathbf{e}_1$ and $\mathbf{e}_1^T(\nabla^2 \mathrm{WA}_\gamma(-\mathbf{x}))\mathbf{e}_1$ are negative for such an a. Therefore,

$$\mathbf{e}_1^T \big(\nabla^2 \overline{\mathrm{NWA}}_\gamma(\mathbf{x})\big)\mathbf{e}_1 = \mathbf{e}_1^T \big(\nabla^2 \mathrm{WA}_\gamma(\mathbf{x}) + \nabla^2 \mathrm{WA}_\gamma(-\mathbf{x})\big)\mathbf{e}_1 < 0$$

so the Hessian $\nabla^2 \overline{\mathrm{NWA}}_\gamma(\mathbf{x})$ is not positive semidefinite for all $\mathbf{x} \in \mathbb{R}^n$. □

Remark. This result cannot be extended to $\overline{\mathrm{NWA}}_\gamma^N$ because with our mathematical description from Section 2.2 a net is allowed to contain only fixed pins. In this case, $\overline{\mathrm{NWA}}_\gamma^N$ would be constant and therefore convex.

Corollary 4.28. *$\overline{\mathrm{NWA}}_\gamma^N$ is bounded by two convex functions:*

$$\overline{\mathrm{HPWL}}^N(\mathbf{x}) - 2\gamma W(\tfrac{n-1}{e}) \leq \overline{\mathrm{NWA}}_\gamma^N(\mathbf{x}) \leq \overline{\mathrm{HPWL}}^N(\mathbf{x}).$$

Proof. $\overline{\mathrm{HPWL}}$ is convex so Lemma 4.18 shows that $\overline{\mathrm{HPWL}}^N(\mathbf{x})$ is convex. The inequalities follow from Theorem 4.7. □

4.4.5 Lipschitz Continuous Gradient

Corollary 4.29. *$\nabla\overline{\mathrm{NWA}}_\gamma^N$ is Lipschitz continuous.*

Proof. $\nabla\overline{\mathrm{NWA}}_\gamma$ is Lipschitz continuous by Theorem 4.12:

$$\begin{aligned}
\sup_{\mathbf{x}\in\mathbb{R}^n} \|\nabla^2 \overline{\mathrm{NWA}}_\gamma(\mathbf{x})\|_2 &= \sup_{\mathbf{x}\in\mathbb{R}^n} \|\nabla^2 \mathrm{WA}_\gamma(\mathbf{x}) + \nabla^2 \mathrm{WA}_\gamma(-\mathbf{x})\|_2 \\
&\leq \sup_{\mathbf{x}\in\mathbb{R}^n} \|\nabla^2 \mathrm{WA}_\gamma(\mathbf{x})\|_2 + \sup_{\mathbf{x}\in\mathbb{R}^n} \|\nabla^2 \mathrm{WA}_\gamma(-\mathbf{x})\|_2 \\
&= \frac{2}{\gamma} + \frac{2+2\sqrt{2}}{\gamma^2}\sqrt{n} \cdot W(\tfrac{n-1}{e}).
\end{aligned}$$

Combining this with Lemma 4.20 gives the claim. □

Remark. Analogous to the remark after Corollary 4.21, we get the following upper bound for the Lipschitz constant of $\nabla\overline{\mathrm{NWA}}_\gamma^N$:

$$\sup_{\mathbf{x}\in\mathbb{R}^n} \|\nabla^2 \overline{\mathrm{NWA}}_\gamma^N(\mathbf{x})\|_2 \leq |N| \left(\frac{2}{\gamma} + \frac{2+2\sqrt{2}}{\gamma^2}\sqrt{n} \cdot W(\tfrac{n-1}{e}) \right).$$

5 Implementation Details

Before we come to the results of minimizing NLSE and NWA on real chips, we need to explain the details of our implementation of the theory that was discussed before.

5.1 Evaluating LSE, WA and Their Derivatives

In this section, the details about how the calculation of the objective function and the gradient was implemented for LSE and WA are discussed. We mostly followed the mathematical description but added a value shift that leaves the result unchanged but avoids overflow errors.

5.1.1 Shifting LSE, WA and Their Derivatives

To start off, it is important to see how shifting all elements in a vector $\mathbf{x}$ by the same amount c affects the output of the two maximum approximations LSE and WA and their derivatives. It turns out that a simultaneous shift in the input is equivalent to a shift in the output for LSE and WA and that such a shift does not alter the value of the derivatives at all.

Lemma 5.1. *Let $\mathbf{x} \in \mathbb{R}^n$ and $c \in \mathbb{R}$ then*

$$\mathrm{LSE}_\gamma(\mathbf{x} + c \cdot \mathbf{1}) = \mathrm{LSE}_\gamma(\mathbf{x}) + c.$$

Proof. By the defintion of LSE we have that

$$\begin{aligned}
\mathrm{LSE}_\gamma(\mathbf{x} + c \cdot \mathbf{1}) &= \gamma \log\left(\sum_{k=1}^{n} \exp((x_k + c)/\gamma)\right) = \gamma \log\left(\sum_{k=1}^{n} \exp(x_k/\gamma)\exp(c/\gamma)\right) \\
&= \gamma \log\left(\sum_{k=1}^{n} \exp(x_k/\gamma)\right) + \gamma \log(\exp(c/\gamma)) \\
&= \mathrm{LSE}_\gamma(\mathbf{x}) + c
\end{aligned}$$

which proves the claim. □

Corollary 5.2. *Let $\mathbf{x} \in \mathbb{R}^n$ and $c \in \mathbb{R}$, then*

$$\sigma((\mathbf{x} + c \cdot \mathbf{1})/\gamma) = \nabla \mathrm{LSE}_\gamma(\mathbf{x} + c \cdot \mathbf{1}) = \nabla \mathrm{LSE}_\gamma(\mathbf{x}) = \sigma(\mathbf{x}/\gamma).$$

Proof. The first and the last equation follow from Theorem 4.2 and the remaining equation follows from the lemma above by taking derivatives on both sides. □

Lemma 5.3. *Let* $\mathbf{x} \in \mathbb{R}^n$ *and* $c \in \mathbb{R}$ *then*

$$\mathrm{WA}_\gamma(\mathbf{x} + c \cdot \mathbf{1}) = \mathrm{WA}_\gamma(\mathbf{x}) + c.$$

Proof. Plugging $\mathbf{x} + c \cdot \mathbf{1}$ into the definition of WA gives us

$$\begin{aligned}\mathrm{WA}_\gamma(\mathbf{x} + c \cdot \mathbf{1}) &= \sigma((\mathbf{x} + c \cdot \mathbf{1})/\gamma)^T(\mathbf{x} + c \cdot \mathbf{1}) \\ &= \sigma(\mathbf{x}/\gamma)^T\mathbf{x} + c \cdot \sigma(\mathbf{x}/\gamma)^T\mathbf{1} \\ &= \sigma(\mathbf{x}/\gamma)^T\mathbf{x} + c\end{aligned}$$

where we used Corollary 5.2 and the fact that the softmax function maps every value to a stochastic vector. □

Corollary 5.4. *Let* $\mathbf{x} \in \mathbb{R}^n$ *and* $c \in \mathbb{R}$ *then*

$$\nabla \mathrm{WA}_\gamma(\mathbf{x} + c \cdot \mathbf{1}) = \nabla \mathrm{WA}_\gamma(\mathbf{x}).$$

Proof. The equation follows from Lemma 5.3 by taking derivatives on both sides. □

5.1.2 Solving Overflow Errors

An overflow error occurs if the result of a computation exceeds the maximal value that can be stored in the chosen numerical datatype. We use the datatype `double` which uses fast floating-point arithmetic and provides the necessary precision for our purposes. By the IEEE 754 Standard for Floating-Point Arithmetic, the maximal value a number of type `double` can store is roughly $2^{1024} \approx e^{710}$ while the smallest positive number is $2^{-1074} \approx e^{-745}$. When evaluating LSE_γ, $\nabla \mathrm{LSE}_\gamma$, WA_γ or $\nabla \mathrm{WA}_\gamma$, the exponential function has to be calculated for inputs of the form x_i/γ and this can fail once x_i/γ is greater than 710. In this case, the result of $\exp(x_i/\gamma)$ is the special `double` value that represents infinity and none of the subsequent calculations work correctly anymore.

One way to tackle the problem would be to increase γ (or equivalently downscale the problem instance) such that $x_{\max}/\gamma$ is small enough, but this comes at the cost of worsening the approximation properties of NLSE and NWA. A second approach would be to modify the problem instance by shifting all positions and the chip area simultaneously such that all position values are sufficiently small. The problem with this approach is that now there might be underflow errors: Imagine a net where all pin positions are so far to the left that after dividing by γ they are all smaller than -745. Evaluating the LSE function with `double` precision will round all $\exp(x_i/\gamma)$ down to zero and then returns $\log(0)$ which is $-\infty$. This result is completely unusable and again breaks all subsequent computations.

The best strategy is to shift all pin positions in a net for each evaluation of one of the functions from the theorems above and then modify the result accordingly. This approach is already in use for the LogSumExp function and is known as

the "LogSumExp trick" in the Machine Learning community (see for example [DPM18, p. 6] and [Coo11, p. 3]). It is standard to shift using $c = -\max(\mathbf{x})$ so that afterwards the maximal x_i is zero. This avoids overflow errors (because $\exp((x_i - c)/\gamma) \leq \exp(0) = 1$) and only produces underflow errors that influence the results very little. Consider for example an evaluation of the LSE function for a net with n pins: The absolute error that occurs when all evaluations of the exponential function except for $\exp(0)$ are rounded down to zero due to underflow errors is at most

$$\log(1 + (n-1)\exp(-745)) - \log(1) \leq (n-1)\exp(-745).$$

Even if the net contained 10,000 pins (which unreasonably large for any practical instance), the error would still be insignificant. Similar observations can be made for the other functions: Once a pin position is so far away from the maximum to produce an underflow error, it hardly influences the result. Of course, this shift requires to calculate the maximum of n values for each use of one of the functions but because this can be done in $O(n)$ it only increases the runtime by a constant factor. In summary, using this shifting technique for each evaluation of LSE_γ, $\nabla\,\mathrm{LSE}_\gamma$, WA_γ and $\nabla\,\mathrm{WA}_\gamma$ is an effective way to avoid significant numerical errors.

5.1.3 Theoretical Runtimes

It might seem like the definitions of $\overline{\mathrm{NLSE}}_\gamma^N$ and $\overline{\mathrm{NWA}}_\gamma^N$ as well as Corollary 4.17 and Corollary 4.26 show that a matrix vector multiplication is needed to evaluate these functions and their derivatives but this is not the case. The introduction of the matrices $A(N)$ was done to abbreviate some statements and definitions and was also useful when proving convexity and Lipschitz continuity. To see what the runtime of the objective functions looks like, we need to first confirm that all of the component functions have linear runtime. This can be checked directly by looking at their definitions:

$$\begin{aligned}
\mathrm{LSE}_\gamma(\mathbf{x}) &= \gamma \log\left(\sum_{k=1}^{n} \exp(x_k/\gamma)\right) \\
\nabla\,\mathrm{LSE}_\gamma(\mathbf{x}) &= \sigma(\mathbf{x}/\gamma) = \frac{1}{\sum_{k=1}^{n}\exp(x_k)}(\exp(x_1), \ldots, \exp(x_n))^T \\
\mathrm{WA}_\gamma(\mathbf{x}) &= \sigma(\mathbf{x}/\gamma)^T\mathbf{x} \\
\nabla\,\mathrm{WA}_\gamma(\mathbf{x}) &= \sigma(\mathbf{x}/\gamma) \odot \left(\mathbf{1} + \mathbf{x}/\gamma - \frac{1}{\gamma}\,\mathrm{WA}_\gamma(\mathbf{x}) \cdot \mathbf{1}\right).
\end{aligned}$$

Calculating the pin positions from the cell positions which was often abbreviated by $\mathbf{x}^P(N) = A(N)\mathbf{x} + \mathbf{x}^P_{\mathrm{offs}}(N)$ in the previous section is straightforward: Loop through all pins $p \in N$ and determine each pin position as the position of the cell $\alpha(p)$ plus the pin offset. This can be done in $O(|N|)$ without any matrix vector

multiplication. Combining these results shows that evaluating one of the functions above with the pin positions as parameter can be done in $O(|N|)$.

Remember that T_{NLSE_γ} and T_{NWA_γ} are the weighted sums of the netlength estimations and serve as objective functions in our case. Because we split up the work in one problem for the x-dimension and one for the y-dimension, it suffices to look at their one-dimensional counterparts which we will denote by $T_{\overline{\mathrm{NLSE}}_\gamma}$ and $T_{\overline{\mathrm{NWA}}_\gamma}$ for the x-dimension (y-coordinates work analogously). What is the theoretical runtime of evaluating one of these objective functions or their gradients? Lets look at the formulas again:

$$
\begin{aligned}
T_{\overline{\mathrm{NLSE}}_\gamma}(\mathbf{x}) &= \sum_{N\in\mathcal{N}} w(N) \sum_{s\in\{-1,1\}} \mathrm{LSE}_\gamma(s\cdot \mathbf{x}^P(N)) \\
\nabla T_{\overline{\mathrm{NLSE}}_\gamma}(\mathbf{x})_c &= \sum_{N\in\mathcal{N}} w(N) \sum_{s\in\{-1,1\}} \sum_{\substack{p\in N\\ \alpha(p)=c}} s\cdot \nabla \mathrm{LSE}_\gamma(s\cdot \mathbf{x}^P(N))_p \\
T_{\overline{\mathrm{NWA}}_\gamma}(\mathbf{x}) &= \sum_{N\in\mathcal{N}} w(N) \sum_{s\in\{-1,1\}} \mathrm{WA}_\gamma(s\cdot \mathbf{x}^P(N)) \\
\nabla T_{\overline{\mathrm{NWA}}_\gamma}(\mathbf{x})_c &= \sum_{N\in\mathcal{N}} w(N) \sum_{s\in\{-1,1\}} \sum_{\substack{p\in N\\ \alpha(p)=c}} s\cdot \nabla \mathrm{WA}_\gamma(s\cdot \mathbf{x}^P(N))_p
\end{aligned}
$$

The s variable was added to abbreviate the two summands that are part of NLSE_γ and NWA_γ. The formulas for the gradients are given as formulas that calculate the gradient for a single cell c. In the implementation one loops over the outer two sums, calculates the pin positions and the gradient of LSE_γ or WA_γ, then loops over all pins in the net and adds the entry of pin p to the entry of cell $\alpha(p)$ if the pin is not fixed. This gives a runtime of $O(\sum_{N\in\mathcal{N}}|N|) = O(|P|)$ for all of the objective functions and their gradients.

5.2 Placing Cells Inside the Chip Area

As mentioned in Section 2.2, it basically suffices to solve an unconstrained optimization problem by dropping the requirement that the cells (or at least their anchor points) stay inside the chip area because any reasonable placement will not place cells outside of the chip area anyways. This knowlegde about how reasonable placements should look like leads to the following improvement of the convex optimization methods. In each iteration, the anchor points (i.e. the midpoints) of all movable cells are projected to the nearest position inside the chip area. In other words

$$
x_c^{(k)} = \begin{cases} x_{\max} & \text{if } x_c^{(k)} > x_{\max} \\ x_{\min} & \text{if } x_c^{(k)} < x_{\min} \\ x_c^{(k)} & \text{otherwise} \end{cases} \quad \text{and} \quad y_c^{(k)} = \begin{cases} y_{\max} & \text{if } y_c^{(k)} > y_{\max} \\ y_{\min} & \text{if } y_c^{(k)} < y_{\min} \\ y_c^{(k)} & \text{otherwise} \end{cases}
$$

for every movable cell $c \in C$ and after every iteration k. This operation is not expensive as it can be combined with the gradient step which also requires to update every cell position. At the same time, it can improve intermediate placements where a few cells are placed outside of the chip "by accident". If these cells were not projected inside the chip area, they would additionally draw other cells towards them so this technique also helps to find good placements faster. Its influence will not be very significant but because it is expected to increase the solution quality with no relevant disadvantages, it is used for all of the results presented in the next chapter.

5.3 The Parameters of the Optimization Process

This section serves as a summary of all the parameters that can be chosen independently from each other when optimizing the total wirelength using either NLSE or NWA as the netlength estimation.

1. The netlength estimation: NLSE or NWA
   a) The value of γ
2. The optimization method: GD or NAG
   a) The start vector $\mathbf{x}^{(0)}$
   b) The first step size h_0
   c) The step size strategy
   d) The precision ε used for the stopping criterion

5.3.1 The Gamma Value

We will try different values for the smoothing parameter γ for both objective functions. The objective functions for x- and y-coordinates will always have the same γ value. To cover a wide range of values, we chose

$$\gamma \in \{10^1, 10^3, 10^5\}$$

where $\gamma = 10$ is extremely small and $\gamma = 10^5$ relatively large. The x- and y-values in the experiments in the following section are roughly between zero and 10^6.

5.3.2 The Start Vector

The two start vectors (for x- and y-direction) for the gradient descent method or Nesterov's accelerated gradient method correspond to an initial placement of all movable cells. It is hard to find an initial placement that is close to the optimal placement when the optimal placement completely unknow. The easiest option is to just select a random placement from the set of all possible placements. We

chose to use a uniform distribution over the chip area for each cell which is the same as a uniform distribution of x-values over the width of the chip and a uniform distribution of y-values over the height of the chip. To be able to compare how the optimization of different objective functions looks like with this start vector, the two different seed values for the pseudo-random number generators for x- and y-coordinates were fixed for each chip.

The second option is to let all cells start at the same place. This is also a simple rule that does not require special knowlegde about the cells. With the (resonable) assumption that the average position of all cells in an optimal placement will be roughly at the center of the chip, the best start vector following this rule is the one that places all cells at the center of the chip:

$$\mathbf{x}^{(0)} = \frac{x_{\min} + x_{\max}}{2} \cdot \mathbf{1} \quad \text{and} \quad \mathbf{y}^{(0)} = \frac{y_{\min} + y_{\max}}{2} \cdot \mathbf{1}.$$

Another two options are to use the optimal placements with respect to HPWL and QCLIQUE as starting positions. Of course, this can increase the runtime considerably but it is of theoretical interest to see how these starting positions are changed by NLSE or NWA. Additionally, because they were found by using information about the cells, pins and nets, they are expected to be good starting points for netlength minimization.

The start values that we will consider are therefore

- RANDOM: Uniformly pseudo-random cell positions with fixed seeds
- CENTER: The placement in which all movable cells are at the center of the chip
- HPWL-OPT: The optimal placement when minimizing T_{HPWL}
- QCLIQUE-OPT: The optimal placement when minimizing T_{QCLIQUE}

5.3.3 The First Step Size

For both convex optimization methods, i.e. gradient descent and Nesterov's accelerated gradient, the very first step from $\mathbf{x}^{(0)}$ to $\mathbf{x}^{(1)}$ is just a gradient step but, as pointed out in Section 3.1.2, the first step size has to be specified. To avoid hardcoding the first step size and being much too low or much too high, we chose to start at

$$h_0 = 10000 \cdot \frac{(x_{\max} - x_{\min})}{|\mathcal{N}|}$$

which should be too high in most cases and then use `backtracking-line-search` to decrease it to a reasonable size.

5.3.4 The Step Size Strategy

The details can be found in Section 3.1.2, this is the list of step size strategies:

- `constant`: $h_k = h_0$
- `scaled-by-sqrt`: $h_k = \frac{h_0}{\sqrt{k+1}}$
- `backtracking-line-search`: $h_k = 2^{-n} h_{k-1}$ with minimal n such that

$$f(\mathbf{x}^{(k)} - h_k \nabla f(\mathbf{x}^{(k)})) \leq f(\mathbf{x}^{(k)}) - \frac{1}{2} h_k \left\| \nabla f(\mathbf{x}^{(k)}) \right\|_2^2$$

- `observed-lipschitz`: $h_k = \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2}{\|\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)})\|_2}$
- `min-observed-lipschitz`: $h_k = \min\left\{ h_{k-1}, \frac{\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2}{\|\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)})\|_2} \right\}$
- `barzilai-borwein`: $h_k = \frac{\left| \left(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right)^T \left(\nabla f(\mathbf{x}^{(k)}) - \nabla f(\mathbf{x}^{(k-1)})\right) \right|}{\|\nabla f(\mathbf{x}^{(k-1)}) - \nabla f(\mathbf{x}^{(k)})\|_2^2}$

5.3.5 The Precision of the Stopping Criterion

For the criterion $\|\nabla f(\mathbf{x}^{(k)})\|_2 < \varepsilon$ introduced in Section 3.1.3 to work correctly it is important that $\|\nabla f(\mathbf{x}^{(k)})\|_2$ converges to zero, so that any choice of ε is a tradeoff between precision and runtime. It is unclear whether the gradient norm will really converge to zero nicely and for most of the comparisons in Chapter 6 runtime is secondary, so it is easier to fix the number of iterations and graph the objective functions iteration by iteration. One can then see how many iterations are reasonable to reach a point where the objective function does not decrease significantly anymore. The difficulty of finding reasonable values for ε that do not need to be fine-tuned for every instance is discussed in Section 6.5.

6 Results

In this chapter, we will use different netlength estimations (NLSE and NWA), optimization methods and parameters to minimize the netlength of multiple chips to see how they perform. This chapter focuses on comparisons between different configurations and will tackle the following questions:

- Are the chosen optimization methods able to produce reasonable results?
- How closely do NLSE and NWA approximate HPWL during the optimization process?
- How do different starting positions influence the results?
- How do different step size strategies influence results and runtime?
- How do the generated placements look like?

6.1 Goal

It is important to understand the goal of investigating the use NLSE and NWA during the very first step of global placement. The goal of the very first step of global placement itself was already stated in Section 2.1: Gain insights about the relative positions of the cells in an optimal solution of the Placement task. Note that all placements with low wirelength (that do not need to obey disjointness constraints) will have lots of overlapping cells and that the more clumped together the cells are the fewer clear insights about relative positions can be gained. Therefore, a balance has to be found between how spread out the cells are and how low the wirelength is.

When using QCLIQUE as the netlength estimation, the minimization problem can be solved extremely efficiently and the resulting placement spreads out the cells over large parts of the chip area. Unfortunately, the real wirelength of a chip depends on the rectilinear distances between pins and not the quadratic ones. This can lead to suboptimal solutions.

Minimizing HPWL does find placements with very low linear wirelength but the cells are mostly concentrated on very few spots of the chip area. If the cells belonging to one of these spots need to be assigned to different regions during partitioning, the assignment becomes almost arbitrary because the cells are so close to each other. Additionally, the running time for this approach is considerably higher than for minimizing QCLIQUE.

The goal of using the netlength estimations NLSE_γ and NWA_γ is to find a middle ground between QCLIQUE and HPWL in terms of approximating linear wirelength while also spreading out the cells on the chip. At the same time, the minimization processes should of course take a reasonable amount of time. This goal will guide us in the following when searching for the optimal parameters.

6.2 A Baseline

To get a feeling for how good or bad the results for minimizing wirelength based on NLSE or NWA are, we will start by evaluating them with the following parameters:

- Start vector: RANDOM
- Optimization method: Nesterov's accelerated gradient method
- The step size strategy: `backtracking-line-search`
- The stopping criterion: After exactly 1000 iterations

While the random start vector is presumably not very close to the optimum, it also has no inherent structure that could accelerate or decelerate the convergence of the optimization methods. The optimization method was chosen to be NAG with `backtracking-line-search` because it is reliable and fast. It was shown by Nesterov in [Nes83] that this method has a similar convergence guarantee as using a constant step size of $1/L$ (see Theorem 3.2). Lastly, we chose to run for 1000 iterations, a very high number, to test what this method is capable of.

All of the following graphs and plots were produced by runs on a rather large chip which we will call `Chip1`. It has roughly 2,000,000 movable cells, 5,600,000 pins and 2,000,000 nets. The placement area of this chip is about 1.3 millimeter wide and 0.9 millimeter high.

For the graphs in Figure 6.1 we optimized T_{NLSE_γ} and T_{NWA_γ} for different values of γ and graphed the value of the objective function that was used by iteration. Note that even though x- and y-direction were optimized independently, we added up the values of the objective functions for both directions to get an overall picture of the progress. Because it is also interesting to see how T_{HPWL} develops for each of these runs, the dashed red line was added for $\gamma = 10^5$. For the other two γ-values, T_{HPWL} is so close to the objective function that the differences are not visible in the plots so these lines were omitted.

What conclusions can be drawn from these pictures? Nesterov's accelerated gradient method is able to optimize T_{NLSE_γ} and T_{NWA_γ} successfully. All six objective functions where minimized such that T_{HPWL} decreased to similar values. This is also true for $\gamma = 10^5$ where a significant difference between the objective functions and T_{HPWL} can be seen. How the different final placements compare when evaluated by different objective functions will be investigated later. When comparing the pictures for NLSE and NWA the similarity is striking. For $\gamma \leq 10^3$ this is explained

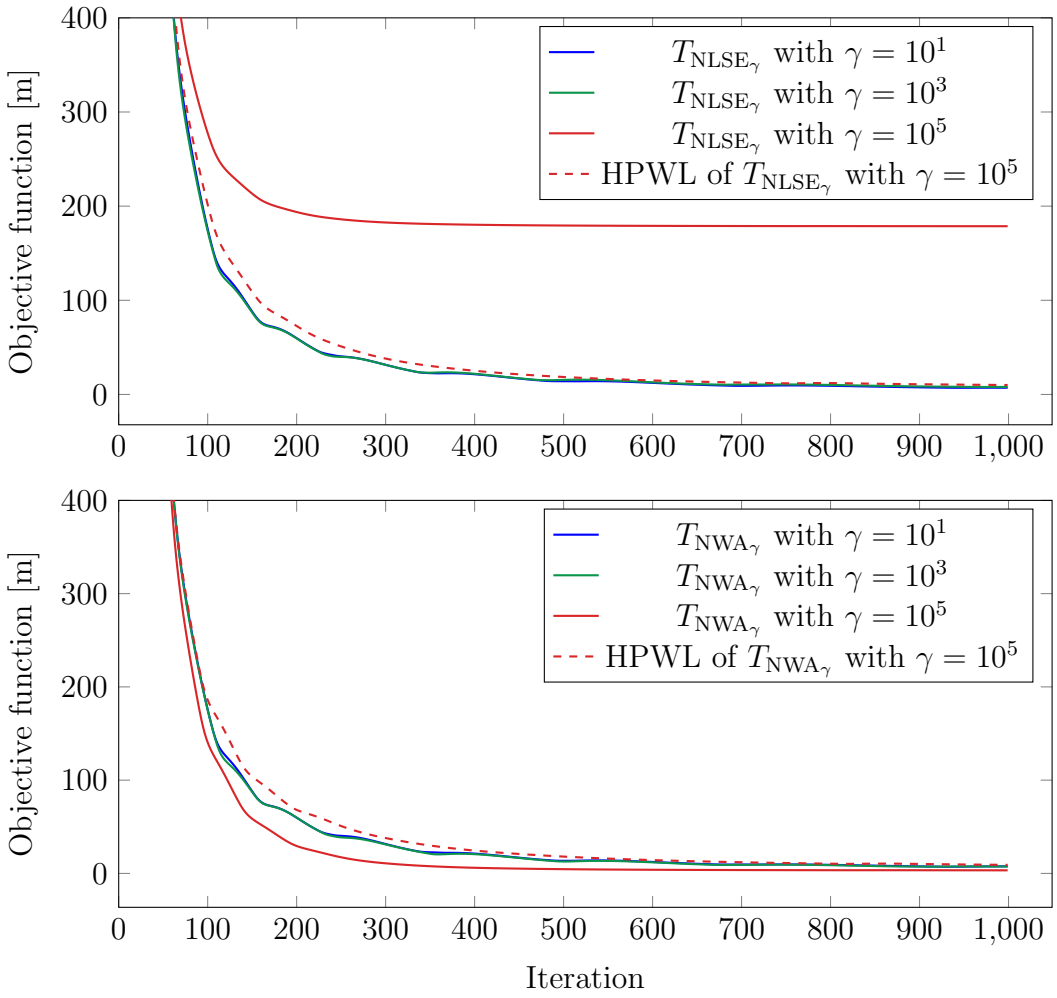


Figure 6.1: Objective function by iteration of NAG depending on γ

by the fact that all of the objective functions are already very close to T_{HPWL} and therefore close to each other. Regarding T_{NLSE_γ} and T_{NWA_γ} with $\gamma = 10^5$, the latter is indeed closer the HPWL objective function as predicted by Corollary 4.23 but the dashed red lines are very similar, so their success at minimizing linear netlength is comparable.

To investigate the convergence properties of these objective functions, we can take a look at the Euclidean norm of the gradients for each NAG iteration of each of these different optimization processes in Figure 6.2. Even though x- and y-direction were optimized independently, the Euclidean norms were added up for both directions to get an overall picture of the progress. Looking at the graphs for each direction individually shows very similar lines. As discussed in Section 3.1.3, the norm of the gradient serves as an indication of how close the optimization process is to reaching the optimum. Only for the highest γ-value of 10^5 a clear decrease down to zero can be seen. This is an important property because if the gradient norm

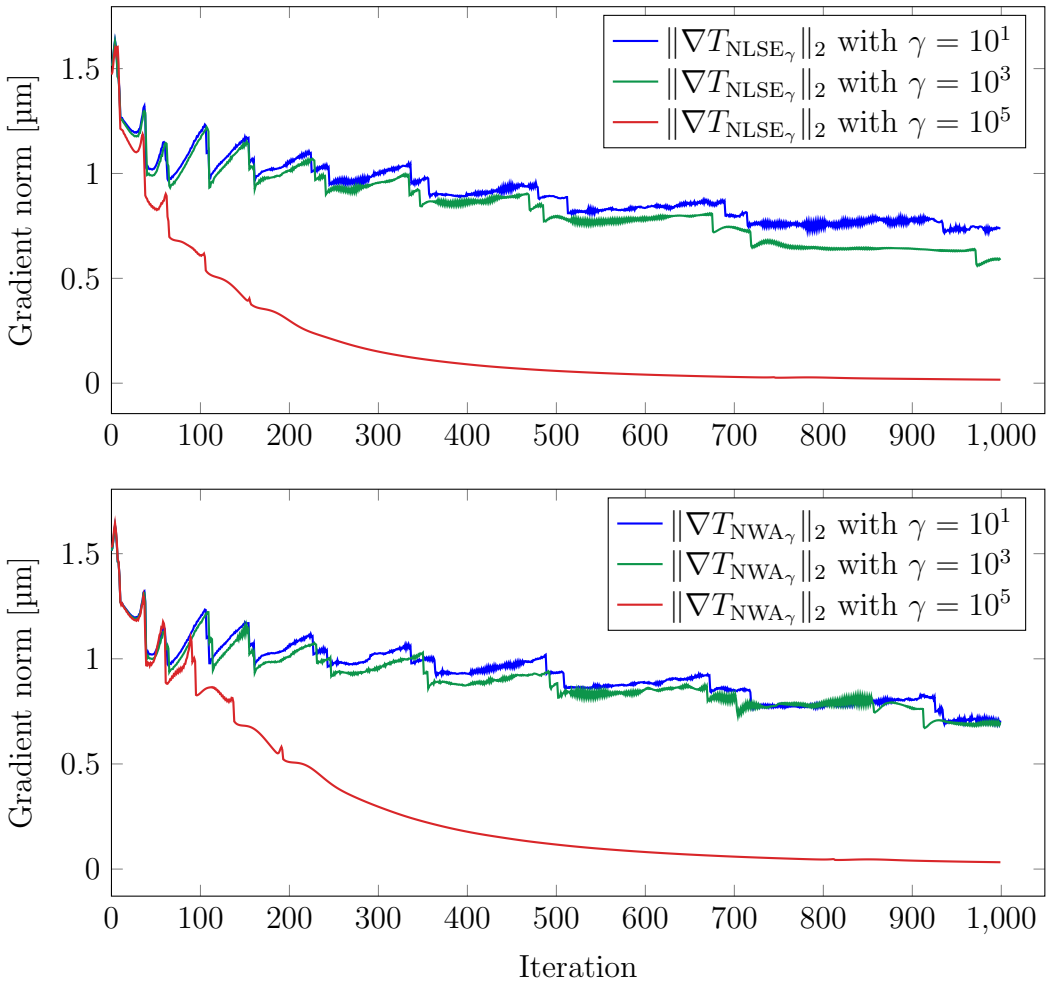
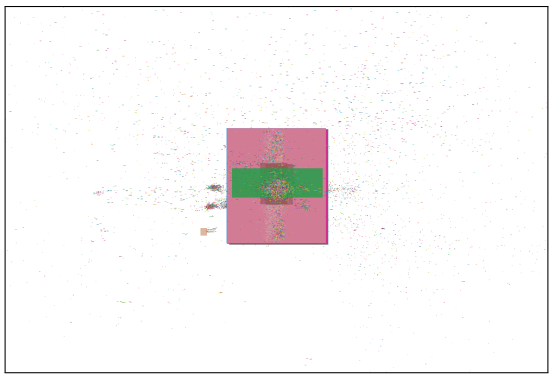


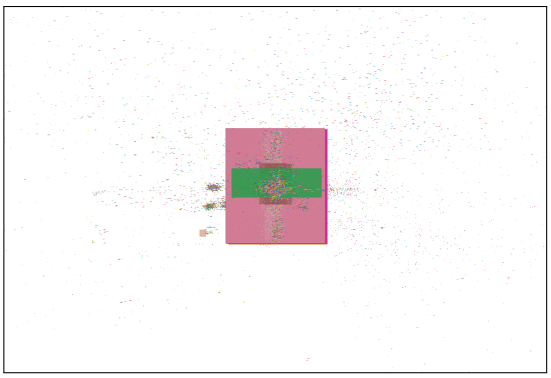
Figure 6.2: Gradient norm by iteration of NAG depending on γ

does not converge to zero, the stopping criterion will not work correctly. It was expected that lower γ-values would lead to worse convergence properties because the Lipschitz constant of the objective functions are higher but it was not clear how big this impact would be. The given graph indicates that they are so bad that γ-values lower than 10^5 cannot be handled by the chosen convex optimization techniques very well. When comparing NLSE_γ and NWA_γ using $\gamma = 10^5$, the gradient of the latter does converge slower to zero than the gradient of the former: Using the same γ-value NLSE_γ converges faster than NWA_γ.

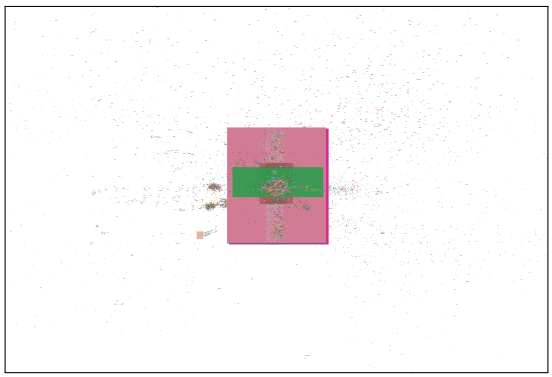
Qualitative insights about the solution qualities of these methods can be gained by inspecting the final placements that are produced by the six objective functions in Figure 6.3. All placements look useful in giving insights about relative positions of the cells. There is no visible difference between the upper four placements because of how close the objective functions are to each other. Only when γ becomes big enough the cells start to spread more over the chip area. While the difference between the plots for NLSE_γ and NWA_γ for $\gamma = 10^5$ is not easily discernable, the



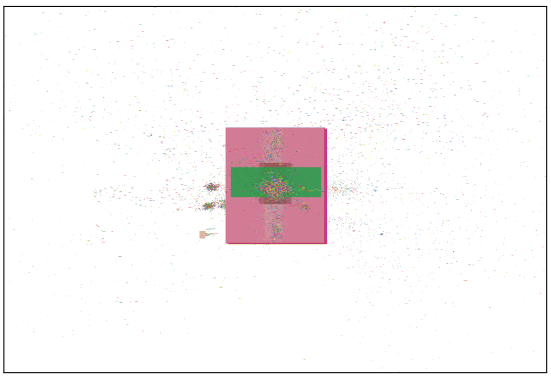
(a) T_{NLSE_γ} with $\gamma = 10^1$



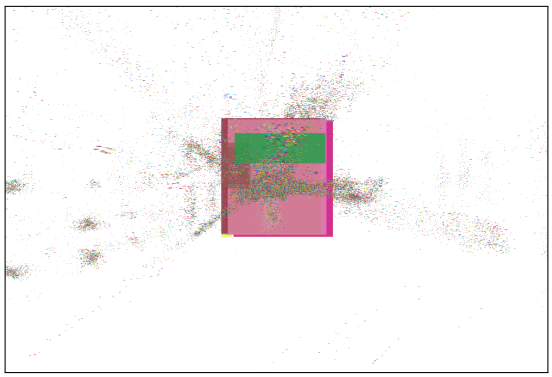
(b) T_{NWA_γ} with $\gamma = 10^1$



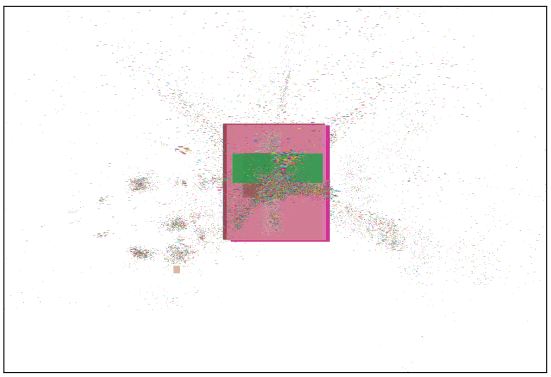
(c) T_{NLSE_γ} with $\gamma = 10^3$



(d) T_{NWA_γ} with $\gamma = 10^3$



(e) T_{NLSE_γ} with $\gamma = 10^5$



(f) T_{NWA_γ} with $\gamma = 10^5$

Figure 6.3: Placement plots after 1000 NAG iterations using the specified objective function, cropped to enlarge the center

right one places more cells close to the center of the chip.

To get a quantitative comparison between the different netlength estimations, the following table lists the evaluation of each netlength estimation (in meters) for the result of optimizing each objective function. The objective functions based on NLSE_γ and NWA_γ were optimized as described above so the resulting placements might not be close to optimal. The objective functions using HPWL or QCLIQUE were optimized by solving a MinCostFlow problem with the network simplex algorithm or a quadratic program with the conjugate gradient method respectively. How to construct the MinCostFlow problem and the quadratic program is covered in [BV08, pp. 5-7] and will not be discussed here. The last line of Table 6.1 which is labeled with "CENTER" does not correspond to an objective function that is minimized but just describes the process of placing all movable cells at the center of the chip area. This shows how small the estimated total wirelengths can get without using any information about the problem instance and without gaining any insights about the optimal cell positions.

	$\gamma =$	HPWL	NLSE 10^1	NWA 10^1	NLSE 10^3	NWA 10^3	NLSE 10^5	NWA 10^5
HPWL		4.020	4.032	4.019	5.663	3.951	197.509	3.643
NLSE	10^1	7.144	7.146	7.143	8.128	6.913	179.292	3.731
NWA	10^1	7.568	7.571	7.567	8.511	7.347	179.309	3.767
NLSE	10^3	7.357	7.357	7.357	7.999	6.962	179.248	3.658
NWA	10^3	7.761	7.761	7.761	8.524	7.392	179.327	3.794
NLSE	10^5	10.016	10.016	10.016	10.356	9.720	178.669	3.447
NWA	10^5	9.164	9.164	9.163	9.522	8.853	178.917	3.295
QCLIQUE		8.690	8.691	8.689	9.693	8.425	178.517	3.723
CENTER		5.346	5.350	5.345	6.838	5.182	180.344	4.626

Table 6.1: Evaluation of the weighted sum of each netlength estimation (column) after optimizing different objective functions (row) in meters

First of all, the table shows how close the objective functions are to T_{HPWL} when γ is small and how far off they are when γ is big. The second ovservation is that optimizing with $\gamma = 10^5$ leads to very low values of the optimized objective function (close to the lowest values in their respective columns) which is not the case for $\gamma \leq 10^3$ where the objective function is lowest when optimizing T_{HPWL} but at the same time choosing the lowest γ-value leads to the best results when measured with T_{HPWL}.

The last aspect of this baseline we want to look at is its runtime. Because x- and y-coordinates are optimized independently both runtimes are shown. Dividing the total runtime by 1000 gives the time that an average iteration needs. This can help to determine how many iterations are feasible. Table 6.2 shows those figures for all the objective functions. It shows that NLSE and NWA are vastly slower than

Netlength estimation		Total [h:mm:ss]		Average iteration [s]	
		x	**y**	**x**	**y**
HPWL		4:10:18	4:48:33		
NLSE	$\gamma = 10^1$	2:20:16	2:20:16	8.416	8.416
NLSE	$\gamma = 10^3$	2:18:35	2:17:55	8.315	8.275
NLSE	$\gamma = 10^5$	2:18:22	2:17:43	8.302	9.263
NWA	$\gamma = 10^1$	2:58:34	2:58:31	10.714	10.711
NWA	$\gamma = 10^3$	2:41:38	2:41:38	9.698	9.698
NWA	$\gamma = 10^5$	2:46:01	2:46:00	9.961	9.960
QCLIQUE		0:00:42	0:00:42		

Table 6.2: Runtime of optimizing different objective functions

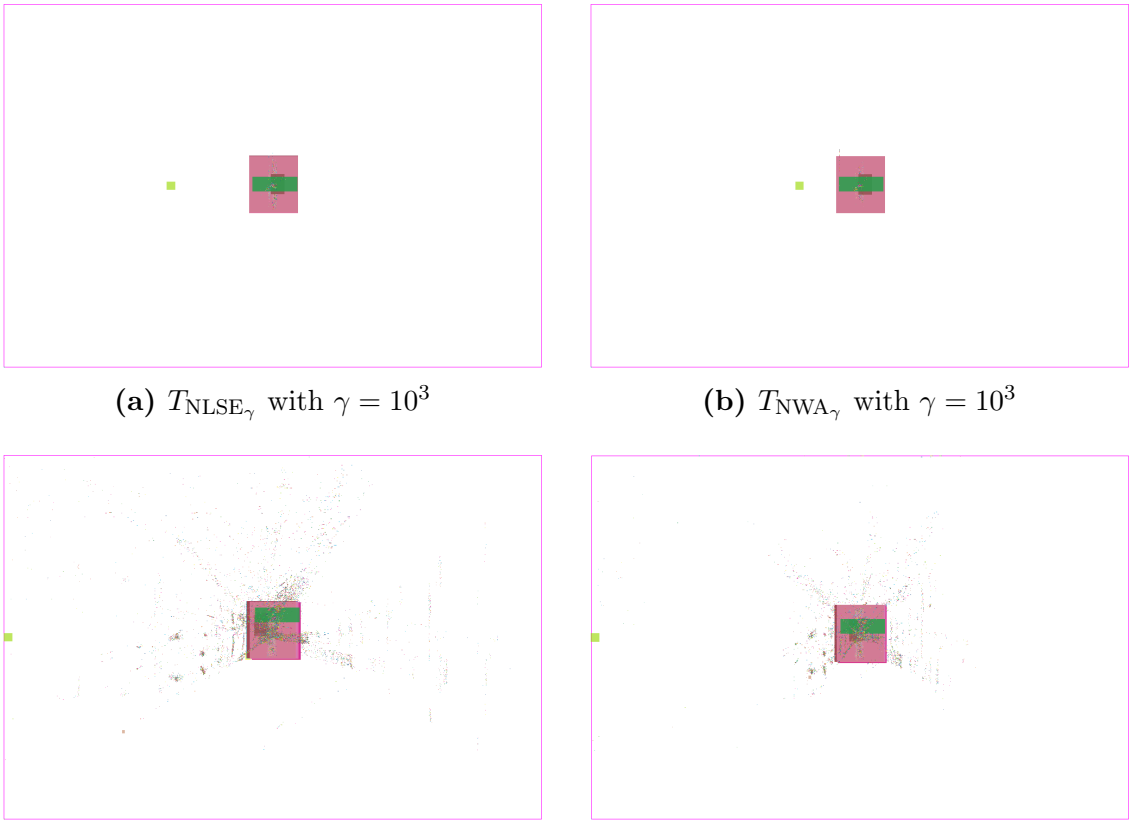
QCLIQUE but still faster than determining a HPWL-optimal placement. There are two factors to keep in mind: Firstly, 1000 iterations are plenty. It is probably possible that much fewer than 1000 iterations suffice for a reasonable placement. Secondly, the `backtracking-line-search` step size strategy is the most expensive of all step size strategies because it requires at least one additional evaluation of the objective function in each iteration, so the runtime of an average iteration will also decrease once we explore other step size strategies. The generally higher runtime of NWA is probably caused by the more complicated evaluation of the gradient while the differences between different γ-values remain unclear.

To summarize, this baseline case leads to the following observations

1. The higher γ is the better are the convergence properties and placements
2. The lower γ is the closer the netlength estimations are to HPWL
3. Optimizing NLSE is faster but optimizing NWA leads to lower HPWL
4. Even after 1000 iterations the resulting HPWL is more than 50% above the optimum when starting with random cell positions

6.3 Changing the Start Vectors

The one obviously suboptimal choice regarding the parameters of the baseline case is to start by placing every cell uniformly random on the chip area. It can be easily seen from Figure 6.1 that the wirelength of this placement is higher than optimum by several orders of magnitude. It could improve the results and also decrease the number of iterations needed to start with a placement that has a lower weighted wirelength (measured with HPWL) because presumably such a placement is also closer to the optimum. Let us start by placing all cells at the center of the chip at the beginning of the optimization. All remaining parameters stay the same as before.



(a) T_{NLSE_γ} with $\gamma = 10^3$

(b) T_{NWA_γ} with $\gamma = 10^3$

(c) T_{NLSE_γ} with $\gamma = 10^5$

(d) T_{NWA_γ} with $\gamma = 10^5$

Figure 6.4: Placement plots after 1000 NAG iterations using the specified objective function

We can skip the graphs for the objective function and gradient norms during the optimization process and directly look at the resulting placements because they show a serious problem. In Figure 6.4, the placements after 1000 NAG iterations are shown for both T_{NLSE_γ} and T_{NWA_γ} and $\gamma \in \{10^3, 10^5\}$ (the plot for $\gamma = 10^1$ is extremely similar to the one with $\gamma = 10^3$). The parameter γ controls the smoothness for both approximations of HPWL. When decreasing γ, the smoothness decreases and `backtracking-line-search` has to reduce the step size more. For $\gamma = 10^3$ the step size is decreased so much that the cells barely move and even after 1000 iterations none of the cells are at one of the edges of the chip. Even though the objective function and HPWL are better than using a random initial placement, the placements are useless in guiding a partitioning process. Therefore, the rest of this section will only investigate the impact of different start vectors with respect to $\gamma = 10^5$.

The details about the different start vectors can be found in Section 5.3.2. In summary, the parameters used for this section are

- Objective function: T_{NLSE_γ} or T_{NWA_γ} with $\gamma = 10^5$

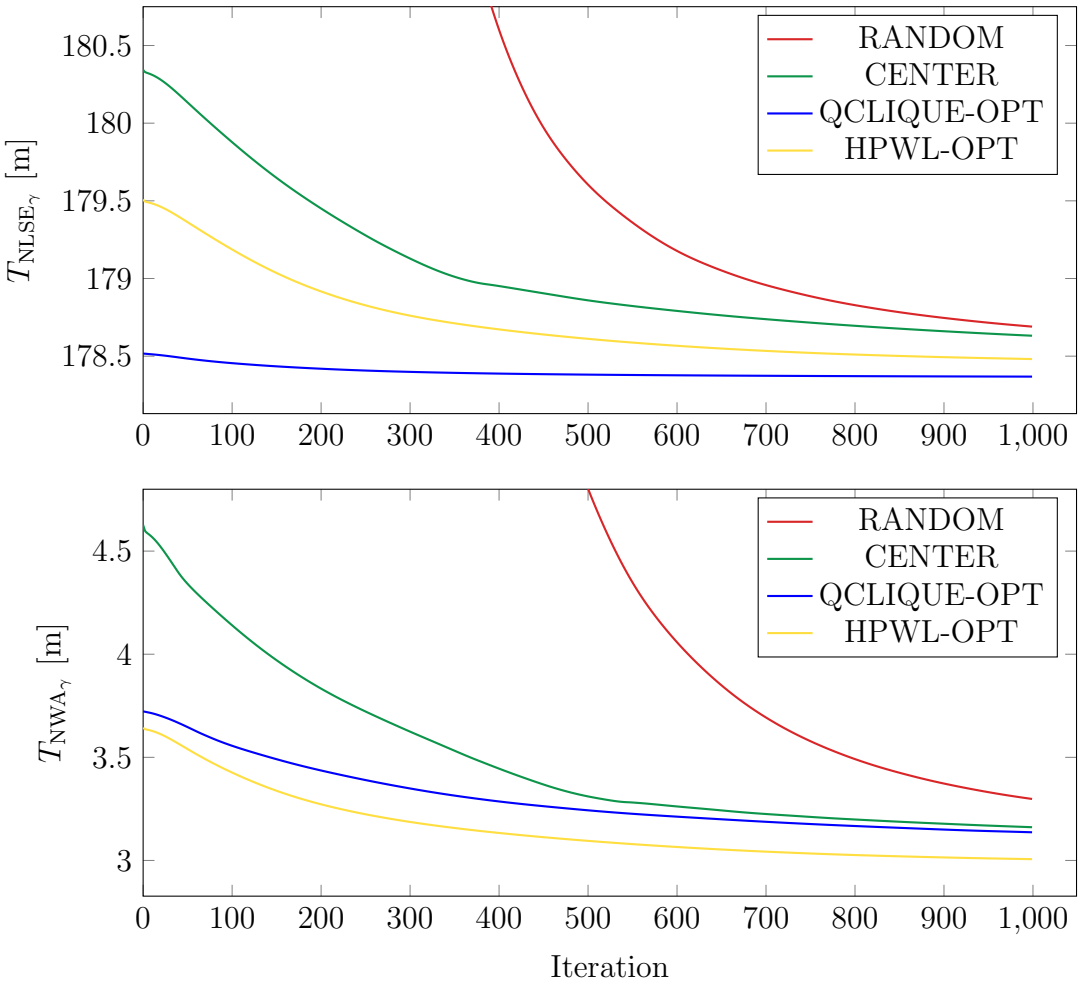
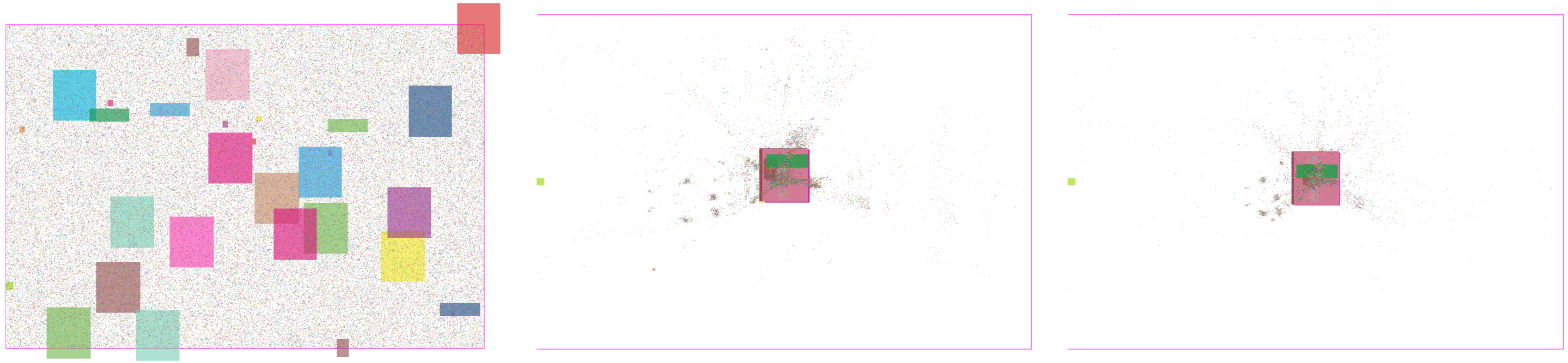


Figure 6.5: Objective function by iteration depending on start vectors

- Start vector: RANDOM, CENTER, HPWL-OPT or QCLIQUE-OPT
- Optimization method: Nesterov's accelerated gradient method
- The step size strategy: `backtracking-line-search`
- The stop criterion: After exactly 1000 iterations

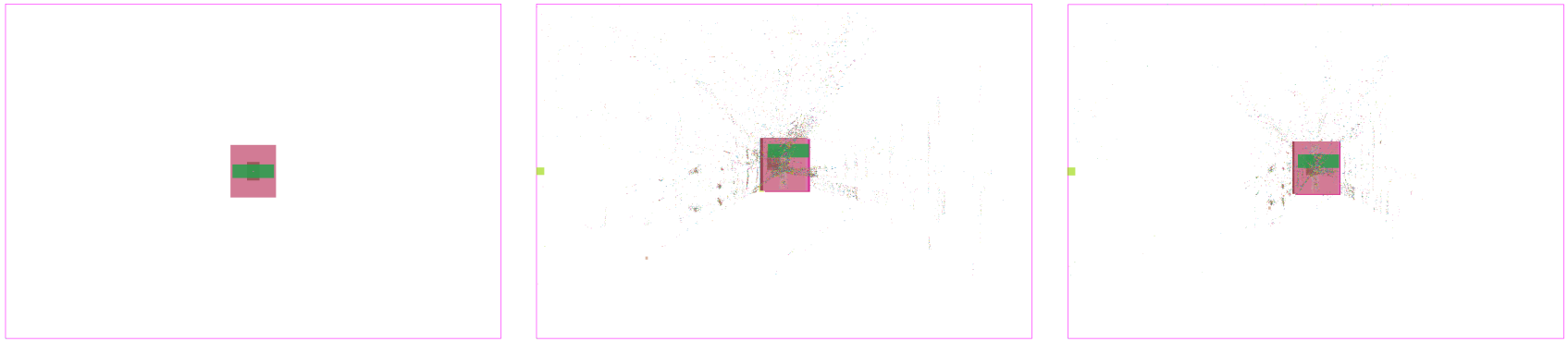
The first observation is that the hope at the start of this section was justified: By not using a random initial placement, the objective function is indeed lower during the whole optimization process. This is true for CENTER and the effect increases when starting with QCLIQUE-OPT or HPWL-OPT as start vectors as shown in Figure 6.5. This was expected as the latter two options already incorporate netlist information. Interestingly, the T_{NLSE_γ} is minimal when using QCLIQUE-OPT and T_{NWA_γ} is minimal when using HPWL-OPT which is another indicator that NWA_γ better correlates with HPWL than NLSE_γ.



(a) RANDOM

(b) T_{NLSE_γ} with $\gamma = 10^5$

(c) T_{NWA_γ} with $\gamma = 10^5$



(d) CENTER

(e) T_{NLSE_γ} with $\gamma = 10^5$

(f) T_{NWA_γ} with $\gamma = 10^5$

Figure 6.6: Placement plots after 1000 NAG iterations using the specified objective function. The start vector is the same as at the leftmost image.

(g) QCLIQUE-OPT (h) T_{NLSE_γ} with $\gamma = 10^5$ (i) T_{NWA_γ} with $\gamma = 10^5$

(j) HPWL-OPT (k) T_{NLSE_γ} with $\gamma = 10^5$ (l) T_{NWA_γ} with $\gamma = 10^5$

Figure 6.6: Placement plots after 1000 NAG iterations using the specified objective function. The start vector is the same as at the leftmost image.

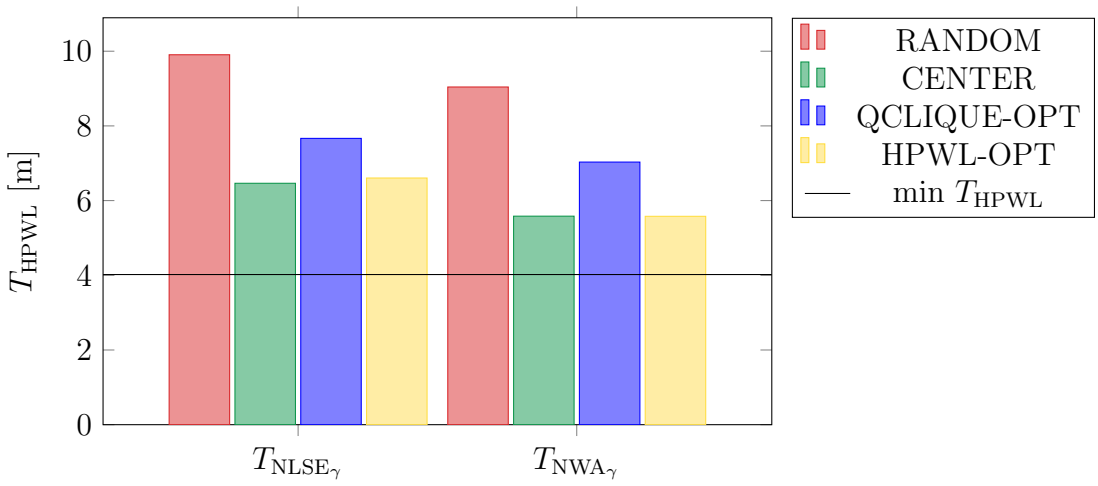


Figure 6.7: Linear wirelength T_{HPWL} of the final placements depending on the start vectors

Figure 6.6 shows that the final placement using CENTER is similar to the one starting with random cell positions, so CENTER achieves much better results during the early iterations and similar ones at the end which makes it the overall better initial placement. Simultaneously, the final placements after starting with CENTER, QCLIQUE-OPT and HPWL-OPT are very different even though Figure 6.5 shows that they have similarly low values of the objective functions. Therefore, the "valley" of optimal placements with respect to T_{NLSE_γ} and T_{NWA_γ} has to be very flat for higher values of γ. While the objective function converges to its minimum regardless of the direction that the valley is approached from, the final placement can wildly differ and in particular the weighted wirelength using HPWL can have very different values.

To investigate the difference between the values of T_{HPWL}, take a look at Figure 6.7. All of the new methods for choosing the start vectors are improving the value that was achieved by random start values. The graph of T_{HPWL} during the optimization procedure is not shown here but it indicates that the value will further decrease with more iterations for QCLIQUE-OPT while it will increase for CENTER and HPWL-OPT. Because CENTER performs considerably better than RANDOM and is much cheaper than the other two methods, it will be the default for our further investigations.

Summarizing the new insights of this section we can say that the NLSE and NWA objective functions using $\gamma < 10^5$ are not smooth enough to be used with NAG, that the CENTER start vectors clearly lead to faster convergence than RANDOM for the objective functions with $\gamma = 10^5$ and that these objective functions become very flat towards their minima such that very different placements can achieve close to optimal values.

6.4 Changing the Step Size Strategies

It is useful to consider other step size strategies than `backtracking-line-search` because it requires at least one evaluation of the objective function for each iteration which makes it very expensive. At the same time, changing the step size strategy will also influence how fast the gradient norm is reduced so one must consider both runtime and convergence speed. To simplify the graphs, we chose to only test with the NWA netlength estimation, the results are similar for NLSE. The parameters for this section are therefore

- Objective function: T_{NWA_γ} with $\gamma = 10^5$
- Start vector: CENTER
- Optimization method: Nesterov's accelerated gradient method
- The step size strategies: `constant`, `scaled-by-sqrt`, `backtracking-line-search`, `observed-lipschitz`, `min-observed-lipschitz`, `barzilai-borwein`
- The stopping criterion: After exactly 500 iterations

In Figure 6.8 and Figure 6.9 the objective function and gradient norm is graphed during the optimization process with different step size strategies. This time, the x-axis does not represent the number of iterations but the runtime which enables us to see the convergence properties and runtime at the same time. For both the gradient norm and the objective function the values for x- and y-direction were added up. Similarly the time values were summed up such that the data point at iteration i is plotted at the time that would be needed to first calculate i iterations for the x-coordinates and then i iterations for the y-coordinates. Due to this, the real time needed for 500 iterations is roughly half of the rightmost point of each graph. Additionally, the second graph of each figure is just a zoomed in version of the graph above to get a more detailed comparison between those step size strategies that perform the best.

One of the first observations is that the runtime indeed decreases significantly when opting for a different step size strategy than `backtracking-line-search`. The runtime for 500 iterations decreases by roughly 30% regardless of which other option was chosen. The cause of the different runtimes between the faster step size strategies remains unclear: An additional square root computation for every iteration does not explain the visible increase in runtime of `scaled-by-sqrt` over `constant`. Similarly, the shorter runtime of `observed-lipschitz` when compared to `min-observed-lipschitz` can not be explained by one missing minimum computation between h_{k-1} and $\frac{\|\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}\|_2}{\|\nabla f(\mathbf{x}^{(k)})-\nabla f(\mathbf{x}^{(k-1)})\|_2}$ per iteration. The differences might be caused by small changes in the runtime of the exp function depending on the input value, which become significant because the function is called millions of times per iteration. If this is correct, the runtime would vary (inside a certain range)

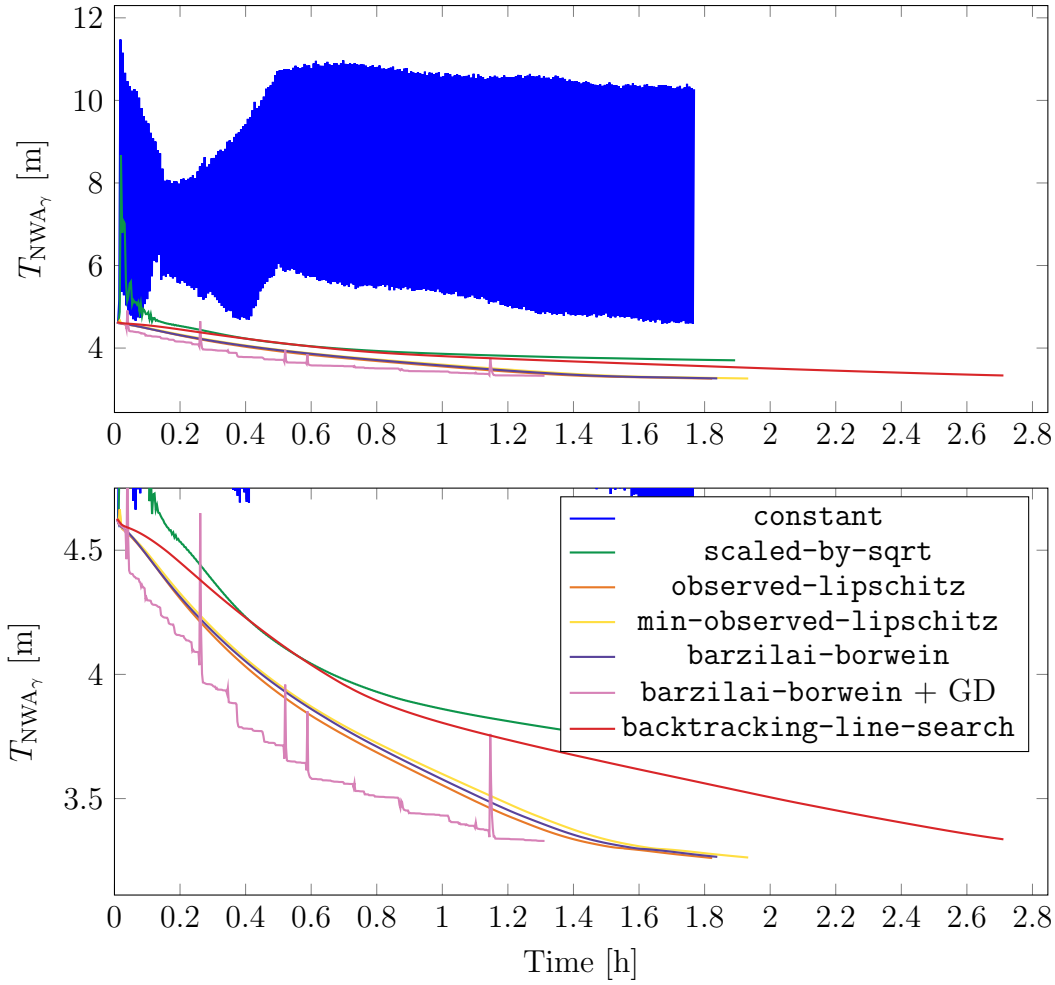


Figure 6.8: Objective function by time depending on the step size strategy

with the minimization path taken and the impact would be unforseeable. This is why no further runtime differences are analyzed. Any step size strategy except for `backtracking-line-search` is considered as having roughly the same runtime.

Let us now analyze the details of graphs: The step size strategy `constant` stands out because the objective function heavily fluctuates during the minimization process and the gradient norm does not converge to zero. This is because the initial step size is determined using `backtracking-line-search` (see Section 5.3.3) and might be alright for the first iteration but is certainly too large for the following iterations. Big fluctuation without relevant decrease of the objective function is a standard problem when the step size is chosen too large. This strong dependence on a heuristic determining the first step size makes this step size strategy unreliable for our purposes.

`scaled-by-sqrt` avoids most of the issues of `constant` but also does perform worse than the other options in terms of gradient norm and objective function.

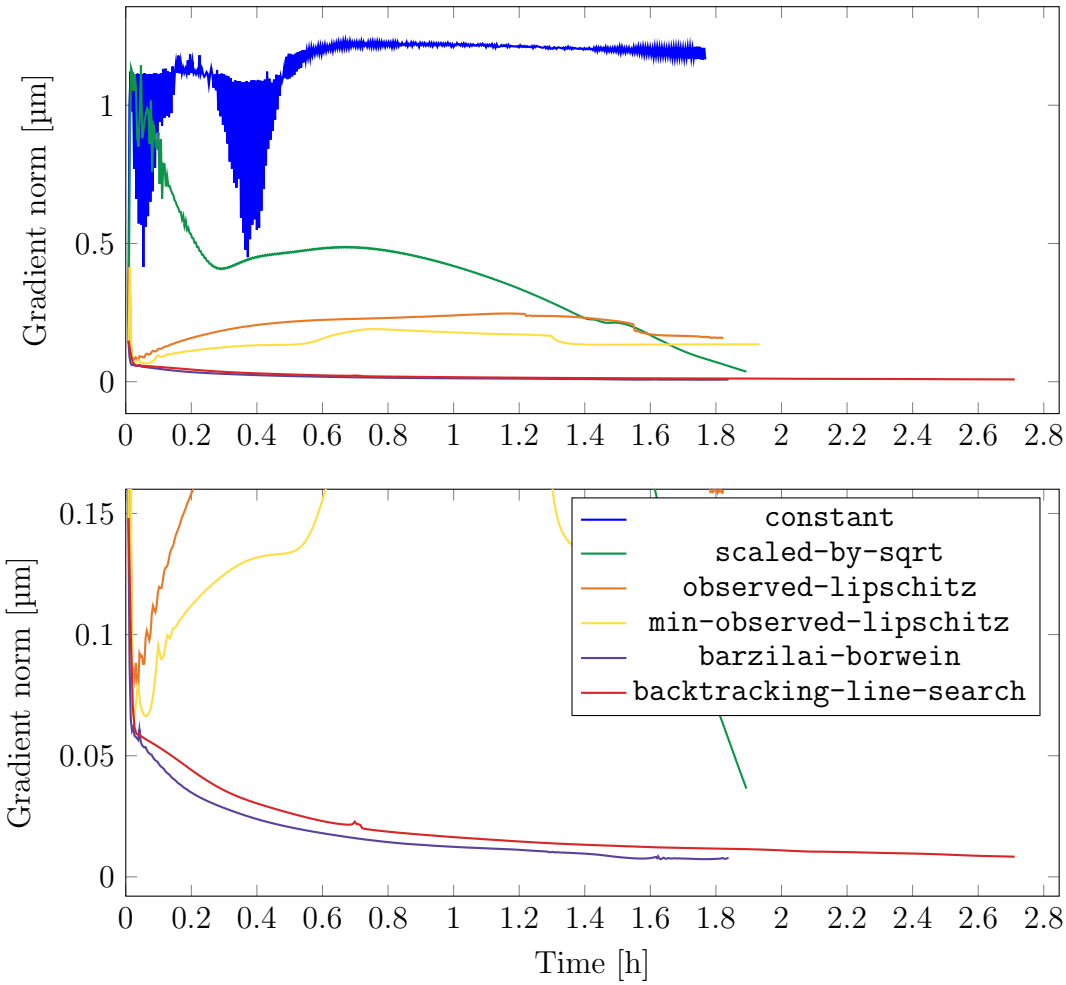


Figure 6.9: Gradient norm by time depending on the step size strategy

The main contenders for the best step size strategy are `observed-lipschitz`, `min-observed-lipschitz` and `barzilai-borwein`. The graphs of the objective functions during the optimization process are very similar for all of them, but the graph of the gradient norm is best for `barzilai-borwein`. It is the only step size strategy apart from `backtracking-line-search` that can consistently decrease the gradient norm. As previously said, this enables the stopping criterion to work correctly in the sense that a continuous decrease of ε leads to continuous increase in runtime.

The question might be raised whether using gradient descent as the method of convex optimization leads to further runtime improvements with comparable success in minimizing the objective function and decreasing the gradient norm. The best step size strategy in conjuction with gradient descent is `barzilai-borwein` and the combination is included in Figure 6.8 as "`barzilai-borwein` + GD". As can be seen, the runtime is indeed reduced further by about 30% but the objective

function does not decrease smoothly. The objective function and the gradient norm both spike at some points during the optimization process which indicates that this method is not stable. (The graph of the gradient norm spikes so much that including it in Figure 6.9 would make the diagram illegible.) The only step size strategy that produces a smooth objective function graph when used with gradient descent is `backtracking-line-search` but the shorter runtime per iteration is outweighed by the slower convergence of the objective function when compared to its optimization with Nesterov's accelerated gradient method. This means that although the gradient descent method is faster, it either makes the objective function converge slower or produces unstable results. For this reason, we will only consider Nesterov's method in the following (as already done in the previous chapters).

In summary, the `barzilai-borwein` step size strategy combined with Nesterov's accelerated gradient is chosen as the best combination of these parameters. It is able to decrease the runtime of the average iteration by about 30% and is subsequently able to reduce the objective function significantly faster than `backtracking-line-search`.

6.5 Choosing the Stopping Criterion Precision

The standard stopping criterion for gradient descent methods is

$$\|\nabla f(\mathbf{x}^{(k)})\|_2 < \varepsilon$$

where f is the objective function and $\mathbf{x}^{(k)}$ is the solution vector after the kth iteration. This section will try to show the difficulty of choosing a reasonable ε for our purposes. The graphs in this section are all generated by the following parameters:

- Objective function: T_{NWA_γ} with $\gamma = 10^5$
- Start vector: CENTER
- Optimization method: Nesterov's accelerated gradient method
- The step size strategy: `barzilai-borwein`
- The stopping criterion: After exactly 500 iterations

Because the choice of ε can depend on the problem instance, it is time to introduce more chips to compare results on. Table 6.3 is an overview of the sizes of these instances. `Chip1` and `Chip2` have the same chip area and similar numbers of cells, pins and nets but differ in the specific way the cells are set up. `Chip3` and `Chip4` are included to see how NWA performs on smaller instances.

The simplest option for choosing ε would be using the same value for all instances. In Figure 6.10, the Euclidean norms of the gradients during the optimization processes are graphed. This time x- and y-direction are considered seperately because

Instance	# Cells	# Nets	# Pins	Width [µm]	Height [µm]
`Chip1`	1,900,000	1,900,000	5,600,000	1300	900
`Chip2`	1,800,000	1,800,000	5,400,000	1300	900
`Chip3`	800,000	600,000	2,200,000	1800	400
`Chip4`	300,000	300,000	800,000	400	200

Table 6.3: Rounded properties of the newly introduced instances

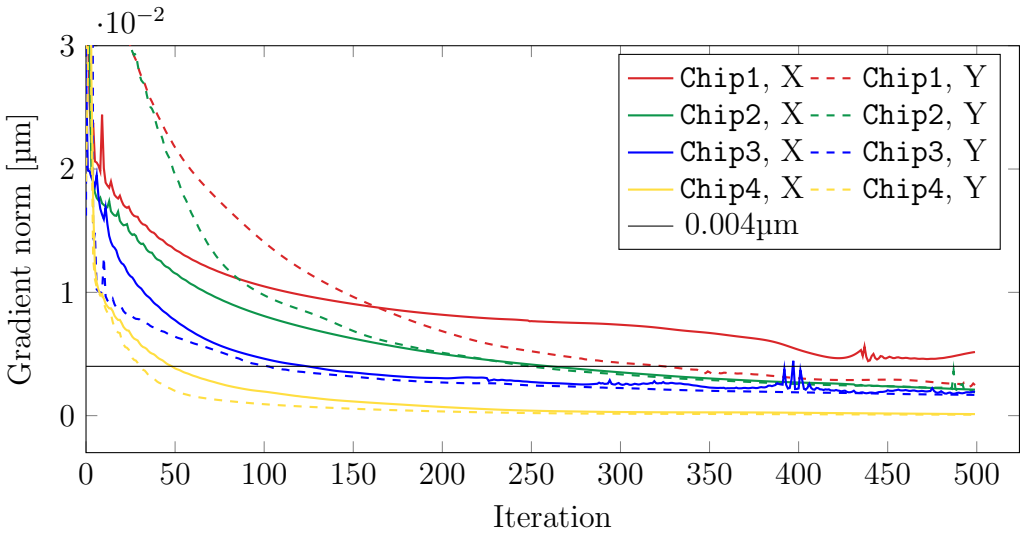


Figure 6.10: Gradient norm by iteration depending on the instance

when the two optimization processes run in parallel the stopping criterion cannot depend on both but has to be considered for each of them seperately. The first clear insight that can be gained by looking at the graph is that, although it was one of the important aspects of choosing the current paramters, the gradient norm can still stagnate above zero for some time. Take a look at the x-direction of `Chip1` for example: After the first 100 iterations the gradient norm is decreasing very slowly and only drops below 0.005µm after 400 iterations. Reaching 0.008µm takes about 200 iterations, reaching 0.006µm about 400 and reaching 0.004µm more than 500 (in fact around 650). In other words, small increases in desired precision lead to great increases in runtime.

The second observation is that a single ε for all these instances does not lead to a comparable number of iterations even for similar instances. Say, for example, that ε is set to 0.004µm (the value is marked in Figure 6.10). Then `Chip2` would reach the stopping criterion after roughly 250 iterations for both directions while `Chip1` would need more than 500 iterations to reduce the gradient norm of both directions below this value. This is the case even though `Chip1` and `Chip2` are very similar and the placements profit similarly from an increased number of iterations. The value of ε is supposed to control the compromise of solution quality and runtime but at least for some values of it the solution quality and runtime differs wildly even

for similar chips which makes this stopping criterion unusable.

It may be possible to scale the gradient norms appropriately so that they all line up and a single threshold becomes viable but is unclear how exactly the scaling factor has to be chosen so that the graphs match up because it is unclear which factors influence the gradient norm in which way. Therefore we chose to not depend on the gradient norm as a stopping criterion but instead fix the number of iterations as before. This way the computation time scales in a predictable way with the instance size and it avoids that one of the two optimization processes on a chip is terminated much earlier than the other.

Now the question becomes how many iterations are reasonable when one has to balance solution quality with runtime. In Figures 6.11 to 6.14, the placement plots after 50, 100 and 200 iterations are shown for each chip. The corresponding runtimes are recorded in Table 6.4.

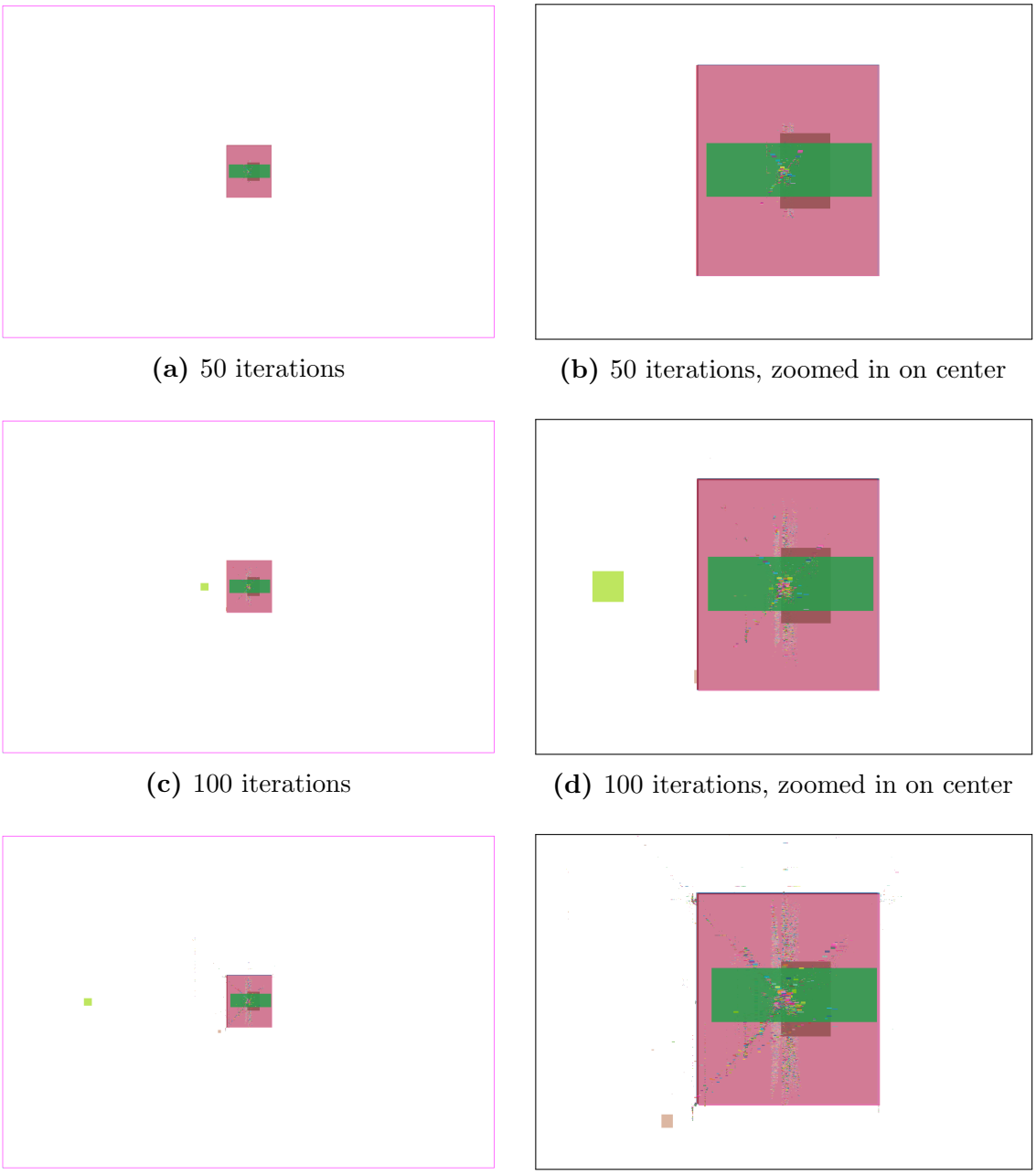
Instance	50 iterations	100 iterations	200 iterations
`Chip1`	230.817	410.340	887.025
`Chip2`	208.221	394.730	755.941
`Chip3`	82.912	167.200	311.384
`Chip4`	30.660	74.842	155.498

Table 6.4: The runtimes of varying numbers of iterations in seconds

It can be seen that there is a huge difference in how much the cells are spreaded between the large chips and the small chips. While even 200 iterations do not suffice to move more than a few cells out of the center for `Chip1` and `Chip2`, (compare these plots with Figure 6.3) the situation is much better for `Chip4` where a good spread can be seen after just 50 iterations. This can indicate that NWA wirelength estimation works better with small chips and that large chips just need more iterations or that some of the parameters need to be chosen differently for large chips. It would be interesting to see, for example, whether increasing the γ value further makes the spreading of cells over the chip area faster. In this case one could adapt the γ value to the size of the chip.

The runtimes in Table 6.4 show that the runtime increases roughly linearly with the number of iterations. It is also roughly linear in the number of pins which is as expected because it can be observed that evaluating the gradient is the most expensive operation of NAG in our case and evaluating the gradient of NWA is done in $O(|P|)$ time (see Section 5.1.3).

It is unclear which number of iterations gives us the best compromise between spread and runtime in general but for the sake of the comparisons with HPWL and QCLIQUE in the next section we will perform 200 iterations on each of the chips.



(a) 50 iterations

(b) 50 iterations, zoomed in on center

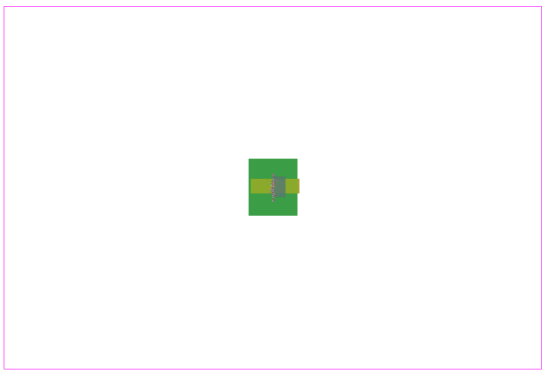
(c) 100 iterations

(d) 100 iterations, zoomed in on center

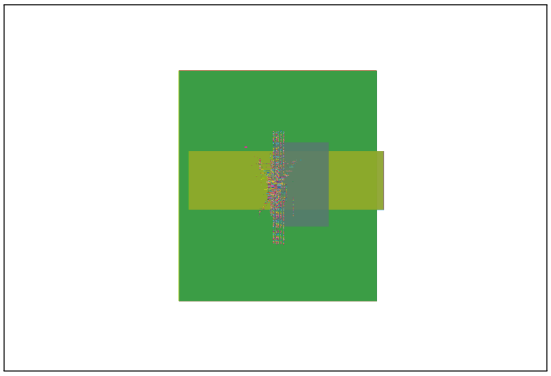
(e) 200 iterations

(f) 200 iterations, zoomed in on center

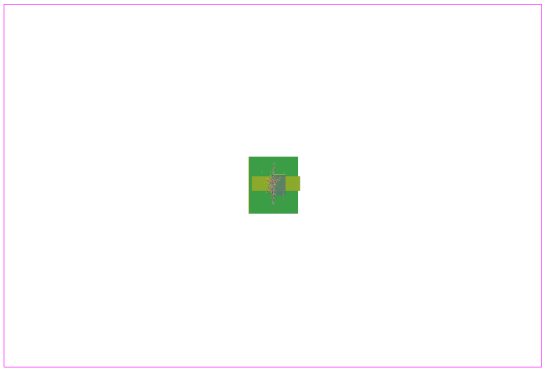
Figure 6.11: Placement plots after 50, 100 and 200 NAG iterations for `Chip1`



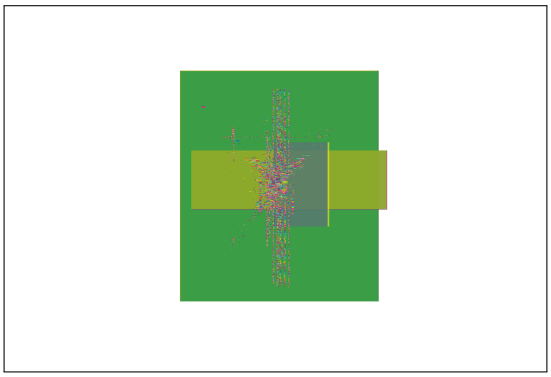
(a) 50 iterations



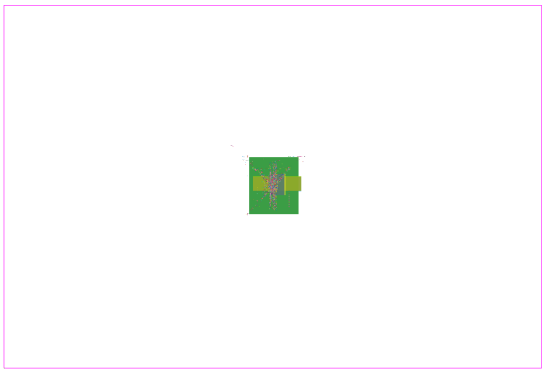
(b) 50 iterations, zoomed in on center



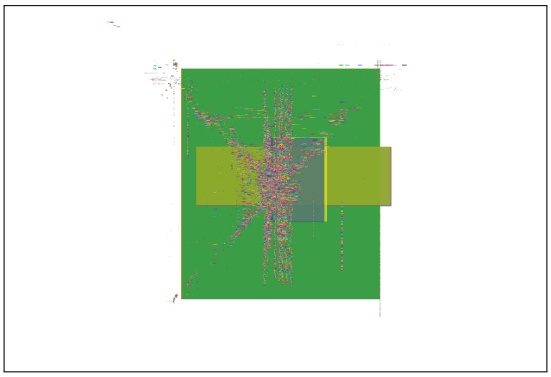
(c) 100 iterations



(d) 100 iterations, zoomed in on center

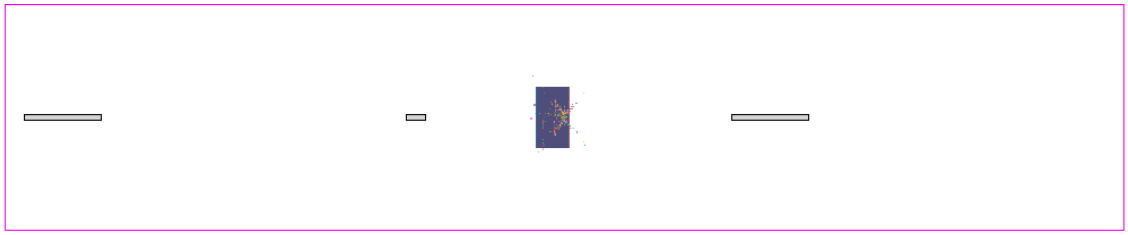


(e) 200 iterations

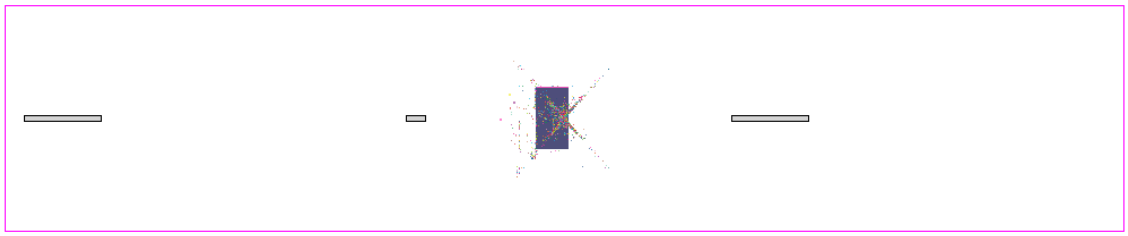


(f) 200 iterations, zoomed in on center

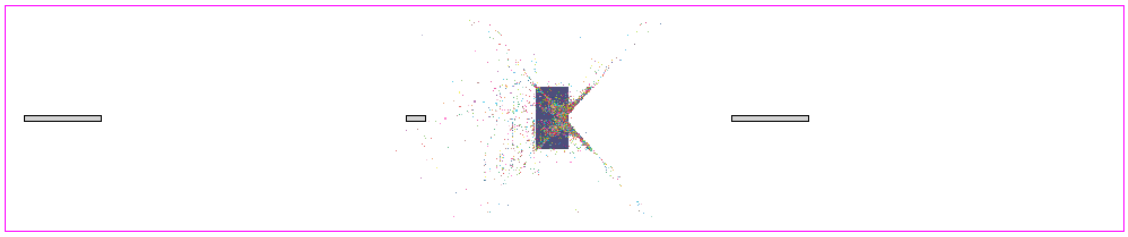
Figure 6.12: Placement plots after 50, 100 and 200 NAG iterations for `Chip2`



(a) 50 iterations

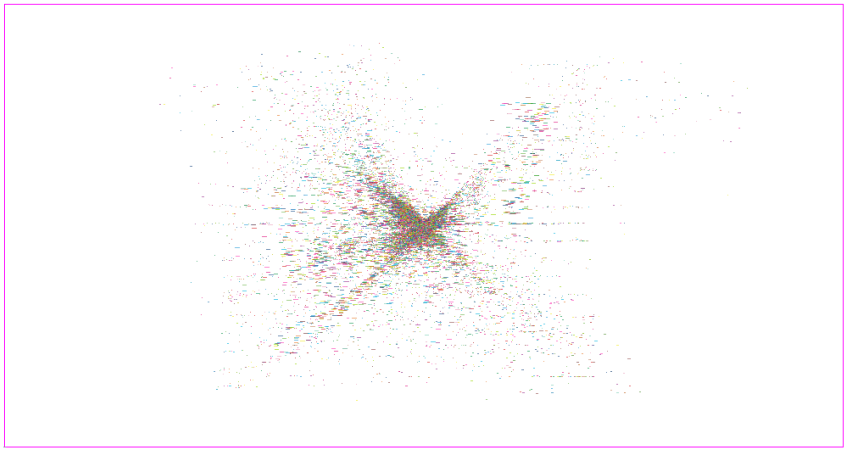


(b) 100 iterations



(c) 200 iterations

Figure 6.13: Placement plots after 50, 100 and 200 NAG iterations for `Chip3`



(a) 50 iterations

(b) 100 iterations

(c) 200 iterations

Figure 6.14: Placement plots after 50, 100 and 200 NAG iterations for `Chip4`

6.6 Comparisons with HPWL and QCLIQUE

In the last section of this chapter, the parameters that were chosen in the previous sections are used to compare the minimization of the NLSE/NWA netlength estimations with the results of minimizing HPWL or QCLIQUE. The parameters that were chosen are:

- Objective function: T_{NWA_γ} with $\gamma = 10^5$
- Start vector: CENTER
- Optimization method: Nesterov's accelerated gradient method
- The step size strategy: `barzilai-borwein`
- The stopping criterion: After 200 iterations

There are three main categories for this comparison, namely solution quality, weighted total HPWL and runtime. We will first investigate the latter two in Tables 6.5 and 6.6. In both tables, the columns specify which objective function was minimized while the rows show the results for different chips. For Table 6.5 the columns are subdivided to show the weighted wirelength based on HPWL and NWA of the final placement.

Instance	min NWA		min QCLIQUE		min HPWL	
	NWA	HPWL	NWA	HPWL	NWA	HPWL
`Chip1`	3.741	5.019	3.723	8.690	3.643	4.020
`Chip2`	3.190	4.353	3.431	8.166	3.330	3.691
`Chip3`	3.971	5.820	4.739	8.577	3.552	3.748
`Chip4`	0.192	1.228	0.188	1.369	0.466	0.594

Table 6.5: The weighted wirelengths after minimizing different objective functions in meters

Instance	# Pins	min NWA	min QCLIQUE	min HPWL
`Chip1`	5,600,000	887.025	42.599	17325.760
`Chip2`	5,400,000	755.941	42.811	17258.766
`Chip3`	2,200,000	311.384	13.402	6248.931
`Chip4`	800,000	155.498	2.433	1588.552

Table 6.6: The runtimes of minimizing different objective functions in seconds

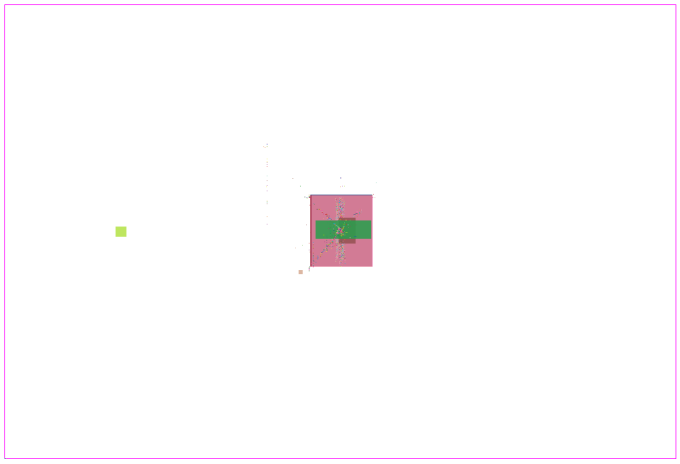
Regarding the weighted HPWL, minimizing NWA produces lower values than minimizing QCLIQUE but the decrease is about 45% for `Chip1` and `Chip2` and roughly 10% for `Chip4`. The closer the cells are to the center the higher the

HPWL decrease so this might be caused by starting at the center and not by NWA being a better approximation of HPWL than QCLIQUE. Also note that for `Chip1` and `Chip3` the NWA wirelength is lower after minimizing QCLIQUE than after minimizing NWA. This certainly underlines one of the earlier observations that very different placements can lead to similarly low values of NWA but it could also indicate that this is not the right minimization technique if it performs worse than a different technique with a completely different objective function. One interesting option would be to minimize QCLIQUE with the CG method first as NLSE/NWA seem to be rather similar to QCLIQUE for high γ-values (also see Figure 6.5) and then optimize the result futher with NLSE/NWA and smaller γ values.

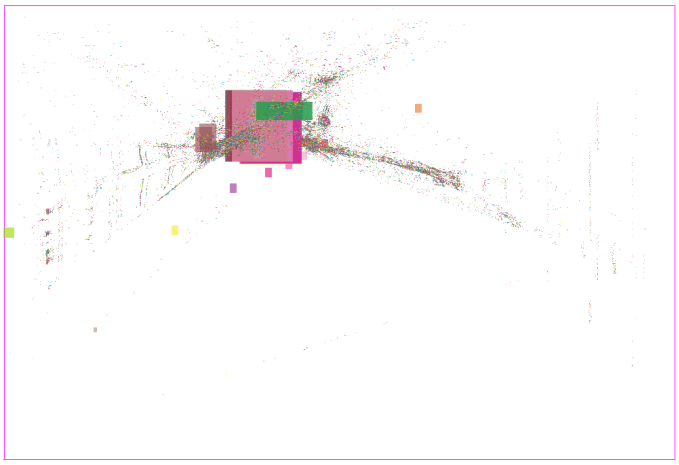
When comparing runtimes, it is clear that minimizing QCLIQUE is much faster than NWA. Only about 3 to 11 NAG iterations are possible in the timespan needed for QCLIQUE to converge. Especially on the larger chips, 11 iterations are almost nothing. Minimizing HPWL optimally on the other hand is much slower than using NWA.

Let us now get to the placement plots that are produced by minimizing these objective functions: Figures 6.15 to 6.18 show the placement plots after minimizing each objective function. In almost all of these cases the QCLIQUE results are much better at spreading out the cells and therefore more useful for a subsequent partitioning step than the placements produced by minimizing NWA. The only exception is `Chip4` on which both netlength estimations produce very similar results but using NWA has a runtime that is 70x higher.

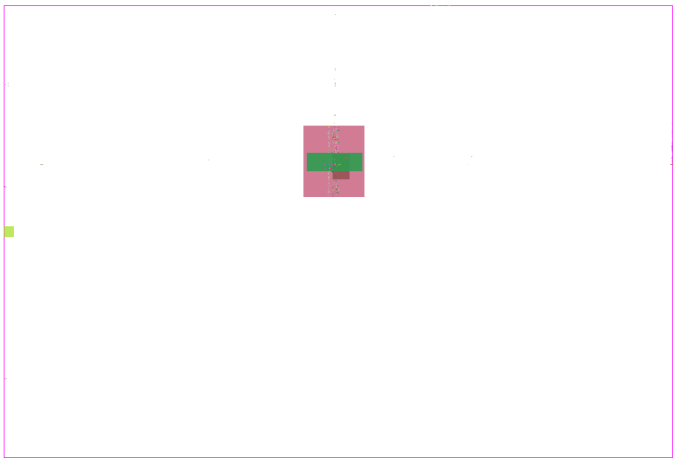
All in all this is rather disappointing. Section 6.2 showed that in general the NLSE and NWA netlength estimations are able to produce somewhat useful placements for a partitioning step but they need an extreme amount of resources to do so. Once the number of iterations is restricted to bring down the runtime the results cannot compete with QCLIQUE anymore.



(a) NWA

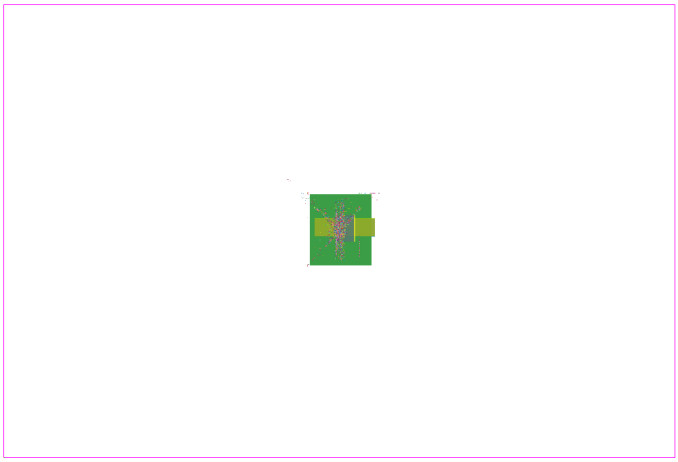


(b) QCLIQUE

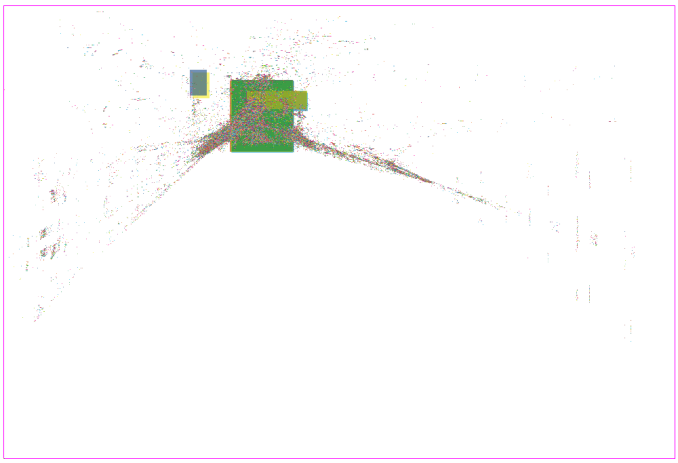


(c) HPWL

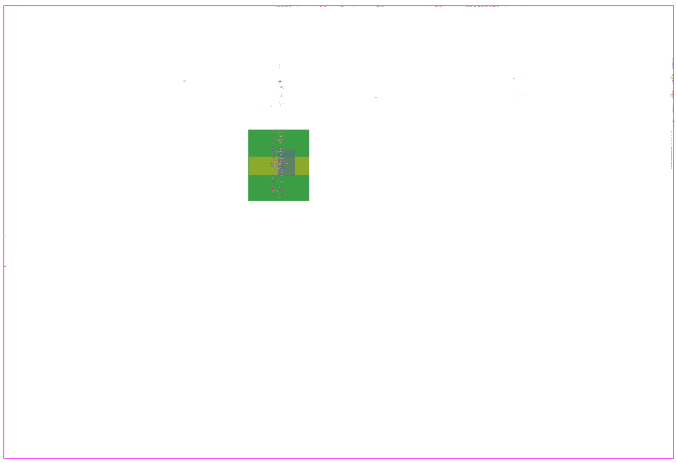
Figure 6.15: Placement plots for `Chip1`



(a) NWA

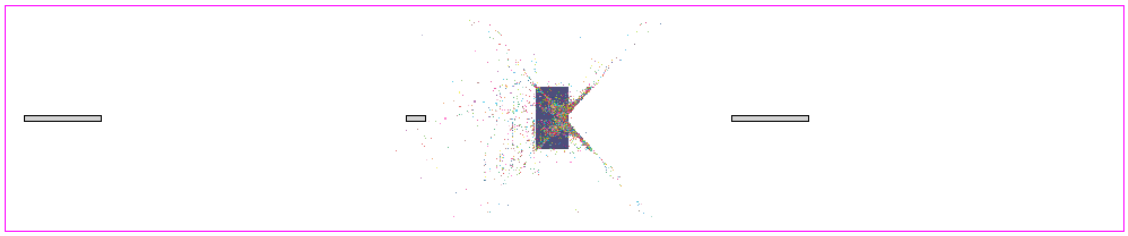


(b) QCLIQUE

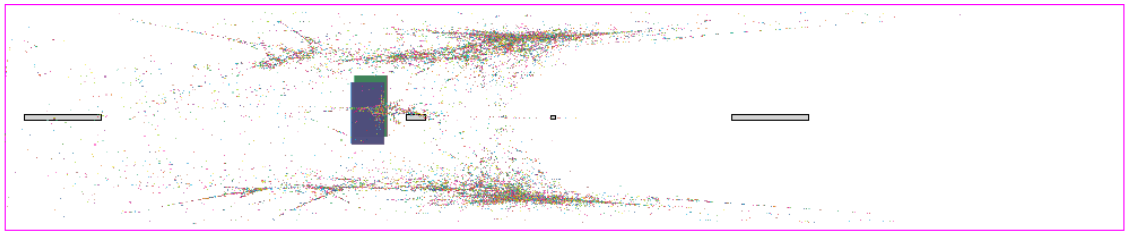


(c) HPWL

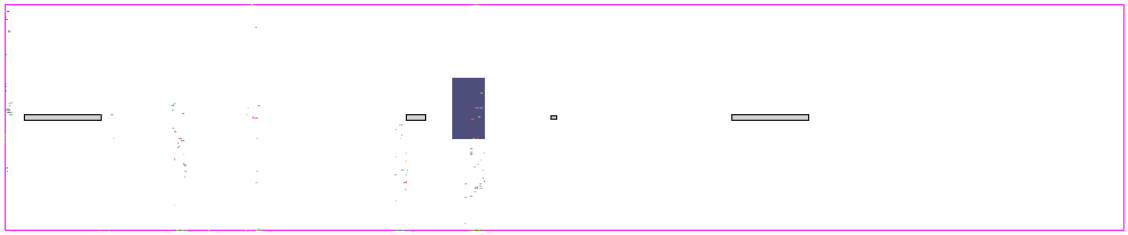
Figure 6.16: Placement plots for `Chip2`



(a) NWA



(b) QCLIQUE



(c) HPWL

Figure 6.17: Placement plots for `Chip3`

(a) NWA

(b) QCLIQUE

(c) HPWL

Figure 6.18: Placement plots for `Chip4`

7 Summary

In this bachelor thesis the use of the NLSE and NWA netlength estimations in a specific subproblem of the global placement task was investigated. The theoretical properties of these two netlength estimations were summarized and expanded to show that the corresponding weighted total wirelength functions can be used as objective functions for the convex optimization techniques gradient descent and Nesterov's accelerated gradient method. Afterwards these convex optimization methods were used in practice on some chip instances to solve the problem and the results were compared with approaches that use the classic netlength estimations QCLIQUE and HPWL.

The main takeaways from the theoretical section are the following: Both NLSE_γ and NWA_γ approximate HPWL closer and closer as γ tends to zero but NWA is always closer to HPWL for the same γ. NLSE is convex and contrary to what was claimed in [HCB11] NWA is not convex but NWA only slightly violates the convexity conditions and can be bounded from above and below by a convex function. Additionally, both netlength estimations have a Lipschitz continuous gradient. All of these properties also extend to the netlength estimations when viewed as functions of cell positions rather than pin positions.

In spite of these positive theoretical results, the pratical results are rather disappointing. The value of γ has to be fairly high in order for the convergence properties of these netlength estimations to become good enough and then the objective functions flatten out so much towards their minima that very different placements can achieve close to optimal values. Even when high γ-values are chosen the convex optimization methods are not able to reliably decrease the gradient norm towards zero so that a low gradient norm fails as a good stopping criterion. If the optimization processes are instead stopped after a fixed number of iterations that keep the runtime in a reasonable range the resulting placements are not able to compete with the results of minimizing QCLIQUE in terms of solution quality or runtime.

8 Zusammenfassung

In dieser Bachelor-Arbeit wird ein Teilproblem des Global Placements betrachtet welches beim Chip-Design-Prozess vorkommt. Während ein Chip designt wird, gibt es einen Zeitpunkt zum dem bereits feststeht welche vorgefertigten Bauteile (bestehend aus einer kleinen Anzahl von Transistoren) für den Chip benötigt werden und wie diese verbunden werden müssen. Diese rechteckigen Bauteile werden Zellen genannt und sollen so überlappungsfrei auf der rechteckigen Chipfläche platziert und mit Leiterbahnen verbunden werden, dass die Länge der Leiterbahnen minimal ist, wobei einige Zellen mit vorplatzierten Verbindungsstellen verbunden werden müssen. Die Verbindungsstellen, die entweder auf den Zellen liegen oder bereits vorplatziert sind, heißen Pins und eine Menge von Pins, die auf dem finalen Chip elektronisch äquivalent seien sollen, heißt Netz. Im ersten Schritt, dem Placement, legt man zunächst nur die Platzierungen der Zellen fest und minimiert dabei eine Schätzung der Gesamtverbindungslänge, die eine gewichtete Summe von den geschätzten Verbindungslängen der einzelnen Netze ist. Erst danach werden die Positionen der Leiterbahnen, die die Pins verbinden, berechnet.

In einer Herangehensweise an das Placement-Problem wird im allerersten Schritt die Bedingung fallen gelassen, dass sich die Zellen nicht überlappen dürfen, und es entsteht ein unbeschränktes Optimierungsproblem. Im Kontext dieses Optimierungsproblems werden in dieser Bachelor-Arbeit zwei Schätzungen der Netzlängen betrachtet, die auf den Maximumsapproximationen LogSumExp und Weighted-Average basieren. Diese wurden bereits in anderen Ansätzen im Chip-Design erfolgreich eingesetzt und sind differenzierbare Approximationen der Standard-Netzlängen-Schätzung HPWL. Ihre Differenzierbarkeit macht es möglich konvexe Optimierungsmethoden wie das Gradientenverfahren und Nesterovs Methode anzuwenden, um sie zu minimieren.

Im ersten Teil (Kapitel 2 bis 4) wird zunächst das Problem in den Kontext des Chip-Designs eingebettet und mathematisch formuliert. Danach werden die verwendeten Verbindungslängen-Schätzungen definiert und die beiden konvexen Optimierungsmethoden vorgestellt. Da nur bestimmte Zielfunktionen kompatibel mit diesen Methoden sind, werden die analytischen Eigenschaften der zwei Netzlängen-Schätzungen (aus denen dann Zielfunktionen werden) zusammengefasst und etwas erweitert. Die auf LogSumExp basierende wird mit NLSE abgekürzt und die auf Weighted-Average basierende mit NWA. Beide approximieren HPWL abhängig von einem Parameter γ beliebig gut, wobei NWA bei gleichem γ den geringeren absoluten Fehler hat. NLSE ist konvex und NWA ist es nicht, obwohl dies in [HCB11] behauptet wird. NWA verletzt die Konvexitätsbedingungen allerdings nur leicht und ist von oben und unten durch eine konvexe Funktion beschränkt. Zusätzlich besitzen

beide Netzlängen-Schätzungen einen lipschitzstetigen Gradienten. Diese Eigenschaften gelten ebenfalls für die Netzlängen-Schätzungen wenn man sie als Funktionen der Zellpositionen anstatt der Pinpositionen betrachtet.

Der zweite Teil (Kapitel 5 und 6) kümmert sich um die praktische Anwendung dieser Netzlängen-Schätzungen im unbeschränkten Optimierungsproblem, das oben beschrieben wurde. Nach ein paar Details zur Implementation werden hier NLSE und NWA mit Hilfe der konvexen Optimierungsmethoden auf mehreren Instanzen und mit mehreren Parameter-Kombinationen minimiert. Die Ergebnisse werden danach verglichen mit denen, die durch das Minimieren von HPWL bzw. quadratischer Netzlänge entstehen. Es stellt sich heraus, dass ein relativ hoher γ-Wert notwendig ist, damit die Konvergenzeigenschaften von NLSE bzw. NWA gut genug werden. Leider flacht mit einem hohen γ Wert die Zielfunktion rund um das Optimum extrem ab und sehr verschiedene Zellplatzierungen können ähnlich niedrige Werte der Zielfunktion annehmen. Außerdem kann es selbst mit hohen γ-Werten passieren, dass die Norm des Gradienten nicht zuverlässig während des Optimierungsprozess zu Null konvergiert, was das Standard-Abbruch-Kriterium unbrauchbar macht. Fixiert man stattdessen eine gewisse Anzahl an Iterationen, die die Laufzeit in einen akzeptablen Bereich bringen, so sind die Resultate sowohl in ihrer Qualität als auch in der benötigten Laufzeit, um diese zu erreichen meist schlechter als Platzierungen, die durch das Minimieren von quadratischer Netzlänge entstehen.

Bibliography

[AMS08] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar. *Handbook of Algorithms for Physical Design Automation*. 1st. 2008 (cit. on p. 3).

[Arm66] L. Armijo. "Minimization of functions having Lipschitz continuous first partial derivatives." In: *Pacific J. Math.* 16.1 (1966), pp. 1–3 (cit. on p. 13).

[BB88] J. Barzilai and J. M. Borwein. "Two-Point Step Size Gradient Methods". In: *IMA Journal of Numerical Analysis* 8.1 (1988), pp. 141–148. DOI: 10.1093/imanum/8.1.141 (cit. on p. 13).

[BS05] U. Brenner and M. Struzyna. "Faster and Better Global Placement by a New Transportation Algorithm". In: *Proceedings of the 42nd Annual Design Automation Conference*. 2005, pp. 591–596. DOI: 10.1145/1065579.1065733 (cit. on p. 8).

[BV01] U. Brenner and J. Vygen. "Worst-case ratios of networks in the rectilinear plane". In: *Networks* 38.3 (2001), pp. 126–139. DOI: 10.1002/net.1031 (cit. on pp. 29, 31, 33, 35).

[BV04] S. Boyd and L. Vandenberghe. "Convex functions". In: *Convex Optimization*. 2004, pp. 67–126. DOI: 10.1017/CBO9780511804441.004 (cit. on pp. 13, 18, 19, 26, 32).

[BV08] U. Brenner and J. Vygen. "Analytical methods in VLSI placement". In: *Handbook of Algorithms for VLSI Physical Design Automation* (2008), pp. 327–346 (cit. on pp. 4, 5, 8, 50).

[Cha+06] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. "MPL6: Enhanced Multilevel Mixed-Size Placement". In: *Proceedings of the 2006 International Symposium on Physical Design*. 2006, pp. 212–214. DOI: 10.1145/1123008.1123055 (cit. on p. 8).

[Che+08] T. Chen, Z. Jiang, T. Hsu, H. Chen, and Y. Chang. "NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.7 (2008), pp. 1228–1240 (cit. on p. 8).

[Coo11] J. D. Cook. "Basic properties of the soft maximum". In: (2011) (cit. on p. 39).

[DPM18] C. Durkan, G. Papamakarios, and I. Murray. "Sequential Neural Methods for Likelihood-free Inference". In: *arXiv e-prints*, arXiv:1811.08723 (2018), arXiv:1811.08723 (cit. on p. 39).

[GJ77] M. R. Garey and D. S. Johnson. "The Rectilinear Steiner Tree Problem is *NP*-Complete". In: *SIAM Journal on Applied Mathematics* 32.4 (1977), pp. 826–834. DOI: 10.1137/0132071 (cit. on p. 7).

[GL13] G. H. Golub and C. F. van Loan. *Matrix Computations*. Fourth. 2013 (cit. on p. 20).

[GP17] B. Gao and L. Pavel. "On the properties of the softmax function with application in game theory and reinforcement learning". In: *arXiv preprint arXiv:1704.00805* (2017) (cit. on pp. 18, 19, 21).

[HCB11] M.-K. Hsu, Y.-W. Chang, and V. Balabanov. "TSV-Aware Analytical Placement for 3D IC Designs". In: *Proceedings of the 48th Design Automation Conference*. 2011, pp. 664–669. DOI: 10.1145/2024724.2024875 (cit. on pp. 9, 22–24, 26, 36, 73, 75).

[Kle+91] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. Antreich. "GORDIAN: VLSI placement by quadratic programming and slicing optimization". In: *IEEE Trans. on CAD of Integrated Circuits and Systems* 10 (1991), pp. 356–365 (cit. on p. 8).

[KW06] A. B. Kahng and Q. Wang. "A Faster Implementation of APlace". In: *Proceedings of the 2006 International Symposium on Physical Design*. 2006, pp. 218–220. DOI: 10.1145/1123008.1123057 (cit. on p. 8).

[Lan+14] M. Lange, D. Zühlke, O. Holz, and T. Villmann. "Applications of lp-Norms and their Smooth Approximations for Gradient Based Learning Vector Quantization". In: *ESANN*. 2014 (cit. on p. 25).

[Lav+16] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer. *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*. 2nd. 2016. DOI: 10.1201/9781315215112 (cit. on p. 3).

[Lu+15] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. "EPlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method". In: *ACM Trans. Des. Autom. Electron. Syst.* 20.2 (2015), Art. No. 17. DOI: 10.1145/2699873 (cit. on pp. 9, 13, 27).

[Nam+05] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. "The ISPD2005 Placement Contest and Benchmark Suite". In: *Proceedings of the 2005 International Symposium on Physical Design*. 2005, pp. 216–220. DOI: 10.1145/1055137.1055182 (cit. on p. 8).

[Nam06] G.-J. Nam. "ISPD 2006 Placement Contest: Benchmark Suite and Results". In: *Proceedings of the 2006 International Symposium on Physical Design*. 2006, p. 167. DOI: 10.1145/1123008.1123042 (cit. on p. 8).

[NDS01] W. C. Naylor, R. Donelly, and L. Sha. “Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer”. In: *US Patent 6,301,693* (2001) (cit. on p. 8).

[Nes04] Y. Nesterov. “Smooth Convex Optimization”. In: *Introductory Lectures on Convex Optimization: A Basic Course*. 2004, pp. 51–110. DOI: 10.1007/978-1-4419-8853-9_2 (cit. on pp. 11, 13, 20).

[Nes83] Y. Nesterov. “A method for solving the convex programming problem with convergence rate $O(1/k^2)$”. In: *Dokl. Akad. Nauk SSSR* 269 (1983), pp. 543–547 (cit. on pp. 14, 15, 46).

[Ray97] M. Raydan. “The Barzilai and Borwein Gradient Method for the Large Scale Unconstrained Minimization Problem”. In: *SIAM Journal on Optimization* 7.1 (1997), pp. 26–33. DOI: 10.1137/S1052623494266365 (cit. on p. 13).

[Sut+13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. “On the importance of initialization and momentum in deep learning”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. 3. 2013, pp. 1139–1147 (cit. on p. 14).